الأكاديمية العربية للعلوم والتكنولوجيا والنقل البحري

Arab Academy for Science, Technology & Maritime Transport

College of Computing & Information Technologies

B. Sc. Final Year Project

Autonomous Underwater Vehicle (AUV) for Underwater Exploration

Presented By:

Ahmed Samy Zaghloul *Mariam Ehab Fayek*

Youssef Ahmed Mohana *Maya Saad Badr*

Haneen Salah El-Din Warda

Supervised By:

Dr. Yasmine Nagy *Dr. Mohamed El-Habrouk*

JULY - 2021

Acknowledgement

We would like to express our sincere gratitude to our supervisors Dr. Yasmine Nagy and Dr. Mohamed El-Habrouk for providing their precious time, guidance, comments, suggestion, and support throughout senior project I and senior project II. We would also thank our department Teacher assistants for constantly helping, motivating, and encouraging us. Finally, we would like to thank our families and friends for their support and attention.

Abstract

Egypt is known for its unique archaeology, oceanography and beautiful landscapes, which makes it an attraction for tourists. Due to COVID-19 pandemic, tourism was hardly affected, and thus affecting Egypt's economy negatively. As a result, this project focuses on thriving tourism in Egypt by making salvage and search operations for archaeological underwater artifacts and inspecting coral reefs. The Autonomous Underwater Vehicle (AUV) software is integrated using Robot Operating System (ROS). Moreover, the vehicle's vision system includes both Stereo Vision and Mono Vision Systems to input the video stream needed. The system then enhances the frames using RGHS technique and uses YOLO for object detection with an accuracy of 96.44%, a precision of 98%, a recall of 97%, an Intersection Over Union (IOU) 80%, and an F1-Score 98% to identify an object and measure the distance in real-time to this object while moving toward it autonomously. Throughout this project, a semi-autonomous underwater vehicle was produced that can be controlled by a pilot using a joystick and move autonomously toward a detected object.

TABLE OF CONTENTS

List of Figures

List of Tables

List of Acronyms

Acronym	Full Name
AUV	Autonomous underwater vehicle
ROS	Robot operating system
IOU	Intersection Over Union
GDP	Gross domestic product
RGHS	Relative Global Histogram Stretching Image Enhancement Technique
YOLO	You Look Only Once (Object Detection)
SIFT	Scale Invariant Feature Transform
HOG	Histogram Oriented Gradient
CNN	Convolutional Neural Networks
YARP	Yet Another Robot Platform
MRPT	Mobile Robot Programming Toolkit
AR	Augmented Reality
VR	Virtual Reality
ESC	Electronic Speed Controller
FC	Fully connected
CSP	Cross Stage partial
MSG	Message
SRV	Service
SRC	Source
CAD	Computer Aided Design
DXF	Drawing Exchange Format
STL	Standard Tessellation Language
UART	Universal Asynchronous Receiver/Transmitter

Chapter 1: Introduction

Chapter 1 illustrates the idea of the proposed project, AUV applications in Egypt, problem statement and motivation, and aims and objectives.

1.1 Introduction

Egypt is globally recognized for its distinguished geographical location that overlooks both the Mediterranean Sea and the Red Sea [1]. Each of which represents an important site for Egypt's tourism. For instance, the Mediterranean Sea is known to be a treasure source of underwater monuments such as shipwrecks, underwater cities, and historical artifacts [2]. Moreover, the Red Sea is known for its numerous and different fish species that live among various kinds of colourful coral reefs [3]. The Red Sea reefs are cited as one of the richest species in the world. Such species are endangered due to climate change and water quality [4].

1.2 AUV Possible Applications in Egypt

There are various applications in which an AUV can be used in Egypt, one of which is **Oceanography** [5], in which water pollution and climate changes can be detected using special tools and equipment in an AUV, and thus detect the factors that might lead to any natural disaster and be ready in advance. **Archaeology and Exploration** [2] is another application, where an AUV can be used to enhance visibility and object detection, in addition to facilitating underwater inspection without time limit or endangering the divers' lives. Furthermore, an AUV can be used to assist in **Ships' Repair Operations** [6] to inspect the ships' body, report its position, and send images of any damages that need special human experts in a less time-consuming process. In addition, using an AUV in **Gas Pipeline Inspection** [7] will provide more effective inspections at lower costs. This project will facilitate Oceanography and Archaeology and Exploration in Egypt, which is discussed in detail in the Aims and Objectives section.

1.3 Problem Statement and Motivation

This project addresses numerous problems. There are various sunken ruins and cities in which to be discovered, for instance the remains of the Pharos' Era in Alexandria that have been discovered [8]. In addition, there is a remarkable reduction in Egypt's tourism due to COVID-19 global pandemic, which affects its economy and reduces its national GDP by 0.7 or 0.8 percent monthly [9]. Moreover, there is a great difficulty in accessing archaeological sites underwater [10], which unfortunately results in the deaths of divers during underwater inspection [11]. The following figures illustrates some of the difficulties and challenges that face these drivers:

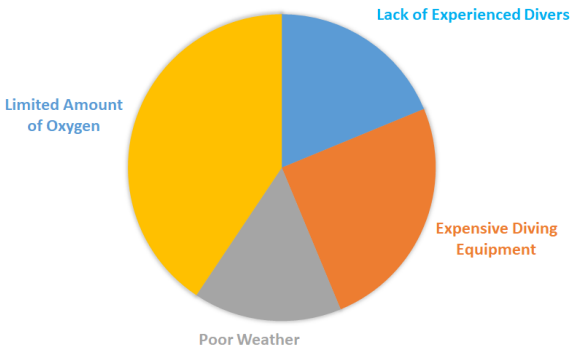


Figure 1-1: Challenges Facing Underwater Archaeology [9]

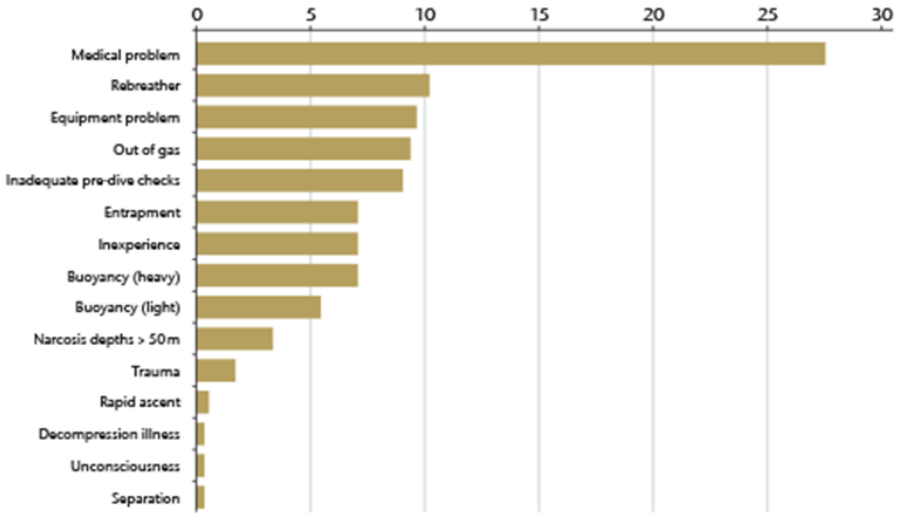


Figure 1-2: Factors Leading to Diving Incidents [9]

1.4 Aims and Objectives

The project aims to encompass thriving tourism by building an autonomous underwater vehicle (AUV) for inspection and exploration of the undiscovered Egyptian historical monuments (underwater cities, shipwrecks, and remnants) in the Mediterranean Sea and the coral reefs in the Red Sea without endangering the divers' lives. These aims are to be achieved through following the following objectives:

- Build the mechanical structure of AUV.
- Analyse, design, and implement the electrical structure.
- Design and program AUV control unit.
- Prepare a test plan for AUV.
- Implement the system outcome of the requirements.
- Test the performance of an Autonomous Underwater Vehicle.

1.5 Project Documentation Outline

In **chapter 2**, the literature survey, related work, and technical background in which the concepts that are close to our own in concept will be discussed.

In **Chapter 3,** the chosen methodology and technology chosen after the literature survey and research will be explained.

In **chapter 4,** the implementation of the AUV will be explained in detail. The project implementation is divided into 3 main systems: the mechanical system, the electrical system, and the software system.

In **chapter 5,** the results, testing and challenges which have faced us will be discussed.

In **chapter 6,** the conclusions, future work which we hope to implement in the future, a summary about the project will be illustrated.

In the appendix, a Gantt chart of Project I and another for Project II, along with the total cost will be displayed.

1.6 Summary

In chapter one, the project main idea and objectives were discussed. In chapter 2, the literature survey conducted will be discussed along with the research conducted on the software technology needed to choose the best for the AUV.

Chapter 2: Literature Survey

2.1 Introduction

In chapter 2, a literature survey about commercial and prototype AUVs will be displayed in section 1 and 2, explaining each AUV's specs and functions. Moreover, section 3 will discuss the technical survey that was conducted to choose the best technology and platforms for the project.

2.2 Commercial AUVs

2.2.1 Solus-LR

Cellula [12] manufactured the Solus-LR AUV in 2019 [13], shown in figure 2-1, and the Imotus Hovering AUV in 2020 [14]. Solus-LR is classified as an extra-Large unmanned Underwater Vehicle (XLUUV). In addition, it has a modular design allowing payloads to be added or removed depending on mission specifications. It is then capable of performing multi-month autonomous missions of surface and subsurface surveillance, deployment and recovery of payload packages, geophysical survey, pipeline and oil field servicing, port-to-port missions, and anti-submarine warfare.

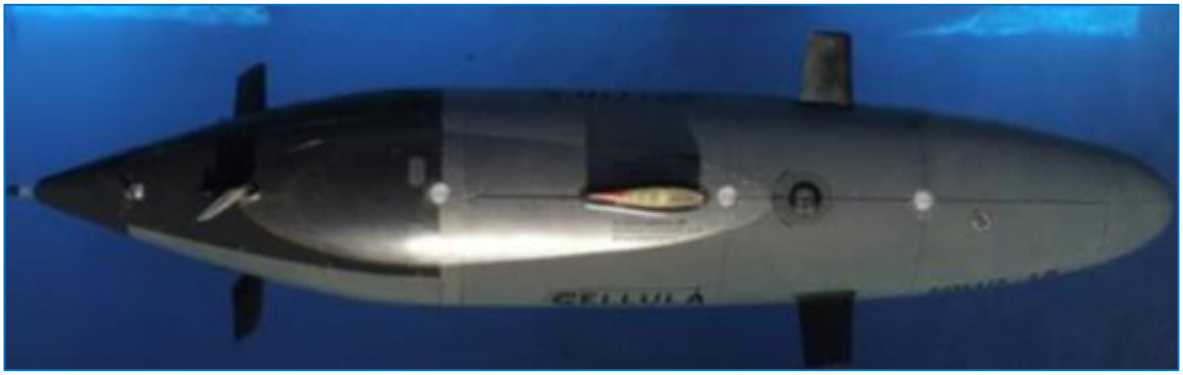


Figure 2-1: Solus-LR [13]

2.2.2 Imotus Hovering

The Imotus Hovering AUV has a similar modular design. It also possesses flexible architecture. It uses Robotic Operating System (ROS) [15] over Ethernet to communicate between navigation sensors, main computer, and payload modules. This enables the vehicle to perform like the previous one.

2.3 Prototype AUVs

Several prototypes have been proposed by research teams all over the globe.

2.3.1 Albatross

Tartan AUV (TAUV) [16] proposed the **Albatross** in 2019 [17] and the **Kingfisher** in 2020 [18]. **Albatross**, depicted in figure 2-2, was the first AUV introduced by the Tartan AUV. Their aim was to build a simple yet effective AUV to gain experience for future work. Albatross's advantages include the simplicity and reliability of the design making it easy to maintain. To provide manoeuvrability, they designed a lightweight and easy to control AUV. Moreover, they tested the vehicle's software offline using simulators such as UUV simulator, which helped the software in decision-making and faster iteration. The only drawback was that Albatross's design was way too simple to base their future work on [19].

Figure 2-2: Albatross [17]

2.3.2 Kingfisher

On the other hand, **Kingfisher**, presented in figure 2-3, has an AUV platform, compared to that of Albatross, to be used in the future. Advantages of Kingfisher's include its extendibility, vehicle design modularity and versatility, ease of maintenance and repair due to the accessibility of internal electronics, with enhanced thermal management of internal electronics. Furthermore, Kingfisher's software includes 3D perception and accurate navigation to support doppler velocity logger (DVL), rotating sonar and stereo vision [20].

Figure 2-3: Kingfisher [18]

2.3.3 Matsya 6

The Indian Institute of Technology Bombay (IITB) [21] proposed **Matsya 6** [22]. Matsya 6, depicted in figure 2-4, was built in 2020 aiming to provide a research platform for underwater robotics and autonomous systems. Matsya 6 is the latest AUV version developed from the Matsya series [23-27]. It included smaller vehicles for better manoeuvrability by repositioning of AUV components, in addition to a simplified mission planner for more reliability. Their design team faced technical problems while building this AUV. Heating issues in the main hull led to multiple electrical failures. This issue was addressed by developing proper heat dissipation mechanism, which included modifications of GPU's base plate and a new hull design for the Electronic Speed Controller (ESC). Moreover, a new camera driver was added to prevent the cameras from disconnecting mid-run [28].

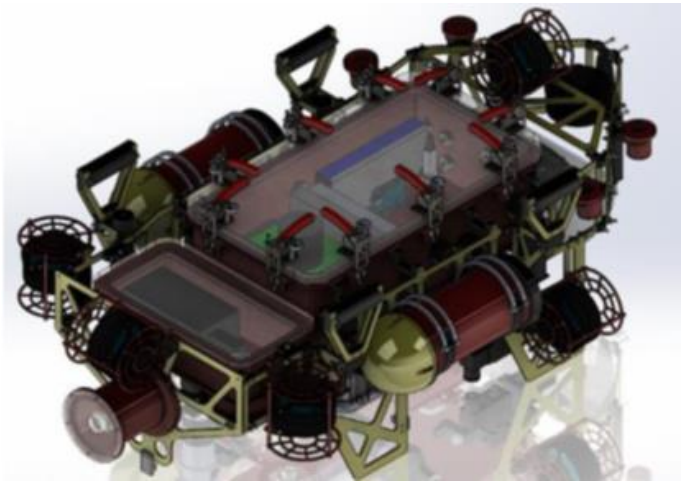


Figure 2-4: Matsya 6 [22]

2.4 Technical Background

Throughout the project, a research phase was continuously ongoing, concerning especially the software, to understand the methodologies by which the previous worked to choose the best ones for the proposed project. Examples of the fields that are needed and in which research were conducted:

1. Image Enhancement Techniques
2. Object Detection Techniques
3. Robot Application Development Platform
4. Virtual Reality and Augmented Reality

2.4.1 Image Enhancement Techniques

Many research papers proved that image enhancement is critical for better results for object detection, especially underwater, since images face light problems and noise. There exists a plethora of underwater image enhancement techniques discussed in paper “A Comprehensive Overview of Image Enhancement Techniques. Arch Compute Methods Eng. (2021)” [29]. The techniques which were searched, analysed, and implemented to choose from include Image enhancement by Histogram Transformation (HE) [30], Log Transformation Function for image enhancement [31], Contrast limited adaptive histogram equalization (CLAHE) [32], Gamma Correction (GC) [33], Underwater Image Enhancement Using an Integrated Colour Model (ICM) [34] and Relative Global Histogram Stretching (RGHS) [35]. RGHS was chosen as the image quality is the best compared with the others previously mentioned techniques.

2.4.2 Object Detection

According to the conducted research, object detection can be achieved through either machine learning approach or deep learning approach. Applying object detection through a machine learning approach is classified into Scale Invariant Feature Transform (SIFT) [36] and Histogram Oriented Gradient (HOG) [37]. While applying object detection through deep learning is classified into convolutional neural networks (CNN), R-CNN family and YOLO. The machine learning approach produces less accurate features than deep learning. As a result, deep learning is applied using the YOLO technique, which is explained in the research paper "You Only Look Once: Unified, Real-Time Object Detection'' [38]. YOLO is considered fast and smart for real time operation, as shown in figure 2-5.

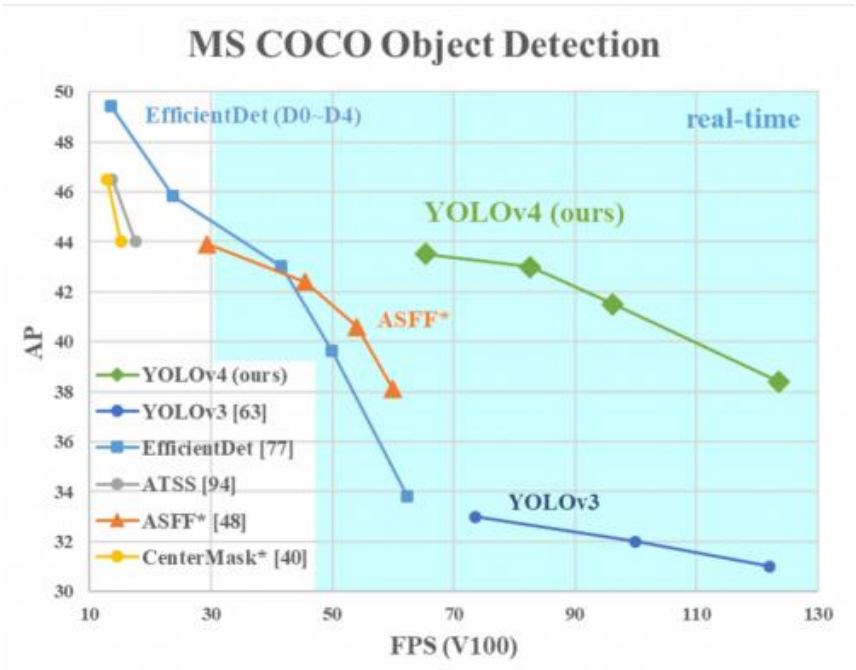


Figure 2-5: Comparison Between the Object Detection Techniques [39]

2.4.3 Robot Application Development Platform

There are various robotic platforms that can be used to build an autonomous mobile robot. For instance, Player, Yet Another Robot Platform (YARP), Orocos, Mobile Robot Programming Toolkit (MRPT) and Robot Operating System (ROS) [40]. ROS is the robotic platform chosen due to its high-end capabilities, modularity, Concurrent resource handling, active community and tons of tools, such as RViz, Gazebo for simulation, rqt_graph in which all the ROS nodes running in the system are shown in the form of a graph with the topics that are connecting them [84], and CV-bridge in which ROS send images in computer vision, which is a field of science involved with how the vehicle perceive the outside world through cameras. The camera stream passes through a pipeline starting from image enhancement to object detection. Those enhancements are made through OpenCV, which is a library involved with computer vision and image enhancement [41]. The detection process is powered by tensor flow, which is an open-source library conducive to designing deep learning and machine learning applications. Its format is in the form of sensor_msg/image message, but this format has to make conjunction with OpenCV, that's why CV-bridge converts image ROS format to OpenCV format.

2.4.4 Virtual Reality and Augmented Reality

2.4.4.1 Virtual Reality (VR)

Virtual Reality is creating a virtual world experience also called computer-simulated reality. It is a computer technology using headsets to generate real sounds, images and other things

that replicate a real environment to create an imaginary world. It is better to engage all five senses (taste, sight, smell, touch, and sound) to make a true VR environment. VR is like any magic trick by fooling the brain into accepting them as reality, to quote Morpheus, “your mind makes it real”. Here is how it happens. [42]

Figure 2-6: Oculus Quest Headset

2.4.4.1.1 Virtual Reality (VR) Displays

Visual VR devices can be divided into three different categories: Mobile phone-based VR, Consumer grade HMDs and CAVE systems, capable of providing feedback for multiple users.

1. **Head-mounted Displays (HMDs):**

They are small displays or projection technology integrated into eyeglasses or mounted on a helmet or hat. An emerging form of heads-up display is a retinal display that “paints” a picture directly on the sensitive part of the user’s retina. This type of display is meant for a total immersion of the user in whatever experience the display is meant for, as it ensures that no matter where the user’s head may turn, the display is positioned right in front of the user's eyes. [43][44]

2. **Mobile phone-based VR:**

In this case a mobile phone is put into a VR Headset which encompasses two lenses to provide a stereo view. Immersion in the VR environment is then achieved by rendering the scene for the point of view of a user determined by the internal sensors of the mobile device. [45]

3. **Cave Systems:**

They have been widely used at universities and research centres in the past, when results with HMDs were not satisfactory for VR. In contrast to HMDs and mobile phone-based VR, the displays are in this case not worn, but surround the users. The position and orientation of the user and (usually) her controllers are determined using motion capture systems; shutter glasses are used to provide a stereoscopic view. [45]

2.4.4.1.2 The main features of VR systems

1. Immersive:

VR needs to interact with user senses to ensure users believe they are physically present in the real world. Some VR devices use various interactive devices such as head-mounted displays and data gloves to seal the viewer's vision, hearing, and other senses and use interactive devices to operate and control the virtual environment, so the viewer truly becomes a participant in the VR system creating a sense of being immersed and fully engaged. [46][47]

2. Interaction:

The virtual world should be interactive and fast to make the experience more realistic. Having the ability to pick up objects in the scene or even interact with characters can improve immersion even further and add to the value of VR simulations. [46][47]

2.4.4.2 Augmented Reality (AR)

Augmented reality is creating digital augmentation overlaid in the real-life environment. It creates a mixture of real and virtual feedback perceived in collected space. Augmented reality appears in direct view of an existing environment and adds sounds, videos, graphics to it. [48-50]

Figure 2-7: Example of Augmented Reality

2.4.4.2.1 Augmented Reality Methodology

Augmented reality needs some components to work, which are:

- **Processor:** It determines if your device can manage the AR requirements. It works like the brain of the device. [50][51]
- **Graphic Processing Unit (GPU):** It handles the visual rendering of a device's display. [50]
- **Camera and Sensors:** It is used to collect data required to make sense of our environment. There are common sensors required for AR include: [48][51]
    a. Depth sensor
    b. Gyroscope
    c. Proximity Sensor
    d. Accelerometer
    e. Light sensor
- **Projection:** It takes data from the sensors and cameras to project the result of processing onto the surface to view. It turns the surface into an augmented reality interactive environment. [48]
- **Reflection:** AR devices use small curved, double-side mirrors to help to path the graphically changed images to the human eyes. Those cameras reflect light to a camera and then to the user's eye. [48]

2.4.4.2.2 The Different Types of AR Experiences

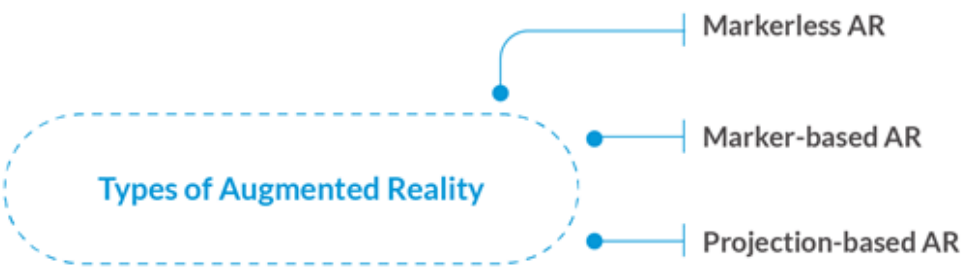


Figure 2-8: Types of Augmented Reality

1. **Marker-based AR:**

It is also called Image Recognition or Recognition based AR. In this type, we need visual marker (QR code) or image that the camera recognizes and processes to have the information about on object. This type is popular and easy to implement. [49][50]

2. **Marker-less AR**

This type we move virtual objects in the physical space. We do not need visual markers for a camera to augment an image. One of the most famous examples of this type is the IKEA app. [49][50][52]

Figure 2-9: Example of Marker-Less AR

3. **Projection-based AR**

This type uses projection technology to display virtual images onto the screen or a physical surface. It is also used in industrial assembly and product visualization. [49][50][52]

Figure 2-10: Example of Projection-Based AR

2.4.4.2.3 Devices Augmented Reality Works On

1. Mobile Devices: Today's mobile apps has all the AR requirements to deliver realistic AR experiences.

2. AR-based Devices: It is also possible to experience AR with the help of special AR devices (wearable smart glasses, head-mounted apparatuses, GPs, CPU, gyroscopes, and displays).

3. AR Glasses: AR glasses are smart glasses that has different shapes and sizes, but they all have in common is the enhancement of reality with digital overlays.

4. AR Contact lenses: This technology is still not being realized. It will be able to display texts, translate speech into captions and has some navigation system. [48]

Figure 2-11: Comparison between VR and AR. [53]

2.5 Summary

In this chapter, the literature and technological surveys were discussed, and the chosen platforms and technology were shown. In chapter 3, the methodology by which each chosen platform, model or technology used will be explained.

Chapter 3: Methodology and Technology Description

3.1 Introduction

In this chapter, explanations about the computer vision system, the Robot Operating System (ROS), AR and VR will be displayed.

3.2 Computer Vision System

Computer vision is a field of science involved with how the vehicle perceives the outside world through cameras, the camera stream passes through a pipeline starting from image enhancement to object detection, those enhancements are made through OpenCV which is a library involved with computer vision and image enhancement [41]. The detection process is powered by tensor flow which is an open-source library conducive to designing deep learning and machine learning applications.

3.2.1 Relative Global Histogram Stretching (RGHS) Image Enhancement Technique

An effective shallow-water image enhancement method called relative global histogram stretching (RGHS) [54] is used to remove the hazing effect. This method consists of two parts: contrast correction and color correction is based on a simple histogram stretching in RGB color model, where it equalizes G and B channels. Then each RGB channel histogram are redistributed with dynamic parameters that relate to the intensity distribution of original image and wavelength attenuation of different colors under the water. After that, the bilateral filter is used to eliminate the noise while preserving the details of the desired colorful underwater image. The color correction is performed by applying a global histogram stretching on the 'L' component of the image and modifying 'a' and 'b' components in CIE-Lab color space in order to improve the saturation and brightness of the image to obtain a more vivid color.

3.2.2 Machine Learning, Deep Learning, Convolutional Neural Networks (CNN)

3.2.2.1 Machine Learning

Machine learning is an emerging branch of computational algorithms that are designed to imitate human intelligence by learning from the surrounding environment [55], where the user desires to input data to achieve a desired task without literally being programmed to produce a particular outcome [55][56]. Machine learning algorithms automatically adapt their

architecture through repetition so they gain experience at achieving the desired task. The process of adaptation is called training [56]. The algorithm keeps evolving so it can generalize the production of the desired outcome from new unseen data [57].

The learning process is done through training. The field of machine learning can be divided into 3 branches: supervised, unsupervised, and semi-supervised. Supervised learning is used to evaluate an unknown input and output mapped from known input and output samples [56] [57], where the output is labeled, whereas in unsupervised learning, input samples are given to the learning system. Semi-supervised learning is a compound of both supervised and unsupervised as part of the data is partially labeled and the labeled part is used to deduce the unlabeled [57].

3.2.2.2 Deep Learning

Undoubtedly deep learning is an extending branch of machine learning that keeps evolving to adequately explore each form of data across the globe [58]59]. This data is known as big data which is driven from the user experience through the internet across different platforms, where indeed this data feeds the multi-layer neural network containing two or more hidden layers [58].

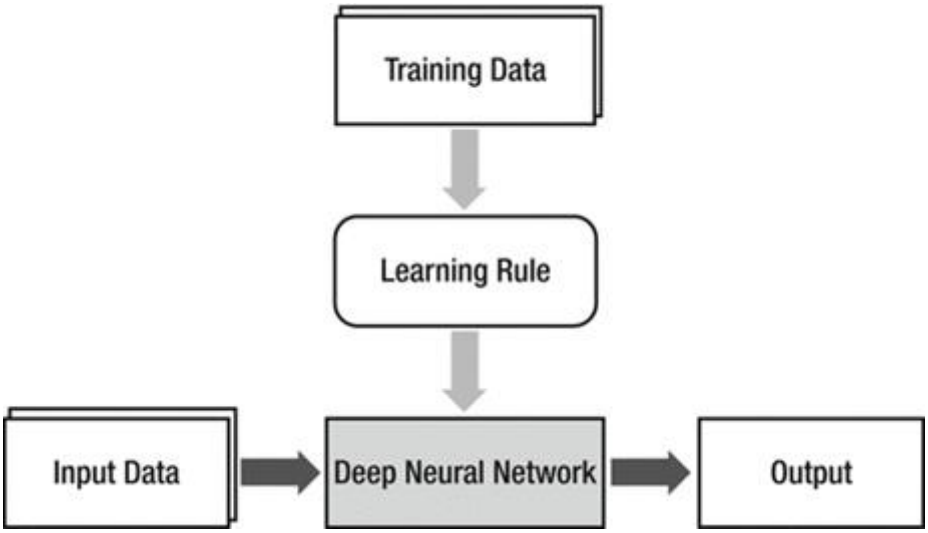


Figure 3-1: Deep Learning Steps

Each layer of this neural network builds on the previous layers with billions of data and features that if it was being processed by a human being would take many years to process [59].

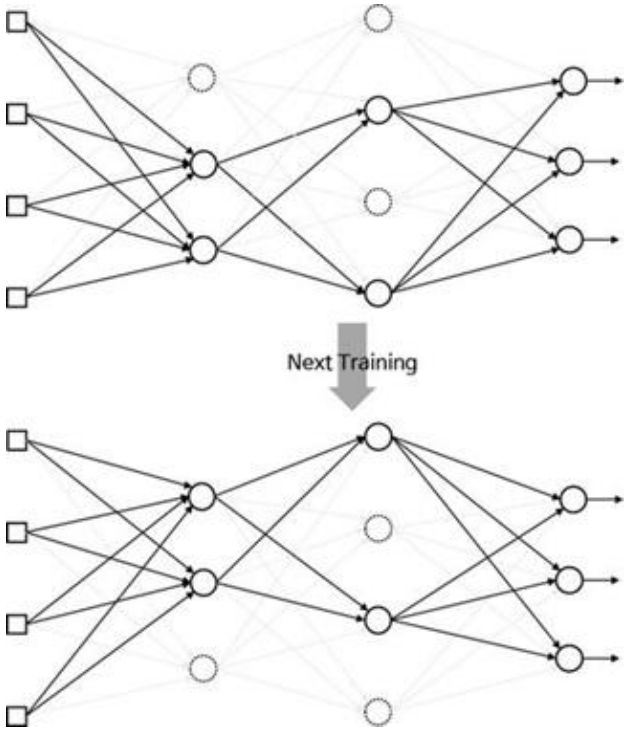


Figure 3-2: Neural Network Training

3.2.2.3 Data Gathering

Data gathering is a major part of the learning process as the quality of data affects the output of the deep learning model and that leads us to two critical topics regarding data gathering which are overfitting and under fitting [58] Overfitting is simply the data being trained too much that it captures noise [60], moreover it causes overall poor performance of the model, which is mostly due to low bias and high variance [60] [61]. However, under fitting is when the model fails to neither capture previously trained data nor generalize to new data, it occurs if the model shows low variance and high bias [60][61]. Previously, it was mentioned the overfitting and under fitting of models and the causes and the consequences of the overall model performance and that drives us to strive to achieve an ideal model which is common ground between bias and variance values [61].

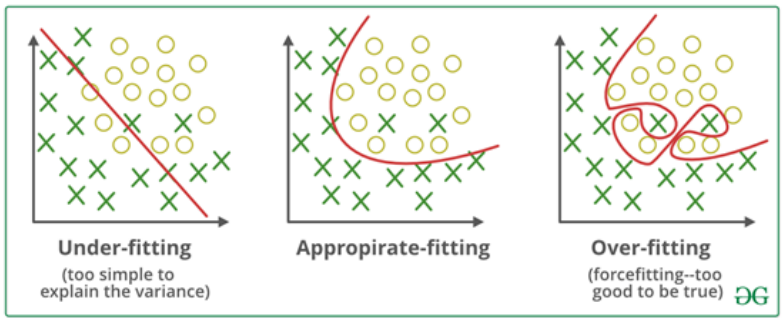


Figure 3-3: Data Gathering

3.2.2.4 Convolutional Neural Networks

Convolutional neural networks are end-to-end AI models that develop their own feature-detection mechanisms, and also recognize features through multiple layers [62].

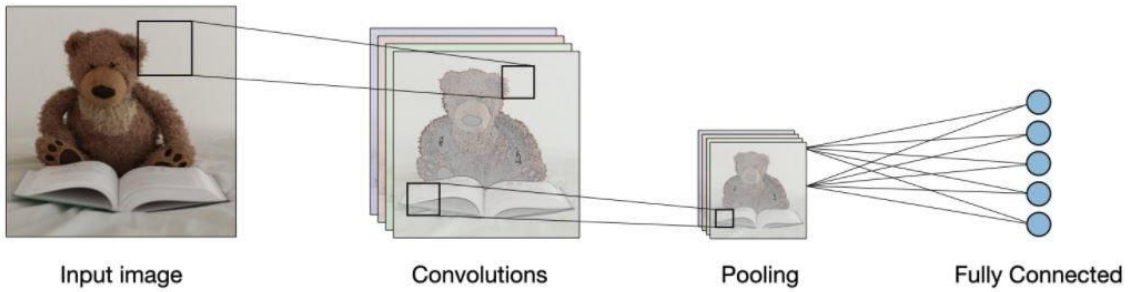


Figure 3-4: Example of Convolutional Neural Networks

Types of layers:

3.2.2.4.1 Convolution layer (CONV)

The convolution layer (CONV) uses filters that perform convolution operations because it is scanning the input I with relevance to its dimensions [62].

3.2.2.4.2 Pooling layer (POOL)

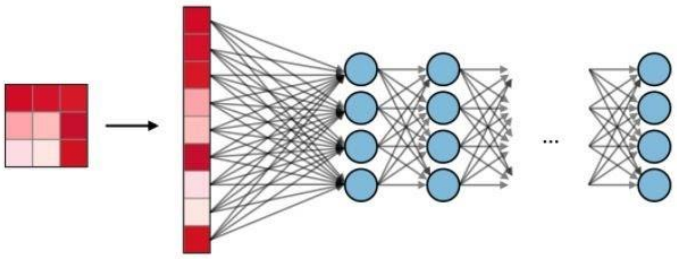
The pooling layer (POOL) may be a down sampling operation, applied following the convolution layer, reducing the spatial size of the extracted Features from the convolution layer [62]. There are two methods in pooling: Max pooling and Average Pooling. Table 3-1 compares between the two methods.

Table 3-1: Comparison between Max Pooling and Average Pooling [62]

Type	Max Pooling	Average Pooling
Purpose	• Each pooling operation selects the maximum value of the current scene.	• Each pooling operation averages the values of this view.
Comments	• Preserves detected features. • Most commonly used act as a Noise Suppressant. • Performs de-noising along with dimensionality reduction.	• Down samples feature map. • Performs dimensionality reduction as a noise suppressant mechanism.

3.2.2.4.3 Fully Connected (FC):

The Fully connected layer (FC) operates on flat input wherever every input is connected to all neurons [62].



Filter hyper parameters:

- Dimensions of a Filter: [62]

A filter of size F×F applied to an input containing C channels is a F×F×C volume that performs convolutions on associate input of size I x I x C so it produces an output feature map (also referred to as activation map) of size O×O×1 [62].

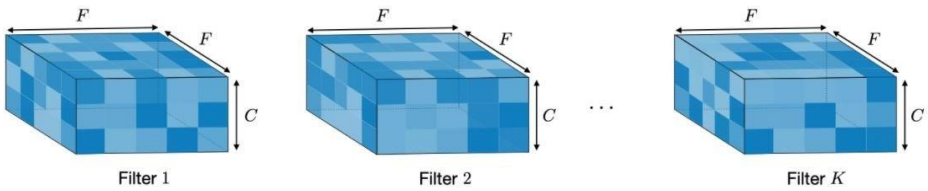


Figure 3-6: Filter Hyper Parameters

- Stride:

For a convolutional or a pooling operation, the stride S denotes the number of pixels by which the kernel moves after each operation [62].

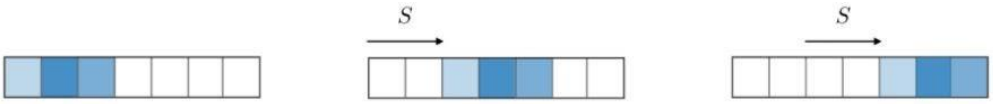


Figure 3-7: Stride

- Zero-padding:

Zero-padding denotes the method of adding P zeroes to every side of the boundaries of the input [62].

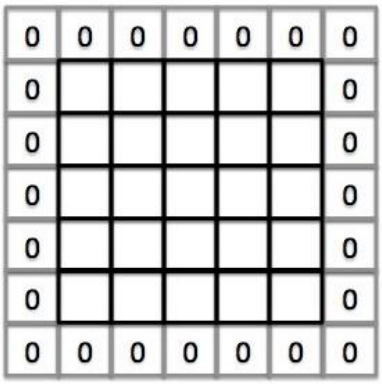


Figure 3-8: Zero Padding

3.2.3 You Only Look Once (YOLO) Object Detection Technique

3.2.3.1 Introduction

Object detection is an important topic to recognize and detect the object. The object detection includes two parts: the first part is classification and second is the localization. To classify objects, need convolution neural networks for classifying objects. To predict the localization of the object the detector use method called bounding box, so the proposed model contains two parts of object detection

3.2.3.2 Definition

YOLO [63] is called the (You only look once) algorithm. YOLO take image and divide it into S*S grid and each grid contain an anchor box, which include a bounding box and the confidence. The confidence reflects the accuracy of bounding if it contains an object or not. YOLO also predicts the classification score for each box for every class in training. The output of algorithm is one-dimension vector contain all object that detected in the image

3.2.3.3 Architecture of Model

This model has high accuracy and high real time speed. The architecture of the model is divided into four blocks shown in figure 3-9.

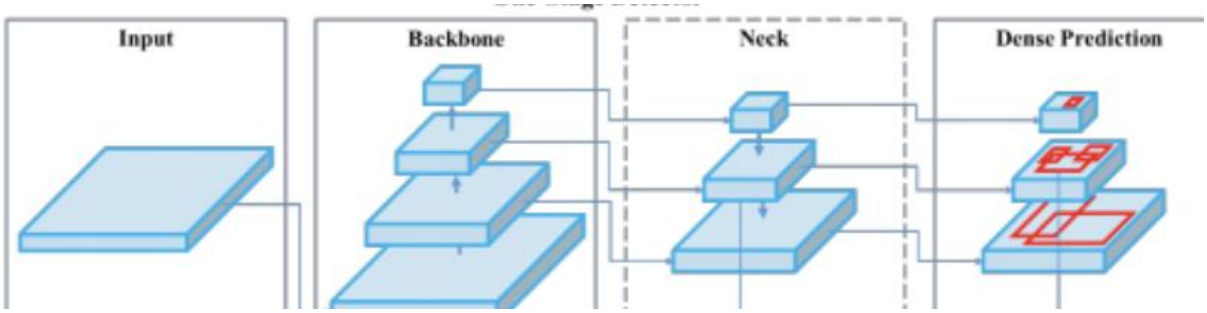


Figure 3-9: YOLO Architecture [64]

Blocks of model

A) Input layer

B) Backbone (**Cross-Stage-Partial Darknet [65]**)

C) Neck (**Path aggregation network [66], Spatial pyramid pooling layer [67]**)

D) Head (**Dense Prediction [68]**)

A) Input layer

This layer works to get an image as the parameter for the network. The layer converts the image into a matrix (Values, size image for network, number of channels for image). After all of this the matrix passed for the Backbone block.

B) Backbone

Backbone refers to the feature extraction block, which contains a network called Cross-stage-partial Darknet 53 [65]. The main components of this network are:

1. Convolution Layer
2. Pooling layer
3. Activation Function
4. Residual Block
5. Dense Block

6. Cross Stage Partial
7. Darknet53 Network
8. CSP Darknet 53

1. Convolution Layer

Convolution layer [69] is a sliding window that moves on the entire image and makes convolution operation (dot product of window with input matrix). Specifying the parameters of this window is used to search in the image for certain features. The result of the convolution layers is the feature map, which defines the main features in the image.

Filter hyperparameter

The convolution layer has a hyperparameter called Filter hyperparameters. This hyperparameter consists of the following parameters.

- **Filter dimension**

The filter has a size of (FF), where F is the value specified for this parameter.

- **Strade**

Strade donates the number of pixels the filter moves after each operation. The strade parameter is used in both the convolution operation and pooling operation.

- **Padding**

Padding is a value specified to define a neglected region at each side of the boundaries of the image. This value can be added manually or automatically. There are different types of padding:

A. Valid

There is no padding for the input image and drops last convolution if dimensions do not match.

B. Same

The padding ensures the output image has the same size as the input image.

C. Full

Full padding ensures each side of the boundaries of the image is added by value. The purpose is math convolution to all the image.

2. Pooling Layer

The pooling layer [70] is located after the convolution layer operations. Because the output feature map from the convolution layer is sensitive for the location of the feature map in the input so the solution is using the pooling layer. Pooling layer do operation called down sampling in order to a summarization of the feature map so the problem of the convolution is solved. There are Two types of pooling: Average pooling and Max pooling. Average pooling makes averaging for the values in the current sliding window and the output from this window is the down sample of the feature map. Max pooling takes the max value in the current window and the main benefit of applying max pooling is for detention of certain features in the feature map.

3. Activation Function

The main purpose of the activation function [71] is for adding non-linearity between the input and output. Nonlinearity solves complex operations like segmentation and detection as the linearity relationship can't solve it. There are different activation functions but the main focus for the YOLO model is Relu activation function, Mish activation function, SoftMax activation function:

A. Relu Activation Function

This activation function rule [72] works if the input is greater than zero then the output is the input and if less than zero the output is zero. Relu [73] works linearly if the input is greater than zero which means the backpropagation in the neural network works by linearity relation and any input less than zero is zero means that the activation function in this part is non-linear. The Relu represents a nearly linear function and properties of linear model help, with gradient-descent method [74]. The main problem of Relu is that it is stuck in the negative side as the Relu neuron will be stuck there and impossible to recover back [74].

B. Mish Activation Function

The mish activation function [75] work by the formula $f(x) = x\ tanh(\text{softplus}(x))$, where $\text{softplus}(x) = \ln(1 + e^{x})$ is the softplus activation [76]. The Mish function works better than Relu as Relu causes some problems in gradient optimization. In the graph of mish at figure 3-10 the left side of the graph shows the output is bounded under the curve that helps for strong regularization effects and reduces overfitting. Right side of the activation function shows this activation function has important properties to avoid saturation that causes slow down training to be near zero gradients. The Mish activation function is better than Relu so the YOLO model uses the mish function for better performance.

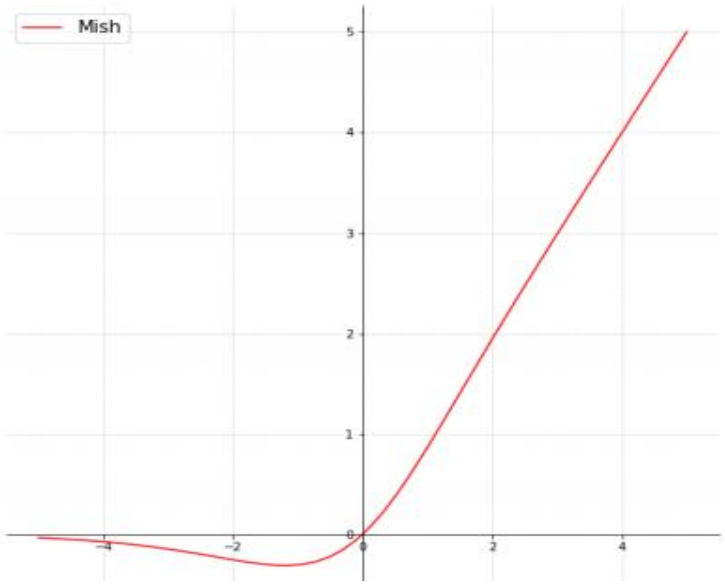


Figure 3-10: Mish Activation Function Graph [75]

C. SoftMax Activation Function

The SoftMax function [74] localized in the output layer in the neural network, which works to get the vector of numbers and convert it to a vector of probabilities, which have values between zero and one. The summation of the vector is equal to one. Where the output vector is the probabilities of each class in the model. The formula of this function is:

$$F(xi)=exp(xi)jnexp(xj)-(1.12)\ [74]$$

The main purpose of the softmax is to be used in models that have multi-class. The YOLO model is multi-class, so the softmax function is included in the model.

4. Residual Block

Residual block [77] created to solve problems called degradation. The degradation, also called vanishing and exploding gradient, this problem is common in deep neural networks especially with the increase in depth. Which the network suffers from trouble reaching convergence due to the rapid degradation The Vanishing gradient mean that weights in the layer is very small near or vanishing, when calculation of stochastic gradient descent by using backpropagation [78] the gradient value for the network very small near to be not a number.

Residual blocks [77] solve this problem, which allow the activation from one layer and fed it for the deeper layer in the network. The activation feed from layer to another in residual block by using skip connotation as shown in figure 3-11, Where the figure shows x is the identity which mean activation function value from this layer and f(x) mean the wight from

previous layer and line take x into addition operator called skip connection and feed the layer by x +f(x).

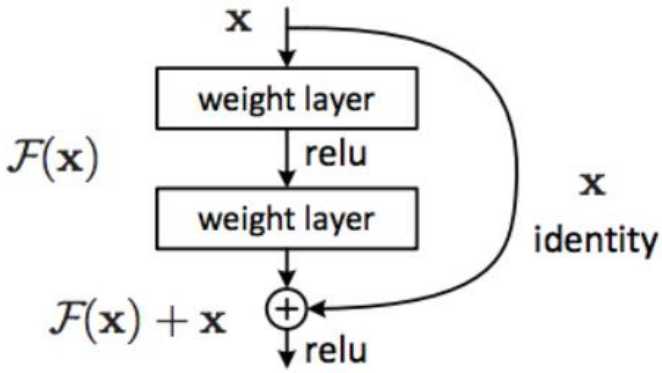


Figure 3-11: Relu Block [77]

5. Dense Block

A Dense Block [79] was produced to solve the problem of vanishing of gradients by using feature maps from the previous layer and concatenating them onto the input future layers in the network. Dense block contains multiple convolution layers with each layer connected to a layer called Hi that consists of both batch normalization and Relu function and followed by convolution. It works like Residual Block [78], which uses a skip connection where it doesn't take output from the last layer only, it takes the output from all previous layers as shown in figure 3-12. The feature map depends on the growth rate of the network. The dense block can be formed as a network called dense net which is composed of multiple dense blocks with a transition layer in between the convolution and pooling. Both the Dense network and the residual network have bottleneck problems, where the computation complexity is still slow, so the solution for this problem is the use of **Cross stage partial connection network [65].**

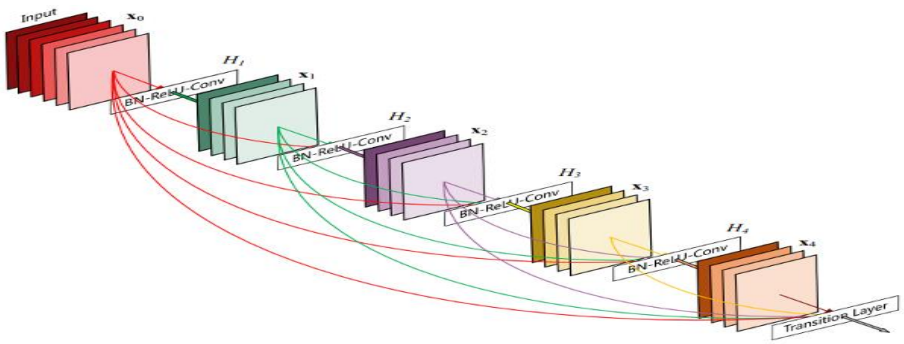


Figure 3-12: Dense Block [79]

6. Cross Stage Partial Connections

This idea depends on Dense net [79] where the input feature maps of the dense Block are separated into two parts, one will go through a block of convolution and one will aggregate the results as shown in figure 3-13. The figure shows the difference between Cross stage dense net and normal densest.

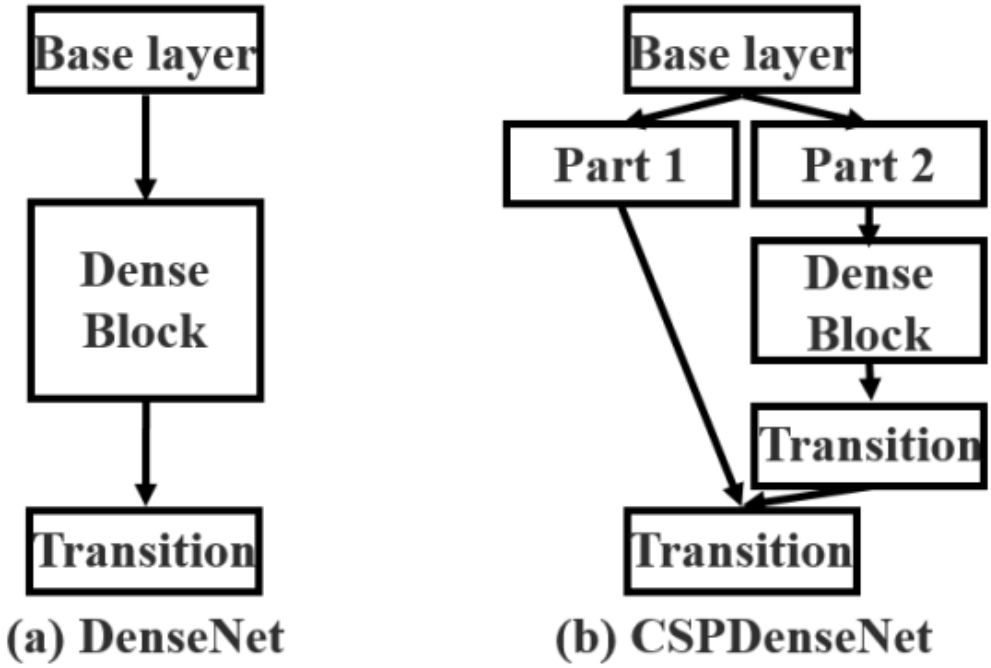


Figure 3-13: Dense Block and CSPDensenet [65]

This design is created to reduce the computational complexity by separating the input into two parts, where one goes through the Dense block, and another goes direct for the transaction layer.

7. Darknet53

This pretrained network [63] architecture contains 53 convolution layers each followed by a Relu activation function and batch normalization. The convolution with stride two is used to sample the feature maps to make the output vector size from the network equal to the size of the input image. To prevent the loss of low-level features and give the model the ability to detect small objects. The network contains residual blocks to solve the vanishing of gradients problem. Figure 3-14 shows the detail of the network where the network contains a three-layer feature map with different scales.

	Type	Filters	Size	Output
	Convolutional	32	3 × 3	256 × 256
	Convolutional	64	3 × 3 / 2	128 × 128
1×	Convolutional	32	1 × 1	
	Convolutional	64	3 × 3	
	Residual			128 × 128
	Convolutional	128	3 × 3 / 2	64 × 64
2×	Convolutional	64	1 × 1	
	Convolutional	128	3 × 3	
	Residual			64 × 64
	Convolutional	256	3 × 3 / 2	32 × 32
8×	Convolutional	128	1 × 1	
	Convolutional	256	3 × 3	
	Residual			32 × 32
	Convolutional	512	3 × 3 / 2	16 × 16
8×	Convolutional	256	1 × 1	
	Convolutional	512	3 × 3	
	Residual			16 × 16
	Convolutional	1024	3 × 3 / 2	8 × 8
4×	Convolutional	512	1 × 1	
	Convolutional	1024	3 × 3	
	Residual			8 × 8
	Avgpool		Global	
	Connected		1000	
	Softmax			

Figure 3-14: Darknet53 [63]

8. CSPDarknet53

The Backbone of this model does not work by darknet53, it works by CSPdarknet53. The CSPDarknet53 [65] is a utilization for darknet 53 [63] where cross stage partial connection Dense network implements it in the entire network. This makes The CSPDarknet53 model have higher accuracy than the Resnet model. This model replaces the Relu activation function with the Mish activation function for better performance. Figure 3-15 shows the structure of the CSPDarkNet53 network.

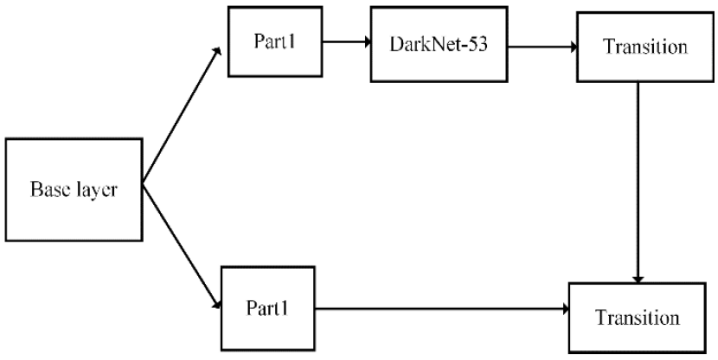


Figure 3-15: The Structure of CSPdarknet53 [80]

C) Neck

The main purpose of the neck on this model is to get the feature map from the backbone and feed it to the head by an input containing rich information of the feature. The main task of the neck is to do the information aggregation. The information aggregated by using:

- Spatial Pyramid pooling layer
- Path Aggregation Network

A. Spatial Pyramid pooling layer

The main purpose of this method [67] is to solve fixed image problems. Since this model contains backbone, where results from feature maps don't require fixed images. The main problem is with the head. The head is the classifier in the model containing the softmax activation function. This classifier requires fixed image size and length as input, so a spatial pyramid pooling layer is included for this problem. To include this method in this model a modification is required. The spatial pyramid pooling used in the model contains dense blocks and three multi-scale max pooling. The output from multi-scale max pooling is their feature map with size (size feature map * size feature map * 512). The output from multi-scale max pooling contact to one feature map with size (sizefmapsizefmap * 2048). Figure 3-16 shows the content of spatial pyramid pooling layer in model.

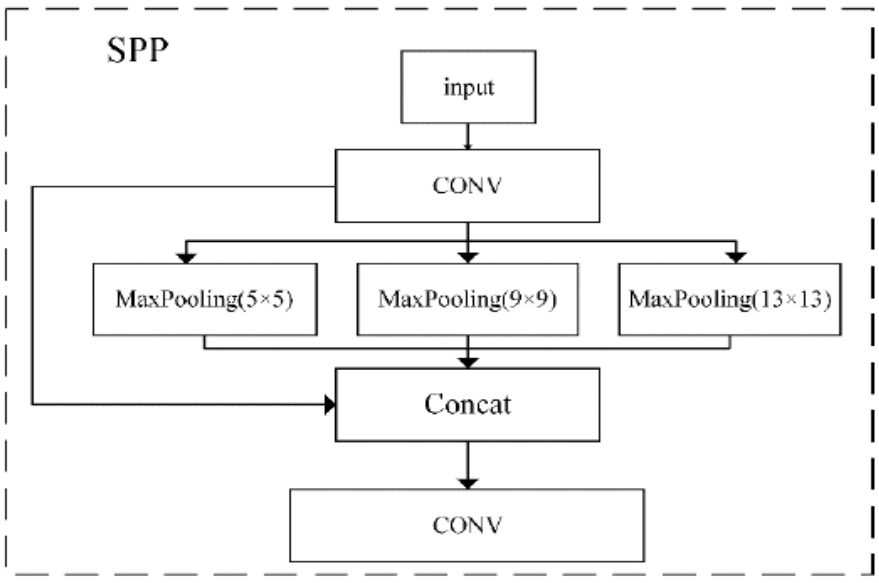


Figure 3-16: SPP Structure [80]

B. Path aggregation Network (PAN)

Path Aggregation Network [66] is used to integrate multi-level features. The low-level information contains the outline of the object. The high-level features contain the detailed information of the object. In the field of object detection, the part of the detector which collects feature maps is usually the neck. The input of the path aggregation network comes from their part, two of which are from the feature layer of the backbone network and the other input from the Spatial Pyramid pooling layer. The output of the path aggregation network is used as input for the head block in the model. Figure 3-17 shows the architecture of path aggregation network contain four parts:

A. Feature Pyramid Network

B. Bottom-up Augmentation

C. Adaptive Feature Pooling

D. Fully-connected Fusion

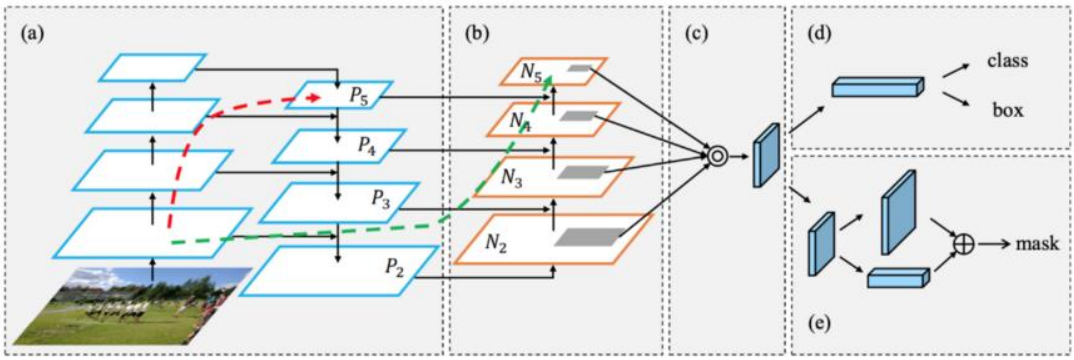


Figure 3-17: Path Aggregation Network Structure [66]

a. Feature Pyramid Network

Feature pyramid network [81] is used to get a multi scale image as input and output is a pyramid of features. The feature pyramid used to make a fast feature pyramid contain information about the objects. Figure 3-18 shows the construction of the pyramid contains a bottom-up pathway (backbone cspdarknet53) and a top-down pathway.

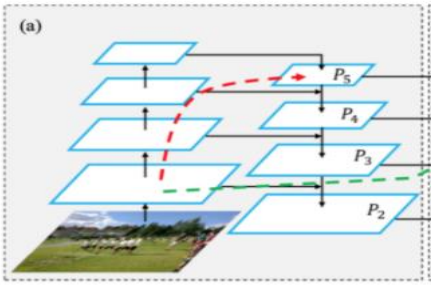


Figure 3-18: FPN Structure [81]

Top-down pathway

It hallucinates higher resolution features by up sampling spatially coarser, but semantically stronger, Feature maps from higher pyramid. These features are then enhanced with features from the bottom-up pathway by using lateral connections. Where the lateral connection merges feature maps of the same spatial size from the bottom-up pathway. In the bottom of the bottom-up pathway lower-level of semantics segmentation. But the lower level helps the detector to localize the object accurately. Figure 3-19 shows the FPN steps for making the feature pyramid.

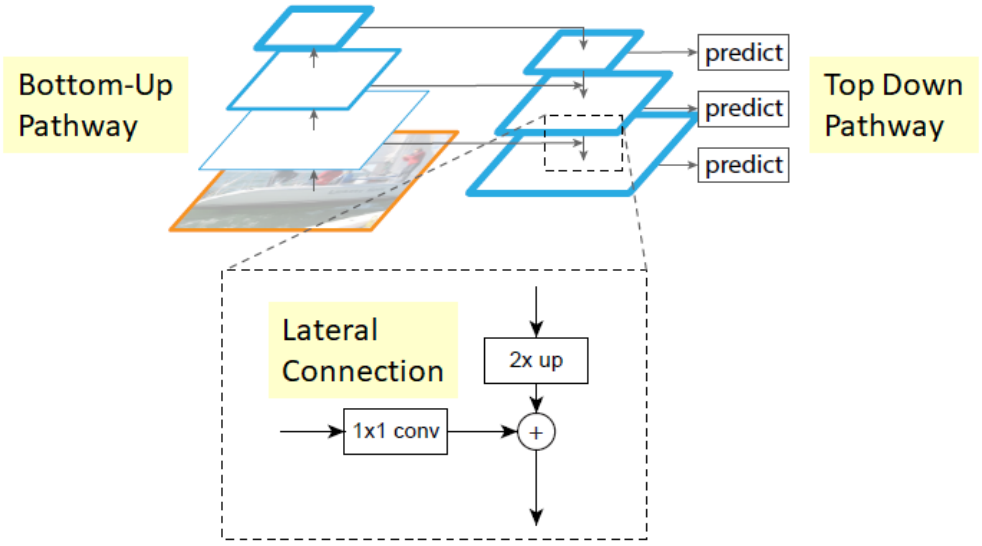


Figure 3-19: FPN Detailed Structure [80]

B. Bottom-up Augmentation

The purpose of bottom-up path augmentation [81] to shorten information path and enhance feature pyramid with accurate localization signals existing in low-level. Figure 3-20 shows the path of bottom-up augmentation, where Nidenote the newly generated feature maps corresponding to p_i means N_iSimply P_i without any processing. Each building block takes a higher resolution map N_i and p_{i+1} goes through lateral connection and generates the new feature map N_{i+1}. N_i goes through convolution layer size of 3*3 with strade by two to reduce the spatial size. After this feature map $P_{i+1}$and the down sample map are added through lateral connection and the results goes through convolution layer size of 3*3 to generate N_{i+1}. Then pass N_{i+1} for adaptive feature pooling.

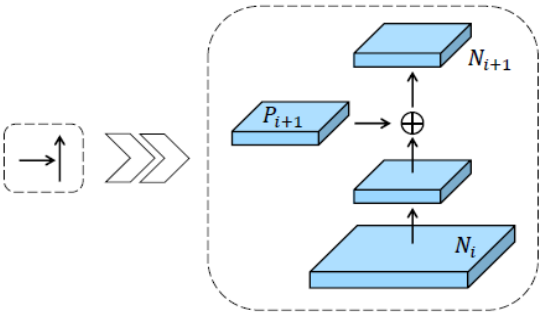


Figure 3-20: Path Aggregation Network [81]

C. Adaptive Feature Pooling

Adaptive feature pooling [66] pools feature from all levels for each proposal in object detection and fuses them for the following prediction. For each proposal, we map them to different feature levels. Adaptive feature pooling extract feature map from each feature map. It performs a region of interest alignment to extract features for the object from the feature map. The aligned feature maps are allowed to go through one parameter layer independently and followed by an operation called fusion operation to compress the features and then flatten the features for passing a flattened matrix to the fully convolutional network.

D) Head

The Head is used as an anchor box method [82]. Each anchor box contains a (pc, bx, by, bh, bw, cn) bound box (b, by, bh, bw), predicted class (Pc) and the class probability (c1,c2,..cn). Prediction of the class is done by using a sliding window method that is done by using logistic regression in a fully convolutional network and the output from the network is a predicate of the class. The bound box computes by using bound box regression on a fully convolutional network. For each created anchor box, the head block calculates which object's bound box has the highest overlap divided by non-overlap.

This is called Intersection Over Union or IOU. Figure 3-21 shows the intersection over union calculation where area of overlap is the area of intersection between the predicted bounding box and the ground-truth bounding box. Area of union is the area that predates the bounding box and ground truth union.

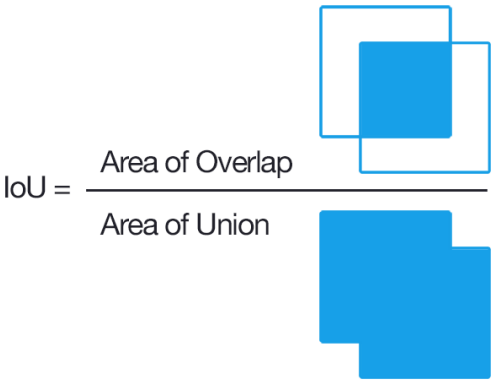


Figure 3-21: Intersection Over Union

Then after calculating the intersection over union if the highest IOU is greater than 50% then the anchor box should detect the object that gave the highest IOU, Otherwise the anchor box should predict that it is not an object. This step is called IOU thresholding where the process to remove all predicted bounded boxes is less than and keep the box with high value after calculating IOU thresholding in the detector use method to detect the best bound box in bounding boxes. The output of head is one-dimensional vector the contain all the anchor boxes detected in the model.

3.3 Robot Operating System (ROS)

Robot Operating System (ROS) is a free and open-source robotics software framework which is used for various applications in commercial and research fields. Before developing ROS, there was no common platform, community, or standard algorithms for robotics development. Furthermore, developers had to develop new software for every robot, which is time and energy-consuming, and couldn't benefit from their previous work. As a result, introducing ROS to the robotics world made a huge difference. [83]

3.3.1 ROS Definition

Robot Operating System (ROS) is a robot application development platform whose aim is to produce software that can run on various robots with little changes to the code through features such as code reusing, distributed computing and message passing. [84][85]

3.3.2 Robot Operating System (ROS) Architecture

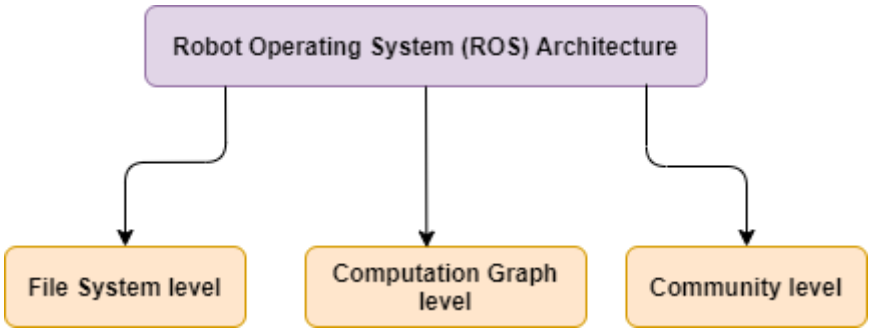


Figure 3-22: ROS Architecture

The **File System Level** explains the internal structure of ROS, including the folder structure and the number of files needed to work.
The **Computation Graph Level** explains how communication occurs between processes and systems and how to handle these processes.
The **Community Level** enables sharing knowledge, code, or algorithms between developers to expand the ROS usage. [85]

3.3.3 ROS File System Level

The following image illustrates the mechanism by which ROS files are managed and organized on the hard disk:

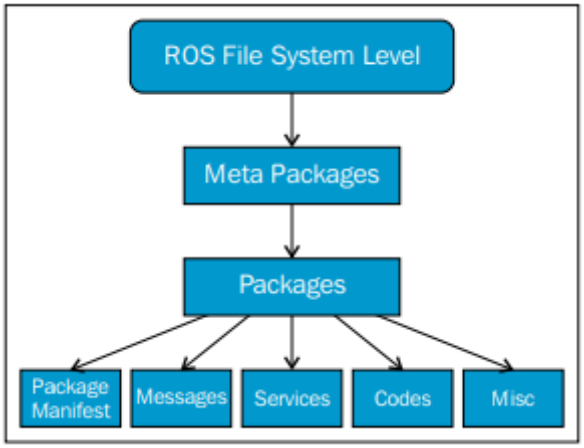


Figure 3-23: ROS File System Level [84]

3.3.3.1 Packages

ROS package is the most basic and minimum unit which includes the structures and content needed to develop a program. It contains nodes (runtime process), libraries and configuration files as a single unit. [84]

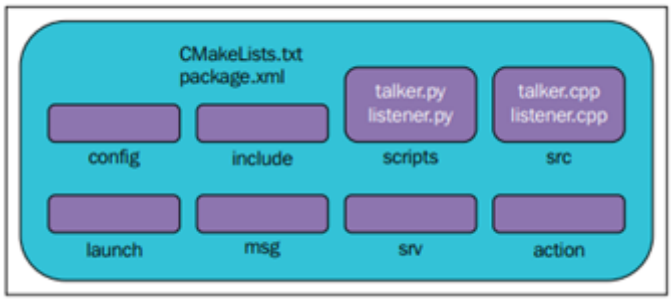


Figure 3-24: ROS Package Structure [84]

3.3.3.2 Messages

It is a type of data that flows between the ROS processes (nodes). In addition to the standard ROS message types, a new custom message type can be defined and stored inside the **msg *folder*** in a package. [85] ROS messages, a simplified message description language, can be published by ROS nodes to describe the types of data to generate the appropriate source in different languages. As mentioned before, the descriptions of the data type of ROS messages are stored in the ***msg folder*** in a ROS package.

To get numbered messages and to be aware of the sender's identity of a message, we use ***headers***. Header is a unique ROS message that carries information like time, frame_id, and sequence number. [84]

3.3.3.3 Services

It enables the interaction between nodes in the form of request/response communication where one node sends and waits till it gets a response from another node. The data types of the request and response are defined in the ***srv folder*** inside a package. [85]

3.3.3.4 The Workspace

A workspace is a folder in which you find and compile the packages and edit the source files. Any typical workspace includes three spaces, each of which has a different role:

1. **Source (*src*) Space:** The packages, projects and clone packages are included in the ***src folder***.

2. **Build Space:** The packages' cache information, configuration and other intermediate files are kept by **Cmake** and **catkin** in the ***build folder***.

3. Development (***devel***) Space: The ***devel folder*** is where the compiled programs or executables are kept skipping the installation step after testing the programs. [85]

3.3.4 ROS Computation Graph Level

The computation graph level is executed through a computation network of processes that is called ROS nodes. These nodes are all connected in this network and thus, can interact with other nodes, share information through transmitting it through the network. [85] This level is based on specific concepts which are ROS Nodes, Messages, Topics, Services, Bags, Master and Parameters. Figure 3-25 shows the ROS communication middleware packages **(ROS Graph Layer)** that are included in the **ros_comm** stack. [84]

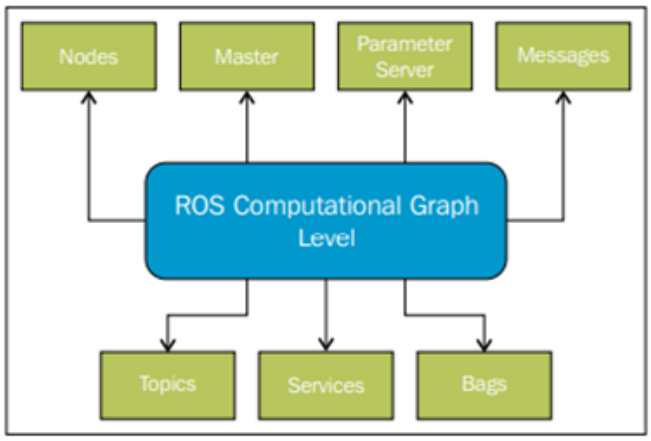


Figure 3-25: ROS Graph Layer Structure [84]

3.3.4.1 ROS Nodes

Nodes are the processes that perform computation. To connect to the ROS network, you need to write a node, using a client library such as **roscpp** or **rospy,** that represents the process you create to perform a specific function. Writing multiple nodes to provide the functionality of a system is better than writing one large node to perform it. [85] As a result, ROS nodes enable you to have a simpler system with easy to debug nodes [85]. For instance, a robot may have one node to process camera images, one node to compute the odometry and another to handle serial data from the robot.

The nodes communicate with each other through sending and receiving ROS Topics, Services and Parameters. You should identify each node with a unique name to be able to communicate with other nodes without ambiguity. [85]

In conclusion, using ROS nodes enables you to create a fault-tolerant system with reduced complexity and increased debug-ability. For example, if a node crashes, the robot system will still be able to work, but without the function that this node was performing. [84]

3.3.4.2 ROS Messages

To communicate, a node must publish a message through a topic to be sent to another node or exchange information through a service where a service message is defined in **srv** file as previously mentioned. The publisher and subscriber must have the same message data type. [84]

3.3.4.3 ROS Topics

Topics are the buses that are used by the ROS nodes to transmit data between them. Topics don't need a direct connection between nodes to be transmitted. A topic can have different subscribers and publishers. As previously mentioned, publishers and subscribers must have the same message type, which means that a node cannot subscribe to a topic unless it has the same message type [85]. Topics communications are unidirectional. So, if you want a request/response communication, you should use a ROS service instead.

3.3.4.4 ROS Services

To define a ROS service for nodes, a user must use a pair of messages, one of which is the request, and the other is the response. Then, define the datatype of them in a **srv file** which is kept in the **srv folder** in a **package.** While using services, one node acts as the server and another node acts as the client that requests a service from the server. After the server finishes the service routine, it sends the result to the client who requested it [84].

3.3.4.5 ROS Master

To enter a ROS system, a node must look for the ROS Master and register its name in it. Moreover, if any changes occur to a node, the ROS Master will be updated with this change through a call-back. This mechanism enables the ROS Master to know the information of all the nodes running on the system which facilitates their connection.

Now let's understand how publishers and subscribers exchange data. The publisher node publishes a topic and sends its details, which includes its name, the message and the message's datatype, to the ROS Master. The ROS Master now will look for another node that subscribed to the same topic. If the subscriber(s) is found, the ROS Master will send the publisher's details to it. The subscriber then will send to the publisher a request to communicate through TCPROS protocol, the publisher will respond and finally the session of transmitting the data will begin. On the other hand, if the subscriber(s) is not found, no connection will be done. Figure 3-26 describes the previous process.

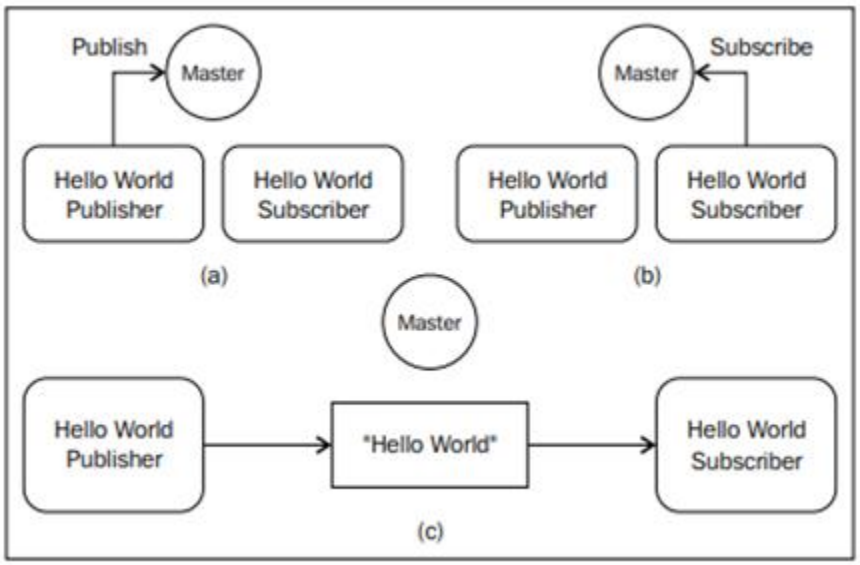


Figure 3-26: Communication between ROS Master, Publisher, and Subscriber [84]

3.3.5 ROS Community Level

The ROS community exchanges software and knowledge through ROS resources such as:

1. **Distributions:** They are meta packages that facilitate the installation and collection of ROS systems.

2. **Repositories:** It is a network of code repositories where different organizations can release their robot SW components.

3. **ROS wiki:** It is the main forum for ROS documentation where developers can document, provide updates, or give tutorials.

4. **Bug ticket system:** This resource is used when a bug is found, or a new feature is needed.

5. **Mailing lists:** It is the primary communication between ROS users where they receive any new updates to ROS or ask questions about it.

6. **ROS Answers:** It is where you ask questions about what you do not understand in ROS or anything related to it. Users can see your question and will give solutions if possible.

7. **Blog:** It provides any news or updates related to the ROS. [84]

3.3.6 ROS Summary

In conclusion, as previously mentioned ROS has a great impact on the robotics development and community. ROS benefits us in many ways. Some of these benefits are the following:

- High-end capabilities (SLAM, AMCL, and Move It packages)
- Tons of tools (rqt_gui, RViz and Gazebo tools)
- Support high-end sensors and actuators
- Inter-platform operability
- Modularity
- Concurrent resource handling [84]

3.4 Virtual Reality and Augmented Reality

3.4.1 Virtual and Augmented Reality Software Development Tools

1) Unity 3D

Unity3D is a tool for designing a 3D application in which interacting in the virtual environment is customizable and the deployment and distribution of the final application is easy across different VR and AR systems. It became the most common design tool for applications due to its good support of VR and AR devices as well as its large user base. Additional functionality can be added to the application by downloading packages, called “prefabs,” from the Unity Store for free or a small fee. Functionality of prefabs can range from simple three-dimensional models to packages used to plot data, e.g., Immersive Analytics Toolkit or packages which simplify interactions in VR and AR like grabbing an object, e.g., Virtual Reality Toolkit [86][87].

2) Unreal Engine 4 (UE4)

One of the main competitors of Unity 3D, Unreal Engine is also a gaming engine with VR integrations, an asset store, and great documentation. It offers a powerful set of VR

development tools. With UE4, you can build VR apps that will work on a variety of VR platforms, e.g., Oculus, Sony, Samsung Gear VR, Android, iOS, Google VR, etc. [86][87].

3) Blender

- It is free and open-source software written in Python and is available for Windows, Mac, and Linux [86][87].
- Blender offers the following features and capabilities:

You can create your 3D pipeline with modeling, rigging, animation, simulation, rendering, composing, and motion tracking. Moreover, Blender supports video editing and the creation of VR video games. If you have an experienced VR developer in your team, then he/she can use its API for Python scripting to customize the application. This allows you to create specialized tools.

4) 3dx Max & Maya

These are Autodesk products for modeling, animation, lighting, and VFX. They do not have VR support by default but through pricey plugins instead. AutoCAD and 3DS Max are long-time standards in the architectural design industry and have some of the most precise tools in their UI. Like almost all GUIs for building 3D environments and drawings, these tend to be quite massive UIs with a lot of tools hidden behind menus, sub-menus, and toolbars [86][87].

5) SketchUp Studio

Google's SketchUp is a basic modeling application with a very low learning curve that can get anyone up and running in a short amount of time. It is useful for use cases like architecture, commercial interior design, landscape architecture, residential construction, 3D printing, and urban planning [86][87].

3.4.2 Software Development Kits (SDKs)

A software development kit (SDK) includes a set of software tools and programs used for specific platforms. It can include libraries, samples, process, guides, tutorials, blueprints and more.

Virtual Reality SDK

1. **Oculus Quest**

Oculus is a line of virtual reality headsets developed and manufactured by Oculus VR, a division of Facebook Inc. This is a feature complete SDK which handles for the developer the various aspects of making virtual reality content, such as the optical distortion and advanced rendering techniques [88][89].

2. **Windows Mixed Reality**

It is Windows VR headset. There are Holographic and immersive headsets. Holographic devices such as the Microsoft HoloLens allow you to see the physical environment around you while wearing the headset, blending the real world with virtual content. You can see-through display that allows you to see the physical environment while wearing the headset [88][89].

3. **Google VR**

It is a virtual reality (VR) platform developed by Google. Named for its fold-out cardboard viewer into which a smartphone is inserted, the platform is intended as a low-cost system to encourage interest and development in VR applications. Users can either build their own viewer from simple, low-cost components using specifications published by Google, or purchase a pre-manufactured one. To use the platform, users run Cardboard-compatible mobile apps on their phone, place it into the back of the viewer, and view content through the lenses [88][89].

4. **OpenVR**

OpenVR is a software development kit (SDK) and application programming interface developed by Valve for supporting the SteamVR (HTC Vive) and other virtual reality headset (VR) devices. The SteamVR platform uses it as the default application programming interface (API) and runtime. It serves as the interface between the VR hardware and software and is implemented by SteamVR [88][89].

3.4.3 Augmented Reality SDK

1. **Vuforia Augmented Reality SDK**

Vuforia is a leading portal for augmented reality application development that has a broad set of features. Vuforia augmented reality SDK:

- Recognizes multiple objects including boxes, cylinders, and toys as well as images.
- Supports text recognition including about 100,000 words or a custom vocabulary.
- Allows creating customized VuMarks, which look better than a typical QR-code.
- Allows creating a 3D geometric map of any environment using its Smart terrain feature
- Turns static images into full motion video that can be played directly on a target surface.
- Provides a Unity Plugin.
- Supports both Cloud and local storage.

Supported platforms: iOS, Android, Universal Windows Platform, Unity [90][91].

2. **ARToolKit**

It is an open-source tool to create augmented reality applications. It provides the following functionalities:

- SLAM tracking (simultaneous localization and mapping) and sensor fusion for creation of location-based AR apps.
- Unity3D and OpenSceneGraph Support.
- Supports both single and dual camera.
- Possibility to create real-time AR applications.
- Integration with smart glasses.
- Multiple Languages Supported

Supported platforms: Android, iOS, Linux, Windows, Mac OS, and Smart Glasses [90][91].

3. **ARCore**

It is Google's proprietary augmented reality SDK. Similar to ARKit, it enables brands and developers to get AR apps up and running on compatible Google smartphones and tablets. This toolkit works with Java/OpenGL, Unity, and Unreal. It provides features such as:

- Device Localization
- Horizontal Plane Detection
- Vertical Plane Detection
- Point Clouds
- Pass-through Camera View
- Light Estimation
- Reference Points
- Oriented Feature Points
- Hit Testing
- Session Management
- ARCore APK On-Demand Installation
- Image tracking
- Face tracking

Supporting devices: Currently: Google Pixel, Pixel XL, Pixel 2, Pixel 2 XL, Samsung Galaxy S7-S8+, Samsung A5-A8, Samsung Note8, Asus Zenfone AR, Huawei P20, OnePlus 5 ARCore is designed to work on devices running Android 7.0 and higher [90-92].

Table 3-2: Comparison between AR SDK [90]

AR SDK	Best for	Supported platforms
Vuforia	Marker-based apps	· iOS · Android · Universal Windows Platform · Unity
ARToolKit	Location-based apps	· Android · iOS · Linux · Windows · Mac OS · Smart Glasses
ARCore	Marker-based apps	· Google Pixel · Pixel XL · Pixel 2 · Pixel 2 XL · Samsung galaxy S8 · Samsung galaxy S9

3.4.4 Application of VR and AR in Robotics Projects

1) Troll Tunnel Inspection ROV Piloted in Virtual Reality Mode

Through the software developed by Morrison McLean, the current position of the ROV can be displayed on the 3D model. The pilot can therefore use the display as a visual guide as he flies the ROV through the tunnel, according to Morrison McLean managing director George Morrison. A tour of inspection can also be planned, with warnings automatically given to the pilot when he is approaching previously spotted irregularities. Again, the model can be used to simulate ROV manoeuvres to check that they are feasible.

Morrison McLean has also developed a multimedia-based reporting system to display inspection results. The system shows features discovered in the previous survey and marks their location on the 3D model. Associated information in the form of video images, audio comments, cathodic protection readings, text files and sonar records can also be accessed. Because the reporting system only shows those parts that have changed or are otherwise of interest, data storage requirements are much reduced, and the survey results could be transferred onto a single CD-ROM, Morrison says. [93]

2) Controlling Robot Swarms with Augmented Reality

At New York University, Jared Alan Frank has turned to augmented reality (AR) to develop a robot control interface that runs on a conventional smartphone or tablet. The system uses the device's camera to capture details from a scene and overlay virtual objects, as other AR applications do. But in this case, you can simply tap and swipe on the screen to make the robots move or pick-up objects.

Using XCode, Apple's software development platform, Frank built an app that can detect robots and objects in the environment and create a virtual grid along with a coordinate system to keep track of those objects on the screen. The user can then manipulate the objects on the device and watch as the robots carry out the desired actions in the real world. Commands from the app are sent via Wi-Fi to the robots, which in the current version use Raspberry Pi as the main controller.

The smartphone or tablet captures the scene using its camera, and the app detects the tags, using that information to keep track of marked objects. These tags, also called fiducial markers, are commonly used in AR apps to integrate physical landmarks and objects into a virtual world.

3) An Underwater Augmented Reality System for Commercial Diving Operations

We accomplished an effective camera calibration that was a critical element for rendering the visualizations proposed initially. The information displayed, navigation aid to reach the worksite, artificial horizon for diver awareness of orientation and positioning, and the memory clues for executing an assembling task and an inspection survey were accomplished [94].

3.5 Summary

In this chapter, all the technology used to implement the AUV were explained. In the next chapter, the entire system implementation will be discussed.

Chapter 4: Implementation

4.1 Introduction

In this chapter, the implementation of the mechanical, electrical and software systems will be explained, each in a section.

4.2 Mechanical Design & Manufacturing Processes

After thoroughly reviewing the tasks required by our vehicle, different design ideas are brainstormed for frame design, tooling, payloads, and their arrangement. Next, a mock-up or a mechanical drawing is presented to discuss with the team members and accommodate their feedback and suggested modifications. The process is repeated until the design is deemed optimal, where every design and implementation process are verified and validated to ensure integration.

One factor that heavily influenced the mechanical design process is our aim to provide a cheap and efficient AUV. Concepts were judged based on size, weight, cost, ease of manufacturing and serviceability, safety, and reliability. Hence, heavy consideration was paid to the available materials in the market along with the different manufacturing techniques available in our locality.

To easily view the design and suggest any beneficial modifications, a computer aided design (CAD) was created, that was shared among all members and on all project platforms. Finally, the designs were converted into Drawing Exchange Format (DXF) and Standard Tessellation Language (STL) files to be sent to the 2D routers, laser cutters, and 3D printers available on campus which our university allowed us to use - an act of generosity from our university which we are greatly thankful for.

4.3 Mechanical Aspects

4.3.1 Frame

The frame consists of a base, two sides, two cradles, two cradle supports, and two horizontal stabilizers. It was decided to maintain the overall frame's design due to its modularity, versatility, and high underwater performance. The entire frame is assembled akin to a Lego structure; every element in the frame has designated ports in which specific elements are to be inserted, enabling us to robustly fit parts in the frame while

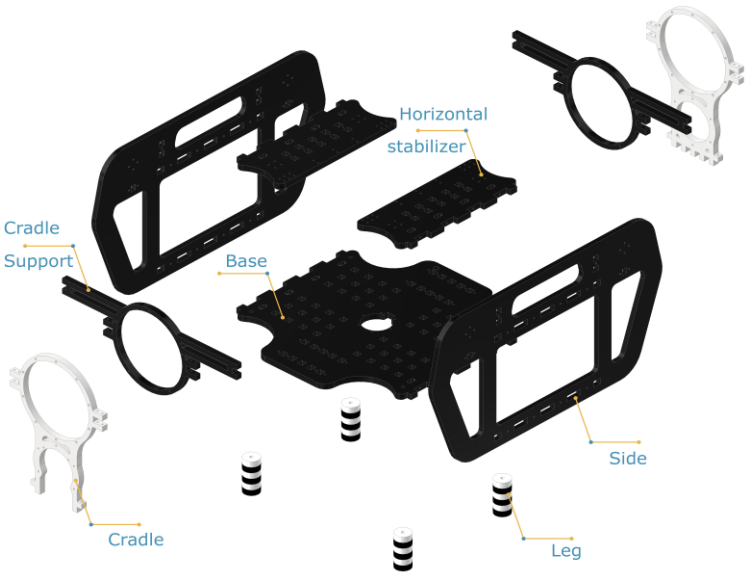


Figure 4-1: Exploded View of the Frame

needing much fewer nuts and bolts. The frame is manufactured using 2D routing machines from 10 mm thick high-density polyethylene “HDPE” costing 1.5$ per kg. HDPE was chosen due to its cheapness, high density-to-strength ratio, and high strength under nominal stresses.

4.3.2 Thrusters

The AUV is equipped with eight T100 Blue Robotics Thrusters. These thrusters provide a remarkable speed as Blue Robotics offers powerful, compact, and efficient thrusters at an affordable price of $119. Four heaving thrusters - one at each corner - and four surging thrusters that are mounted at 45° around the central electronics housing at each corner of the AUV.

Figure 4-2: T100 Thruster

The centre of mass of the AUV is aligned with the centre of thrust to maximize stability and obtain a stable vector drive, allowing all thrusters to contribute to the total propulsion in all cardinal directions and minimizing flow interference with the electronics housings in the centre of the vehicle. Four -rather than two- thrusters were used for vertical control to enable rolling and pitching the vehicle whenever needed while executing missions.

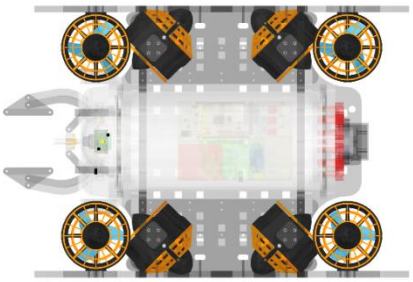


Figure 4-3: Thrusters Layout

4.3.3 Enclosures & Sealing

The AUV comprises three sealed enclosures: Control and Power Housing, a separate USB camera, and a ZED camera. The Control and Power Housing carries all control and power subsystems in a cylindrical-shaped enclosure. The cylinder is made of Poly (methyl methacrylate) (PMMA) acrylic which was chosen for its transparent nature; allowing us to instantly view the status of debugging LEDs in our systems. It is capped with two aluminium flanges and an acrylic dome at the front-to-house additional cameras.

Figure 4-4: ZED Camera

Since high-quality waterproof cameras are costly, we chose to seal a USB camera in house, a black Polyamide-6 (PA6) cylinder was machined to house the USB camera with a transparent 8 mm acrylic sheet that is used to seal the camera and acts as a lens.

Figure 4-5: USB Camera

Similarly, The ZED Camera enclosure is sealed in house, a slot shaped - just like the camera - enclosure was manufactured from white Polyamide-6 (PA6) material using 3 Axes CNC machine, courtesy of the college of Engineering at AASTMT, Industrial Department. A transparent 8 mm acrylic sheet is used to seal the camera and acts as a lens.

Figure 4-6: O-rings

Water Sealing mechanisms are done using three main sealing methodologies: compressed tori joints “O-rings”, Marine Epoxy, and mechanical seals. Housings are sealed with compressed O-rings, such that all cables run through penetrators which in turn are sealed with marine epoxy. Penetrators were used instead of cable glands as they provide a better water seal.

4.4 Electrical & Control System

4.4.1 Electrical & Control System Overview

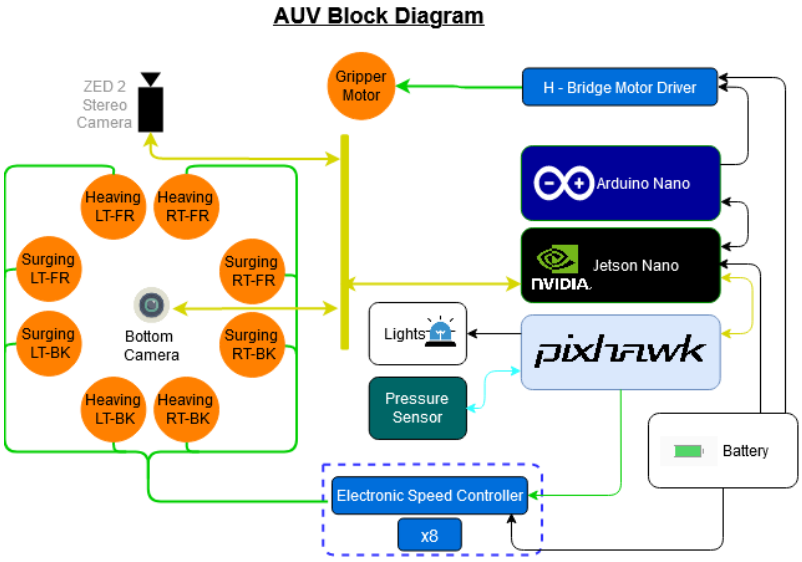


Figure 4-7: Block Diagram

Figure 4-7 represents the block diagram which illustrates an overview of the AUV system. The Nvidia Jetson Nano is the processor responsible for controlling the AUV while executing missions as well as processing the computer vision algorithms. The Jetson Nano controls an Arduino Nano microcontroller, a Pixhawk flight control computer and processes the images and data from both the ZED 2 stereo camera and the USB camera, it is responsible for each move of the vehicle. The Jetson Nano communicates with the different system components as follows:

4.4.1.1 Arduino Nano

The Jetson Nano communicates with the Arduino Nano over UART protocol, which in response communicates with an **H-Bridge motor driver** to move the **gripper** accordingly and communicate over I2C to get the IMU readings.

4.4.1.2 Pixhawk Flight Controller

The Jetson Nano communicates with the Pixhawk through MAVLink protocol. Firstly, the Pixhawk is connected with **8 T100 Thrusters** through an **Electronic Speed Controller**, which provides PWM control of the motor speed. Secondly, it is connected with a **pressure sensor** to measure the pressure of water and thus calculate the depth. Finally, the Pixhawk is connected with **LED lights** which are used for lighting when sufficient natural light is not present.

4.4.1.3 Cameras

Jetson Nano sends and receives feedback from 2 cameras, **ZED 2 Stereo Camera** and another **USB camera**. The ZED 2 Stereo Camera provides a forward viewing angle of 120 degrees, while the bottom camera provides inspection beneath the AUV.

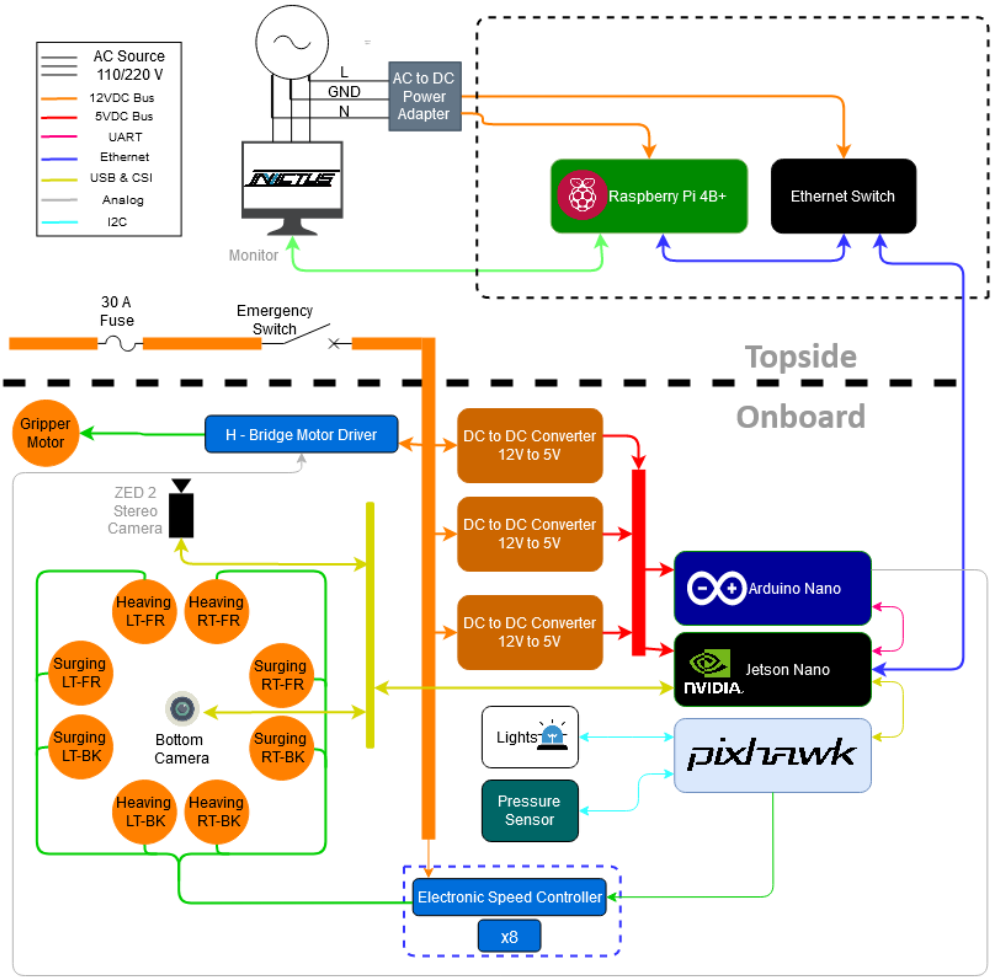


Figure 4-8: System Interconnection Diagram

4.4.1.4 Topside Monitoring Unit (TMU)

On the surface side, we have a 12v/30A power supply that supplies the vehicle with power, a single board computer (SBC) for the vision system by receiving camera feeds through an ethernet connection that is reliable and preserves the image quality & a monitor that displays the cameras feedback in the testing & debugging phase.

4.4.1.5 Onboard Electronics

We start by taking the 12v/30A on the power distribution board and regulating the power for the 3 different controllers (Raspberry pi 3b+, Arduino Nano and Jetson Nano) that use 5v to operate.

Pixhawk is also a controller that operate by plugging it in the Jetson Nano processor for taking commands from it as well as taking back any feedback from the motors since we are using MAVLink protocol, Pixhawk linked with the 8 thrusters we have through 8 Electronic speed control (ESC) also the pressure sensor and lights are connected to Pixhawk.

4.4.1.6 Tether

A VideoRay tether was selected as it has a good electrical layout as well as neutral buoyancy and flexibility that allows the AUV worry-free manoeuvring. A section of VideoRay contains two pairs of 16 AWG silicon wire to carry power, two 120 ohm twisted pairs for ethernet and data transmission which are inherently immune to noise and one 75-ohm pair which is ideal for signal integrity and is suitable for backup video transmission.

Figure 4-9: Tether Cross Section

The surface station, an enclosure splits the tether into a power cable with an inline 30A fuse as a safety measure and a signal-carrying-cable bundle that goes to the TMU. The same enclosure has a big red emergency stop switch that is always in everyone's reach and cuts off the power immediately should anyone feel the necessity to.

4.5 Electrical & Control Components

Table 4-1: Electrical and Control Hardware Components

Electronic Components	
NVIDIA Jetson Nano Developer Kit	
Pixhawk	
Arduino Nano	
ZED 2 Stereo Camera	
USB Camera	
Electronic Speed Controller (ESC)	
BlueRobotics T100 Thrusters	
LogiTech Extreme 3D Pro Joystick	

4.6 Software System

4.6.1 Computer Vision System

The computer vision system is divided into two systems: stereo vision system and mono vision system. The mono vision system is made for streaming and recognizing objects. The stereo vision system is made for obstacle avoidance and object with distance estimation.

4.6.1.1 Monovision Nodes

The mono vision system is made up of three ROS nodes: USB camera stream node, Image processing node and object detection node. The output is a bound box with coordinates x and y for the detected objects as shown in figure 4-10.

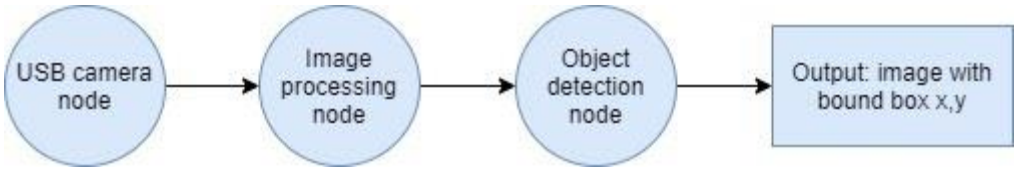


Figure 4-10: Monovision System

A. USB Camera Node

The USB camera node is built by ROS to open USB camera by using OpenCV to get frame by frame from camera and convert the image from OpenCV format to ROS format by using CV_bridge in ROS, then it publishes the output from CV_bridge to the ROS system to any subscribing node waiting for the camera stream as shown in figure 4-11.

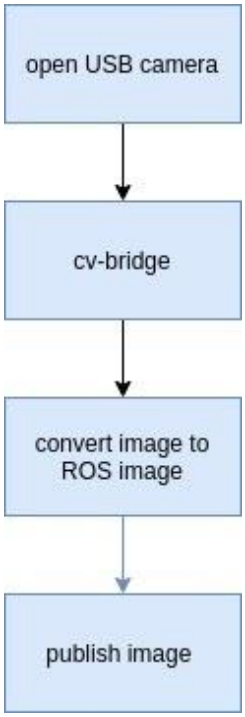


Figure 4-11: USB Camera Node

Published Topics

- bottom_cam/image_raw(sensor_msgs/Image)

B. Image Enhancement Node

The image enhancement node uses the RGHS Technique, implemented as shown in figure 4-12, to enhance the quality of received images from the USB camera node. Image enhancement nodes subscribe to the image from the USB camera node in ROS format. Then, it uses cv-bridge to convert the ROS format into OpenCV format to be able to apply the RGHS on this image. Afterwards, the enhanced image is published to the object detection node as shown in figure 4-13.

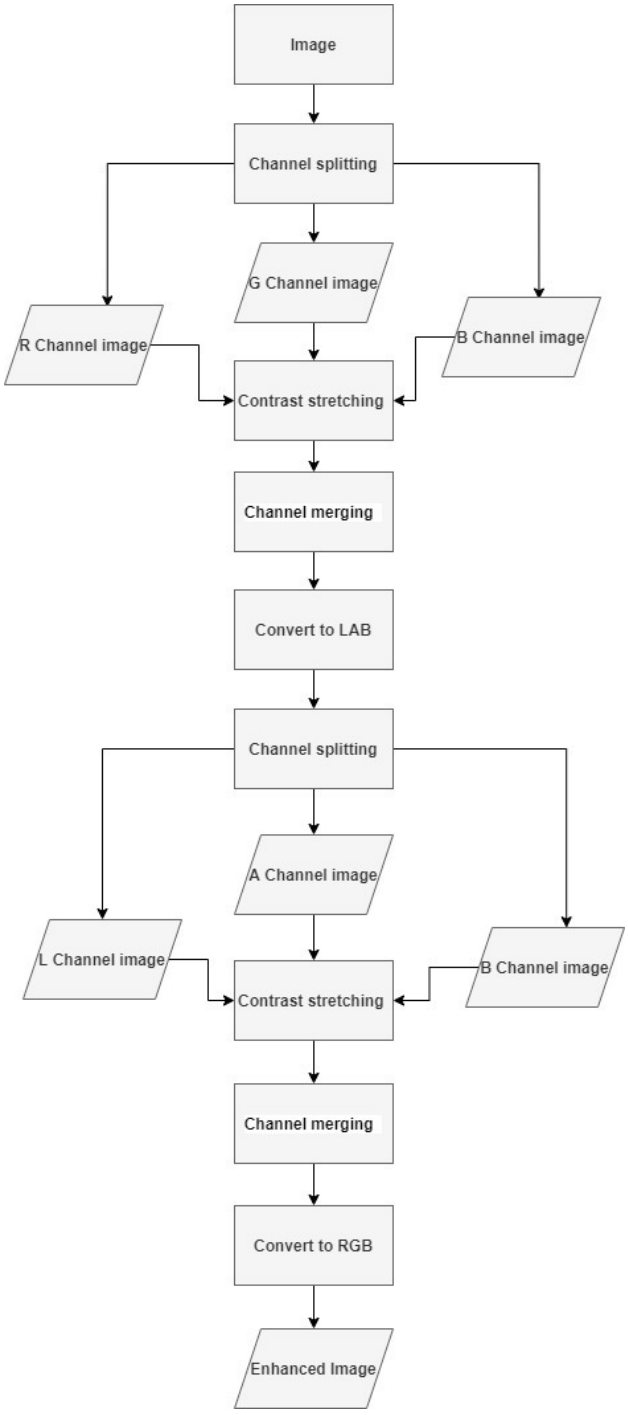


Figure 4-12: RGHS Image Enhancement Implementation

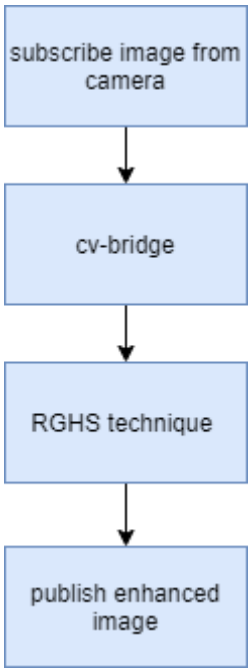


Figure 4-13: Image Enhancement Node Implementation

Published Topics

- bottom_cam/preprocessed/image(sensor_msgs/Image)

Subscribed Topics

- bottom_cam/image_raw(sensor_msgs/Image)

C. Object Detection Node

Before building an object detection node, some prerequisites are needed to be implemented before building the object detection node in ROS system as shown in figure 4-14.

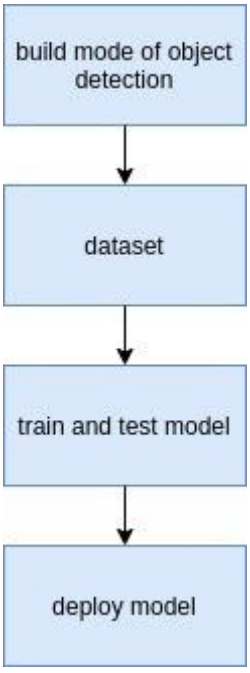


Figure 4-14: Steps of YOLO Implementation

1. Model of Object Detection

The model used in object detection is YOLO, as mentioned in chapter 3. This approach needs Cross Stage partial connections for feature extraction, Spatial pyramid pooling layer and Path aggregation network. Then after getting flatten, the feature map will be one vector. Then use the Dense prediction to classify an object as shown in figure 4-15 and then apply non-maximum suppression to get the best bound box to detect an object.

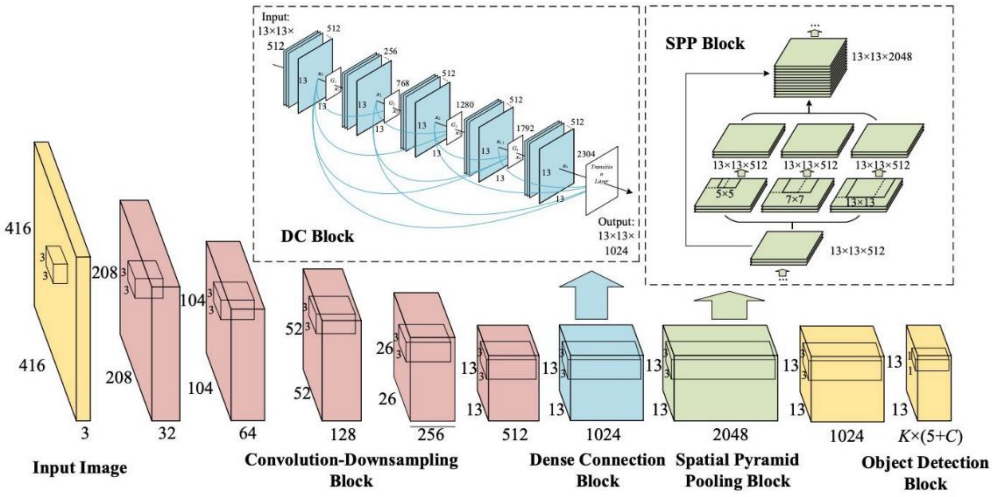


Figure 4-15: Architecture of YOLO v4

2. Dataset

The dataset collected are from different classes including starfish, tree, lines in 3d printed. The total images collected per class are 300 images. The data collected in the pool was by using a sealed USB camera. The images are labelled by drawing a bound box around the desired object by using the label image tool as shown in figure 4-16. The dataset split to 70% training dataset, validation dataset 20% and test dataset 10%.

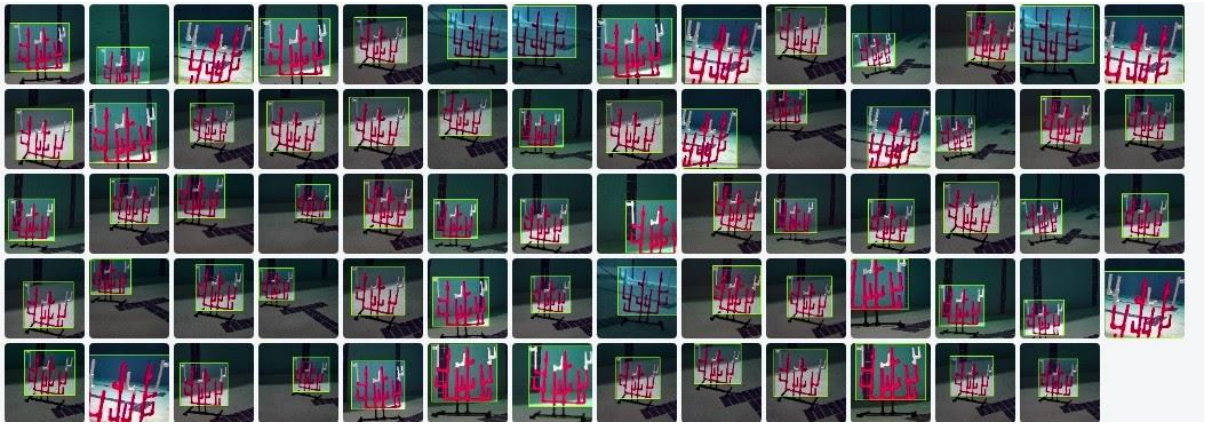


Figure 4-16: Collected Dataset

3. Training and Test Model

After training the model on 1500 iterations, it successfully reached an accuracy of 96.44 % and after testing results in test data set the model fit the output result as shown in figure 4-17.

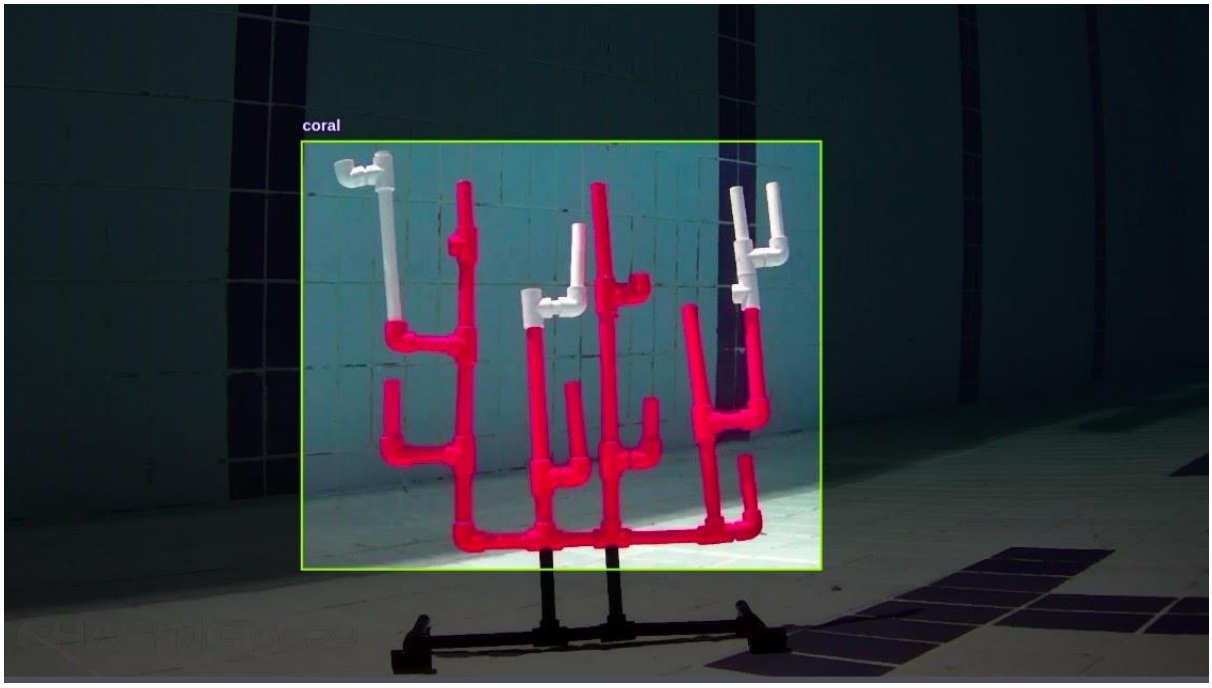


Figure 4-17: Test Results of YOLO

4. Deploy Model

To use the model for real time tasks, one needs to deploy the model by converting the model to TensorFlow light then save the weights of the model for Future Inference then deploy the model in hardware by using OpenCV library.

The object detection node subscribes to the enhanced image from the image processing node and converts an image to OpenCV format by using cv-bridge. then integrates the deployed model with the node to get the detected object as a bound box with x, y coordinates, width, height, class and class property. Then, the previous data are used to draw the bound boxes. The node then publishes the image message and the bound box message in the form of x, y coordinates, width, height, class and class property as shown in figure 4-18.

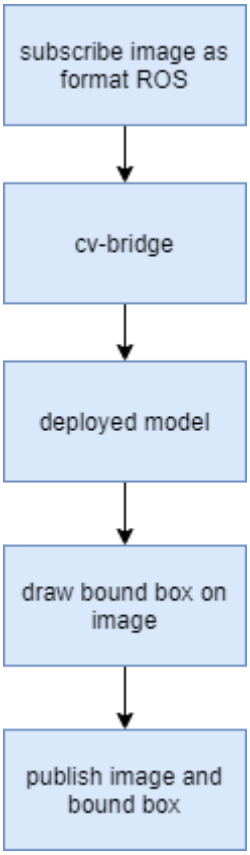


Figure 4-18: Object Detection Node Implementation

Published Topics:

- object_detector (std_msgs::Int8)
- Publishes the number of detected objects.
- bounding_boxes (yolo 2d::BoundingBoxes)
- Publishes an array of bounding boxes that gives information of the position and size of the bounding box in pixel coordinates x, y.
- detection_image (sensor_msgs::Image)
- Publishes an image of the detection image including the bounding boxes.

Subscribed Topics

- bottom_cam/preprocessed/image(sensor_msgs/Image)

4.6.1.2 Stereo Vision Nodes

The stereo vision system is made up of three ROS nodes: stereo camera node, Image processing node and object detection node. The output is a bound box with coordinates x, y and z for detected objects as shown in figure 4-19.

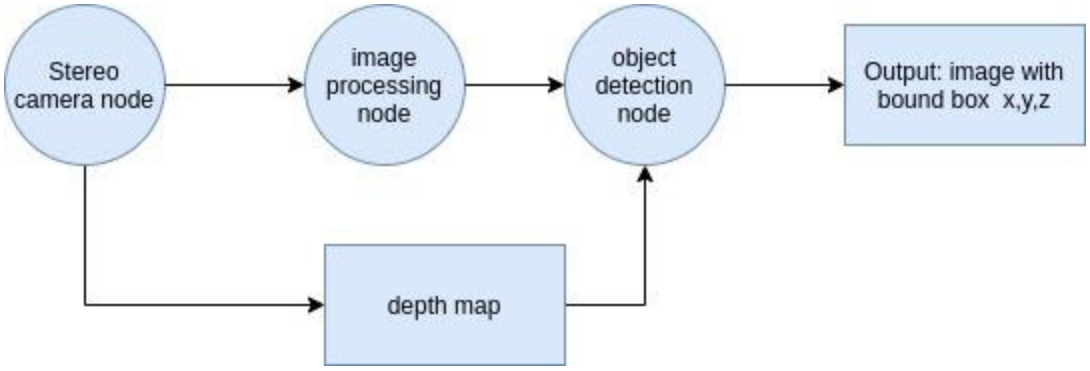


Figure 4-19: Stereo Vision System

A. Stereo Camera Node

The stereo camera node is based on the ZED ROS wrapper package which is built by the stereolab team developer [95]. This package contains a node to make an interface between the ZED SDK and ROS framework to publish some topics.

Published Topics

- **Left camera**
  - /zed2/zed_node/rgb/image_rect_color (type :sensor_msgs/Image)
  - /zed2/zed_node/rgb_raw/image_raw_color (type:sensor_msgs/Image)
  - /zed2/zed_node/left/image_rect_color (sensor_msgs/CameraInfo)
- **Right camera**
  - /zed2/zed_node/right/image_rect_color (sensor_msgs/Image)
  - /zed2/zed_node/right_raw/image_raw_color (sensor_msgs/Image)
- **Depth and Point Cloud**
  - depth/depth_registered (sensor_msgs/Image): Depth map image registered on left image (32-bit float in meters by default).

B. Image Processing Node

The image processing node is implemented similar to the one in the Monovision system, where it subscribes to the topic /zed2/zed_node/rgb/image_rect_color and then convert the image format from ROS to openCV format by using cv-bridge. Afterwards, it applies the same image enhancement technique RGHS as in Mono vision system, then convert the image to ROS format and publish it in ROS format, as shown in figure 4-20.

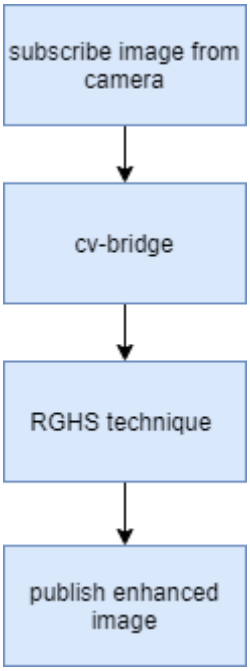


Figure 4-20: Image Enhancement Node in Stereo Vision System

Published Topics

- preprocess/right/image_rect_color(sensor_msgs/Image)

Subscribed Topics

- /zed2/zed_node/rgb/image_rect_color(sensor_msgs/Image)

C. Object Detection Node

The object detection node in the stereo vision system is similar to the one in the Monovision system. It applies the object detection technique (YOLO), but this node is a subscriber node waiting for both preprocess/right/image_rect_color topic from image enhancement node and /zed2/zed_node/depth/depth_registered topic from stereo camera node. After subscribing to the image and depth map, they are converted to opencv using cv_bridge, where the model detects if an object is found in the image as a bounded box with coordinate x,y, width, height, class, class property. The bounded box coordinates are used to get the center of the image by applying (xmin+xmax)/2 and (ymin+ymax)/2. After getting the center, the node extracts the depth of the object by using x,y center as z value in meters. If an object is detected, then a bounded box is drawn in the image by x,y coordinates and the bounded box message is publish in the form of xmin, xmax ,ymin,ymax and z in meters as shown in figure 4-21.

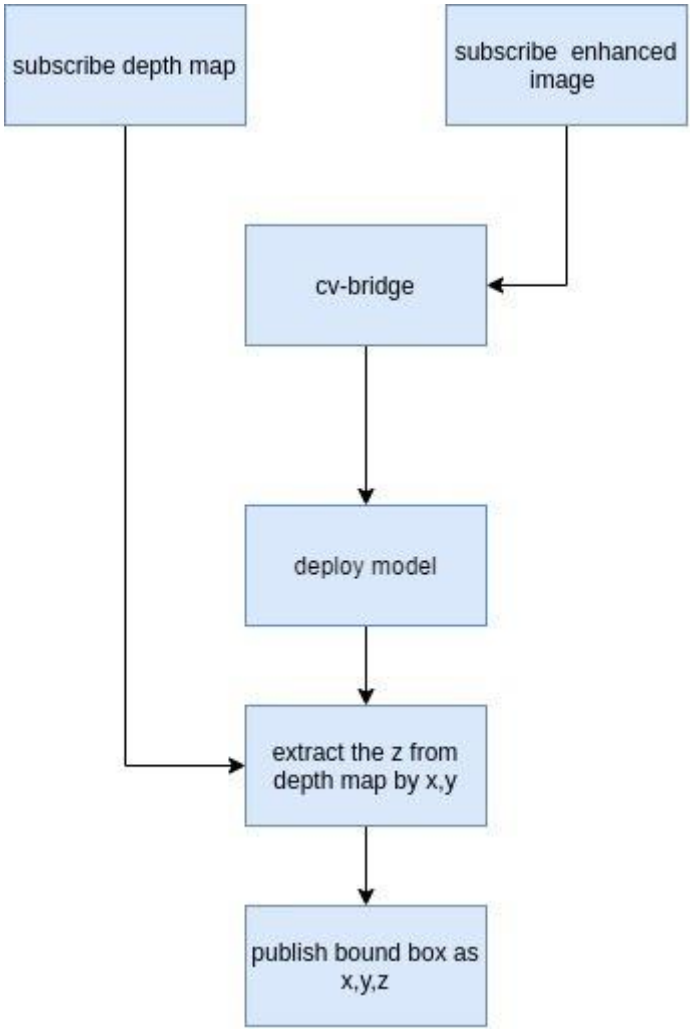


Figure 4-21: Object Detection Node in Stereo Vision System

Published Topics

- object_detector (std_msgs::Int8)
    - Publishes the number of detected objects.
- bounding_boxes (yolo3d::BoundingBoxes)
    - Publishes an array of bounding boxes that gives information of the position and size of the bounding box in pixel coordinates x,y,z in metter.
- detection_image (sensor_msgs::Image)
    - Publishes an image of the detection image including the bounding boxes.

Subscribed Topics

- preprocess/right/image_rect_color(sensor_msgs/Image)

4.6.2 Control System

4.6.2.1 Joystick Node

This node works as an application programming interface (API) in the ROS system where it links the Joystick to the system by publishing a message. The message built in the ROS system called “Joy" message contains a Header, an array that holds the axes of the joystick and an array that holds buttons of the joystick.

Published Topics

- /Joy (sensor_message/Joy)

4.6.2.2 Manual Control Node

This is a subscriber node which waits to get a topic /Joy message and then convert this message into a MAVROS format message called rc_overide/in. The node then publishes this message on a topic called mavros/rc_override/in to a MAVROS subscriber node for this topic as shown in figure 4-22.

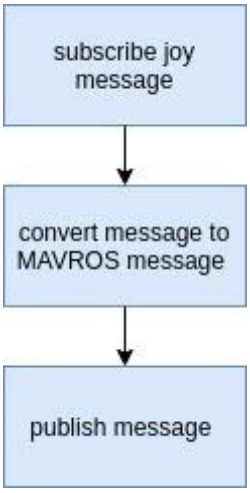


Figure 4-22: Manual Control Node Implementation

Published Topics

- rc/in (mavros_msgs/RCIn)

Subscribed Topics

- /Joy (sensor_message/Joy)

4.6.2.3 MAVROS Node

The MAVROS node is a communication driver that acts as both a subscriber and a publisher node. It is a subscriber node waiting for **rc/override/in** topic (mavros_msgs/RCIn) that is received from a publisher node such as manual_control node or stabilize_control node. It also acts as a publisher node that transmits **rc/override/out** topic (mavros_msgs/rcout) which is converted to the Pixhawk as a UDP MAVLink stream, as illustrated in figure 4-23.

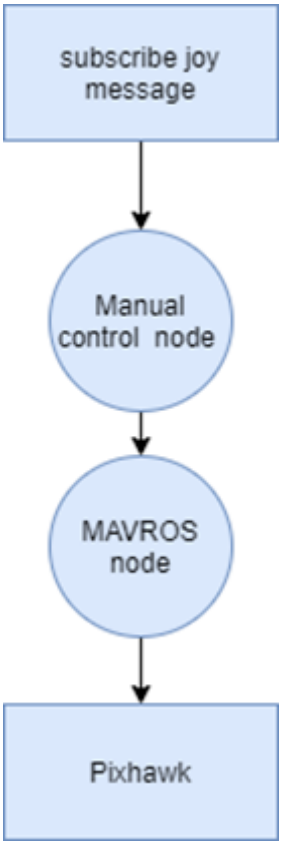


Figure 4-23: Control System Overview

Published Topics

- rc/in (mavros_msgs/RCIn)
    - Publish RC inputs (in raw milliseconds)
- rc/out (mavros_msgs/RCOut)
    - Publish FCU servo outputs

Subscribed Topics

- rc/override (mavros_msgs/OverrideRCIn)
    - Send RC override message to pixhawk

4.6.2.4. Arduino Node

The IMU sensor is connected to Arduino Nano through I2C protocol to send its readings (accelerometer, gyroscope and temperature) to the Arduino Nano, which will publish these readings into ROS through rosserial protocol through a node called serial_node. In this node, the readings are converted to String to be sent in a concatenated String (std_msg) to another node called imu_subscriber node, in which the String readings will be converted to integers once again as shown in figure 4-24.

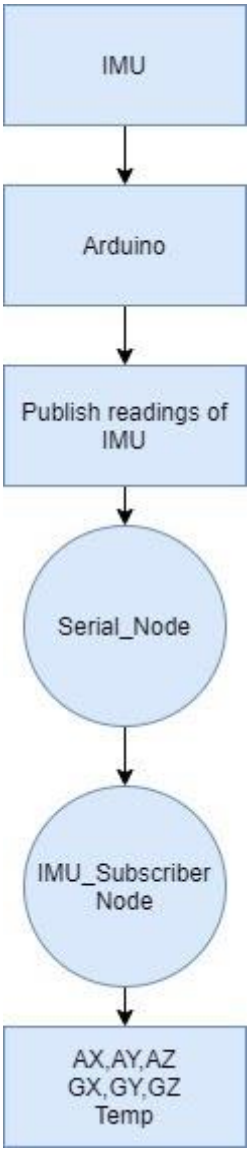


Figure 4-24: IMU and Arduino Implementation

4.6.3 Augmented Reality (AR) System

4.6.3.1 ZED Stream to Web Server Node

Web Server Node

This node is a built-in package in the ROS system where this node opens HTTP streaming for all the images that are published in the ROS system as Stereo camera node and USB camera node. The streamed data is in Format JPEG.

4.6.3.2 AR Implementation in Windows

Cameras stream from ROS to Unity:

As mentioned before, cameras' streaming is uploaded on a web server in real-time. Streaming is transmitted from both stations ubuntu and windows on a single network – on a single URL -, which is received by split camera software that retrieves each camera stream separately from the URL. Unity then accesses the stream through the split camera software, as illustrated in figure 4-25.

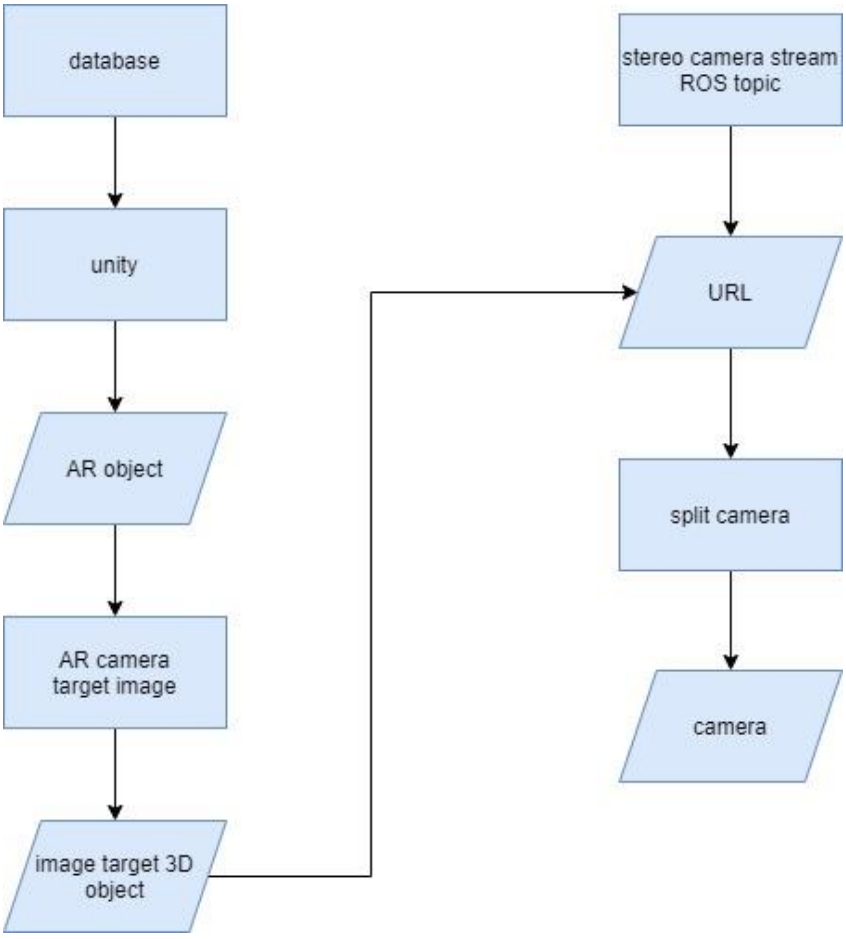


Figure 4-25: Augmented Reality and Virtual Reality Implementation

Unity 3D:

We used unity because:

- It is highly effective while rendering 2D and 3D scenes.
- Unity has a large and varied asset store
- Unity is good for VR developers
- It is a multi-Platform

Software Development Kits used in Project (SDKs):

Vuforia Engine is a software development kit (SDK) for creating Augmented Reality apps. Developers can easily add advanced computer vision functionality to any application, allowing it to recognize images and objects, and interact with spaces in the real world. Vuforia Engine supports AR app development for Android, iOS, Lumin, and UWP devices.

1. Adding Vuforia Engine to a Unity Project

Download Vuforia SDK for unity, import it unity and the package will automatically add the latest Vuforia Engine version to your Unity project, as shown in figure 4-26.

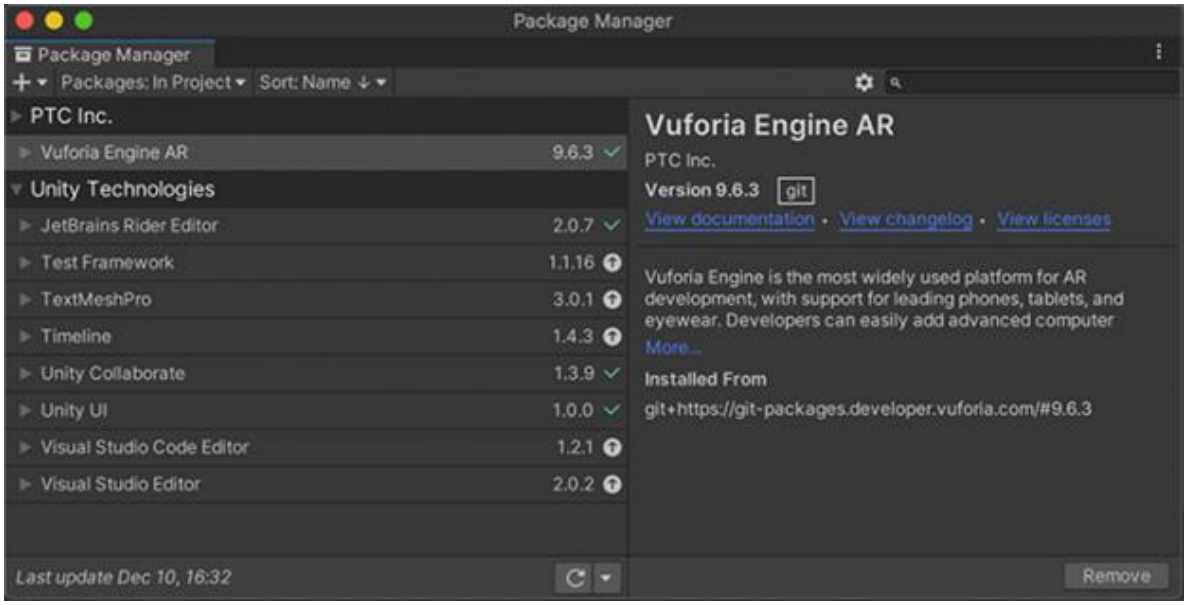


Figure 4-26: Importing Vuforia to Unity

2. Vuforia Engine Setup

The Vuforia Engine will be visible in the GameObject Menu, as shown in figure 4-27.

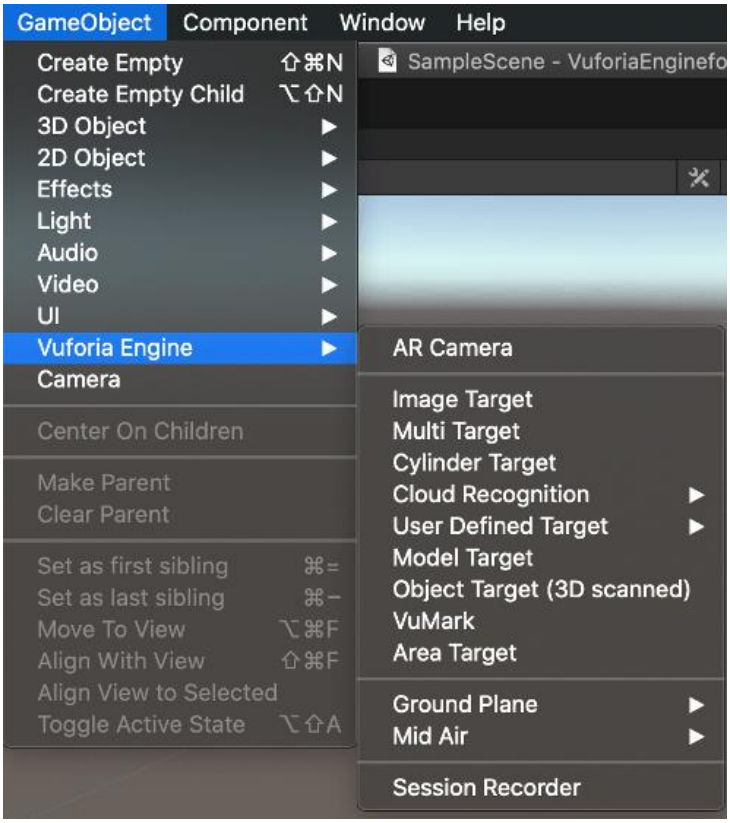


Figure 4-27: Vuforia Engine in GameObject Menu

Start by adding an ARCamera and delete the Main Camera. This is a Unity camera game object that includes the VuforiaBehaviour to add support for augmented reality apps for both handheld devices and digital eyewear.

3. Creating Vuforia License key

We created a license key from the License Manager section of the Vuforia Developer Portal. We put this into Unity's Vuforia configuration settings to build our application with Unity.

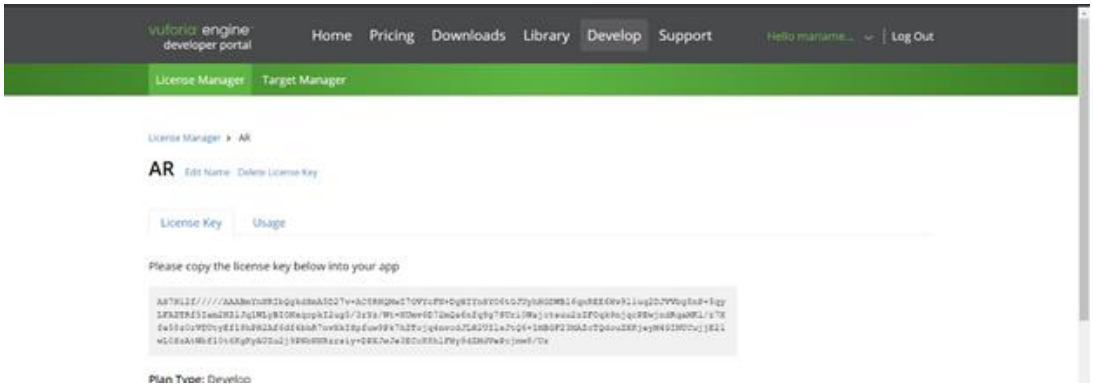


Figure 4-28: Vuforia License

Copy the license key to the clipboard and navigate back to your Unity Project and paste it in Vuforia configuration in unity.

4. Creating Database for Target Images

We added our target in Vuforia website, as shown in figure 4-29. Our target can be single image, cuboid, cylinder, or 3D object, then we added the database in our unity project.

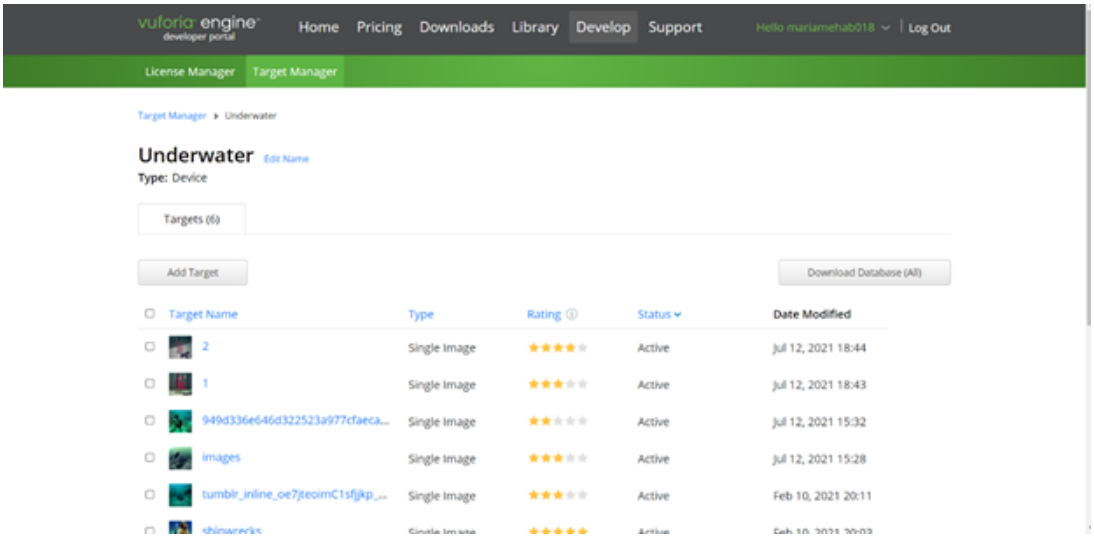


Figure 4-29: Adding the Target in Vuforia Website

Target Image

From Vuforia engine in unity we added as GameObject image, and this image will become our target image from our database. We can add any 3D object in the target image, and it will appear once the camera sees the image.

1. Building and running your app

Vuforia Engine-powered Unity apps are built and run in the same way as other Unity apps for Android, iOS, and UWP

· Oculus Quest 2 SDK

The Following metrics are used for evaluation of the efficiency and accuracy

1. Unity Asset Store

We imported the oculus integration package. This step ensures that you are using the Oculus Utilities plugin that is bundled with the package you are installing. If you choose not to update the plugin at this point, you need to manually update it later.

2. Import XR plugin

Use the XR Plug-in Management package to help streamline XR plug-in lifecycle management and potentially provide users with build time UI through the Unity Unified Settings system.

4.7 Summary

In chapter 4, the implementation of each of the 3 major systems was explained. In the next chapter, the results of these implementations will be displayed, along with the test types performed and the challenges which have faced us throughout the project.

Chapter 5: Results, Tests and Challenges

5.1 Introduction

In this chapter, the results of the system are displayed starting with YOLO model results. Afterwards, the results of the whole system integrated with ROS are displayed.

5.2 YOLO

Evaluation Metrics Of the current model:

1. Recall and precision

Precision is the probability of the predicted bounding boxes matching actual ground truth boxes. The Recall is the true positive rate, also referred to as sensitivity, measures the probability of ground truth objects being correctly detected.

Precision = TPTP+FP (1)

Recall = TPTP+FN (2)

True positive (TP) represents the samples that are predicted to be correct and actually positive. False positive (FP) represents the samples that are predicted to be positive, but actually negative. In addition, False negative (FN) represents the samples that are predicted to be negative and actually negative.

2. The Average precision

The average precision calculated for all class in the model by using the following formula:

Average precision = 01p(r)dr (3)

The formula calculates area under curve of Precision and Recall

3. Mean Average Accuracy (mAP)

mAP means that the average accuracy of all categories is added, and it is divided by the number of categories, shown in formula:

mAP = I = 1NAPiN (4)

Where **N** is the number of object classifications. We use two ranges of average precision mean, map@0.5, and map@0.5:0.95. The mAP@0.5 represents the average accuracy of the confidence threshold of 50%.

5.2.1 YOLO Test and Results

A. Results

Table 5-1: Precision Testing Results on different Classes

Class	Average Precision
Coral	96.8%
Sponge	95.2%
Coral bed	97.0%
starfish	95.8%

Precision: 95%**, Recall:** 97%**, F1 Score:** 95%**, Intersection of Union:** 80% and **Mean Average Precision (mAP@0.50) =** 0.960218

Evaluation Matrix Graphs

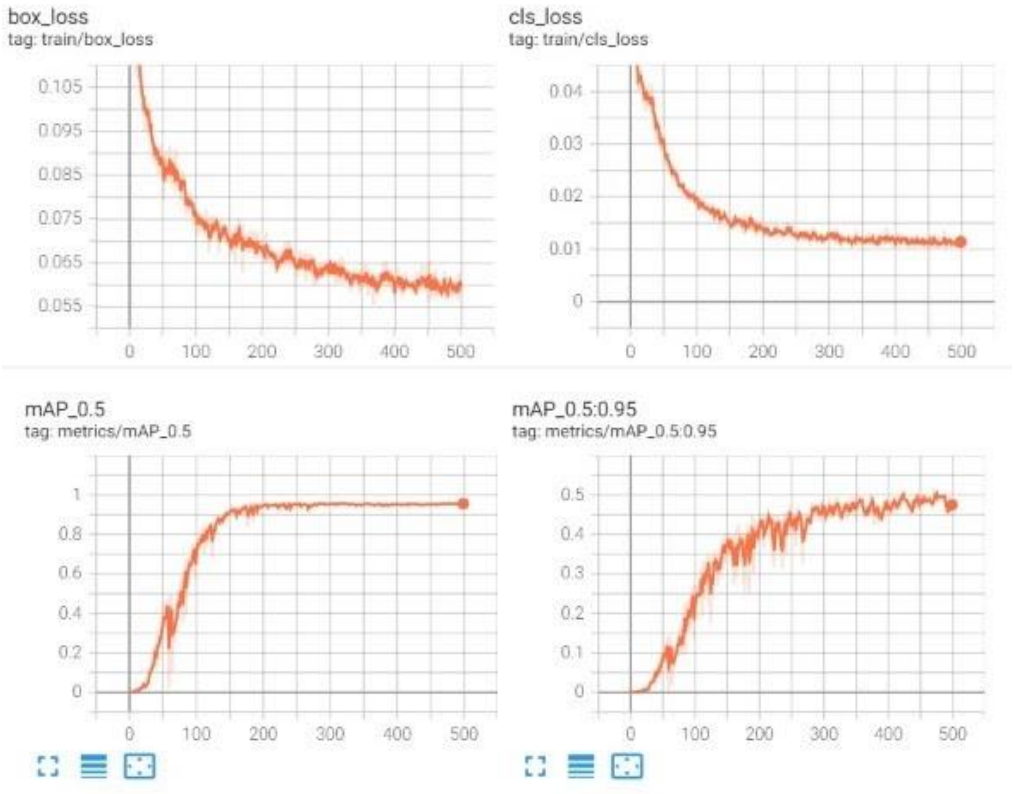


Figure 5-1: Evaluation Matrix Graphs

The following dataset, shown in figure 5-2 and figure 5-3, is used to test the YOLO Model. Figure 5-4 and figure 5-5 are the results of YOLO tested on the coral reefs dataset.

Figure 5-2: Coral Reefs Dataset 1

Figure 5-3: Coral Reefs Dataset 2

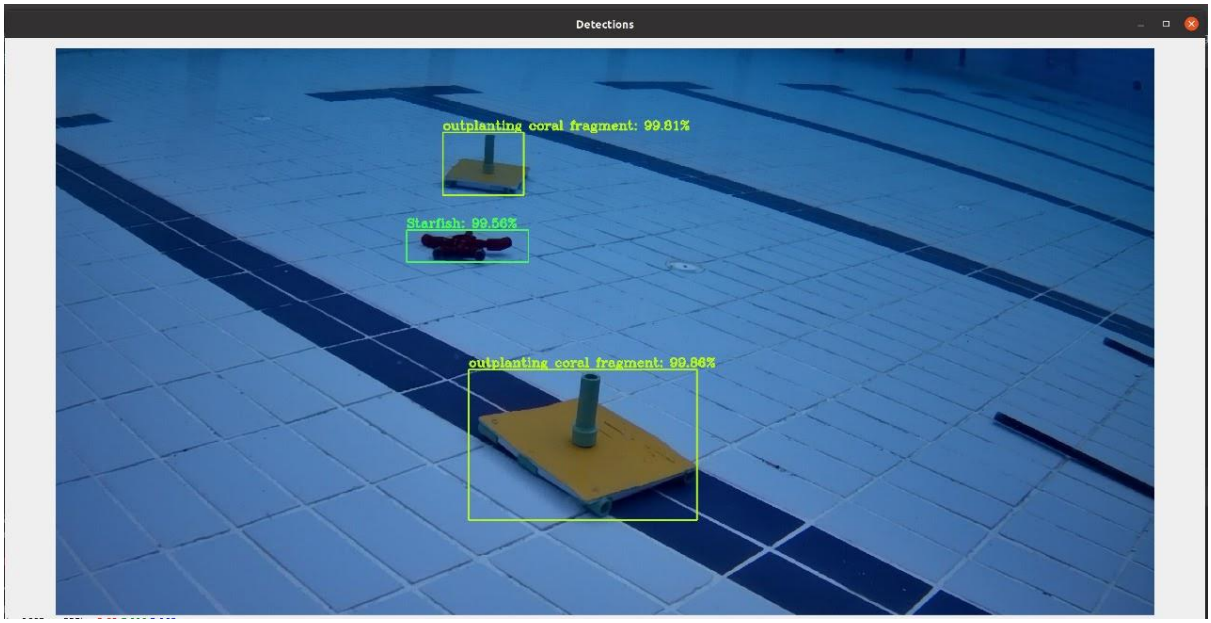


Figure 5-4: Detected Results 1

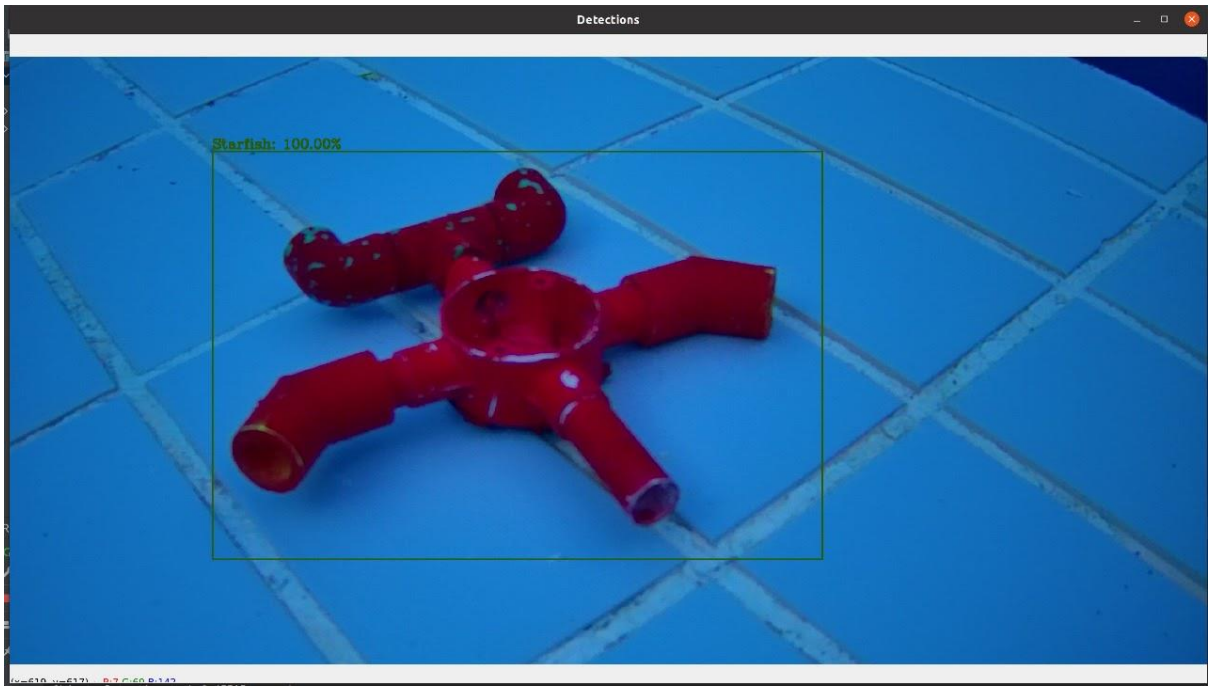


Figure 5-5: Detected Results 2

5.3 System Overview

Figure 5-6 shows the implemented system overview using the rqt_graph tool in ROS.

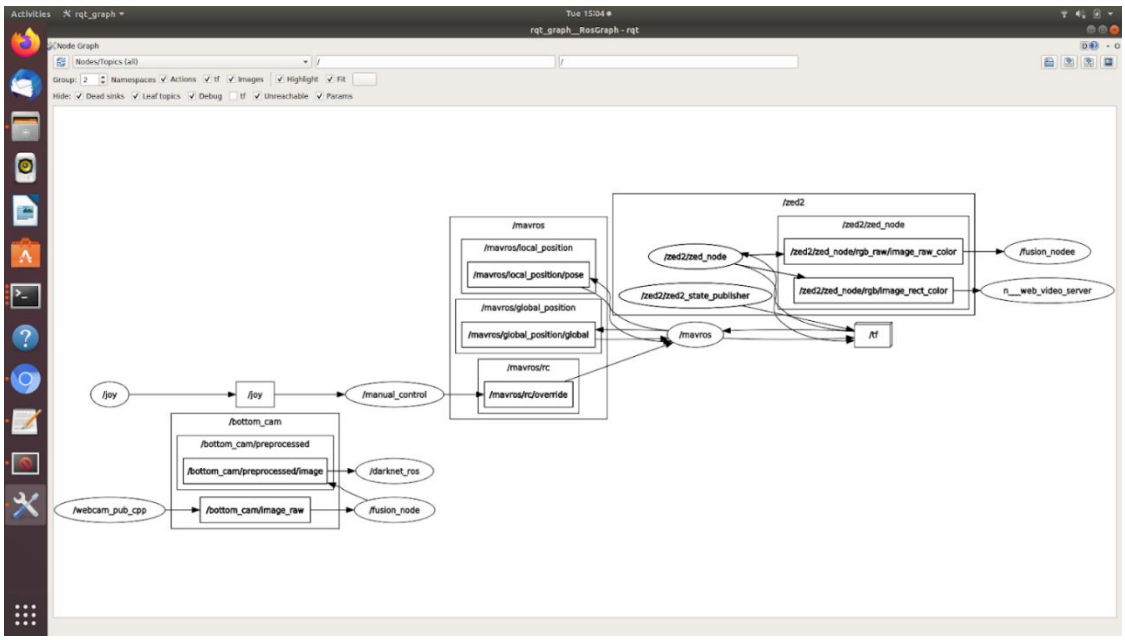


Figure 5-6: ROS System Overview

5.3.1 Computer Vision System

5.3.1.1 Monovision Nodes

The following figures are the results of the implemented Monovision nodes. The following figures show the implemented Monovision system through using rqt_graph tool.

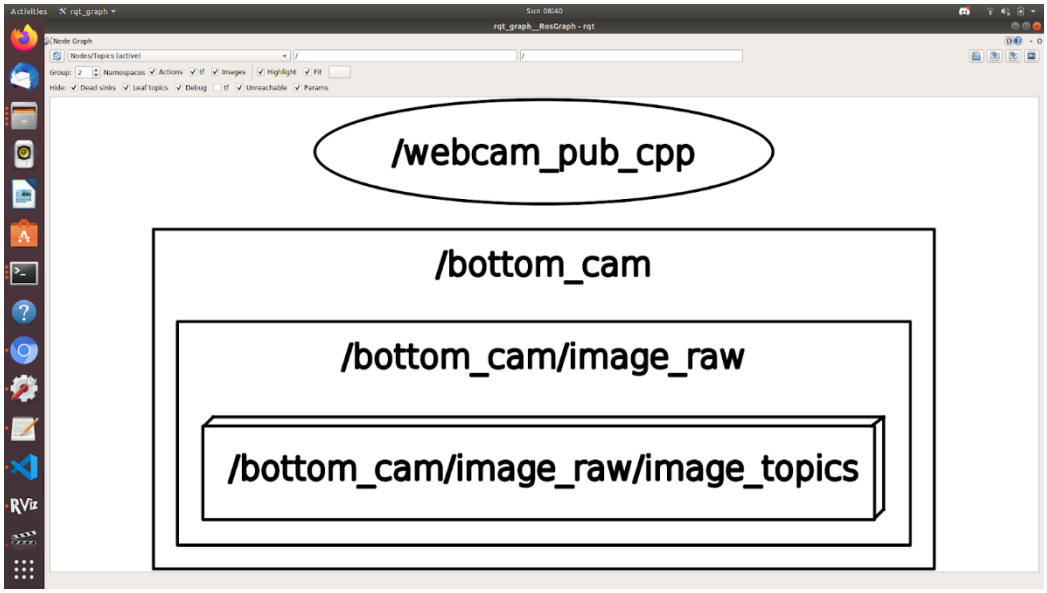


Figure 5-7: Monovision Node Topics

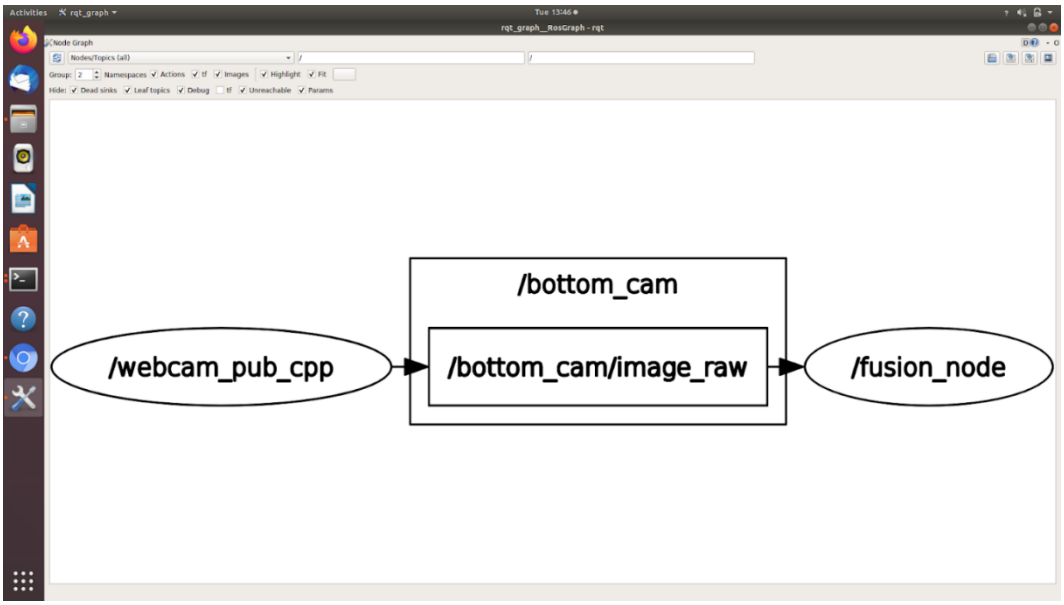


Figure 5-8: USB Camera Node Connected with Image Enhancement Node in ROS

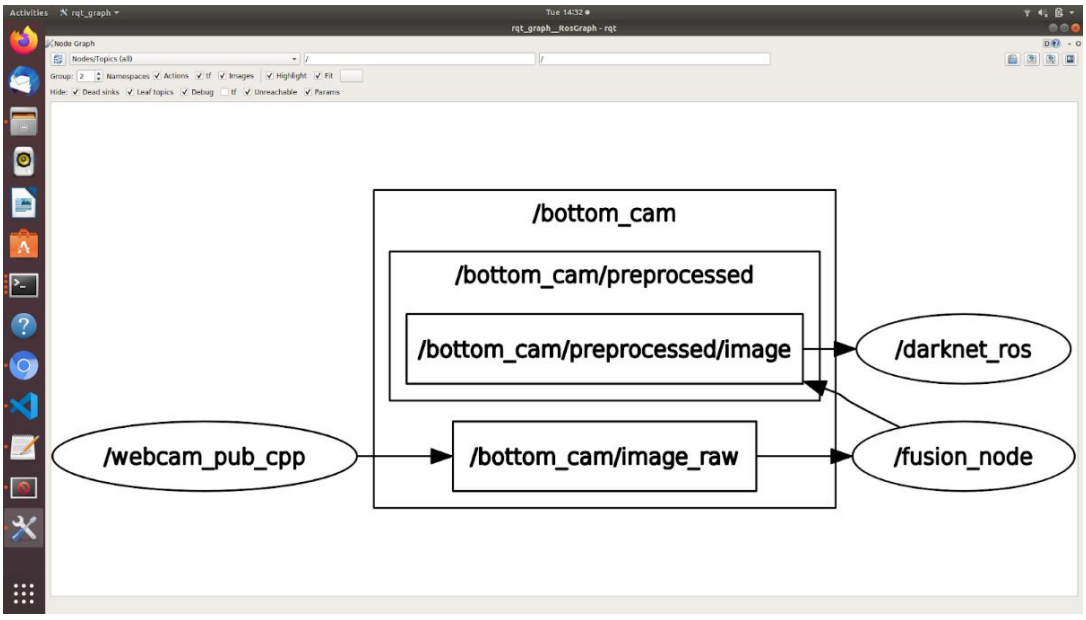


Figure 5-9: USB Camera Node Connected with Image Enhancement Node and Object Detection in ROS

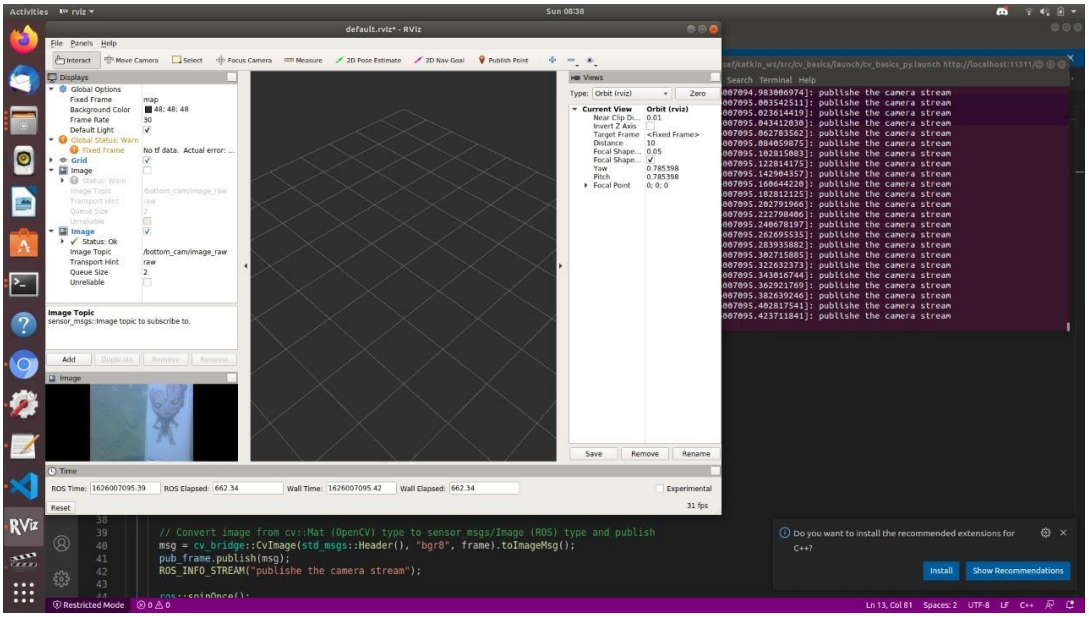


Figure 5-10: USB Camera Running in ROS

5.3.1.2 Stereo Vision Nodes

The following figures are the results of the implemented of Stereo Vision nodes. The following figures show the implemented Stereo Vision system through using the rqt_graph tool.

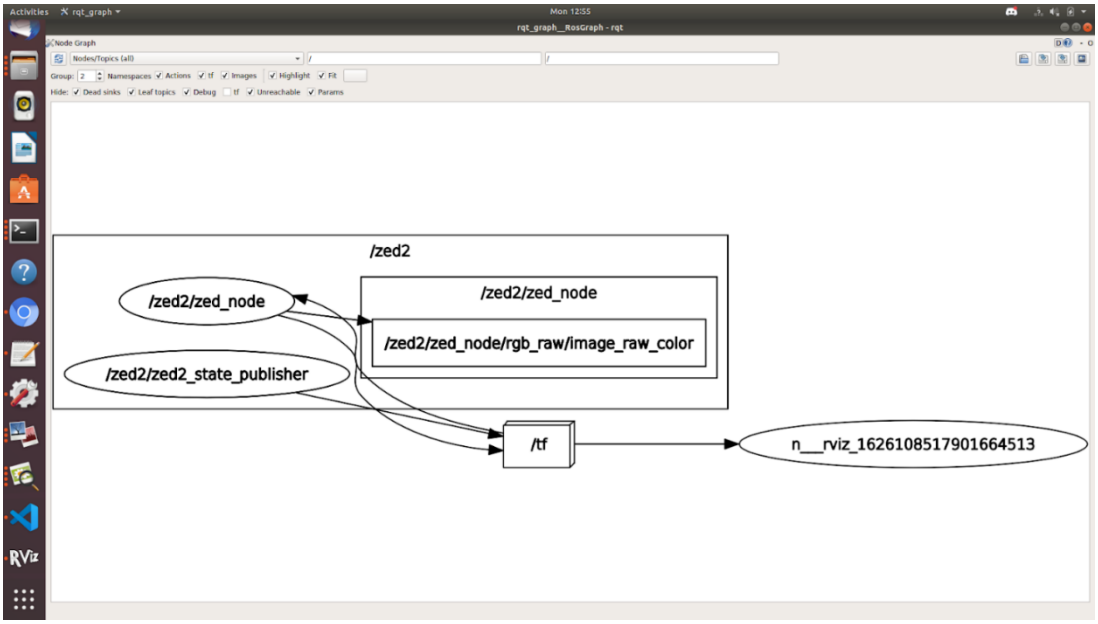


Figure 5-11: Stereo Vision Node Topics

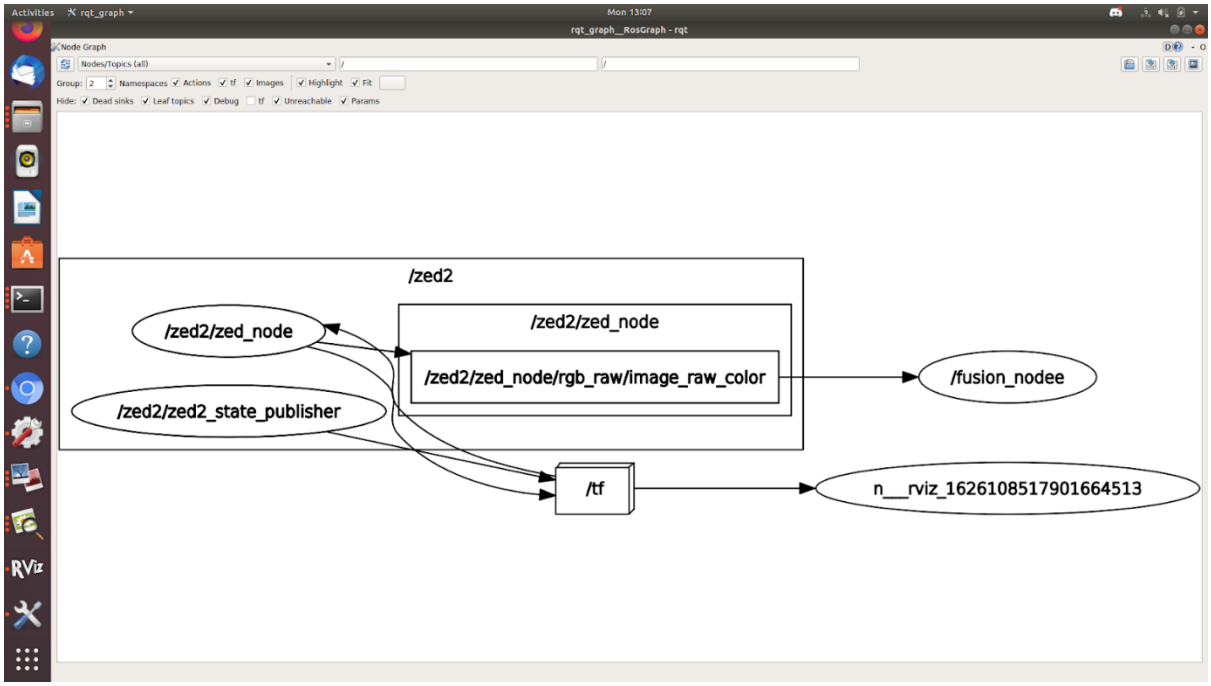


Figure 5-12: ZED Stereo Camera Node Connected with Image Enhancement Node in ROS

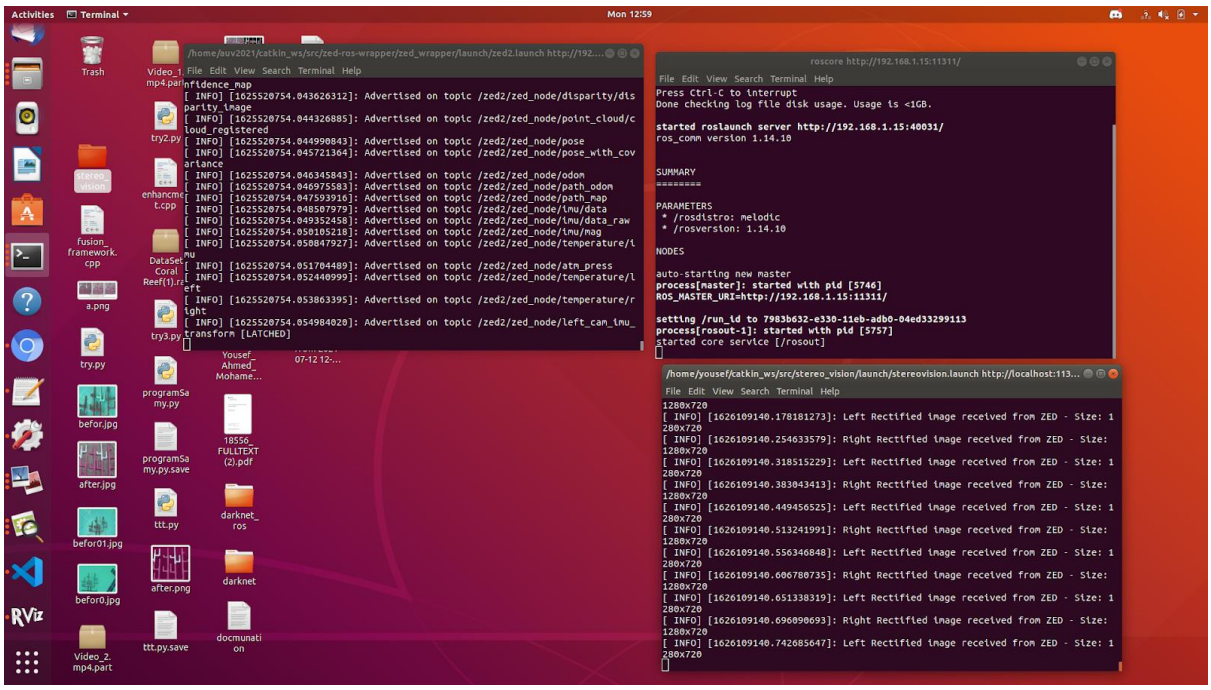


Figure 5-13: ZED Stereo Camera Running with Image Enhancement in ROS

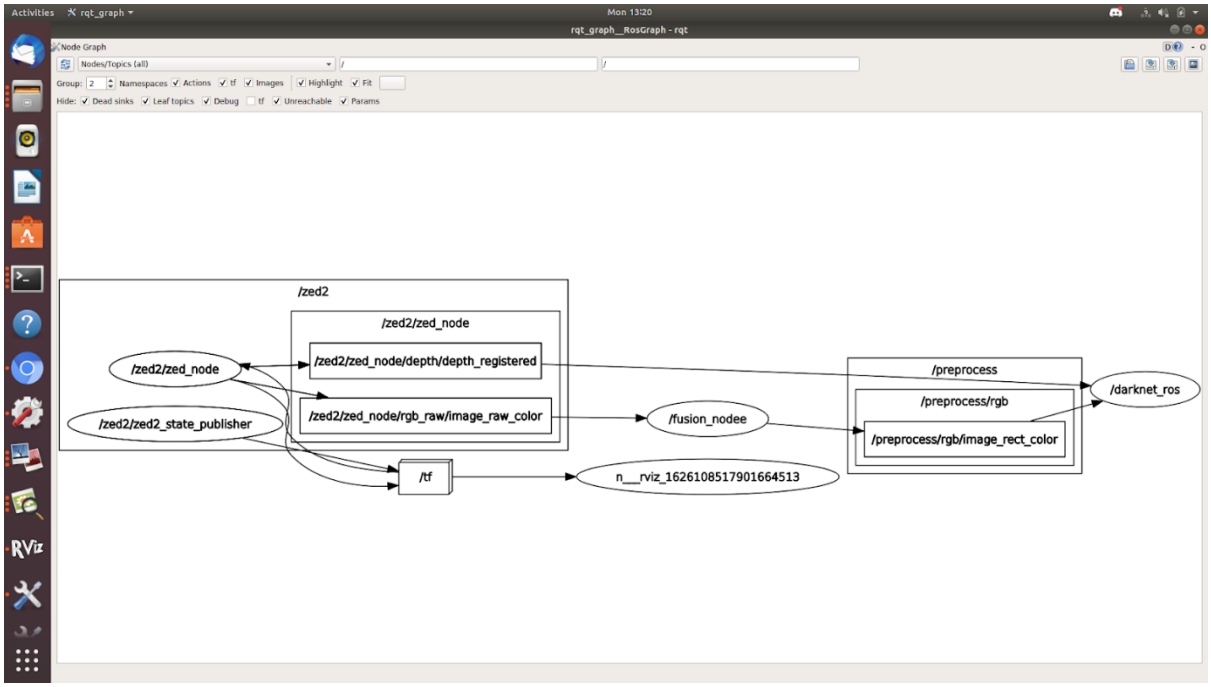


Figure 5-14: ZED Stereo Camera Connected with Image Enhancement Node and Object Detection Node in ROS

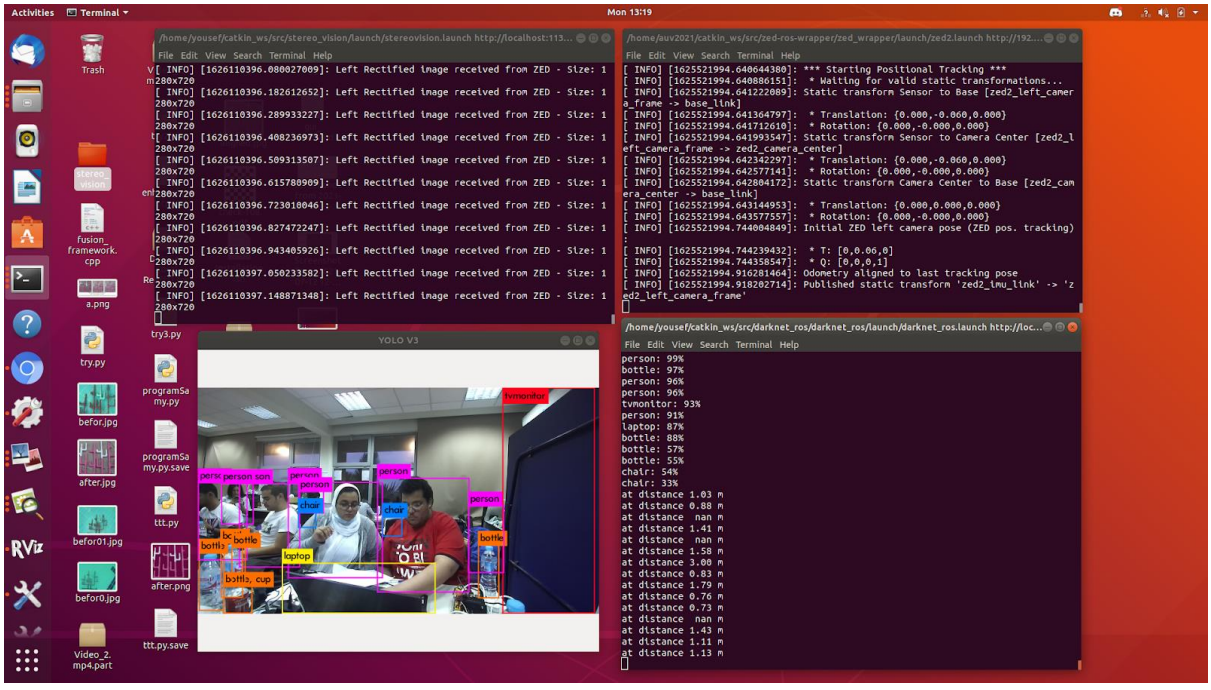


Figure 5-15: ZED Stereo Camera Running with Image Enhancement and Object Detection in ROS

5.3.2 Control System

5.3.2.1 Joystick, Manual Control and MAVROS Nodes

The following figures are the results of the implemented Joystick, Manual Control and MAVROS nodes which illustrates how they're integrated. The following figures show these nodes and how they're implemented through using the rqt_graph tool.

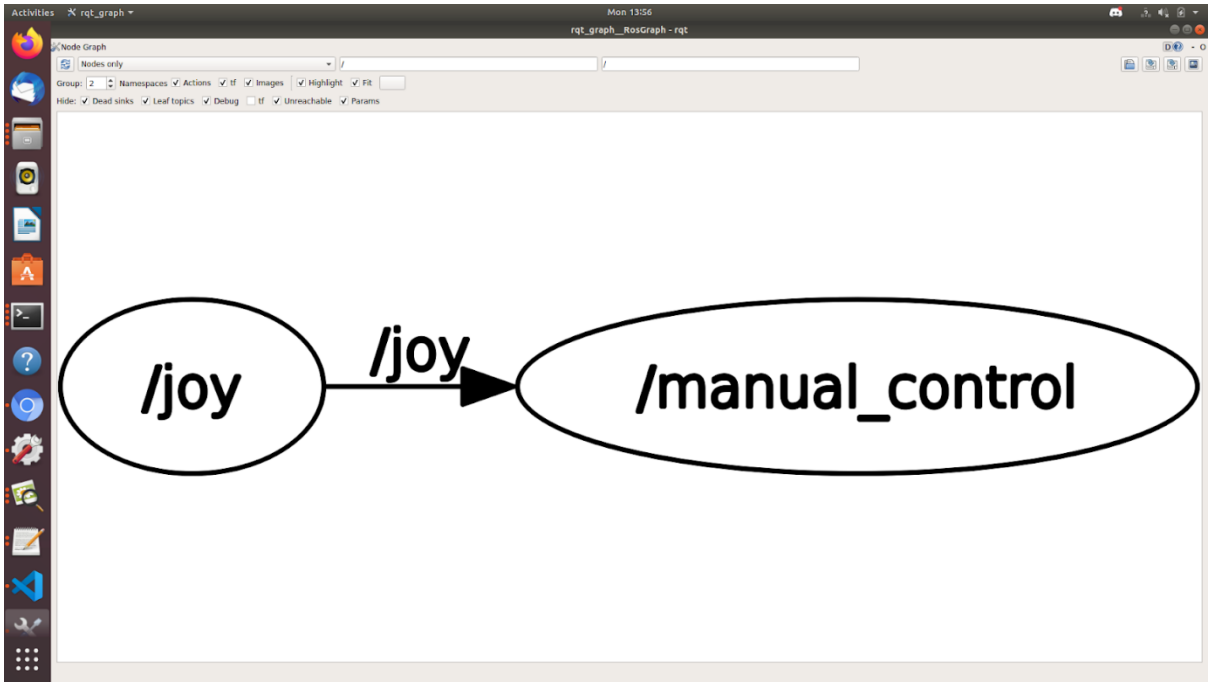


Figure 5-16: The Joystick Publishes /joy Message to manual_control Node in ROS

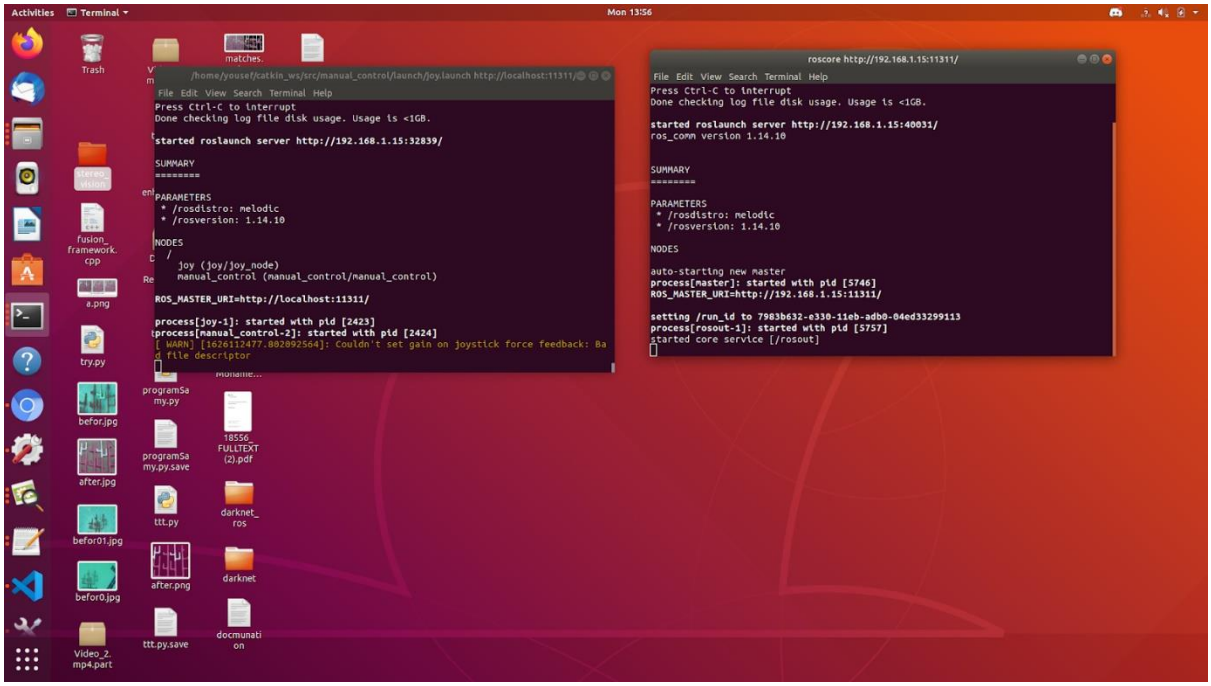


Figure 5-17: Joystick and manual_control Running in ROS

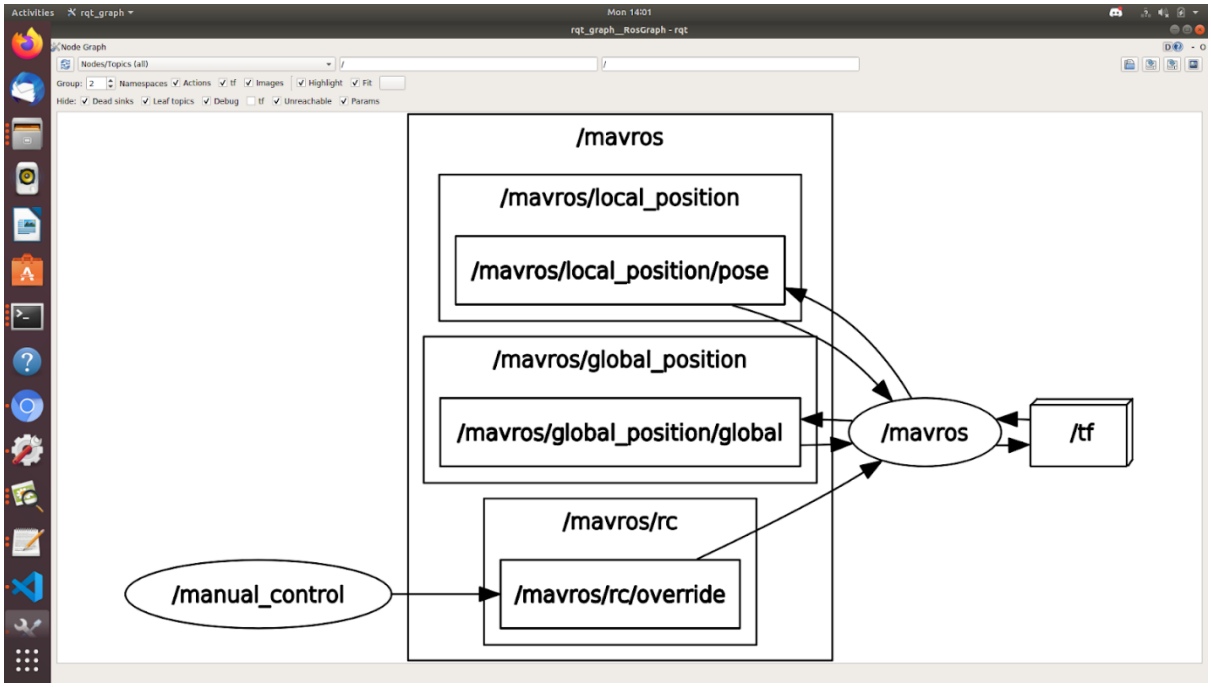


Figure 5-18: Joystick Connected with Manual Control and MAVROS Nodes in ROS

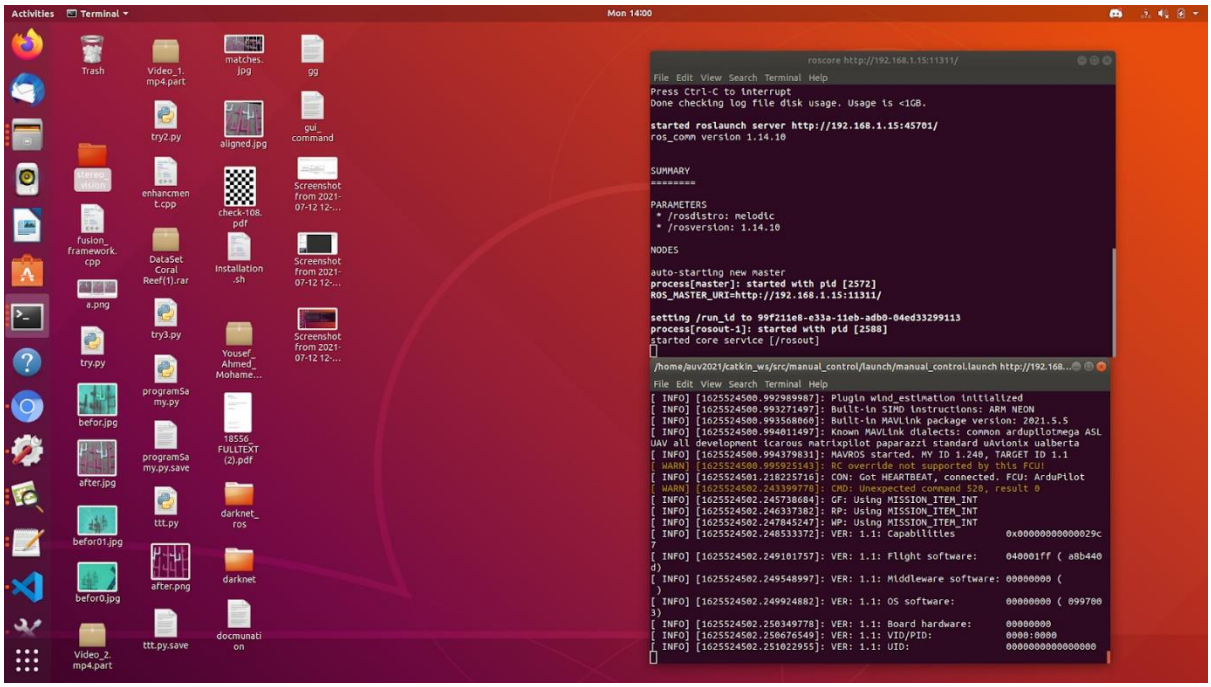


Figure 5-19: roscore and MAVROS Running in ROS

5.3.2.2 Arduino Node

The following figures are the results of the implemented Arduino node. The following figures how they're implemented through using the rqt_graph tool.

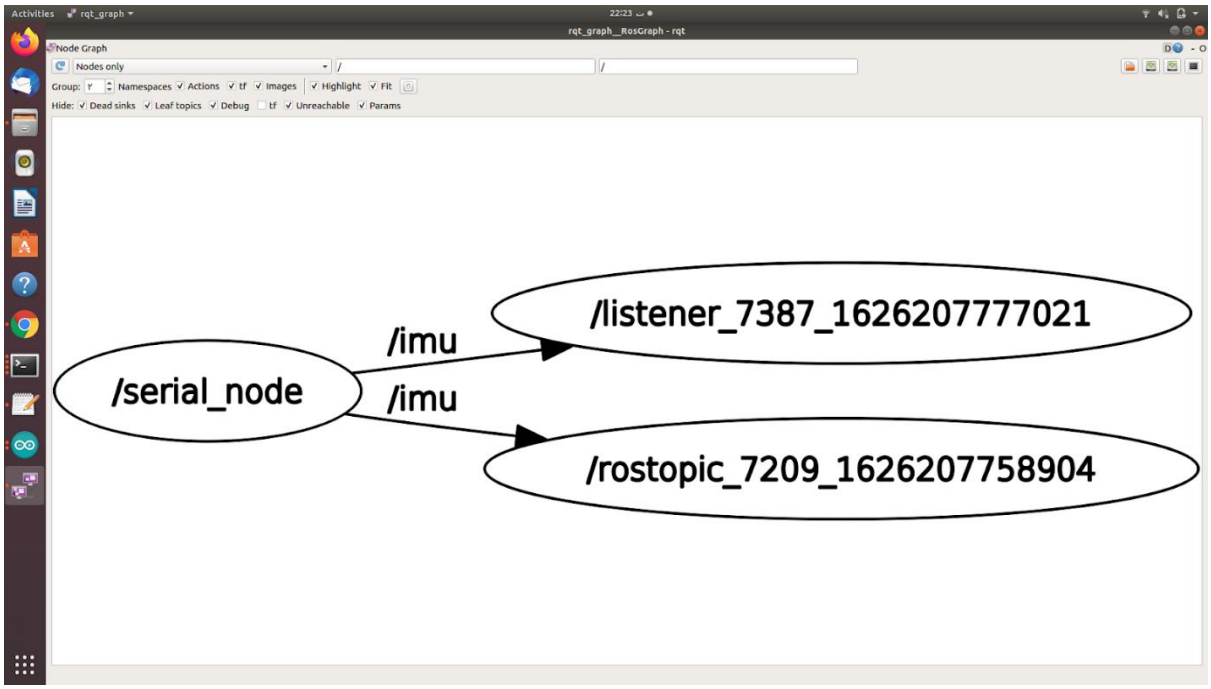


Figure 5-20: Arduino Nano Node Publishing IMU Readings Through rosserial Protocol in ROS

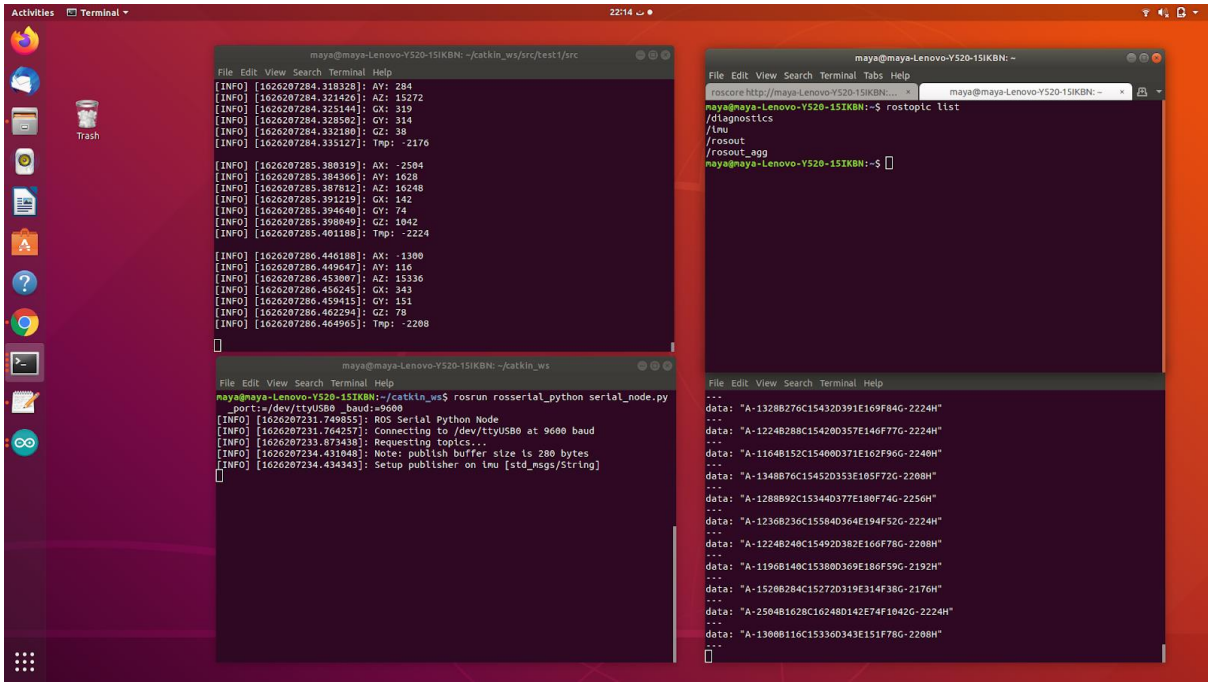


Figure 5-21: Arduino Nano Node Running in ROS

5.3.3 GUI

The following figure shows the GUI running in ROS with all the topics in the system.

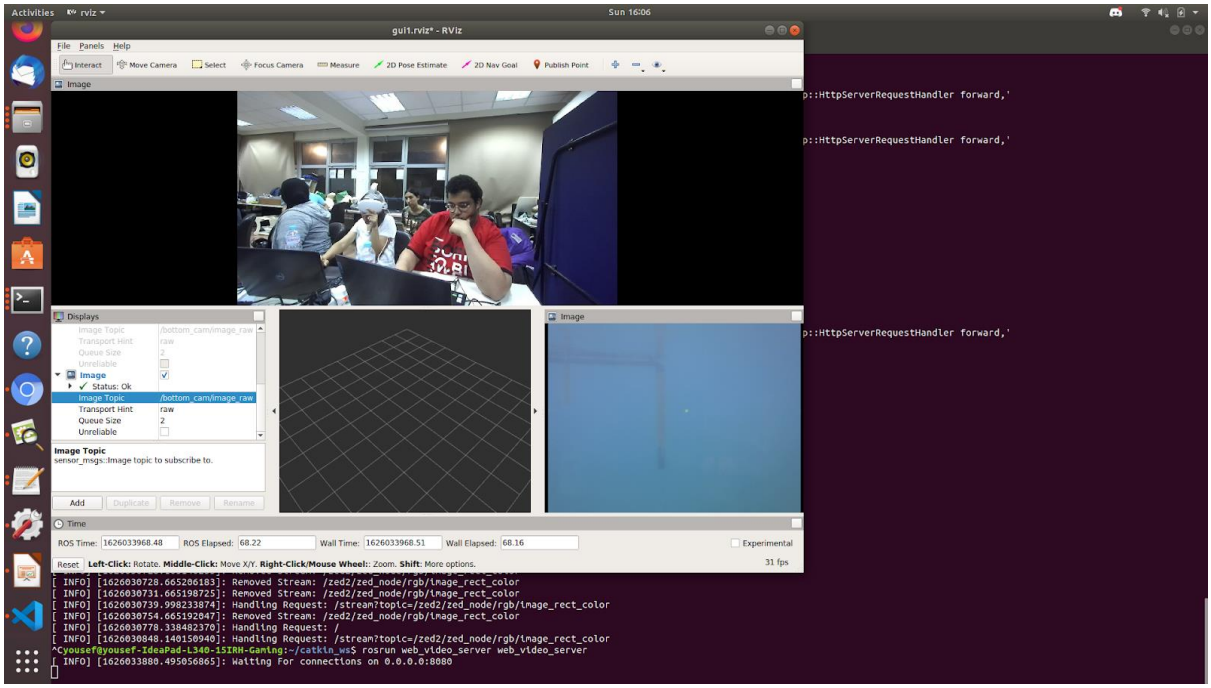


Figure 5-22: GUI Running in ROS

5.3.4 Augmented Reality (AR) System

The following figures show the web server results after being streamed in real-time from ZED Stereo Camera.

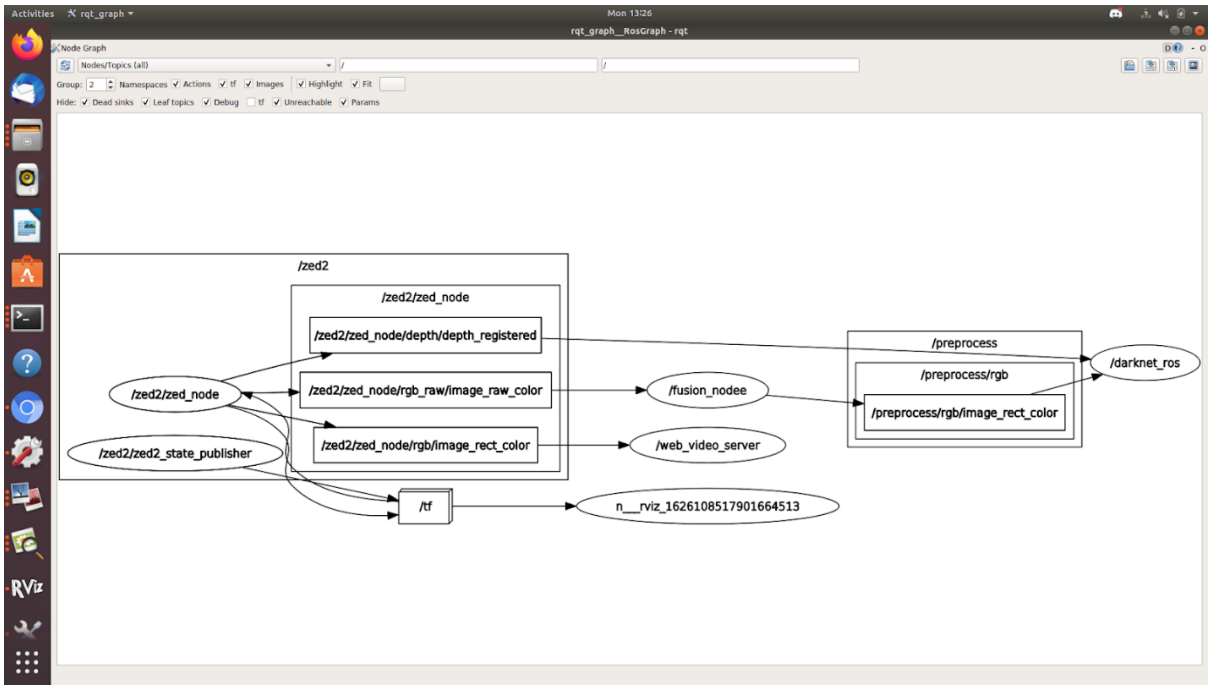


Figure 5-23: Stereo Vision Node Connected with Web Server in ROS

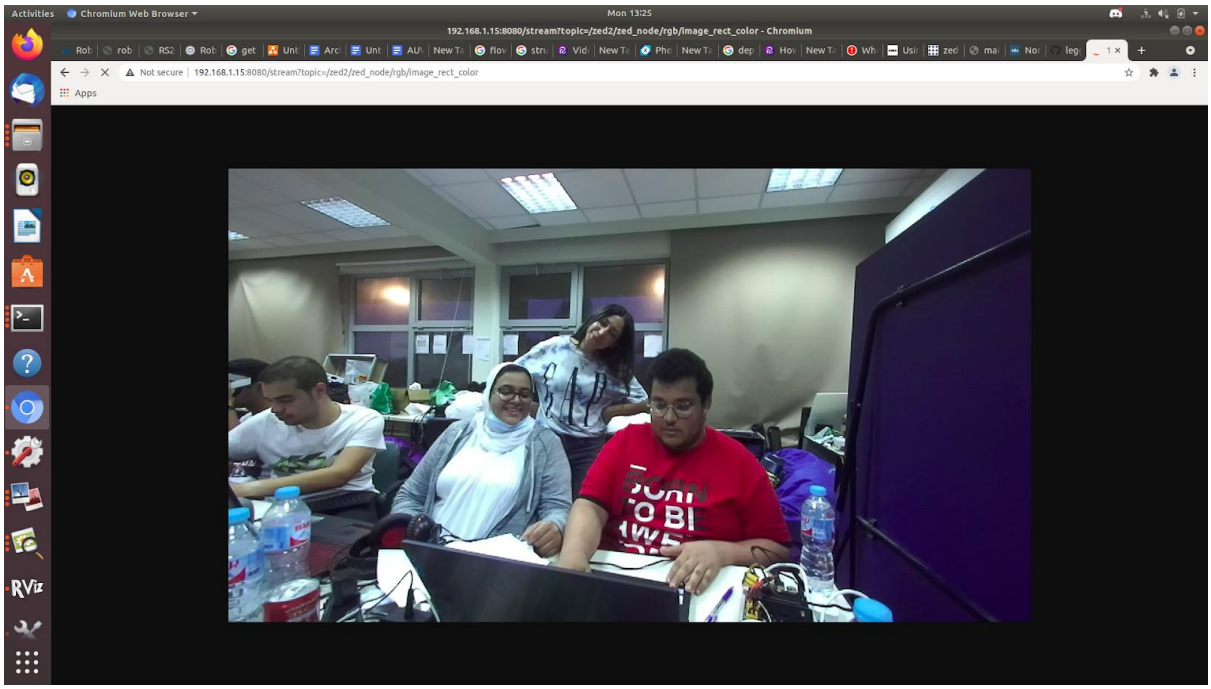


Figure 5-24: Web Server Streaming ZED in ROS

The following figures show the AR implemented in unity.

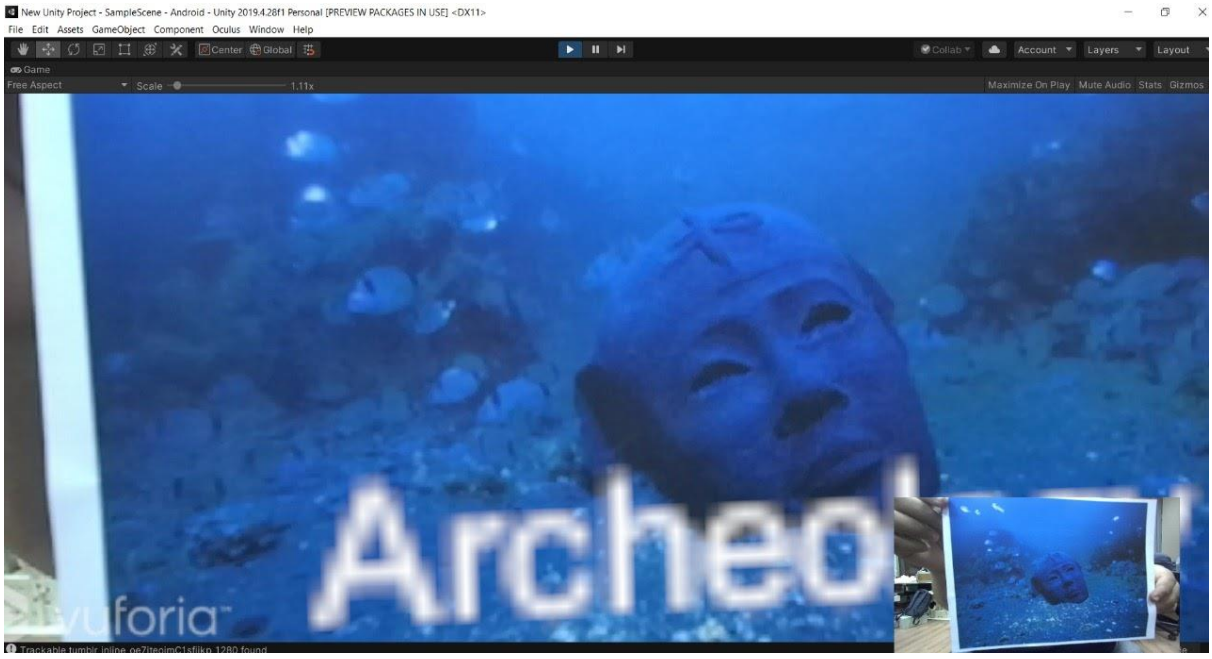


Figure 5-25: Implemented AR Output on Vuforia 1

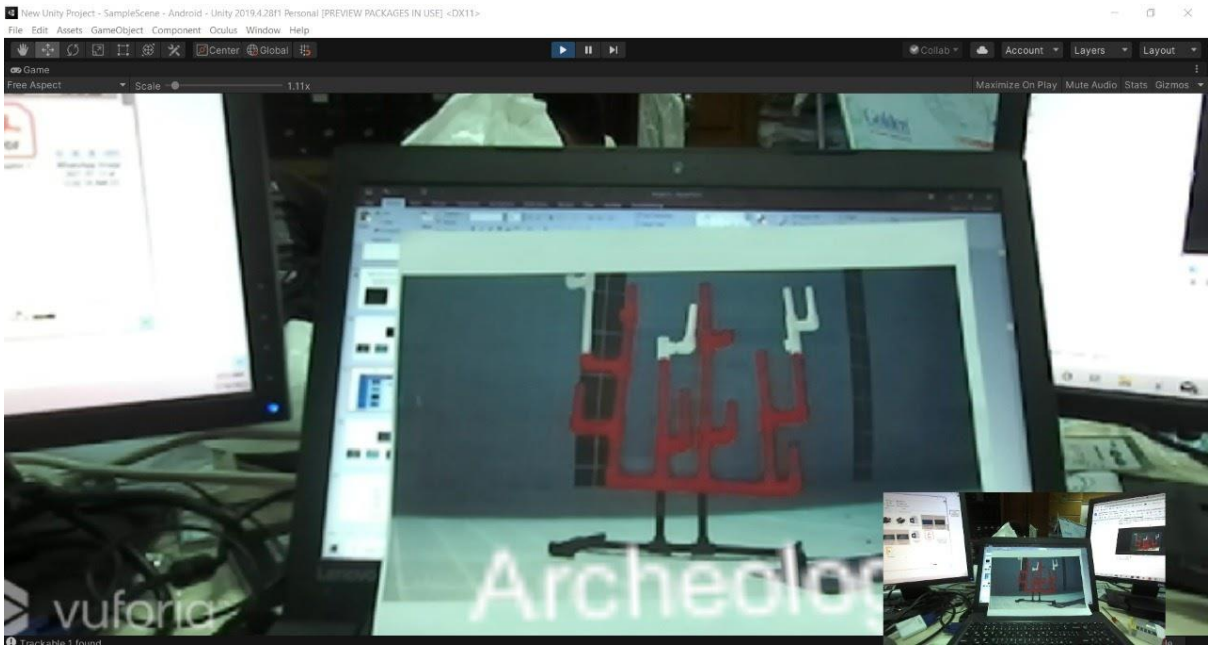


Figure 5-26: Implemented AR Output on Vuforia 2

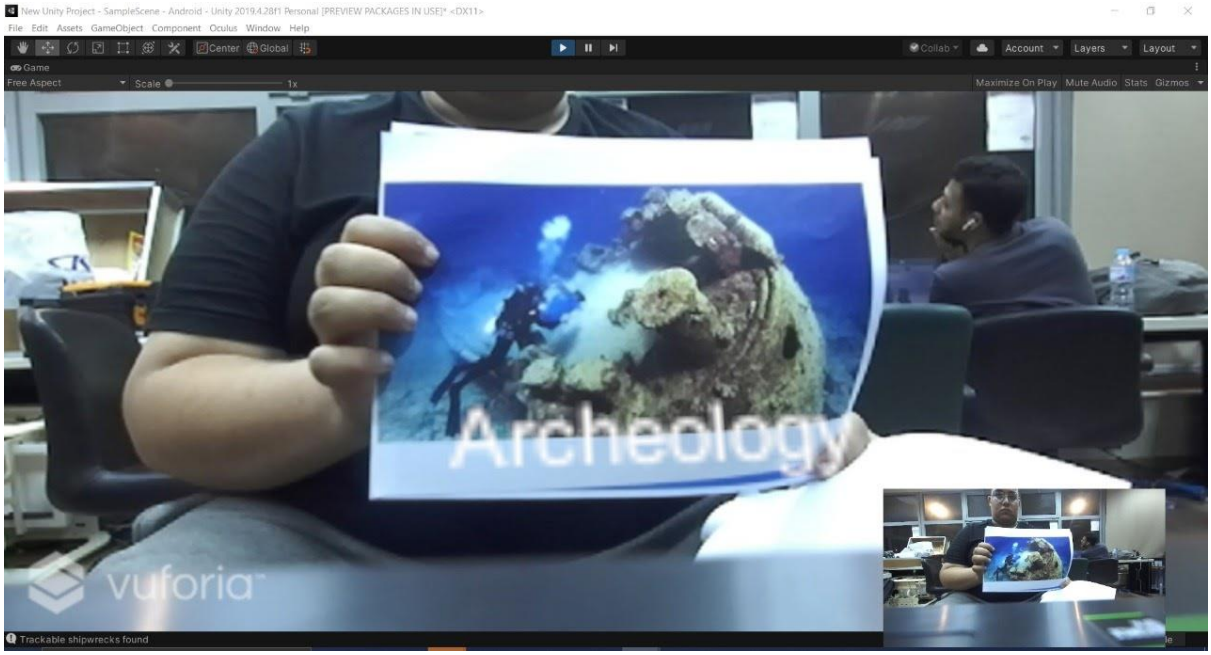


Figure 5-27: Implemented AR Output on Vuforia 3

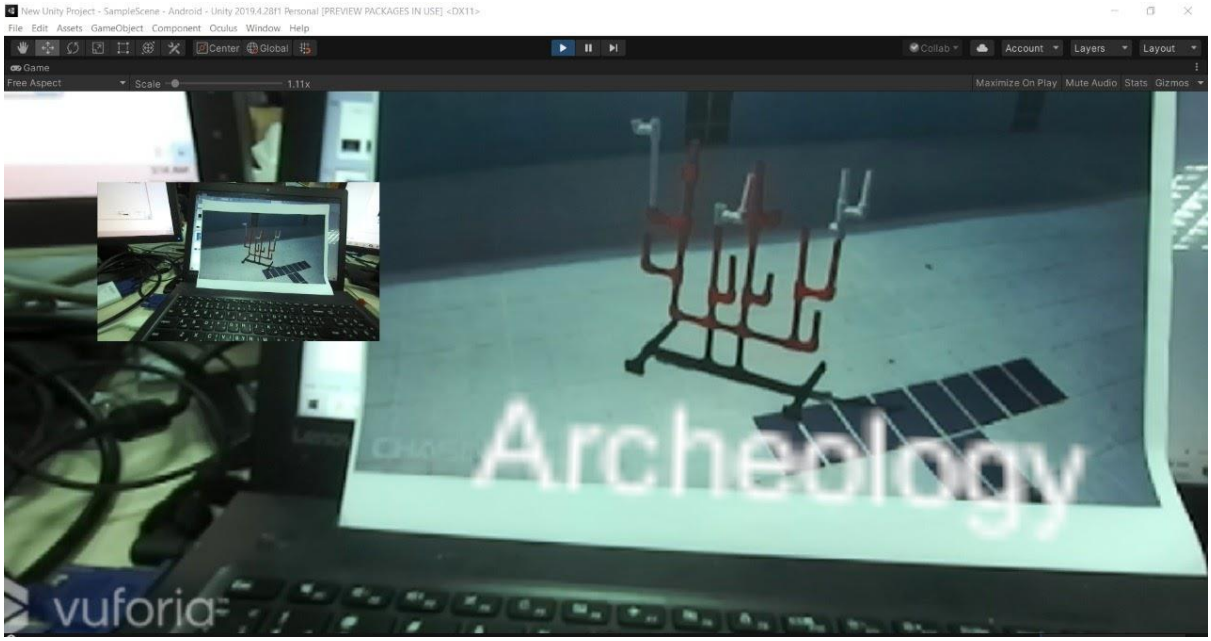


Figure 5-28: Implemented AR Output on Vuforia 4

5.4 Testing

5.4.1 Testing and Troubleshooting:

Testing and troubleshooting were an objective for us from the smallest mechanical part to the largest functional code. In line with our safety protocol and the implementation of the V model, vehicle testing consists of a four-step verification and validation process that is carried out prior to attempting mission runs.

Phase one – Unit testing: For mechanical systems, 3D printed models were used to prototype design ideas in order to understand the nature of their assembly in the real world. Also, all sealed enclosures undergo a water test for a various number of hours to ensure water sealing before any electronic components are placed within. Moreover, in the real world they are tested individually in an isolated environment to avoid any major accidents. As for software, algorithms are tested by implementing “Statement Coverage” and “Decision Coverage” techniques where all statements, especially decision-based ones are tested.

Phase two – Integration testing: Once a new unit passes its unit test, it is integrated with the relevant subsystem. The subsystem is tested where specific inputs – obtained from truth tables, equivalence partition and boundary values techniques - are fed, and monitoring the corresponding outputs to ensure it works as expected. Furthermore, interpreters are used to run partial code snippets to ensure the absence of critical bugs and guarantee safe operation of the code.

Phase Three – System testing: After testing all subsystems fully in simulation, and once the vehicle has been assembled electrically and mechanically, it is tested in the real world. It has a dry run phase test where it is tested in a safe and functional environment on land. Once the dry run is complete, phase four of the testing protocol begins to test the vehicle in water.

Phase Four – Acceptance Testing: The AUV is stress-tested underwater for many consecutive days to ensure that it can achieve its full capabilities. In case of any underwater malfunctions, a root cause analysis (RCA) is carried out to locate the malfunction. Once it is located, solutions are brainstormed which are subsequently tested through the same four phases to ensure optimum functionality. If no malfunctions are found, the operators use the AUV to perform all missions. The obtained performance in each mission is used to validate the acceptance test meeting the requirements set by clients.

5.5 Challenges

5.5.1 Challenges in case of Hardware Failure

Using hardware in any project presents a risk due to being prone to hardware failure at any time. To be ready for such problems, alternatives have been purchased to almost every main component.

5.5.2 Challenges in Integrating the ROS System

The ROS community is not well supported and faced a lot of bugs that consumed much effort to solve because the ROS version is in the beta version.

5.5.3 Challenges in Dataset Collection

It took time in collecting dataset with high quality that can be used in training the model to get the best accuracy.

5.5.4 Challenges in Manufacturing

Workshops were under lockdown and a limited number of technicians were authorized to help in the project due to the COVID-19 pandemic, which caused a slight delay in the manufacturing timetable.

5.5.5 Challenges in Testing

It is hard to test the vehicle in seawater. As a result, the test takes place in the AASTMT swimming pool, but it is inconvenient sometimes due to the COVID-19 pandemic.

5.5.6 Challenges in Budget

Since the project is self-funded and due to purchasing extra components, providing a sufficient budget has been difficult.

5.6 Summary

This chapter explained everything that has been implemented throughout this project. The next chapter will introduce the future work and what we concluded and learned through this project.

Chapter 6: Conclusion and Future Work

6.1 Introduction

In this chapter, the future work that is hoped to be achieved will be briefly listed, along with a brief conclusion of what has been achieved.

6.2 Future Work

AUVs are now at an early stage of acceptance. As they work their way into the phase of industrial acceptance on the commercial side, their numbers will grow immediately. Also, maritime operations expand into deep waters, more remote locations or more complex conditions, operators need more intelligent, safer, more cost-effective solutions. Some of the most important features to be added:

1. Localization and Navigation System

simultaneous localization and mapping (SLAM) techniques applied for above ground applications are being increasingly applied to underwater systems. The result is that limited error and accurate navigation for the AUVs is becoming possible with less cost and overhead. Also, we can use Sonar for detecting and locating long range objects. Sonars are devices that produce sound waves of controlled frequencies, and listen for the echoes of these produced sounds returned from objects in the water.

2. 360° View.

We can integrate an omnidirectional underwater camera with an underwater autonomous vehicle. The integration of omnidirectional cameras with underwater robots is expected to have a large impact in both Remotely Operated Vehicles (ROVs) and Autonomous Underwater Vehicles (AUVs).

3. Piloting with the VR Headset Joysticks

VR headset has a more important job than only viewing the underwater scene; it also can be used for piloting the AUVs and see some dashboard parameters which can be used for 360° subsea virtual tours which can capture, record and transmit a live mission feed directly to a supported Virtual Reality (VR) headset.

4. Automatic Mechanical Storage System.

Adding a storage system in AUV can be very helpful in some missions such as: underwater rescue missions, picking, dropping objects and cable burial.

5. Obstacle Avoidance

It is very important to add an obstacle avoidance feature. The goal is to detect obstacles that represent a threat to the AUV to be more efficient and to avoid hitting any objects in the

specific path. This includes use of computer vision techniques for detecting and measuring obstacles and the design of autopilot controllers for efficient avoidance behaviours.

6.3 Conclusion

In conclusion, the aim of this project is to assist the authorities in thriving the tourism in Egypt through exploring the seabed to search for underwater archaeology and oceanography to increase the number of touristic diving areas. The achieved vehicle is a semi-autonomous vehicle that can be controlled by a pilot using a joystick and move autonomously toward a detected object. In the future work, the proposed vehicle is planned to be fully autonomous.

7. References

[1] Society, N.A., “Underwater Archaeology: The NAS Guide to Principles and Practice, 2nd Edition”, Wiley-Blackwell, 2008

[2] http://www.getty.edu/publications/artistryinbronze/large-scale-bronzes/3-koutsouflakis/

[3] https://www.redsea-divingsafari.com/eco-efforts/marine-life

[4] Fine, M., Cinar, M., Voolstra, C.R., Safa, A., Rinkevich, B., Laffoley, D., Hilmi, N., Allemand, D., “Coral reefs of the Red Sea — Challenges and potential solutions”, Regional Studies in Marine Science, 2019.

[5] https://www.marineinsight.com/types-of-ships/everything-you-ever-wanted-to-know-about-autonomous-underwater-vehicle-auv/#:~:text=For%20Military%20Purposes%3A%20Where%20AUVs,be%20used%20for%20underwater%20purposes.

[6] Inzartsev, A. and Alexander Pavin. “AUV Application for Inspection of Underwater Communications.”, Institute of Marine Technology Problems (IMTP FEB RAS), Russian Academy of Sciences, Russia, 2009.

[7] Niu, H., Adams, S., Hussain, T., Bose, N., & Lee, K., “Applications of Autonomous Underwater Vehicles in Offshore Petroleum Industry Environmental Effects Monitoring”, Canadian International Petroleum Conference, June 2007, Calgary, Alberta, Canada, doi:10.2118/2007-116

[8] http://www.unesco.org/new/en/culture/themes/underwater-cultural-heritage/underwater-cultural-heritage/

[9] https://www.ifpri.org/publication/covid-19-and-egyptian-economy-estimating-impacts-expected-reductions-tourism-suez-canal

[10] https://www.nationalgeographic.com/culture/archaeology/underwater-archaeology/

[11] https://www.genre.com/knowledge/publications/uwfocus152beijeen.html#:~:text=Medical%20problems&text=It%20is%20important%20to%20note,cause%20of%20death%20in%20divers.

[12] http://www.cellula.com

[13] http://www.cellula.com/solus-lr

[14] http://www.cellula.com/imotus-1

[15] Joseph, L., “Mastering ROS for Robotics Programming”, UK: Packt Publishing Ltd., 2015.

[16] https://tartanauv.com/

[17] https://tartanauv.com/albatross/

[18] https://tartanauv.com/kingfisher/

[19] Wang, E., Scherlis,T., Jain, R., Gandhi, N., Nabah, R., “Tartan Autonomous Underwater Vehicle Design and Implementation of TAUV-19: Albatross”, Carnegie Mellon University, 2019

[20] Scherlis,T. , Wood, J.,Sethuraman, A., Sodon, J., Sayeed, M., Jain, R. and Agarwal,S., “Tartan Autonomous Underwater Vehicle Design and Implementation of TAUV-20: Kingfisher”, Carnegie Mellon University, 2020.

[21] https://www.auv-iitb.org/index.html

[22] https://www.auv-iitb.org/vehicles2.html#

[23] Sneh R. Vaswani, others, “System Design and Implementation of Autonomous Underwater Vehicle “Matsya” “, AUV-IITB, Indian Institute of Technology, Bombay, India, 2012.

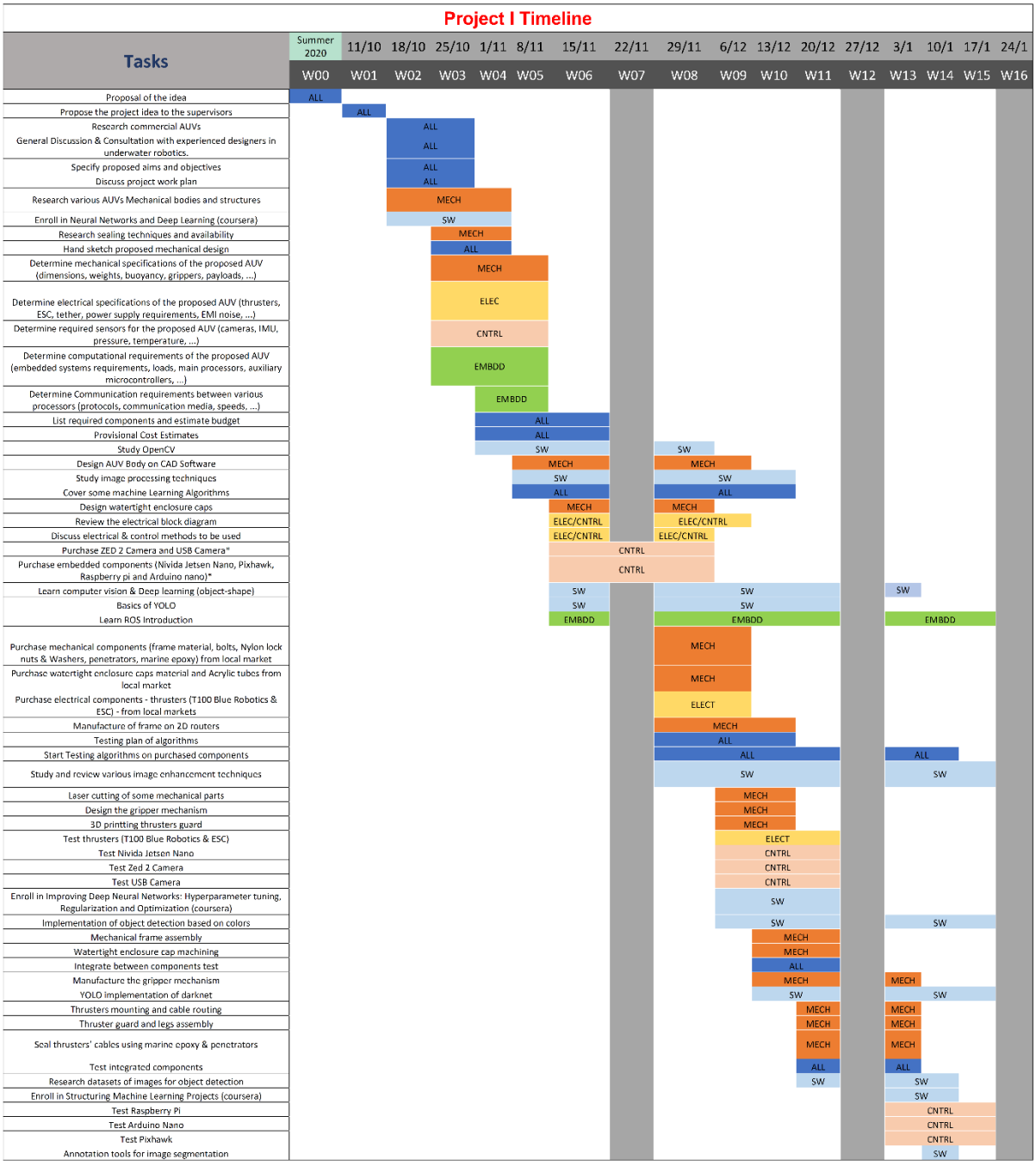
[24] Prashant Iyengar, others, “System Design and Implementation of Matsya 2.0, a Technology Demonstrating Autonomous Underwater Vehicle”, AUV-IITB, Indian Institute of Technology, Bombay, India, 2013.
[25] Vachhani, L., Hemendra, A., Kartik, V., others,” Research and Development of Matsya 3.0, Autonomous Underwater Vehicle”, AUV-IITB, Indian Institute of Technology, Bombay, India, 2014
[26] Vachhani, L., Hemendra, A., Kartik, V., others,” Research and Development of Matsya 4.0, Autonomous Underwater Vehicle”, AUV-IITB, Indian Institute of Technology, Bombay, India, 2015
[27] Vachhani, L., others,” Technical Design Report of Matsya 5A, Autonomous Underwater Vehicle”, Technical Design Report of Matsya 5A, AUV-IITB, Indian Institute of Technology, Bombay, India, 2017.
[28] Vachhani, L., Hemendra, A., others, “Technical Design Report of Matsya 6, Autonomous Underwater Vehicle”, Technical Design Report of Matsya 6, AUV-IITB, Indian Institute of Technology, Bombay, India, 2019.
[29] Qi, Y., Yang, Z., Sun, W. et al. A Comprehensive Overview of Image Enhancement Techniques. Arch Computat Methods Eng (2021).
[30] R. Garg, B. Mittal and S. Garg, “Histogram Equalization Techniques for Image Enhancement,”International Jour-nal of Electronics and Communication Technology, (2011).
[31] Urvashi Manikpuri, Yojana Yadav, Image Enhancement Through Logarithmic Transformation, (2014)
[32] Xu, Z.; Liu, X.; Ji, N. Fog Removal from Color Images using Contrast Limited Adaptive Histogram Equalization. International Congress on Image and Signal Processing, (2009).
[33] Shanto Rahman, Md Mostafijur Rahman, M. Abdullah-Al-Wadud, Golam Dastegir Al-Quaderi and Mohammad Shoyaib,”An adaptive gamma correction for image enhancement”,(2016)
[34] Kashif Iqbal,R. Salam ,Azam Osman ,”Underwater Image Enhancement Using an Integrated Colour Model”,(2007).
[35] Dongmei Huang, Yan Wang, Wei Song, Jean Sequeira, Sébastien Mavromatis,”Shallow-water Image Enhancement Using Relative Global Histogram Stretching Based on Adaptive Parameter Acquisition”,(2018).
[36] D. G. Lowe, "Object recognition from local scale-invariant features," Proceedings of the Seventh IEEE International Conference on Computer Vision, 1999
[37] Ramík, D.M., Sabourin, C., Moreno, R. et al. A machine learning based intelligent vision system for autonomous object detection and recognition. Appl Intell 40, 358–375 (2014).
[38] J. Redmon, S. Divvala, R. Girshick and A. Farhadi, "You Only Look Once: Unified, Real-Time Object Detection," 2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016
[39] https://www.nakasha.co.jp/future/ai/vol13_darknet_yolov4_with_opencv4.html
[40] L. Joseph. Mastering ROS for Robotics Programming. UK: Packt Publishing Ltd., 2015.
[41] https://opencv.org/
[42] https://filmora.wondershare.com/virtual-reality/how-does-vr-work.html
[43] https://www.gartner.com/en/information-technology/glossary/head-mounted-displays-hmd
[44] https://www.techopedia.com/definition/2342/head-mounted-display-hmd
[45] https://www.frontiersin.org/articles/10.3389/fmars.2019.00644/full
[46] https://www.smlease.com/entries/technology/what-is-virtual-reality/

[47] https://vrty.io/2020/09/15/vr-features/
[48] https://litslink.com/blog/what-is-augmented-reality-and-how-does-it-work
[49] https://www.techslang.com/how-does-augmented-reality-work/
[50] https://www.constructdigital.com/insight/how-does-augmented-reality-ar-work
[51] https://www.bitdegree.org/tutorials/what-is-augmented-reality/#how-does-augmented-reality-work
[52] https://www.immersiv.io/blog/what-is-augmented-reality-definition/
[53] https://www.softwaretestinghelp.com/ar-vs-vr-comparison/
[54] Zuiderveld, K. Contrast Limited Adaptive Histogram Equalization. Academic Press Inc.,(1994).
[55] El Naqa I., Murphy M.J. (2015) What Is Machine Learning?. In: El Naqa I., Li R., Murphy M. (eds) Machine Learning in Radiation Oncology. Springer, Cham.
[56] H. Wang, C. Ma and L. Zhou, "A Brief Review of Machine Learning and Its Application," 2009 International Conference on Information Engineering and Computer Science, 2009
[57] A. Bansal and A. Singhrova, "Performance Analysis of Supervised Machine Learning Algorithms for Diabetes and Breast Cancer Dataset," 2021 International Conference on Artificial Intelligence and Smart Systems (ICAIS), 2021
[58] Bote-Curiel, Luis, Sergio Muñoz-Romero, Alicia Gerrero-Curieses, and José L. Rojo-Álvarez 2019. "Deep Learning and Big Data in Healthcare: A Double Review for Critical Beginners" Applied Sciences 9
[59] X. Du, Y. Cai, S. Wang and L. Zhang, "Overview of deep learning," 2016 31st Youth Academic Annual Conference of Chinese Association of Automation (YAC), 2016
[60] Gu, Yingxin, Bruce K. Wylie, Stephen P. Boyte, Joshua Picotte, Daniel M. Howard, Kelcy Smith, and Kurtis J. Nelson 2016. "An Optimal Sample Data Usage Strategy to Minimize Overfitting and Underfitting Effects in Regression Tree Models Based on Remotely-Sensed Data" Remote Sensing 8
[61] V. L. Berardi and G. P. Zhang, "An empirical investigation of bias and variance in time series forecasting: modeling considerations and error evaluation," in IEEE Transactions on Neural Networks, vol. 14, no. 3, pp. 668-679, May 2003
[62] Alzubaidi, L., Zhang, J., Humaidi, A.J. et al.” Review of deep learning: concepts, CNN architectures, challenges, applications, future directions”. J Big Data 8, 53 (2021).
[63] Joseph Redmo , Santosh Divvala, Ross Girshick, Ali Farhadi,”You Only Look Once: Unified, Real-Time Object Detection”,2016.
[64] Alexey Bochkovskiy,Chien-Yao Wang,Hong-Yuan Mark Liao ,”YOLOv4: Optimal Speed and Accuracy of Object Detection” 2020.
[65] Chien-Yao Wang, Hong-Yuan Mark Liao, Yueh-Hua Wu, Ping-Yang Chen, Jun-Wei Hsieh, and I-Hai Yeh. CSPNet: A new backbone that can enhance the learning capability of cnn. Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshop (CVPR Workshop), 2020.
[66] Shu Liu, Lu Qi, Haifang Qin, Jianping Shi, and Jiaya Jia. Path aggregation network for instance segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2018.
[67] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Spatial pyramid pooling in deep convolutional networks for visual recognition. IEEE Transactions on Pattern Analysis and Machine Intelligence (TPAMI), 2015.

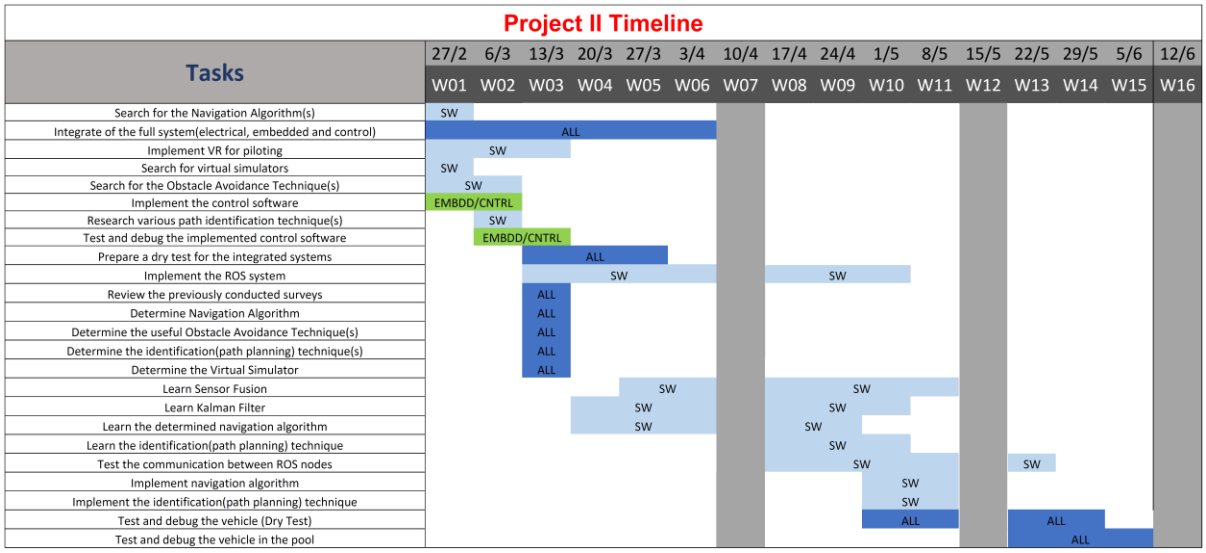
[68] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollar. Focal loss for dense object detection. In ´ Proceedings of the IEEE International Conference on Computer Vision (ICCV), 2017.
[69] Rikiya Yamashita, Mizuho Nishio , Richard Kinh Gian Do , Kaori Togashi,"Convolutional neural networks: an overview and application in radiology",2018.
[70] Sakshi Indolia, Anil Kumar GoswamiS, P. MishraPooja Asopa,"Conceptual Understanding of Convolutional Neural Network- A Deep Learning Approach",2018.
[71] Chigozie Enyinna Nwankpa, Winifred Ijomah, Anthony Gachagan, and Stephen Marshall, "Activation Functions: Comparison of Trends in Practice and Research for Deep Learning" 2018.
[72] V. Nair and G. E. Hinton, "Rectified linear units improve restricted boltzmann machines," Haifa, 2010.
[73] P. Ramachandran, B. Zoph, and Q. V. Le, "Searching for Activation Functions" 2017.
[74] I. Goodfellow, Y. Bengio, and A. Courville, "Deep learning." MIT Press, 2016.
[75] Diganta Misra,"Mish: A Self Regularized Non-Monotonic Neural Activation Function", 2020.
[76] C. Dugas, Y. Bengio, F. Bélisle, C. Nadeau, and R. Garcia, "Incorporating Second-Order Functional Knowledge for Better Option Pricing," in Advances in Neural Information Processing Systems (NIPS), 2001.
[77] Kaiming He Xiangyu Zhang Shaoqing Ren Jian Sun,"Deep Residual Learning for Image Recognition", 2015.
[78] Y. LeCun, B. Boser, J. S. Denker, D. Henderson, R. E. Howard, W. Hubbard, and L. D. Jackel. Backpropagation applied to handwritten zip code recognition. Neural computation, 1989.
[79] Gao Huang,Zhuang Liu,Laurens van der Maaten,"Densely Connected Convolutional Networks",2018.
[80] Bo Daji Ergu, Ying Cai, Bo Ma "A Method for Wheat Head Detection Based on Yolov4",2020.
[81] Tsung-Yi Lin, Piotr Dollar, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. Feature pyramid networks for object detection. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2017
[82] Tong Yang, Xiangyu Zhang, Zeming Li, Wenqiang Zhang, Jian Sun, "MetaAnchor: Learning to Detect Objects with Customized Anchors",2018.
[83] L. Joseph. Robot Operating System for Absolute Beginners: Robotics Programming Made Easy. US: Springer Science+Business Media, 2018.
[84] L. Joseph. Mastering ROS for Robotics Programming. UK: Packt Publishing Ltd., 2015.
[85] E. Fernández, L. S. Crespo, A. Mahtani, and A. Martinez. Learning ROS for Robotics Programming, 2nd edition. UK: Packt Publishing Ltd., 2015.
[86] https://www.devteam.space/blog/10-great-tools-for-vr-development/
[87] https://www.lullabot.com/articles/11-tools-for-vr-developers
[88] https://www.techleer.com/articles/617-best-virtual-reality-sdks-to-build-vr-apps/
[89] https://thinkmobiles.com/blog/best-vr-sdk/
[90] https://www.infoq.com/articles/augmented-reality-best-skds/
[91] https://dzone.com/articles/12-best-augmented-reality-sdks
[92] https://docs.unity3d.com/Packages/com.unity.xr.arcore@2.1/manual/index.html
[93] https://www.offshore-mag.com/subsea/article/16759458/troll-tunnel-inspection-rov-piloted-in-virtual-reality-mode

[94] https://spectrum.ieee.org/video/robotics/robotics-software/controlling-robot-swarms-with-augmented-reality
[95] https://www.stereolabs.com/blog/use-your-zed-camera-with-ros

Appendix A: Project I Timeline



Appendix B: Project II Timeline



Appendix C: Cost

ARAB ACADEMY FOR SCIENCE, TECHNOLOGY, AND MARITIME TRANSPORT

COLLEGE OF COMPUTING AND INFORMATION TECHNOLOGY

Academic Year: 2020/2021 Semester: 8th semester of July

Senior Project Summary Report

Project Title	**Autonomous Underwater Vehicle (AUV) for Underwater Exploration**	
Supervisor(s)	**Dr. Yasmine Nagy and Dr. Mohamed El-Habrouk**	
Team members:	**Names** **1.** Ahmed Samy Zaghloul **2.** Haneen Salah El-Din Warda **3.** Mariam Ehab Fayek **4.** Maya Saad Badr **5.** Youssef Ahmed Mohana	**Registration Numbers** **1.** 17100794 **2.** 17100492 **3.** 17100106 **4.** 17100473 **5.** 17101722
Project Deliverables	The project aims to assemble an autonomous underwater vehicle (AUV) with no need for manual piloting and with an efficient software using robot operating system (ROS) to assist in specific missions in underwater exploration, such as in underwater archeology and oceanography.	
Team Organization	**1. Ahmed Samy Zaghloul:** Software, Mechanical, Embedded **2. Haneen Salah El-Din Warda:** Software, Control, Embedded **3. Mariam Ehab Fayek:** Software, Mechanical, Electrical **4. Maya Saad Badr:** Software, Control, Electrical **5. Youssef Ahmed Mohana:** Software, Control, Embedded	
Ethical Considerations	----------	
Social Impact	The AUV will help in thriving tourism through search and salvage operations for archelogy and oceanography, and thus thrive with Egypt's economy that has been severely affected by COVID-19 pandemic.	

Supervisor Name	Signature
Dr. Yasmine Nagy	
Dr. Mohamed El-Habrouk	