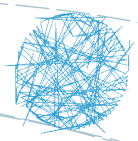
DM



LARSyS
Laboratory of Robotics
and Engineering Systems

Marine Biodiversity Assessments Using Aquatic Internet of Things

MASTER DISSERTATION

Carlos Diogo Félix Martins
MASTER IN INFORMATICS ENGINEERING

UNIVERSIDADE da MADEIRA
A Nossa Universidade
www.uma.pt

December | 2020

Marine Biodiversity Assessments Using Aquatic Internet of Things

MASTER DISSERTATION

Carlos Diogo Félix Martins
MASTER IN INFORMATICS ENGINEERING

ORIENTATION
Marko Radeta

CO-ORIENTATION
Filipe Magno de Gouveia Quintal

Marine Biodiversity Assessments using Aquatic Internet of Things

Dissertação de Mestrado em Engenharia Informática sob orientação do Professor Doutor Auxiliar Convidado Marko Radeta e co-orientação do Professor Doutor Filipe Magno de Gouveia Quintal, apresentada a prova pública, na Universidade da Madeira, a 7 de abril de 2021, perante seguinte júri:

Professor Doutor Auxiliar Convidado Marko Radeta (Vogal);

Professora Doutora Frederica Margarida Camacho Gonçalves, Universidade da Madeira (Arguente);

Professor Doutor Eduardo Miguel Dias Marques, Universidade da Madeira (Vogal e Presidente de Júri).

Abstract

While Ubiquitous Computing remains vastly applied in urban environments, it is still scarce in oceanic environments. Current equipment used for biodiversity assessments remain at a high cost, being still inaccessible to wider audiences. More accessible IoT (Internet of Things) solutions need to be implemented to tackle these issues and provide alternatives to facilitate data collection in-the-wild. While the ocean remains a very harsh environment to apply such devices, it is still providing the opportunity to further explore the biodiversity, being that current marine taxa is estimated to be 200k, while this number can be actually in millions.

The main goal of this thesis is to provide an apparatus and architecture for aerial marine biodiversity assessments, based on low-cost MCUs (Microcontroller unit) and microcomputers. In addition, the apparatus will provide a proof of concept for collecting and classifying the collected media. The thesis will also explore and contribute to the latest IoT and machine learning techniques (e.g. Python, TensorFlow) when applied to ocean settings. The final product of the thesis will enhance the state of the art in technologies applied to marine biology assessments.

Keywords: Ubiquitous Computing · Aerial Assessments · Wildlife Monitoring · Machine Learning · Internet of Things

Resumo

A computação ubíqua é imensamente utilizada em ambientes urbanos, mas ainda é escassa em ambientes oceânicos. Os equipamentos atuais utilizados para o estudo de biodiversidade são de custo alto, e geralmente inacessíveis para o público geral. Uma solução IoT mais acessível necessita de ser implementada para combater estes problemas e fornecer alternativas para facilitar a recolha de dados na natureza. Embora o oceano seja um ambiente severo para aplicar estes dispositivos, este fornece mais oportunidades para explorar a biodiversidade, sendo que a taxa de marinha atual é estimada ser 200 mil, mas este número pode estar nos milhões.

O objetivo principal desta tese é fornecer um aparelho e uma arquitetura para estudos aéreos de biodiversidade marinha, baseado em microcontroladores low-cost e microcomputadores. Adicionalmente, este aparelho irá fornecer uma prova de conceito para coletar e classificar a mídia coletada. A tese irá também explorar e contribuir para as técnicas mais recentes de machine learning (e.g. Python, TensorFlow) quando aplicadas num cenário de oceano. O produto final desta tese vai elevar o estado da arte em tecnologias aplicadas a estudos de biologia marinha.

Keywords: Ubiquitous Computing · Aerial Assessments · Wildlife Monitoring · Machine Learning · Internet of Things

Acknowledgements

First of all, I would like to thank my thesis supervisor Prof. Marko Radeta (Wave Lab Coordinator) and a researcher at ITI/LARSyS. Anytime I ran into problems or had any questions his door was always open. I would also like to thank my thesis co-supervisor Prof. Filipe Quintal which was always willing to help in any way he could.

I would also like to thank the experts who were involved in this research project. Without their passionate participation and input, this project could not have been successfully conducted.

I thank my friends who shared this five year journey with me and always made it a lot more enjoyable than going through it alone.

Last but not least, I must express my very profound gratitude to my parents for always supporting and encouraging me throughout my years of study, and through the process of researching and writing this thesis. This accomplishment would not have been possible without them. Thank you.

This study is part of LARGESCALE project with grant no. PTDC/CCI-CIF/32474/2017 by Fundação para a Ciência e a Tecnologia (FCT) and Portuguese National Funds (PIDDAC). It is also supported by the funding by project UID/EEA/50009/2019 and UIDB/50009/2020 (ARDITI/LARSyS) as well as by Programa de Cooperación INTERREG V-A España-Portugal MAC (Madeira-Azores-Canarias) 2014-2020, throughout project INTERTAGUA (Interfaces Aquáticas Interativas para Deteção e Visualização da Megafauna Marinha Atlântica e Embarcacoes na Macaronésia usando Marcadores Rádio-transmissores), with Ref: MAC2/1.1.a /385. Moreover, conducted studies were supported by the additional funding from project MITIExcell - EXCELENCIA INTERNACIONAL DE IDT&I NAS TIC (Project Number M1420-01-01450FEDER0000002), and project SUBMARINE (Simulators for Understanding the Biodiversity using Mobile and Augmented Reality as Interactive Nautical Explorations), provided by the Regional Government of Madeira. Also, it was supported by the project INTERWHALE - Advancing Interactive Technology for Responsible Whale-Watching, with grant no. PTDC/CCI-COM/0450/2020 by Fundação para a Ciência e a Tecnologia (FCT). Finally, project is supported by the Wave Lab[1] - interactive technologies for depicting the anthropogenic impact on marine biosphere.

Carlos Diogo Félix Martins

[1] `http://wave.arditi.pt`

Table of Contents

List of Figures ....

List of Tables ....

1 Introduction 1

1.1 Motivation 2

1.1.1 Understanding Aquatic Biodiversity 2

1.1.2 Overview of Issues to be Explored 3

1.2 Research Questions and Contributions 4

1.3 Objectives 4

1.4 Organization of the Manuscript 4

2 Related Work 6

2.1 State of the Art 6

2.1.1 Tools for Aerial Assessments 6

2.1.2 Overview of Machine Learning Software 11

2.1.3 Types of Imagery 13

2.1.3.1 Passive Imagery 14

2.1.3.2 Active Imagery 16

2.1.4 Multispectral Imagery Cameras 17

2.2 Prior Research 19

2.2.1 Aerial Image Observations 20

2.2.2 IoT devices in Marine Environments 21

2.2.3 Detecting Aquatic Species using Machine Learning 22

2.3 Overview of Multi-Spectral Imagery 24

2.4 Summary of the Related Work 26

3 Methodology 27

3.1 User Pre-Study 27

3.2 Machine learning 28

3.2.1 Data gathering 28

3.2.1.1 Data collection 29

3.2.1.2 Data annotation 30

3.2.1.3 Data Augmentation 31

3.2.2 Model training 32

3.2.2.1 Tools 33

3.2.2.2 Training hardware . . . 33
3.2.2.3 Model Quantization . . . 35
3.2.2.4 Training protocol . . . 36
3.2.3 TensorFlow Lite conversion . . . 37
3.2.4 Trained model performance on microcontrollers . . . 38
3.3 Summary of the Methodology . . . 39
4 System . . . 39
4.1 Hardware . . . 39
4.2 Software . . . 41
4.2.1 Flowchart/Algorithm . . . 41
4.2.2 System Architecture . . . 42
4.2.3 Database . . . 43
4.2.3.1 API . . . 46
4.2.4 Scripts . . . 48
4.2.4.1 Real time object detection . . . 48
4.2.4.2 Video streaming . . . 51
4.2.4.3 Supervisor . . . 52
4.2.4.4 WiFi Access Point . . . 53
4.2.5 Mobile App . . . 54
4.2.5.1 Detections . . . 55
4.2.5.2 Video Feed . . . 57
4.2.5.3 Statistics . . . 58
4.3 Summary of the System . . . 58
5 Experiments . . . 60
5.1 Sample . . . 60
5.2 Study Protocol . . . 60
5.2.1 User validation . . . 60
5.2.2 Human versus Machine . . . 62
5.3 Summary of the Experiments . . . 63
6 Results . . . 64
6.1 Human versus Machine . . . 64
6.2 User Validation . . . 64
6.3 Summary of the Results . . . 71
7 Discussion . . . 72
7.1 Finding Analysis . . . 72
7.2 Limitations . . . 73

7.3 Future work 74
8 Conclusions 75

References 77

List of Figures

1 SPACEWHALE training image down-sampling example. [Source: SPACEWHALE - MAPPING WHALES FROM SPACE] 7

2 Footage gathered from an HiDef aircraft. [Source: HiDef Aerial Surveying sample video images] 8

3 Public Lab's Balloon Mapping Kit. [Source: Public Lab Store] 8

4 False color image captured by a DJI P4 drone that highlights the vegetation making it easier to see the weaker areas. [Source: DJI P4 MULTISPECTRAL AGRICULTURE DRONE] 10

5 Example of a false color image that highlights water as light blue. [Source: Assessing Storm Impact in Puerto Rico with Remote Sensing and Digital Volunteers] 10

6 Image captured through a panchromatic band. [Source: From panchromatic to hyperspectral EARTH OBSERVATION IN A MYRIAD OF COLORS] 14

7 Comparison between natural color (left) MSI and a highlighted vegetation false-color (right). [Source: False color] 15

8 Example of pan-sharpened image. [Source: Spear Pan Sharpening] 15

9 Aerial-trained deep learning pipeline's automated workflow. [Source: Aerial-trained deep learning networks for surveying cetaceans from satellite imagery] 23

10 Multispectral images showing four different bands (in order blue 488nm, blue-green 532nm, green 550nm, red 600nm). As we can see, the second and third bands show a few more animals than the other two. [Source: Spectral detection and monitoring of marine mammals] 24

11 Sperm whale detection by the automated MSI system. [Source: Automated Detection of Marine Animals Using Multispectral Imaging] 25

12 Long range whale imagery [Source: Multispectral Observations of Marine Mammals.] . . 25

13 Whale imagery captured at night using an IR sensor [Source: Multispectral Observations of Marine Mammals] 26

14 Custom dataset stored and managed in supervise.ly. 29

15 Image annotated using supervise.ly 30

16 Image augmentation by flipping it horizontally. 31

17 Image augmentation using color transformation. 31

18 Image augmentation using noise. 32

19 Image augmentation using blur. 32

20 Bandwidth comparison between CPUs and GPUs [Source: Do we really need GPU for Deep Learning? - CPU vs GPU] . . . 34
21 Jupyter Notebook hosted in Google Colaboratory which provides easy access to high performance hardware. . . . 36
22 TensorFlow Lite work flow. . . . 37
23 Coral Dev Board with a Coral Camera connected. . . . 40
24 GPS receiver GP-735 used to collect location data. . . . 40
25 Real time object detection performed by the apparatus using COCO SSD MobileNet and an infrared camera module. . . . 41
26 System Flowchart, apparatus's behaviour on the left and mobile app's behaviour on the right. . . . 42
27 Apparatus system architecture. . . . 42
28 Mobile app system architecture. . . . 43
29 Database schema. . . . 44
30 Coordinates of the bounding box. . . . 45
31 Original image on the left and annotated on the right . . . 50
32 Real time object detection streaming to a browser. . . . 52
33 RaspAP dashboard's homepage displaying information about the hourly traffic and connected devices. . . . 54
34 Detections page trying to fetch data on the left and failed connection on the right. . . . 55
35 Detections page loading some images on the left and fully loaded on the right. . . . 56
36 Detection Feedback page allowing the user to input their feedback and zoom in/out feature shown on the right. . . . 57
37 Video Feed page displaying the apparatus' current video feed performing object detection. . . . 57
38 Page displaying statistics about the detections stored on the apparatus on the left and no data in a certain time frame on the right. . . . 58
39 User holding imagery of the marine mega fauna in front of the apparatus. . . . 61
40 Footage gathered from a sea vessel displaying a dolphin swimming automatically annotated. . . . 63
41 Human versus Machine experiment: Detection time comparison of the marine mega fauna in seconds. . . . 64
42 Age distribution of the participants. . . . 65
43 Gender distribution of the participants. . . . 65
44 Technological proficiency of the participants. . . . 66
45 System usability scale question 1 results. . . . 66

46 System usability scale question 2 results. 67
47 System usability scale question 3 results. 67
48 System usability scale question 4 results. 68
49 System usability scale question 5 results. 68
50 System usability scale question 6 results. 69
51 System usability scale question 7 results. 69
52 System usability scale question 8 results. 70
53 System usability scale question 9 results. 70
54 System usability scale question 10 results. 71

List of Tables

1 Comparison of Technologies and Techniques for Aerial Assessments 6
2 Comparison of Machine Learning software 12
3 Comparison between different Multispectral Imagery cameras. 17
4 Overview of studied projects. 19
5 Comparison between automatic and manual whale detection. (Adapted from Whales from Space: Counting Southern Right Whales by Satellite) 20
6 Number of collected images per each taxa. 29
7 Differences between CPUs and GPUs. 33
8 Model training time using different hardware configurations. 34
9 Types of quantization available in TensorFlow Lite. 36
10 Latency and accuracy results for post-training quantization and quantization-aware training. 36
11 Inference performance results from Jetson Nano, Raspberry Pi 3, Intel Neural Compute Stick 2, and Google Edge TPU Coral Dev Board. 38
12 Inference time (in milliseconds) benchmark between different model architectures and devices. (Source: Edge TPU performance benchmarks) 51
13 Sources of the collected test footage and their duration. 62

Nomenclature

AI Artificial Inteligence

ANN Artificial Neural Network

API Application programming interface

CDB Coral Dev Board

CNN Convolutional neural network

$CO2$ Carbon dioxide

$COCO$ Common objects in context

CPU Central processing unit

CSS Cascading Style Sheets

DTL Data Transformation Language

EO Earth Observation

$GNSS$ Global Navigation Satellite System

GPS Global Positioning System

GPU Graphics processing unit

HD High-Definition

$HDAS$ HiDef Aerial Surveying

$HTTP$ Hypertext Transfer Protocol

IMF International Monetary Fund

IR Infrared

JS JavaScript

$JSON$ JavaScript Object Notation

LEO Low Earth Orbit

$LIDAR$ Light Detection And Ranging

$LoRa$ Long-Range

$LSTM$ Long short-term memory

$LWIR$ Long wave Infrared

ML Machine Learning

MSI Multi-spectral imagery

$NDRE$ Normalized Difference Red Edge

$NDVI$ Normalized Difference Vegetation Index

OID Open Images Dataset

PAM Passive Acoustic Monitoring

$RCNN$ Region Based Convolutional Neural Networks

$REST$ Representational state transfer

RF Random Forest

SAR Synthetic Aperture Radar

SoA State of the Art

SSD Single Shot MultiBox Detection

SUS System Usability Scale

SVM Support Vector Machines

TPU Tensor Processing Unit

UAV Unmanned Aerial Vehicle

UID Unique identifier

$UNSDG$ United Nation Sustainable Development Goal

URL Uniform Resource Locator

UV Ultraviolet

VHR Very High Resolution

$WSGI$ Web Server Gateway Interface

1 Introduction

Our oceans, coasts, and estuaries are home to diverse living beings. These organisms take many forms, from the tiniest single-celled plankton to the largest animal on Earth, the blue whale[2] [1]. Understanding the life cycles, habits, habitats, and inter-relationships of marine life contributes to the reduction of CO_2 as well as to our understanding of the planet as a whole, which was recently confirmed not just by scholars [2], however also by IMF (International Monetary Fund)[3]. Us humans are constantly being influenced and rely on these species, and there is a need for their better understanding and care. The changing climate caused by humans is a major risk to all marine life and can be the final factor to cause extinction on species already at risk due to overfishing [3].

Taking into consideration these concerns, the main goal of this thesis is to gain a better understanding of surface aquatic life, focusing on local and migratory marine megafauna species. These species are known to the Madeira archipelago, which is a historically touristic island destination located in the North-East Atlantic, known for significant occurrence of marine life [4,5]. This thesis studies to which extent it is possible to detect the whales, dolphins, turtles, sea lions, and sea birds using IoT. Although these species make only a small sample size of the total species population that occupies the oceans, nevertheless, they remain connected through diverse food chains or migrations, and this thesis addresses the software and hardware necessities which can provide more insights into the evolution of marine biodiversity.

Focusing on these species, this thesis will produce a low-cost aerial imagery apparatus, that can be attached to the mast of a sea vessel, during the visual surveys. This device will consist mainly of a MCU, using accessible IoT devices such as *Raspberry Pi*,[4], a *NVIDIA's Jetson Nano*[5] or a *Coral Dev Board*[6], and a camera, pointed at the sea below it, to assess the marine species. One of the important benefits of this apparatus is to obtain a more robust aerial imagery of marine life (a top-down view). This allows collecting the marine-life top-down imagery, which usually remains

[2]The blue whale (Balaenoptera musculus) is a marine mammal that can grow up to 29.9 meters (98 ft) in length and with a maximum recorded weight of 173 tonnes (190 short tons), is the largest animal known to have ever existed. Blue whales were abundant in nearly all the oceans on Earth until the beginning of the twentieth century. For over a century, they were hunted almost to extinction by whaling until protected by the international community in 1966.

[3]Recent report of whale impact by IMF, accessed Dec 30, 2019, from `https://www.imf.org/external/pubs/ft/fandd/2019/12/natures-solution-to-climate-change-chami.htm`

[4]The Raspberry Pi is a low cost, credit-card sized microcomputer capable of supporting computer monitor or TV. It is a little device which enables people of all ages to explore computing, and to learn how to program in languages like Scratch and Python [6]

[5]NVIDIA Jetson Nano Developer Kit is a small, powerful computer allowing multiple neural networks in parallel for applications such as image classification, object detection, segmentation, and speech processing, using 5 watts [7]

[6]The Dev Board is a single-board computer that targets performing fast machine learning inferencing in a small form factor, it uses an onboard Edge Tensor processing unit to achieve this performance [8].

inaccessible (from on-board the sea vessel). Moreover, such a system is designed to support marine biologists to easily determine the abundance of species, using image vision algorithms for detection and classification.

1.1 Motivation

This manuscript is motivated to increase marine literacy and impact the reduction of marine litter given the targets by the UNSDG (United Nation Sustainable Development Goal) 14. Life Below Water. From hereinafter, the manuscript describes some of the facts and figures regarding biodiversity in oceanic settings, as well as an outline of the important issues which remain unsolved when performing the biodiversity assessments.

1.1.1 Understanding Aquatic Biodiversity

The oceans cover 71% of the Earth's surface, they contain 97% of the Earth's water and represent 99% of the living space on the planet. They absorb around 30% of the carbon dioxide produced by humans, which buffers the impact of global warming. They contain nearly 200 000 identified species, however, the actual numbers are unknown and may be in the millions[7]. More than 3 billion depend on marine and coastal biodiversity for their livelihood. Less known is the fact that the oceans are the world's largest source of protein [9]. Moreover, the global market value of marine and coastal resources and industries is estimated at $3 trillion per year, which is equivalent to about 5% of the global GDP. Over 200 million people are directly or indirectly employed by marine fisheries [9].

Even though some existing policies and treaties encourage responsible use of ocean resources and protected areas for marine biodiversity, they are not enough to combat the negative effects of overfishing, the acidification of the ocean, and the worsening coastal eutrophication[8]. The ocean acidification is due to the ocean absorbing the atmospheric CO2 (Carbon dioxide) which changes the chemical composition of the water. Over the past 30 years observing the ocean acidity we can see an average increase of 26% since the pre-industrial times, and at this rate, by the end of the century, it is predicted to increase between 100 to 150% [10].

Our oceans are polluted with a wide variety of marine litter ranging from soda cans and plastic bags to derelict fishing gear and abandoned vessels. Everyday trash is entering the oceans at an alarming rate, it is estimated that by 2025 more than 250 trillion tons of plastic will make its way into the sea. Marine litter beyond being unsightly, it's dangerous to marine life, hazardous

[7] Assessed at 30 Dec 2019 - recent UNSDG Life Below Water facts and figures at `https://www.un.org/sustainabledevelopment/oceans/`

[8] Eutrophication is when a body of water becomes overly enriched with minerals and nutrients which induce excessive growth of algae. This process may result in oxygen depletion of the water body

to human health. Marine animals can mistake marine litter for food, which is often fatal, or become entangled in it. Furthermore, divers, swimmers, and beachgoers can be directly harmed by encounters with marine litter or its toxins. The environmental damage caused by plastic litter alone is estimated at $13 billion a year [11].

1.1.2 Overview of Issues to be Explored

Given the aforementioned importance of issues in aquatic settings, this manuscript leverages the potential of aerial imagery, capable to assess marine biodiversity. Thus, it is important to outline the three main issues when obtaining the biodiversity imagery from sea-vessels:

(i) **Biodiversity Population Estimation is Based on Rule-of-Thumb**
As it was observed when attending one of the whale-watching trips, when counting the marine taxa from the sea vessel, marine biologists usually follow the rule of thumb technique to estimate the population. This typically means that two or three biologists (including a skipper) suggest a number from which an average is decided [12]. This technique allows to collect the data, however yielding a very significant standard deviation, where more tools should be implemented to obtain the fine resolution, producing a more correct estimation of the number of species.

(ii) **UAV (Unmanned Aerial Vehicle) Marine Assessments Continue to Be Obtrusive**
As it was also assessed from one of the whale watching trips, due to the high impact of the wind factor [13], (e.g. above Beaufort[9] 2), it is very challenging to operate a state of the art drone[10] in these conditions, as they tend to drift. Moreover, drones which are capable of landing on the water surface or which can withstand the waterproofing, remain at a very high cost[11].

(iii) **Low-Cost Aerial Survey Mechanisms Remain Unexplored in Ocean Setting**
As it will be addressed throughout this manuscript, typical aerial assessments are obtained using satellites imagery [14], which hinders access to the data or are costly to a typical marine biologist. Therefore, the manuscript explores the low-cost solutions based on IoT for assessing marine biodiversity, as well as the pros and cons of using such equipment on typical sensing marine units (e.g. sea-vessel mast, UAV, etc), as some of the recent work suggests it as an alternative approach to the obtained aerial imagery.

[9]The Beaufort scale is an empirical measure that relates wind speed to observed conditions at sea or on land, it ranges from 0 (calm) to 12 (Hurricane force).

[10]Tested with `https://www.dji.com/pt/phantom-2`

[11]At the date of the manuscript, cost of the typical aerial drone is approximately EUR 1k. While claimed to be waterproof, it is to some extent water safe. See e.g. HEX2O kit at `https://www.quadh2o.com/hexh2o/hexh2o-kit/`. Retrieved at 30 Dec 2019

1.2 Research Questions and Contributions

To address the previously outlined issues, this manuscript studies and provides empirical results for assessing the low-cost marine biodiversity assessments, using aerial imagery. Studies reported in this thesis support the next four research questions:

(i) **[RQ1.] How to create low-cost aquatic IoT apparatus for assessing marine biodiversity using aerial imagery?**
In this research question, this thesis will depict the best practices for creating the IoT device capable of collecting, classifying, and reporting the confidence rates when detecting the species from imagery gathered aboard sea vessels.

(ii) **[RQ2.] To which extent the proposed apparatus can support the marine biologist in counting the species?**
Several user studies will be performed, mainly on usability and user experience, outlining the pros and cons of using such a system in real-time, on-board the sea vessels.

(iii) **[RQ3.] How accurate is the software for classification prediction and counting the marine taxa?**
This particular question will analyze, to which extent the automatic classification performs compared to that of humans. More insights will be provided on understanding who performs better in the classification of imagery, being a human, a computer, or combined (human and computer).

1.3 Objectives

Practical goals of this manuscript are therefore directly tailored to the aforementioned research questions and challenges and include next objectives:

- **[O1.]** Prototyping IoT apparatus for top-view surveys
- **[O2.]** IoT Software Development of the device
- **[O3.]** Mobile Application Software Development
- **[O4.]** Image vision classifiers to detect the marine mega fauna
- **[O5.]** Usability and HCI studies from deployment

1.4 Organization of the Manuscript

This manuscript is organized as follows: (i) **Introduction** section outlined the research questions, problems and objectives of the thesis; (ii) **Related work** section portrays the analysis of current

SoA (State of the Art) of all IoT devices available on market as well as the current techniques for image vision biodiversity assessments in ecology; (iii) **Methodology** section describes the used apparatus, comprised from system architecture and used algorithms for image vision classification; (iv) **Results** section provides insights to the collected data as well as to the feedback from the marine biologists; (v) **Discussion** will thoroughly describe collected results and provide novel insights from the assessed imagery; and finally (vi) **Conclusion** will outline the constraints, contributions, and future works of developing both hardware and software for performing biodiversity assessments in oceanic environment.

2 Related Work

Before providing the solution to assess the aerial imagery, the thesis reports the technology and techniques (**Section 2.1**) as well as an overview of the previous works reported by the scholars when deploying and testing IoT devices in aquatic setting (**Section 2.2**).

2.1 State of the Art

As aforementioned, current technologies to assess the marine megafauna exist, however remaining at the significant high cost. Manuscript provides an overview of the SoA: (i) **Section 2.1.1** depicts technology and techniques currently applied to the aquatic wildlife monitoring. It also includes an overview of best practices in assessing aerial imagery; (ii) **Section 2.1.2** benchmarks the prior machine learning techniques in detecting marine taxa; (iii) **Section 2.1.3** reports the types of imagery which can be collected; (iv) **Section 2.1.4** outlines the diverse MSI (Multi-spectral imagery) cameras accessible on market.

2.1.1 Tools for Aerial Assessments

To better understand how to design for biodiversity assessments using aerial imagery, an overview of some technologies and techniques for aerial assessments is depicted in the following table 1. The provided list of works is also outlined in the remainder of the section.

Table 1. Comparison of Technologies and Techniques for Aerial Assessments

Producer	Cost	Type	Data Type	Software	Data Retrieval
SPACEWHALE [15]	*	Satellite	Images	Data report	HTTP
HDAS [16]	*	UAV	Video	Data report	*
Balloon Mapping kit [17]	$ 100	Balloon	Images/ Video	MapKnitter	SD card
DJI P4 [18]	$ 6499	Drone	MSI	-	SD card
RedEdge [19] + DJI M200 [20]	$ 6900 + £ 3200	Drone	MSI	-	SD card
AEROKATS [21]	$ 165	Kite	Images/ Video	AEROKATS Mission Mapper	SD card
HEXH2O^{TM} [22]	$ 1049	Drone	Images/ Video	-	SD card

* Information is unavailable.

Among the plethora of technologies to study biodiversity, SPACEWHALE is a service that provides a tool for automatic detection and identification of large whale species using VHR (Very High Resolution) satellite imagery[12], providing the information to the end-user through a report. It

[12]e.g. WorldView 3, a commercial EO (Earth Observation) satellite

uses an algorithm, based on deep-learning techniques using CNN (Convolutional neural network), that is trained using a large dataset of high-resolution aerial imagery gathered from aircraft surveys. This footage gathered from aircraft surveys has a resolution of 2cm which is down-sampled to 31cm so that it can match the WorldView 3 satellite imagery, as can be seen in the following figure 1. Once a whale is identified, that image is used to re-train the algorithm, improving the accuracy throughout time [15].

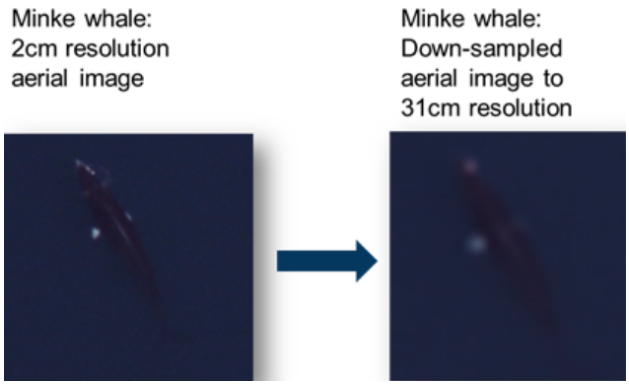


Fig. 1. SPACEWHALE training image down-sampling example. [Source: SPACEWHALE - MAPPING WHALES FROM SPACE]

While this system provided a proof of concept for applying the space technology in detecting the aquatic species, this manuscript argues that an apparatus of such kind has several issues: (i) it provides solely the back-end side (sole SQL database as for the report), where offline analysis need to be performed; (ii) data remains inaccessible, as there are no existing simplified pipelines to the satellite imagery; and (iii) collected imagery remains at a high cost.

HDAS (HiDef Aerial Surveying) is a UK based company which provides aerial surveying imagery to third parties. Their system is deployed around the globe on an exclusive territory basis, offering a HD (High-Definition) video platform deployed via an aircraft. On every contract, they gather video footage and data that is later analyzed to identify the species that are captured in the images. Such imagery can also assess the smaller taxa, such as sea birds, calculating the bird's flying heights. They also claim to have a unique technology that enables them to monitor birds' flight paths to avoid collisions. Most of their clients are energy companies that survey the environment around their offshore oil plants or offshore wind farms [16]. In the following figure 2 an example of the footage gathered by this company can be seen.

Fig. 2. Footage gathered from an HiDef aircraft. [Source: HiDef Aerial Surveying sample video images]

Although they produce very high-quality footage from their aerial surveys, these surveys can be very costly. Moreover, their footage analysis and specie identification are performed offline after the surveying mission is complete, tagged by the experts from the field, which takes a long time depending on the length of the footage. Thus, this manuscript proposes an automatic image analysis (with some input from end-users), and classification to be performed online.

Nevertheless, not all of the technologies assessing biodiversity need to be expensive. For instance, Public Lab's balloon mapping kit is a DIY solution that enables users to take their aerial photos up to 305m. This kit consists of a balloon, a Dacron line, and a few accessories to help set it up, that can have a camera attached to take aerial pictures. For windy days, they offer the alternative Kite Mapping Kit. After taking the photos they can be added onto Public Lab's tool MapKnitter to stitch them into an online map, effectively making a crowdsourced map [17].

Fig. 3. Public Lab's Balloon Mapping Kit. [Source: Public Lab Store]

Weather balloons are also used for telecommunication where the IoT reaches the low-altitudes [23]. Recently, the world record for the longest distance communication using LoRa (Long-Range). The

balloons were tracked using The Things Network[13], plain LoRa® and by APRS and satellite using Spot[14]. The first balloon the break the world record in 2019 had three direction 3D printed Moxon Antennas placed at 120º and a reaction wheel(a type flywheel used for altitude control) which stabilizes and prevents the probe from spinning. In this scenario keeping the probe from spinning is very important since the LoRaWAN transmissions they used can take about a second to finish. This balloon broke the world record several times during its flight however the final record was established when it contacted Lisboa at 741km [24].

The second balloon, which holds the current world record, was aiming to reach the highest altitude possible (>40 000m) and for that purpose, it was developed with optimizations to its lift and the capsule weight. After the launch, due to an incoming storm, the balloon did not reach the intended altitude, reaching up to 28 000m and then dropping to 18 000m before stopping the transmissions. However, two days later the balloon flew by Açores at 33 200m sending its last transmission before disappearing somewhere in the Atlantic. The record was set when the balloon was flying at an altitude of 24 859m and sent a message that was received by 24 gateways, the farthest of which was at a distance of 766km [24].

Balloons have been used in research for monitoring air pollution back to 1981 [25]. These balloons are a great way to monitor air pollution because they have very high vertical and horizontal mobility, and can monitoring at specific altitudes for extended periods. And the main benefit of the balloons is that they are versatile and can get emission samples from both point and nonpoint sources where the ground, tower, or aircraft sampling would have been inadequate or impractical.

The DJI P4 is an all-in-one solution that includes a drone with a multispectral camera attached to it and software to analyze the imagery. The drone has a transmission range of 7km and a flight time of 27 minutes, which is not nearly enough for our purpose that might require more than a couple hours of flight at a time. It is targeted at farmers to gather data about their plots of land and the vegetation health [18]. The data gathered by this device can be imported into free software like DJI Terra or Pix4D for further analysis.

[13]An open-source and tech-savvy community, allowing participants to deploy IoT nodes and gateways and to use open radio communication. (`https://www.thethingsnetwork.org/`)

[14]SPOT (is a french acronym that means "Satellite for observation of Earth") is a commercial high-resolution optical imaging hyperlinkabbrEO satellite system operating from space.

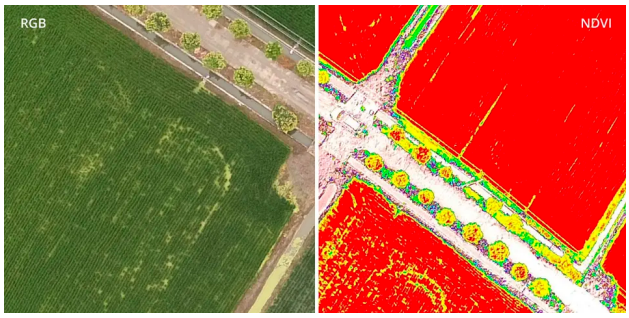


Fig. 4. False color image captured by a DJI P4 drone that highlights the vegetation making it easier to see the weaker areas. [Source: DJI P4 MULTISPECTRAL AGRICULTURE DRONE]

Another solution that is mostly targeted at agriculture is the RedEdge multispectral camera attached to a drone. Besides the false-color image that highlights vegetation, mentioned in the previous solution, this one can also capture a false-color image that highlights water, as shown in figure 5. However, this product is sold only as a camera and needs to be integrated with a drone making it more expensive [26]. The data gathered by this device can be imported into free software like Pix4D for further analysis.

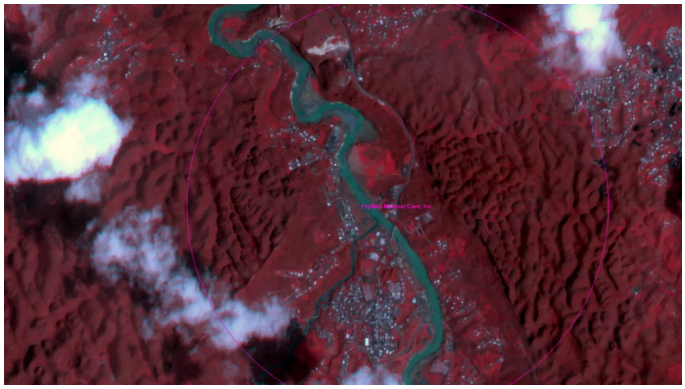


Fig. 5. Example of a false color image that highlights water as light blue. [Source: Assessing Storm Impact in Puerto Rico with Remote Sensing and Digital Volunteers]

Alongside the aerial surveys, the AEROKATS is an adequate solution offering simplistic aerial surveys and kits at a low-cost. These kits use aeropods[15] with some type of camera attached, supporting the aerial imagery. They also have a website called AEROKATS Mission Mapper, where people can see all the data submitted from previous AEROKATS missions through the Survey 123 app [21].

The HEXH2O^{TM} is a kit that contains all the body and electrical components to build a drone designed to be used in an ocean environment. It has a flight time of 25 minutes, can achieve speeds

[15] Aerodynamically stabilized instrument platform for kites and tethered balloons

of 55km/h, and carry payloads up to 2kg. Since it is designed to be water-resistant it can land on the water to capture underwater imagery [22].

Although most of these solutions are adequate for our purpose some of them have significant drawbacks. The SPACEWHALE and HDAS solutions provide great imagery however are not available to the public and even though the pricing is not available, we can estimate that they have a higher cost than the other solutions mentioned. The solutions involving drones are not ideal for our surveys because they are very costly and drones are hard to control even on land so deploying from a boat would be a greater challenge. Finally, the solutions using balloons or kites are adequate, however, they use proprietary software which would hinder our data analysis. Considering all of this the solution we propose is a custom-designed apparatus, with a MCU and open source software, that can be deployed in diverse locations (e.g. attached to a balloon/kite or the mast of a vessel).

2.1.2 Overview of Machine Learning Software

Nowadays, AI (Artificial Inteligence) is everywhere and one of its most popular applications is ML (Machine Learning), in which computers, software, and devices perform tasks via cognition similar to the human brain. ML is integrated with many services that people use every day, for example, virtual personal assistants, traffic predictions while commuting, social media services (people you may know, face recognition, etc), email spam filtering, search engine results refining, product recommendations, etc [27].

To support these applications there are many engines and tools that developers can use to implement machine learning models in several different ways. The following table 2 presents an overview of several tools used for ML, with information pertaining in which programming languages their interface can be used, their availability, release date and if they are open source.

Below, a brief overview of machine learning software is provided, covering some of the benefits of each tool researched.

TensorFlow is an easy to use open-source ML framework, allowing the user to deploy across a variety of platforms. It is one of the most well-maintained and used frameworks. It also allows developing neural networks using flowgraphs [28].

OpenCV is a computer vision tool that is prebuilt with all the necessary techniques and algorithms to perform several image and video processing tasks. As TensorFlow, it is an out-of-the-box solution, however, takes a performance hit when working with massive data sets or very large images [29].

Table 2. Comparison of Machine Learning software

Software	Release	Open Source	Written in	Interface	Price
TensorFlow	2015	Yes	C++, Python, CUDA	Python, C/C++, Java, Go, JS, R, Julia, Swift	Free
OpenCV	2000	Yes	C/C++	C++, Python	Free
SimpleCV	2012	Yes	Python	Python	Free
MATLAB's Deep Learning Toolbox	2016	No	C++, Java, C, MATLAB	MATLAB	1150 €
PyTorch	2016	Yes	Python, C, C++, CUDA	Python, C++	Free
Keras	2015	Yes	Python	Python, R	Free
Accord.NET	2010	Yes	C#	C#	Free
Chainer	2015	Yes	Python	Python	Free
Caffe	2017	Yes	C++	Python	Free
Deeplearning4j	2018*	Yes	Java, CUDA, C, C++	Java, Scala, Python, Clojure and Kotlin	Free
BigDL	2017*	Yes	Scala, Python	Scala, Python	Free
Apache MXNet	2017	Yes	C++, Python, R Java, Julia, Scala JavaScript, Go, Perl	C++, Python, Scala MATLAB, R, Julia Perl, Clojure, JavaScript	Free

* Beta version.

SimpleCV is a simple tool for machine learning targeted at users that do not wish to get into the depths of image processing, however focusing on quick prototyping [30].

MATLAB's Deep Learning Toolbox is another adequate tool for image processing in a research environment. It allows quick prototyping and its' code is concise, making it easy to read and debug. On the other hand, it can get a decrease in speed during execution suggesting it not be a handy tool for a production environment [31].

PyTorch is a python library that enables users to build Deep Learning models and use them in various applications. It focuses on the ease of use and makes it possible for even users with very basic programming experience to use Deep Learning in their projects, which makes it a great first tool to use for ML [32].

Keras is an open-source library design to simplify the creation of deep learning models, it can be deployed on top of other AI tools like TensorFlow. It is a good tool to use when you need an easy and fast prototype and is very user-friendly, modular, and versatile [33].

Accord.NET is a machine learning framework suitable for production-grade scientific computing. You can build various applications in ANNs (Artificial Neural Network) statistical data processing, image processing, between others. It is also one of the few tools that can be used with C# [34].

Chainer is an open-source deep learning framework written entirely in Python on top of the NumPy and CuPy libraries. It is notable for its early adoption of the "define-by-run" scheme, along with its performance on large scale systems. The first version was released in June 2015 and has gained large popularity in Japan since then [35].

Apache MXNet is an open-source deep learning software framework used to train and deploy deep neural networks. It allows fast model training due to being scalable and supports a flexible programming model and also multiple programming languages, listed in table 2. The MXNet library is portable and can scale to multiple GPUs and multiple machines [36].

CAFFE (Convolutional Architecture for Fast Feature Embedding) is an open-source deep learning framework, originally developed at the University of California, Berkeley. It supports several different types of deep learning architectures with a focus on image classification and segmentation. It supports CNN, RCNN (Region Based Convolutional Neural Networks), LSTM (Long short-term memory) and fully connected neural network designs [37].

Deeplearning4j, a deep learning programming library written in Java for the Java virtual machine, is a framework with wide support for deep learning algorithms. It includes implementations of the restricted Boltzmann machine, deep belief net, deep autoencoder, stacked denoising autoencoder, recursive neural tensor network, word2vec, doc2vec, and GloVe [38].

BigDL is a distributed deep learning library for Apache Spark which enables users to write their deep learning applications as standard Spark programs, that can directly run on top of existing Spark or Hadoop clusters [39].

For this thesis, TensorFlow will be analyzed as it provides threefold contribution: (i) it is a ML tool with an accessible and readable syntax; (ii) provides adequate features and services compared to other popular tools; and (iii) allows access to the documentation, tutorials, and out-of-the-box examples. Another important feature is that it provides the TensorFlow Lite, which is a lightweight machine learning solution for on-device inference allowing deployments and execution on mobile and embedded devices, which seem adequate to be coupled with the used MCU proposed by this manuscript.

2.1.3 Types of Imagery

While there are diverse possibilities to obtain the accuracy when detecting the species, image classification remains limited as the collected image is subject to the passive (see **Section 2.1.3.1**) or active imagery [40] (see **Section 2.1.3.2**). The main difference between these two is that passive sensors are designed to detect electromagnetic emissions, these can be the result of the reflected sunlight or produced locally (e.g. thermal radiation from vegetation in the infrared spectrum). And

the active sensors, which are not dependent on the solar illumination, that consist of a transmitter that sends out a specific electromagnetic signal and a sensor receiving the interaction of the signal with the Earth's surface. In this manuscript, a brief overview of both is outlined below, where an emphasis on using the Passive Imagery is taken, as such technique can be easily manufactured [40].

2.1.3.1 Passive Imagery

A candidate of passive imagery types starts with panchromatic imagery. Panchromatic images are created when the imaging sensor is sensitive to a wide range of wavelengths of light, typically spanning a large part of the visible part of the spectrum. The panchromatic band is essentially black and white, it has a wide bandwidth usually between 100-450 nanometers. The wide bandwidth allows it to have a high signal to noise[16], that is why panchromatic data is often available at the highest spatial resolution.

Fig. 6. Image captured through a panchromatic band. [Source: From panchromatic to hyperspectral EARTH OBSERVATION IN A MYRIAD OF COLORS]

The MSI, which is another passive imagery type, captures image data within specific wavelength ranges across the electromagnetic spectrum. These wavelengths can be separated with filters or instruments that are sensitive to particular wavelengths. The most common use of MSI is the photos which most people take every day, the production of natural color by combining 3 bands of the visible spectrum (red, green, and blue wavelengths). However, MSI is not limited to the visible spectrum, different spectrums can be combined like IR (Infrared), UV (Ultraviolet), microwave or others to allow observations which are not visible to the naked eye. A common example of this is

[16]Signal-to-noise ratio is used in imaging to characterize image quality. The sensitivity of an imaging system (digital or film) is typically described in the terms of the signal level that yields a threshold level of Signal-to-noise.

false-color[17], combining the IR, green and red spectral bands to show the vegetation highlighted in red as shown can see in the following figure 7.

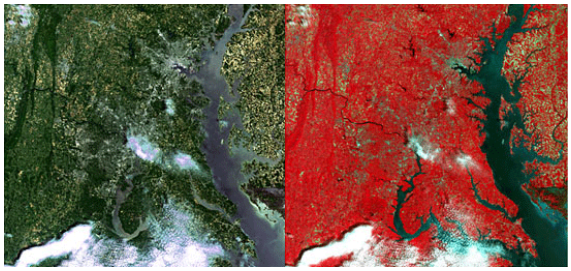


Fig. 7. Comparison between natural color (left) MSI and a highlighted vegetation false-color (right). [Source: False color]

A common combination of passive imagery is Pan-sharpened imagery, which is a process that merges panchromatic imagery with MSI. It aims to deliver the high-resolution images captured by panchromatic bands and add color to them with the help of MSI, as can be seen in the following figure 8.

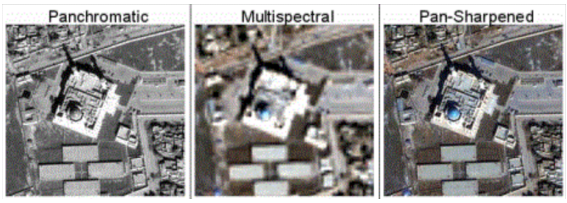


Fig. 8. Example of pan-sharpened image. [Source: Spear Pan Sharpening]

Another example of passive imagery is Hyperspectral imagery, it collects and processes information from across the electromagnetic spectrum. The main goal of this imagery is to collect the spectrum for each pixel in the image of a scene, to find objects or identifying materials. Since certain objects leave unique "fingerprints" in the electromagnetic spectrum, they can be identified by this imagery. For example, using a spectral signature for oil can help geologists find new oil fields.

Finally, there is Microwave Radiometry which is the measurement of the energy emitted at millimeter-to-centimeter wavelengths (frequencies of 1-1000GHz) also known as microwaves. This imagery can measure thermal electromagnetic radiation emitted by atmospheric gases and are useful in a lot of fields like weather forecasting, climate monitoring, radio astronomy, etc.

[17]False color images are a representation of a multispectral image produced using any bands other than visible red, green, and blue as the red, green and blue components of the display. False-color composites allow us to visualize wavelengths that the human eye can not see (i.e. near-infrared and beyond).

2.1.3.2 Active Imagery

In terms of active imagery, there is the SAR (Synthetic Aperture Radar), which is the most common active sensor used to observe the Earth from satellites. It is a sensor that emits electromagnetic pulses towards the Earth's surface and detects the return pulses which were reflected or scattered by the surface features. It can be used to detect ships or oil spills or to monitor sea ice, forests, soil moisture, critical infrastructure, etc.

Similar to SAR there is LIDAR (Light Detection And Ranging) which has the same principle however, it works with IR, UV or visible wavelengths. LIDARs are used to measure precisely topographic features, monitor growth or decline of glaciers, profile clouds, measure winds, study aerosols, and quantifying various atmospheric components.

Radar Altimetry uses the ranging capability of radar to measure the surface's topography along a satellite track. They do this by measuring the time interval between the transmission and reception of the electromagnetic pulse emitted by the sensor. A lot of different information can be gathered from these measurements, such as the ocean topography, the lateral extent of sea ice and altitude of large icebergs above sea level, the topography of the land, ice sheets, or the seafloor, and it can also provide information for mapping the sea-surface wind speeds and significant wave heights.

A relatively new category of satellite navigation applications is GNSS (Global Navigation Satellite System) reflectometry, which involved transmitting and receiving microwave signals reflected from various surfaces to extract information from various surfaces. This process involves the GNSS satellite acting as the transmitter and an airplane or a LEO (Low Earth Orbit) as the receiver. The receiver can also be placed on land if the goal is an altimetry application. The possibilities with this GNSS reflectometry include wide-swath altimetry, sea-wind retrieval, humidity measurements over land, and measurement of seawater salinity and ice-layer density.

Finally, Radar Scatterometry, which uses a microwave radar sensor to measure the reflection or scattering effect produces while scanning the surface of the earth from an aircraft or a satellite. This imagery can provide measures of wind speed and direction near the sea surface and information such as sea ice coverage.

Although the active imagery could be very useful in what this thesis aims to achieve since these sensors are very pricey they will not be used seeing as the goal of this thesis is to develop a low-cost solution. Furthermore, they consume more power because they need an emitter component as well as the receiver, which is another big downside since good energy autonomy is a must. Considering this research into MSI, this thesis will explore the development of a modular apparatus that

can be equipped with different cameras seeing as different light spectrums can contain valuable information to aid the machine learning classification.

2.1.4 Multispectral Imagery Cameras

As stated previously in section 2.1.3, this thesis will focus on collecting the MSI, specifically the RGB, the IV and the UV bands. There is a lot of data that can be gathered with this combination of bands, and they can be used to more easily identify the species in a marine environment. In the following table 3 there is a list of solutions already in the market to capture this type of imagery.

Table 3. Comparison between different Multispectral Imagery cameras.

Producer	Cost	Bands(nm)	Application
Micasense RedEdge [41]	$ 5,195.00	400-1500	Agriculture
MAIA WV [42]	13,500.00 €	390-950	Agriculture, Geology, Industry, Nature
MAIA S2 [42]	15,200.00 €	433-900	Agriculture, Geology, Industry, Nature
MAIA M2 [42]	2,400.00 €	395-950	Agriculture, Geology, Industry, Nature
DJI P4 Multispectral [43]	$ 6,499.00	434-866	Agriculture
Parrot Sequoia+ [44]	$ 3,800.00	510-830	Agriculture
SlateRange 4P [45]	$ 4,950.00	410-950	Agriculture
SlateRange 4P+ [46]	$ 5,750.00	410-950	Agriculture
Sentera 6X [47]	$ 6,349.00	445-860	Agriculture
Sentera Double 4K [47]	$ 3,469.00	446-840	Agriculture
Sentera AGX710 [47]	$ 4,299.00	446-850	Agriculture
Sentera NDVI or NDRE [47]	$ 1,999.00	400-750	Agriculture
Sentera Quad Sensor [47]	$ 4,599.00	450-825	Agriculture

Below, a brief overview of multi-spectral cameras is provided, covering some of the benefits of aerial assessments.

Micasense RedEdge is a professional multispectral camera built with a metal case for extreme durability, can capture five narrow spectral bands, generates plant and health indexes, and can operate up to 60°C.

MAIA WV is a multispectral camera that is equipped with the same wavelength intervals as the WorldView 2 satellite. It has an array of 9 sensors (1 RGB and 8 monochromes with relative band-pass filters) to detect multispectral imagery between the 390nm and 950nm bands. Similar to the MAIA WV we have the MAIA S2 which was built with the same wavelength intervals as the Sentinel 2 satellite. It is an advanced multispectral camera with two narrow bands in Red Edge regions and bands in Violet and Blue regions. For people that don't need as many sensors,

there is the MAIA M2, which is a modular multispectral camera that has 2 modular sensors that can be changed to match the user's needs.

An all-in-one solution that includes a multispectral camera and a drone, to complete aerial surveys, is the DJI P4 Multispectral. This multispectral camera has 6 sensors, one of which is RGB, and the other 5 cover the bands Blue, Green, Red, Red Edge, and Near IR.

Parrot Sequoia+ was designed to be an affordable multispectral camera that is compatible with all types of drones. With its multispectral and sunlight sensors, this camera analyses the plants' vitality by capturing the amount of light they absorb and reflect.

SlantRange 4P is a multispectral sensor consisting of the three RGB bands, a NIR, a Red Edge, and another Red band. It specializes in capturing high-resolution MSI which they claim is a major benefit in an agriculture environment because it enables the users to measure the size and shape of each plant, which provides more options for image analysis. The SlantRange 4P+ provides the same features as the previously mentioned sensor however with twice the resolution.

Sentera provides a multitude of sensors ranging in capabilities and price, the most expensive is the Sentera 6X which can capture MSI in the blue, green, red, red edge and NIR bands. Although other sensors can capture MSI from five different brands, the Sentera 6X has separate high-quality optical hardware to capture each band. Both the Sentera Double 4K and Sentera AGX710 can capture MSI in the same five bands, as the previous sensor, using only two sensors which results in lower quality results, they can also be easily swapped with other sensors that use Sentera's Lock-and-Go technology. Sentera Quad Sensor has four fully-customizable multispectral imagers that can be used to collect NDVI (Normalized Difference Vegetation Index), Green NDVI, NDRE (Normalized Difference Red Edge), and high-resolution color data, all in a single flight. Finally, the Sentera NDVI or Sentera NDRE is a single sensor that has a variant that can capture only NDVI and another that captures NDRE, it is the most affordable solution for people that can work using only this data type.

After researching several commercially available multi-spectral cameras, no solution can target the easy-to-setup application on the sea vessels as well as affordable price. Although most of these cameras would probably work for the project, these cameras were not tested to detect marine megafauna and remain mostly applied in agriculture. Conversely, the ocean is a rough environment and water splashes can easily damage electronics or objects that fall into the sea, hence, additional waterproofing and custom solutions should be invented to secure the deployments in aquatic settings.

2.2 Prior Research

In this section, this thesis will explore prior research on the following topics: (a) **Section 2.2.1**, depicting the aerial image observations; (b) **Section 2.2.2**, providing the research in deploying the IoT devices in marine environment; and (c) **Section 2.2.3**, reporting the overview of works in image vision classification and algorithms, important to detect and classify the species. In the following table 4 we can see a general overview of the projects studied.

Table 4. Overview of studied projects.

Author	Project	Data type	Setting	Data
Nieukirk et al. [48]	Drone up! Quantifying whale behavior from a new perspective improves observational capacity	Video	Aquatic	Offline
Borowicz et al. [49]	Aerial-trained deep learning networks for surveying cetaceans from satellite imagery	Video	Aquatic	Offline
Murgai et al. [50]	Development of an Automatic Classification System for Cetaceans Using their Vocalizations	Audio	Aquatic	Offline
Lopez et al. [51]	Automated detection of marine animals using multispectral imaging	MSI	Aquatic	Offline
Armstrong et al. [25]	Tethered balloon sampling systems for monitoring air pollution	Air Samples	Urban	Offline
Fretwell et al. [14]	Whales from space: counting southern right whales by satellite	Satellite Imagery	Aquatic	Offline
Radeta et al. [52]	SeaMote-Interactive Remotely Operated Apparatus for Aquatic Expeditions	Video /Audio	Aquatic	Offline
Schoonmaker et al. [53]	Spectral detection and monitoring of marine mammals	MSI	Aquatic	Offline
Kuznetsov et al. [54]	Red balloon, green balloon, sensors in the sky	Air Samples	Urban	Offline
Jensen et al. [55]	Detecting the attributes of a wheat crop using digital imagery acquired from a low-altitude platform	MSI	Rustic	Offline
Saghri et al. [56]	BalloonSat: design, implementation, and application of a low-cost tethered weather balloon remote sensing station	MSI	Urban	Offline
Radeta et al. [57]	POSEIDON-Passive-acoustic Ocean Sensor for Entertainment and Interactive Data-gathering in Opportunistic Nautical-activities	Video /Audio	Aquatic	Offline

2.2.1 Aerial Image Observations

The two most common ways to obtain aerial imagery in oceanic settings are aircraft surveys and satellite imagery. There have been several research projects done on the topic of whale detection from satellite imagery. In this case, the first step to perform this detection is to collect aerial imagery of whales and train a classification model to correctly identify them. Two of the projects ([49] and [15]) used the same method to collect such imagery. They first obtained aerial imagery from an aerial surveying company (see HDAS mentioned in section 2.1.1), serving as a baseline, and then downsampled the images to simulate the satellite imagery.

Another work gathered satellite imagery, from the WorldView2[18] satellite[19], surveying a certain region in the Golfo Nuevo Bay which is a bay that separates the Península Valdés from Argentina's mainland. The reason for the region chosen is that it is the largest breeding aggregations of the whale species that was being studied [14]. After the imagery gathering, they ran five analyses through the data (a manual analysis, two unsupervised classification techniques, and two Thresholding [20] analysis) and compared the results against the manual analysis as we can see in the following table 5, which shows that the Threshold Panchromatic analysis was the most accurate, however, the manual still got more total whale detections.

Table 5. Comparison between automatic and manual whale detection. (Adapted from Whales from Space: Counting Southern Right Whales by Satellite)

		Unsupervised		Threshold	
	Manual	**iso means**	**kmeans**	**Panchromatic**	**band 5**
Total signals	91	158	102	64	101
Probable	55	44	42	43	49
Possible	23	16	11	14	15
Band 5 only	13	1	0	0	13
Total found	-	61	53	57	77
Found (%)	-	67.0	58.2	62.6	84.6
Total missed	-	30	38	34	14
Missed (%)	-	33.0	41.8	37.4	15.4
False positives	-	97	49	7	24
False positives(%)	-	61.4	48.0	10.9	23.8
Correct(%)	-	38.6	52.0	89.1	76.2

[18] DigitalGlobe's WorldView-2 is a satellite sensor that was launched October 8, 2009, and provides a high-resolution panchromatic band and eight (8) multispectral bands; four (4) standard colors (red, green, blue, and near-infrared 1) and four (4) new bands (coastal, yellow, red edge, and near-infrared 2), full-color images for enhanced spectral analysis, mapping and monitoring applications, land-use planning, disaster relief, exploration, defense and intelligence, and visualization and simulation environments.

[19] This imagery was bought from the provider Digital globe.

[20] Image thresholding is a simple, yet effective, way of partitioning an image into a foreground and background. This image analysis technique is a type of image segmentation that isolates objects by converting grayscale images into binary images.

Besides satellites and aircraft surveys, several research projects used balloons. One of these projects used white balloons and a triple colored led to indicate the air quality in a public park. Using a volatile organic compounds sensor or a dual function diesel/exhaust sensor, these devices change the color of the led depending on the air quality which could be optimally displayed at night because the white balloon easily reflects the led color [54]. A big problem of these balloons is that sometimes after a mission their location is unknown, so to tackle this problem a research project developed a balloon that combined LoRa, cellular IoT, and live video transmission which allows for better real-time monitoring of the apparatus, reducing the risk of losing the balloon [23]. Another project used a weather balloon with a multispectral camera to gather imagery of a parking lot and count the number of occupied spots using machine learning [56]. Similar to this work, there was a project that gathered multispectral imagery, also with a balloon and a camera, of a wheat field to analyze crops health [55].

Aside from satellite, aircraft, and balloon, drones were also used in the analysis of whale to assess the footage from an aerial view. During the whale sightings research spent 594 minutes with whales, however, the whales were only visible from the traditional horizontal view for sole 104.8 minutes. On the other hand, with the footage gathered from the drone, the whales were visible for 300.6 minutes which is an increase of 3 times from the traditional techniques [48].

While these works provide important ways to assess the aerial imagery, most of them remain inaccessible or remain at a high cost. In the case of former, such as satellite or aircraft imagery, they are not easily accessible to a wider audience. In this thesis, the proposed system opens the door to wider audiences to design, deploy, and test the proposed system. As for the latter case, it is possible to estimate that, even using the cheapest off the shelf aerial assessment technique (Balloon mapping kit) in table 1, as well as the cheapest multispectral camera in table 3, the total price would be in the thousands of euros. Furthermore, most of these tools require manual and offline analysis of the data. The proposed research in this study will provide an automatic tool that analyses the data in near real-time, allowing the user input for improving the classification model.

2.2.2 IoT devices in Marine Environments

The internet of things, or IoT, is a system of interrelated computing devices, mechanical and digital machines, objects, animals, or people, that are provided with UIDs (Unique identifier). They can transfer data over a network without requiring human-to-human or human-to-computer interaction. These devices seem to be an adequate way to monitor diverse environments (such as aquatic ones) and gather data [58].

Nowadays IoT devices can be deployed in diverse settings, including the seas. Due to the rapid advancement in technology in the last two decades, the possibilities for devices to monitor the ocean in general, or the water quality, fish farms, coral reefs, waves, and currents, among others, are higher than ever before. This research targeting marine environment monitoring is crucial to better understand the seas or climate change causes and effects. This results in a rise IoT devices being created to expand the human knowledge of the oceans and take better care of it [58].

This research [58] studied 40 of the IoT devices developed for use in a marine environment, depicting where this technology currently stands. Most of these devices were used for monitoring or data gathering, and the environments they were deployed in range from the sea to rivers, pools, lakes, and some were only tested in a lab environment.

Besides data gathering, some of the devices deployed in a marine environment are used to provide a better experience to end-users like the POSEIDON (Passive-acoustic Ocean Sensor for Entertainment and Interactive Data-gathering in Opportunistic Nautical-activities). The device consists of a capsule that has two media acquisition tools, a hydrophone which will be submerged and is connected to it through a 10m cable and a GoPro camera that will gather underwater video samples, and connecting to it through a mobile app gives the user access the media that is being collected. This PAM (Passive Acoustic Monitoring) device was used to enhance the whale watching boat trips, in which people spend a few hours in a boat sailing, however only see whales for a couple of minutes. With this device, during the downtime between whales surfacing the users can use the mobile app to interact with the data collected [57]. An example of how a more active device can be used in a marine environment is the SeaMote, which is a small remotely operated surface vehicle that can be controlled through a mobile app [52].

Nevertheless, one of the biggest problems of these devices is the battery capacity. Since they are deployed in a marine environment, they are not easily connected to a power outlet, thus most of them work on a battery for a limited amount of time. This is the main reason why most of such devices are only used for monitoring since this purpose doesn't require much power and the information gathered can be useful [58]. Provided research is an adequate contribution to this field and provides an insight into the problems this thesis will have to overcome during the development of its' apparatus.

2.2.3 Detecting Aquatic Species using Machine Learning

As this thesis explores the usage of an IoT device and algorithms for image vision, it is important to note that there is a large number of diverse aquatic animals, providing difficulty in counting and classifying the exact number of taxa. A marine biologist needs a better resolution when counting these species, as traditionally, visual surveys are conducted with one observer [4]. Conversely, some

efforts used semi-automated pipelines to identify only cetaceans, using their most recent location using satellite imagery and a convolutional neural network (CNN). The satellite imagery is divided into tiles and the trained CNN would then classify whether a tile contains a whale. Their best model managed to correctly classify 100% of the tiles that had whales and 94% of the tiles that only contained water [49].

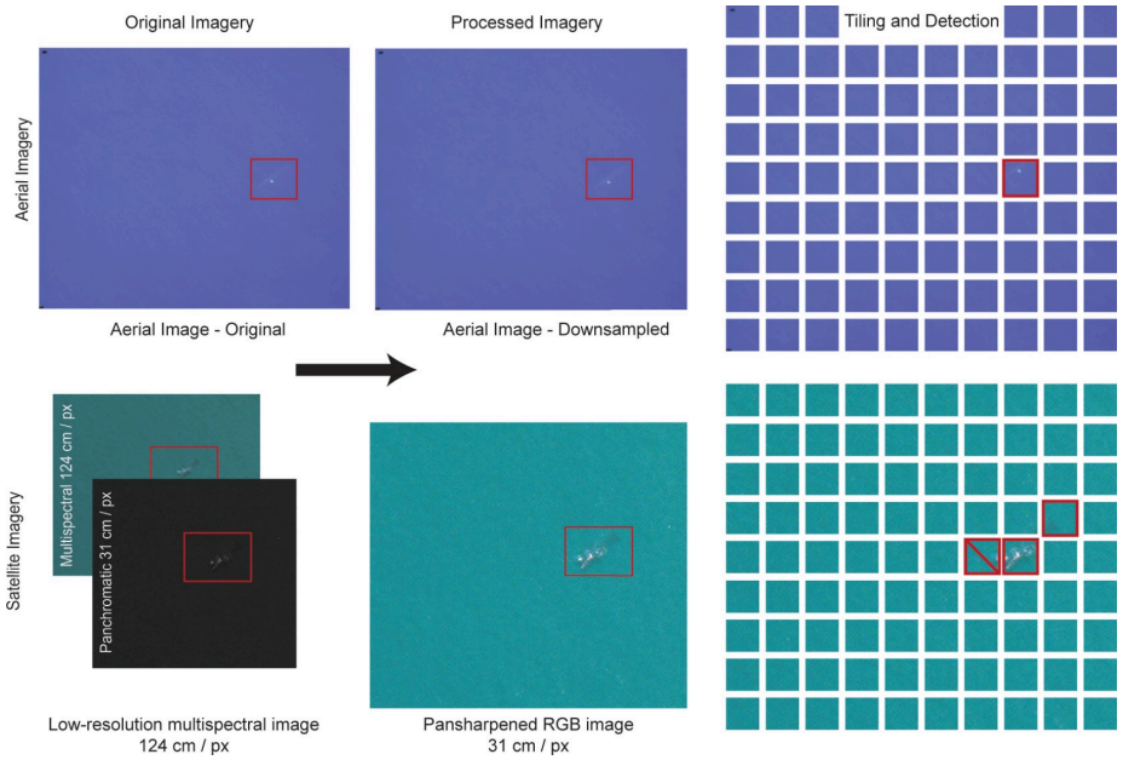


Fig. 9. Aerial-trained deep learning pipeline's automated workflow. [Source: Aerial-trained deep learning networks for surveying cetaceans from satellite imagery]

Not only images can be used for training the models. Even though the sound is very distant from images to a neural network they are just numbers. In a research paper about developing an automatic classification system for cetaceans [50], authors describe the system which classifies different whale species based on the vocalization acoustic signals (either raw or preprocessed). The dataset they used consisted of five different cetacean species, namely: blue whale, fin whale, Cuvier's Beaked whale, Sperm whale, and porpoise. For this classification task, they extracted thirty-four features specific to acoustics. Besides the time domain and frequency features, there are many features inspired by speech recognition and music classification applications, similar work has been done in POSEIDON [57]. The classifiers used were the SVM (Support Vector Machines) and the RF (Random Forest) [50].

Murgai et al provides insight to three diverse tests [50]. The first consisted of a binary classification that had the system detect if the sound was from a blue or a fin whale. These test results yielded an accuracy of 95% for both classifiers. The second test had the system classify four species (excluding the sperm whale which was not inculcated into the system). The results of this test were successful as well with an accuracy of 93.2% for the RF and 91.1% for the SVM [50]. The final test was an experiment to remove 15 features that were thought to be redundant. The results prove this point because although the accuracy dropped to 87% for the SVM, the RF remained at 93% [50].

Although this research used acoustics instead of imagery, which is the selected medium for this thesis, it is of a contribution to know that machine learning systems can identify different species with greater accuracy [50].

2.3 Overview of Multi-Spectral Imagery

As this manuscript explores the usage of aerial imagery, it is important to note several diverse kinds of imagery inputs. Instead of relying solely on traditional colored photos (RGB), it is also possible to use multispectral sensors to gather several filtered images. Such imagery conveys a lot more data that is not present in the visible light spectrum[21]. These filtered images can also improve the whale classification algorithms, executed by the image vision tools. In research about detecting marine fauna using MSI, the study found that certain bands can detect submerged animals more easily, as we can see in figure 10 [53].

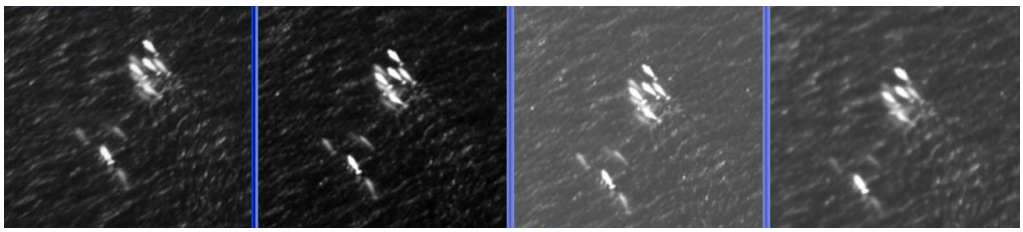


Fig. 10. Multispectral images showing four different bands (in order blue 488nm, blue-green 532nm, green 550nm, red 600nm). As we can see, the second and third bands show a few more animals than the other two. [Source: Spectral detection and monitoring of marine mammals]

Lopez et al developed a system to automatically detect marine fauna using multispectral imagery [51]. The system starts by capturing four multispectral images that are then flat field-corrected and calibrated to put into reflectivity units, and finally, they are used to generate a composite image. This final image is then analyzed pixel-by-pixel by the system, which compares it against several spectral and spatial filters. When this analysis is done the scores of the pixels

[21] The visible light spectrum is the segment of the electromagnetic spectrum that the human eye can view. Typically, the human eye can detect wavelengths from 380 to 700 nanometers.

are blobbed together and only the detections that score high in spectra, shape, size, and density that match a target animal target are retained as detections. After this data is generated the final image is sent to the user which can validate the detection.

Fig. 11. Sperm whale detection by the automated MSI system. [Source: Automated Detection of Marine Animals Using Multispectral Imaging]

Another group developed an apparatus with two multispectral sensors that collect data with sensors designed for the 480nm and 550nm bands, a LWIR (Long wave Infrared) and a commercial color video camera. Instead of using this apparatus for aerial surveys, like the previous works, it was designed to be mounted on solid ground and gather long-range imagery from land thus evaluating the ability to use MSI and IR sensors to detect marine mammals (and their blows, breaches, etc.) from a low-grazing angle system at great distances. This device was tested on the coasts of Hawaii and demonstrated the ability to detect marine mammals at up to 8 US miles as seen in fig. 12, furthermore it could gather imagery at night or with poor light conditions thanks to the IR sensor as seen in fig. 13 although it has resolution and range limitations [59].

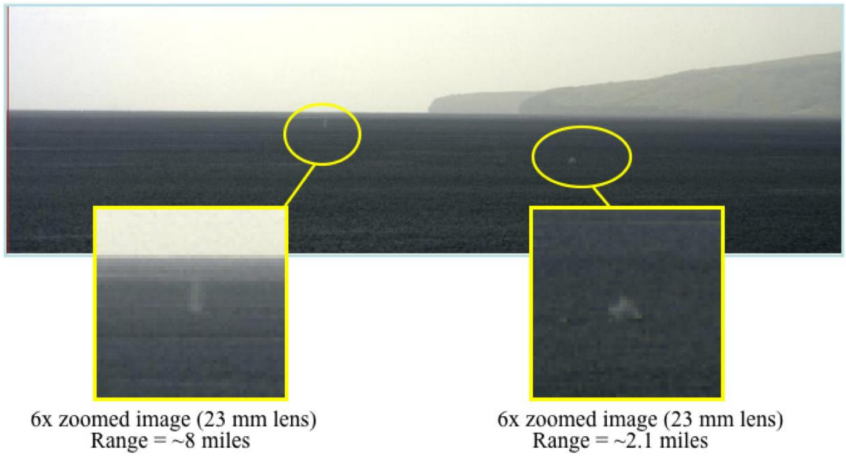


Fig. 12. Long range whale imagery [Source: Multispectral Observations of Marine Mammals.]

Fig. 13. Whale imagery captured at night using an IR sensor [Source: Multispectral Observations of Marine Mammals]

As it was found during this research, MSI will be a great tool to capture imagery of marine fauna that is more easily detected compared to the traditional RGB imagery. Furthermore, the IR band yielded great results regardless of the light conditions, which can't be said of the RGB bands. Therefore, the apparatus being modular is a must, seeing as we can replace the camera to fit the survey conditions (e.g. replacing the RGB camera module with an IR one such as a Pi NoIR Camera [60]).

2.4 Summary of the Related Work

As it can be assessed in the table 4, most of the aforementioned works collected data in an offline manner, obtaining the data after the deployment. In this manuscript, the contribution of an apparatus is to portray the data will be in real-time. Moreover, the aforementioned technologies and techniques for assessing aerial imagery remain at a high cost (see Table 1), this manuscript will provide a low-cost apparatus with a custom attachment solution, as it is a more accessible way to collect the aerial imagery from the mast of sea vessels. Moreover, although MSIs remain at a significantly high cost (as in Table 3) proposed apparatus will provide a modular design allowing for several cameras to be swapped when necessary, providing low-cost alternatives to assess imagery. Lastly, used algorithms to accurately classify and count the species will be based on TensorFlow Lite, an open-source existing image vision and machine learning technique, allowing the ease of access and programming languages (see Table 2).

3 Methodology

This section reports the motivation to develop an apparatus to support marine visual surveys (**Section 3.1**), followed by the data treatment using machine learning (**Section 3.2**), outlining the data gathering, collection, annotation, augmentation, as well as the steps for training the models of marine megafauna and used software. Moreover, it provides an overview into the used apparatus for performing the aerial surveys (**Section 4**), serving as an interface to support the collection of data by marine biologists. Lastly, the study setup is described, providing the methods, obtained data inquiry, experiments, and depicting insights into collecting, interpreting, and displaying the accuracies of detected species (**Section 5**).

3.1 User Pre-Study

To understand the overall challenges which marine ecologist experience during their visual surveys, two field trips, and two focus groups were performed with the marine ecologists (N=6 in both groups). Where the first focus group explored the idea of an IoT apparatus capable of detecting marine megafauna and the second focus group brought about the idea of a mobile marine assistant tool, namely the mobile app. Additionally, the groups helped understand the challenges of deploying such systems, for example, the severe weather conditions.

To better understand the issues with visual surveying methods, the first focus group was asked the following questions, yielding the responses:

- **(Q1)** How proficient are you in reporting the sightings/estimating the species from the boat?
  R: All six subjects had the expertise and proficiency in doing visual surveys from the sea-vessel.
- **(Q2)** Which protocol do you use when reporting the sightings/estimating the species from the boat?
  R: Most subjects used the CETUS Protocol[61], when cetaceans are observed on another observer is called to confirm the sighting and specie then marking it down on paper.
- **(Q3)** Which tool do you use when reporting sightings/estimating the species from the boat?
  R: All subjects reported the usage of binoculars.
- **(Q4)** Which pain(s) do you encounter when reporting the sightings/estimating the species from the boat?
  R: Most of the subjects pointed to subjective and rule of thumb estimation of the population as an issue making it difficult to correctly count the wildlife.
- **(Q5)** Which factors do you feel may influence the accuracy of reported sightings/estimating the species from the boat?

R: The main reported difficulties are weather conditions (sun glare and sea state) and distance to the species.

- **(Q6)** Do you think that an automated process of reporting the sightings/estimating the species from the boat would be helpful (as an assisting tool)?
  R: Most of the subjects pointed that it could be used if complemented with other survey inputs(behavior, group size, number of calves) as well as that it may serve the non-expert persons who are performing the surveys on regular ferries.
- **(Q7)** If yes, how do you envision the usage of such an automated system?
  R: Most of them indicated that the system can be scouting the greater distance and that it is a complementary tool.
- **(Q8)** If you obtained the important data in such a way, how would you use such collected data afterward?
  R: Having a database may be very helpful to determine migration routes, abundance, and richness of species which helps to understand how the populations are varying through the years.

The second focus group served as a design session where a prototype of a mobile app was discussed. The main goal of this app would be to interface with the apparatus and provide real-time imagery and statistics about the detections. This data will aid the marine ecologists in their visual surveys mainly in reaching agreements in terms of the abundance of the marine megafauna as well as confirmation of the classification.

3.2 Machine learning

Regarding machine learning, this chapter reports the performed process to achieve the trained model that is capable of detecting the five different marine megafauna taxa (Turtle, Bird, Mysticeti, Odontoceti, Pinniped). Firstly, it describes the tools and methods used to gather and treat the vast amount of data (**Section 3.2.1**). Also, it provides the training protocol (**Section 3.2.2**). Lastly, the process to convert the resulting model to a version compatible with the proposed apparatus is described (**Section 3.2.3**).

3.2.1 Data gathering

In order to train the models, it is necessary to collect the imagery of marine megafauna. In this section, process of data collection is described (**Section 3.2.1.1**), including the data annotation (**Section 3.2.1.2**). Lastly, data augmentation process (**Section 3.2.1.3**) is depicted, allowing the increase of the imagery, used for training the model.

3.2.1.1 Data collection

The first step to achieve the desired model is gathering a vast amount of marine megafauna data that will be later used for training the model. Images that depict any of the five marine megafauna taxa (Turtle, Bird, Mysticeti, Odontoceti, Pinniped) were gathered from the internet and stored. The total obtained imagery was 11971, averaging at around 2300 per species. The breakdown per species can be seen in the following table 6.

Table 6. Number of collected images per each taxa.

Species	Images
Mysticeti	2002
Odontoceti	1998
Pinniped	2380
Bird	3807
Turtle	1784
Total	**11971**

The imagery was collected from two different sources: the OID (Open Images Dataset) and Google Image Search. To manage all the data, the platform supervise.ly[22] was used. This particular platform was used as it provides several tools to support the model training process, ranging from image annotation[23] to training and validating the ANN. Although it was possible to use it for all the data treatment steps (data gathering, training, and testing), only data gathering, annotation, and augmentation were chosen, due to the easiness of using the other platform for training and validation (more description is provided in Section 3.2.2.1). After uploading all the aforementioned collected imagery to supervise.ly, data annotation is described in the next step.

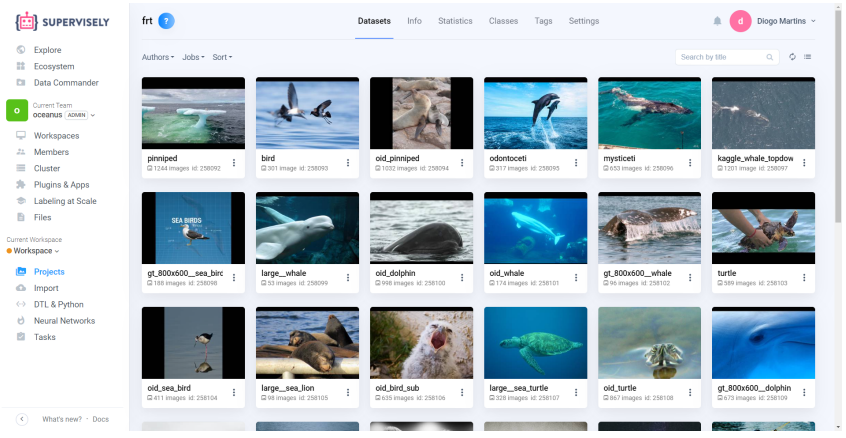


Fig. 14. Custom dataset stored and managed in supervise.ly.

[22] Accessible at `https://supervise.ly/`

[23] Image annotation is the human-powered task of annotating an image with labels and bounding boxes. These labels are predetermined by the AI engineer and are chosen to give the computer vision model information about what is shown in the image.

3.2.1.2 Data annotation

For the ANN to correctly detect objects, it needs to be trained with annotated images. In other words, every picture needs to have the selected objects (in this case, using a bounding box) and a label indicating its type. An example of such annotation can be seen in Figure 15. To ensure that the dataset would have an adequate number of images per marine megafauna taxa class, manual data collection and annotation were performed using Google Image Search. However, most of the imagery was collected from Open Images Dataset. The latter one was favorable as it provides the data which are already annotated automatically by a neural network. Nevertheless, it is important to outline several constraints, as this automated annotation process resulted in two issues: **i)** double object annotation, meaning that an object was detected as belonging to two different classes[24]. Moreover, a case of **ii)** grouped object annotation was spotted, which occurred when several instances of an object were included in the same bounding box annotation. To mitigate such issues, marine megafauna taxa classes were defined to mitigate the first issue, by matching the visually similar species by grouping them. For instance, the Mysticeti parvorder is a group of species known also to be baleen whales, named after their filter-feeder system. Also, the Odontoceti parvorder is comprised of all toothed cetaceans, where some taxa are commonly known as "whales" are included, such as the sperm whale and the killer whale (otherwise known as "orca"). This distinction will therefore yield adequate results, as Mysticeti taxa have visual features that are quite distinctive from those of Odontoceti taxa.

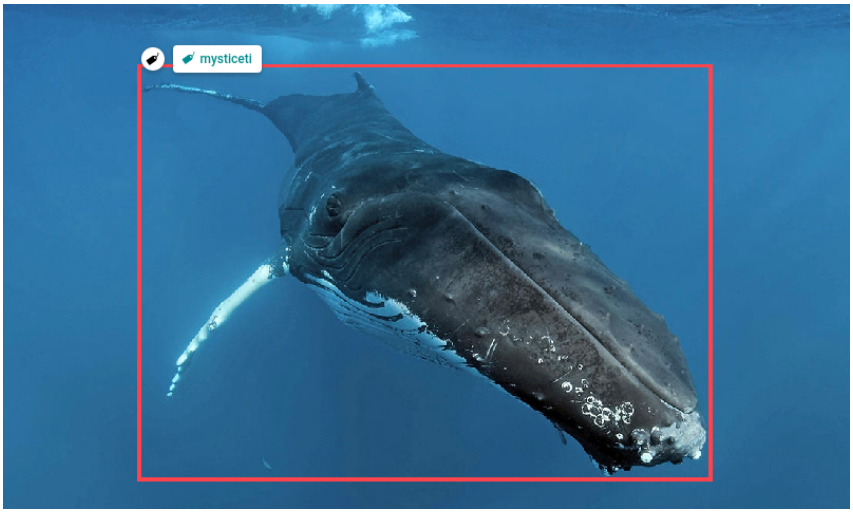


Fig. 15. Image annotated using supervise.ly

[24]For example, killer whales being tagged as both "whale" and "dolphin", due to the ongoing debate in marine biology when classifying this particular specie.

3.2.1.3 Data Augmentation

After data collection and data annotation, to increase the aforementioned dataset with more versatile imagery in the training phase, data augmentation was performed. This method was chosen as it increases the amount and diversity of the dataset without a need of searching and collecting more imagery data. This was achieved by transforming the current dataset with different operations. The most common operations to augment image datasets are rotation, shearing, zooming, cropping, flipping, and changing brightness, however, some image augmentation packages offer more than 60 operations, such as in the case of imgaug [25] package.

Normally, by applying a single transformation to every image, it was possible to effectively double the dataset, however seeing as the dataset already has an adequate quantity of imagery there was no need to apply a great number of transformations. Therefore, the transformations were applied to half of the images at random to simulate different conditions. Applied transformations were:

i) flipping the image, which will simulate imagery of the same object from different angles (**figure 16**);

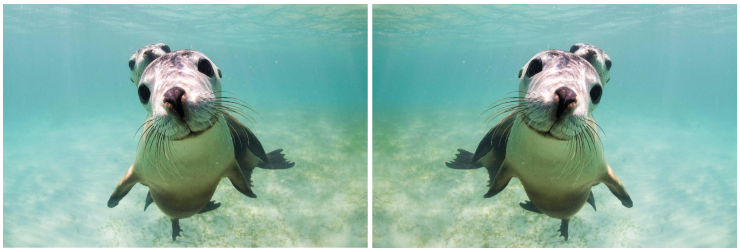


Fig. 16. Image augmentation by flipping it horizontally.

ii) modifying the image's hue[26], which will simulate different lighting conditions (**figure 17**);

Fig. 17. Image augmentation using color transformation.

[25] Accessible at `https://github.com/aleju/imgaug`

[26] Hue, in the context of color and graphics, refers to the attribute of visible light due to which it is differentiated from or similar to the primary colors: red, green and blue. The term is also used to refer to colors that have no added tint or shade.

iii) applying a noise filter to the image, which will simulate low lighting conditions or bad camera quality (**figure 18**);

Fig. 18. Image augmentation using noise.

iv) applying blur to the image, which will simulate fast moving objects that cannot be clearly captured by a camera sensor (**figure 19**);

Fig. 19. Image augmentation using blur.

Using supervise.ly's DTL (Data Transformation Language) a query is defined to perform data augmentation in a view to prevent overfitting. The query steps are the following:

- image resize to 480x320 dimensions
- filtering out images in which the object area is less than 10% of the total image area
- random color jitter and noise
- randomly assign pictures for validation, train and test sets via tagging

3.2.2 Model training

This section describes process of training the model for object detection, firstly the tools used (**Section 3.2.2.1**), then the training hardware (**Section 3.2.2.2**) and finally the training protocol (**Section 3.2.2.4**).

3.2.2.1 Tools

After gathering the required data and proceeding to the training, Google's Colaboratory[27] platform was used. This platform allows the execution of jupyter notebooks[28] which can be used to manage the data files and run TensorFlow's training functions. This platform was used due to the hardware they provide, in the following section (**Section 3.2.2.2**) the importance of the hardware is highlighted.

3.2.2.2 Training hardware

When running machine learning applications the hardware plays an important role in deciding its performance or time to complete. When performing object detection it can improve or decrease the detection rate depending on its power. If an object that we want to detect is moving with some speed and the hardware that is performing the object detection is only capable of processing one frame per second (as it was the case with a standalone RPi3) it will be difficult to accurately identify the object. Furthermore, when training a model the entire process has to be executed so by having powerful hardware this process can be drastically faster.

Seeing as a python script is enough to perform the training, it could be run in most computers or even some MCU like the CDB[29] (Coral Dev Board) however it would take a lot longer to finish the process. Additionally the performance in machine learning tasks varies drastically between a CPU (Central processing unit) and a GPU (Graphics processing unit), since they are design very differently. In the following table 7 we can see an overview of their differences:

Table 7. Differences between CPUs and GPUs.

CPU	GPU
Central Processing Unit	Graphics Processing Unit
Several cores	Many cores
Low latency	High throughput
Good for serial processing	Good for parallel processing
Can do a handful of operations at once	Can do thousands of operations at once

[27] Colaboratory is a free Jupyter notebook environment requiring no setup and running entirely in the cloud. Colaboratory allows the writing and executing of code, saving and sharing analyses, and accessing powerful computing resources, all from a browser.

[28] The Jupyter Notebook is an open-source web application that allows the creation and sharing of documents that contain live code, equations, visualizations, and narrative text. Uses include data cleaning and transformation, numerical simulation, statistical modeling, data visualization, machine learning, and much more.

[29] The Coral Dev Board is a single-board computer that is adequate to perform machine learning inferencing in a small form factor. This dev board also has an integrated Edge TPU (Tensor Processing Unit) coprocessor, which is a small application-specific integrated circuit that provides high performance ML inferencing with a low power cost.

The facts stated in the previous table 7 result in an extreme difference in the bandwidth provided by these components, which is a major factor for machine learning tasks. In the following figure 20 we can see the bandwidth of some common CPUs and GPUs that are optimized for high-performance and general-purpose computing.

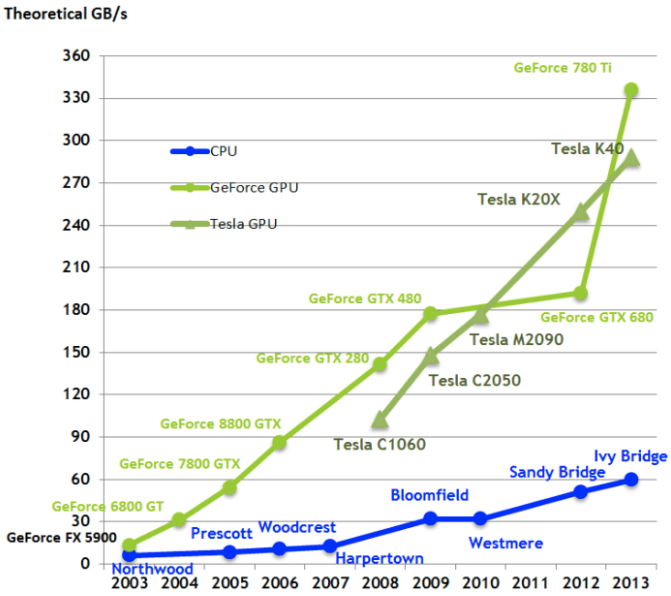


Fig. 20. Bandwidth comparison between CPUs and GPUs [Source: Do we really need GPU for Deep Learning? - CPU vs GPU]

Therefore, before executing the jupyter notebook in Google Colaboratory, the runtime type can be edited. In terms of programming language, it provides the choice between Python3 and Python2 and regarding the hardware the choices available are None (only CPU), GPU and TPU. The GPUs available in Colab often include Nvidia K80s (480 GB/s), T4s (320 GB/s), P4s (192 GB/s) and P100s (82 GB/s). In the following table 8 we can see the results of an experiment performed where as model was trained with 1000 training steps and 50 evaluation steps using the three different hardware configurations available, however due to the used versions of TensorFlow and model architectures, the TPU times are higher than expected due to the configuration files not being adequate to properly utilize this hardware.

Table 8. Model training time using different hardware configurations.

Hardware:	**CPU**	**TPU**	**GPU**
100 Steps (s)	790.49	948.41	48.31
200 Steps (s)	773.98	999.31	39.11
300 Steps (s)	766.27	907.89	39.69
400 Steps (s)	838.05	1001.57	40.19
500 Steps (s)	792.19	964.29	41.82
Total time (s)	3960.97	4821.47	209.12

Furthermore, to keep their server usage under control Google implemented a runtime limit for each instance of the Colaboratory platform. Google Colab notebooks have an idle timeout of 90 minutes and an absolute timeout of 12 hours. Meaning, if a user does not interact with his Google Colab notebook for more than 90 minutes, its instance is automatically terminated. Also, even if the user keeps interacting with the notebook the absolute limit will only allow its execution up to 12 hours.

Ultimately, within the 12 hours time limit a TensorFlow model was trained using our custom dataset and the following parameters: 70 000 training steps and a batch size of 12. Although the training steps could not be increased within this time frame, an accuracy analysis of this trained model yielded adequate results. This analysis was accomplished by using the resulting model to examine several pictures of the marine megafauna and list the existing species in each alongside an accuracy score. However, only an optimized model can be loaded by our target framework, TensorFlow Lite, therefore further steps need to be performed to achieve it.

3.2.2.3 Model Quantization

Seeing as the CDB is a low powered device, low inference performance is expected therefore some steps can be taken during the training phase to improve the overall performance of the apparatus. Tensorflow Lite and the Tensorflow Model Optimization Toolkit provide tools to minimize the complexity of optimizing inference, one of these tools is quantization, which works by reducing the precision of the numbers used to represent a model's parameters (default are 32-bit floating-point numbers). Resulting in smaller model size and faster computation. Inference efficiency is particularly important for edge devices, such as mobile and Internet of Things (IoT) since they have many restrictions on processing, memory, power consumption, and storage for models. Also, some optimizations (i.e. quantization) are required to allow the use of specialized hardware for accelerated inference like the Edge TPU present in the CDB. Therefore, our training uses a quantization optimized model architecture based on the MobileNet V2.

It's recommended to consider model optimization during the application development process, seeing as post-training optimizations are possible but not as effective as optimization aware training. In the following tables 9 and 10 we can see an overview the quantization types and benchmarks of an example model's accuracy and latency using different optimizations. It should be noted that not all quantizations support the intended hardware, in our case the TPU and that even though post-training quantization is possible it should be avoided in order to minimize the accuracy loss.

Table 9. Types of quantization available in TensorFlow Lite.

Technique	Data requirements	Size reduction	Accuracy loss	Supported hardware
Post-training float16 quantization	No data	Up to 50%	Insignificant	CPU, GPU
Post-training weight quantization	No data	Up to 75%	Noticeable	CPU
Post-training integer quantization	Unlabelled representative sample	Up to 75%	Small	CPU, EdgeTPU, Hexagon DSP
Quantization-aware training	Labelled training data	Up to 75%	Minimal	CPU, EdgeTPU, Hexagon DSP

Table 10. Latency and accuracy results for post-training quantization and quantization-aware training.

Model	Accuracy			Latency (ms)		
	Original	Post Training Quantized	Quantization Aware Training	Original	Post Training Quantized	Quantization Aware Training
Mobilenet-v1	0.709	0.657	0.70	124	112	64
Mobilenet-v2	0.719	0.637	0.709	89	98	54
Inception_v3	0.78	0.772	0.775	1130	845	543
Resnet_v2	0.770	0.768	N/A	3973	2868	N/A

3.2.2.4 Training protocol

In this section steps taken to complete the training, in the jupyter notebook mentioned earlier, are briefly described. The notebook is set up so that it can easily be deployed in any machine and executed as soon as needed since it downloads all the required files and libraries automatically.

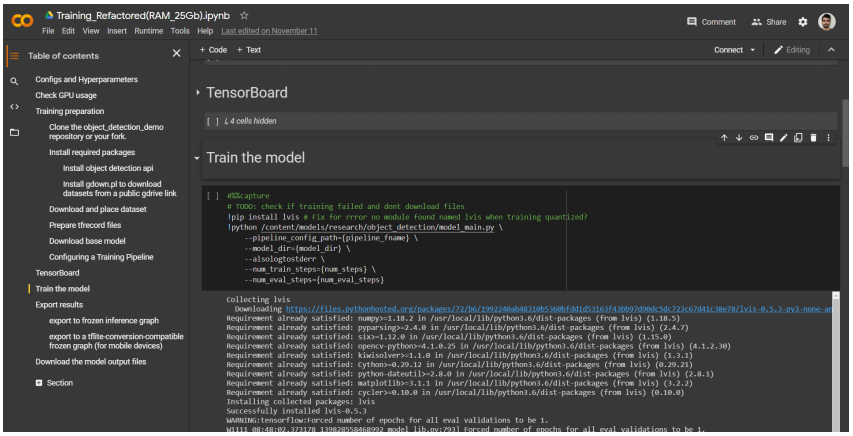


Fig. 21. Jupyter Notebook hosted in Google Colaboratory which provides easy access to high performance hardware.

This notebook was developed to automatically complete the training and download the resulting model files. The user only needs to set up the training parameters, namely the dataset download URL, number of training steps, batch size, and finally the model architecture. The

dataset needs to be exported from supervise.ly and zipped before being uploaded to google drive, where the download URL is provided and inserted into the parameters of the training. If the training completes, a zip file containing the trained TensorFlow model files is automatically downloaded, and considering the platform's time constraint a quantized model of 70 000 training steps was the highest possible to achieve in the 12 hours. However, this model is not compatible with the used tool TensorFlow Lite so a conversion is performed.

3.2.3 TensorFlow Lite conversion

In the previous section, after training is completed there are several resulting files, a typical TensorFlow model consists of the following files: *model-ckpt.meta*, *model-ckpt.data-xxxx-of-xxxx*, *model-ckpt.index* and *checkpoint*.

Using the previous files, a frozen graph was exported which is capable of being converted to TensorFlow Lite allowing it to run in our apparatus. To perform this conversion TensorFlow provides a tool called TensorFlow Lite Converter that can get as an input a SavedModel, a tf.keras model file or a frozen graph, and outputs a **tflite** file. The resulting file is a TensorFlow Lite FlatBuffer, it can be shipped to client devices, generally mobile devices or low-powered computers like our CDB, where the TensorFlow Lite interpreter handles them on-device. This flow is represented in figure 22 below.

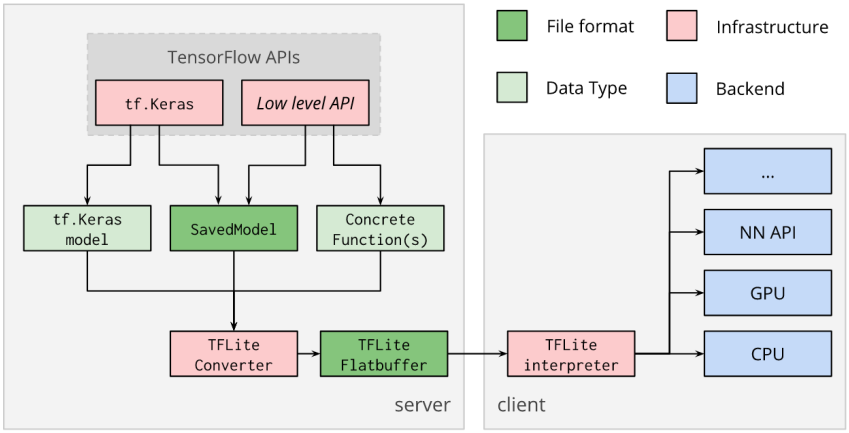


Fig. 22. TensorFlow Lite work flow.

To run these tools some requirements need to be met, firstly the TensorFlow Lite Converter tool needs to be downloaded from its repository[30] then it needs to be compiled using bazel[31]. To simplify this process a Docker container was created that already contains TensorFlow Lite Converter built with Bazel. Docker uses OS-level virtualization to deliver software in packages called containers. Containers are isolated from one another and bundle their software, libraries,

[30] Accessible at `https://github.com/tensorflow/tensorflow/tree/master/tensorflow/lite/toco`
[31] Bazel is a free software tool that allows for the automation of building and testing of software.

and configuration files. Meaning, instead of installing several tools and dependencies in every machine that might be used, we can simply install docker and build the container[32]. Afterward, the only steps needed to convert the model files are to edit the script *tflite_ graph2tflite.sh* with the path to the original model files and execute it, the resulting TensorFlow Lite model will be created in a tflite folder inside the original one.

3.2.4 Trained model performance on microcontrollers

A first prototype of the IoT apparatus was developed using a RPi3. To achieve real time object detection a script that fetches each frame captured by the video camera and analyses it was used. The image inference time in this first prototype was not adequate resulting in very low frame rates for the annotated video (reaching as low as 0.8 frames per second). This low performance was not adequate resulting in a noticeable delay in the real-time object detection and blurry video which adds further difficulties in classifying the objects on the screen.

To improve the performance of the apparatus several models of microcomputers were considered. In the following table 11 we can see a performance benchmarks overview of different devices performing object detection inference with the TensorFlow framework [62].

Table 11. Inference performance results from Jetson Nano, Raspberry Pi 3, Intel Neural Compute Stick 2, and Google Edge TPU Coral Dev Board.

Model	NVIDIA Jetson Nano	RPi 3	RPi 3 + Intel Neural Compute Stick 2	Coral Dev Board
SSD ResNet-18 (960x544)	5 FPS	DNR	DNR	DNR
SSD ResNet-18 (480x272)	16 FPS	DNR	DNR	DNR
SSD ResNet-18 (300x300)	18 FPS	DNR	DNR	DNR
SSD Mobilenet-V2 (960x544)	8 FPS	DNR	1.8 FPS	DNR
SSD Mobilenet-V2 (480x272)	27 FPS	DNR	7 FPS	DNR
SSD Mobilenet-V2 (300x300)	39 FPS	1 FPS	11 FPS	48 FPS

DNR - Did not run.

The did not run results occurred frequently originated from limited memory capacity, due to most of the models having a much higher resolution than the used 300x300. Additionally, unsupported network layers or hardware/software limitations may have been the cause. Nevertheless, seeing as our custom model used the SSD Mobilenet V2 (300x300) architecture we can focus on

[32] Accessible at `https://bitbucket.org/wave-oceanus/docker-tf-odapi/src/master/`.

these results which reveal that the CDB provides the better performance in the same or smaller form factor compared to the other devices. The final IoT prototype will use a CDB as the processing unit for the object detection.

3.3 Summary of the Methodology

This section firstly reported on the user pre-study to understand issues with the existing surveying methods, followed by an overview of the process used to achieve a trained ML model capable of identifying the marine megafauna. The first step to create a model consists of creating a dataset with a significant number of images depicting the megafauna, as reported in table 6. Furthermore, several image transformations were used to artificially increase this dataset size, as seen in figures 16, 17, 18 and 19. Then the model training was described, as well as the tools and hardware used which were both provided by Google's Colaboratory platform. Furthermore, the decision making process for choosing a model architecture was described, resulting in the Mobilenet_v2 being selected as it provides an adequate performance for low powered devices. Finally, the performance of several models and microcontrollers was analyzed, as seen in table 11, influencing the decision on the hardware used for the system.

4 System

In short, the goal of this apparatus is to collect imagery of marine fauna and correctly classify them. Furthermore, after classification, the system requests the user feedback to confirm the automatic classification.

This section describes the apparatus and its components. It can be mainly divided into two parts: (i) the hardware (**Section 4.1**), referring to the physical components of the system and (ii) the software (**Section 4.2**) referring to the data collecting, machine learning, and the mobile app.

4.1 Hardware

The apparatus' hardware consists of three main components: (i) the CDB, which is the microcontroller responsible for collecting the data and detecting the species present in the imagery; (ii) the camera, a Coral Camera Module is connected to the CDB, this camera will be used to gather imagery of the marine fauna; and (iii) the mobile phone which is used to communicate with the apparatus through the mobile app.

Fig. 23. Coral Dev Board with a Coral Camera connected.

A CDB is adequate for this system since it has an onboard Edge TPU co-processor capable of performing 4 trillion operations tera-operations per second. Furthermore, it is very efficient, using only 0.5 watts for each tera-operation per second. Therefore it was integrated into this system seeing as it can execute state-of-the-art mobile vision models in a power-efficient manner and at a high frame rate.

Furthermore, a GPS (Global Positioning System) GP-735 module was attached to the CDB which is used to fetch the location of the apparatus during each detection. This data can then be used to draw maps about the routes species travel or heat maps displaying where there is a higher concentration of individuals.

Fig. 24. GPS receiver GP-735 used to collect location data.

Furthermore, the modularity of the apparatus is displayed in the following figure 25 where we can see the object detection being performed with an infrared Camera Module (Pi NoIR v2) and a pre-trained model provided by TensorFlow called COCO (Common objects in context) SSD (Single Shot MultiBox Detection) MobileNet. It is a model trained with common objects, meaning it can be tested mostly anywhere unlike our trained model which will only detect the predetermined species.

Fig. 25. Real time object detection performed by the apparatus using COCO SSD MobileNet and an infrared camera module.

4.2 Software

This section describes the operation of the system in general, by describing every software component and how they interact with each other. Firstly, system's behaviour is explained through a flowchart (**Section 4.2.1**), then a diagram of the system architecture is described (**Section 4.2.2**). Regarding the software, there are three main components: (i) the database (**Section 4.2.3**), (ii) the scripts running on the apparatus (**Section 4.2.4**) and lastly (iii) the mobile app (**Section 4.2.5**).

4.2.1 Flowchart/Algorithm

This apparatus, in further, will have a simple behavior, the object detection will be performed in real-time and if any object of interest is detected in the frame the unedited picture is saved, furthermore a bounding box and a label will be drawn to identify the marine fauna present, this image will also be saved and its data is stored in the database. Every picture that has marine fauna identified will be added to a list called Unreviewed Detections which are the detections waiting for user review. In a separate process, this list of images will then be displayed in the mobile app and can be further analyzed by a user, they can then confirm or contest the automatic classification performed by the system. Therefore, the detections can be either unreviewed or reviewed depending on whether they received user feedback already. Furthermore, two images are associated with every detection, one is the raw picture taken and the other is the edited image with the bounding box around the objects and a label with its name.

In the following figure 26 we can see the flowchart representing the apparatus' operation on the left referring to the machine learning and data collecting. Also, the mobile app's behavior related to user feedback is displayed on the right.

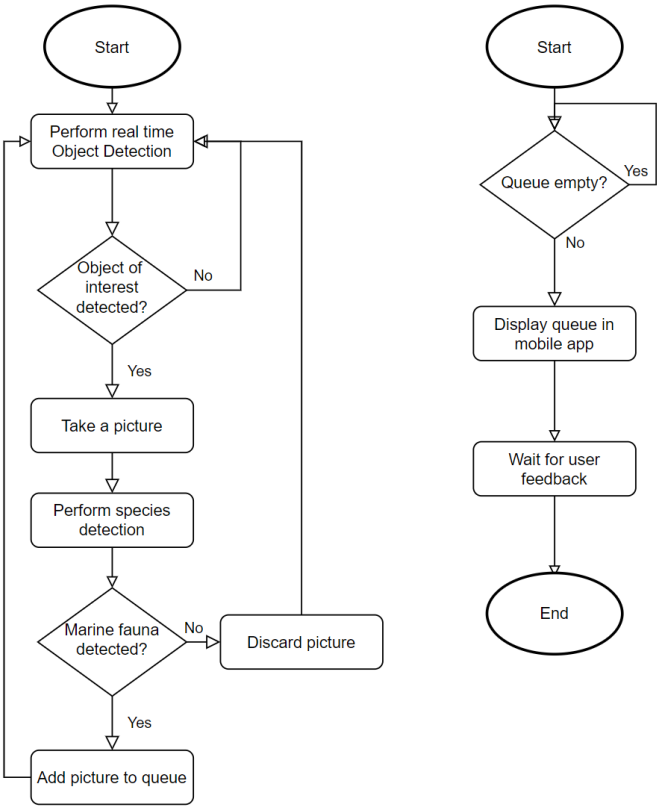


Fig. 26. System Flowchart, apparatus's behaviour on the left and mobile app's behaviour on the right.

4.2.2 System Architecture

As stated previously in **section 4.1**, the apparatus is composed of three components: (i) the CDB microcontroller, responsible for all the processing in the system from the object detection with TensorFlow to the data storing in the database and requesting user feedback through the mobile app; (ii) the camera module, using a Coral Camera Module[33] and the GPS GT-735 assuring the collection of geolocation data of the detections. An overview of the system architecture can be seen in figure 27.

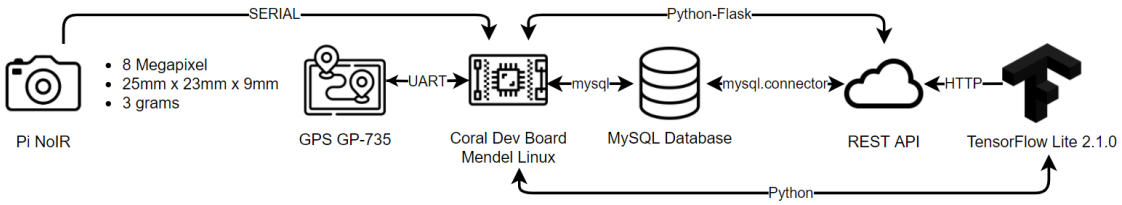


Fig. 27. Apparatus system architecture.

[33]The Coral Camera is a 5MP camera designed for use with the Coral Dev Board, providing input for computer vision use cases.

The 24-pin serial connection between the camera and the CDB uses the MIPI CSI-2 communication protocol, which stands for Mobile Industry Processor Interface Camera Serial Interface 2. Its high performance and low power, alongside low electromagnetic interference, are the fundamental features of this connection. For the GPS module, the Universal asynchronous receiver-transmitter communication protocol is used, the size of the data it supports is much more limited but almost all microcontrollers have dedicated hardware built into their architecture that supports this protocol. Moreover, it only requires two wires to allow for communication between two devices.

All software components communicate with the database through the REST (Representational state transfer) API (Application programming interface). It is developed with Python and the web application framework Flask and enables reading and writing to the database using the library MySQL connector.

The mobile app allows the user to view the automatic detections, the video feed of the apparatus, and the statistics about the machine learning's performance. Furthermore, users can input their feedback about the validity of automatic detection. The following figure 28 shows how the app communicates with the apparatus to make these features work.

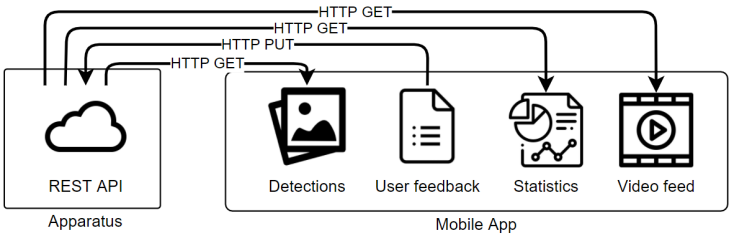


Fig. 28. Mobile app system architecture.

4.2.3 Database

Seeing as the main goal of this system is to collect vast amounts of data automatically a database was structured to store all the information needed. All details about the detections are stored in the database, to accommodate this data a final database with three tables was designed represented in figure 29.

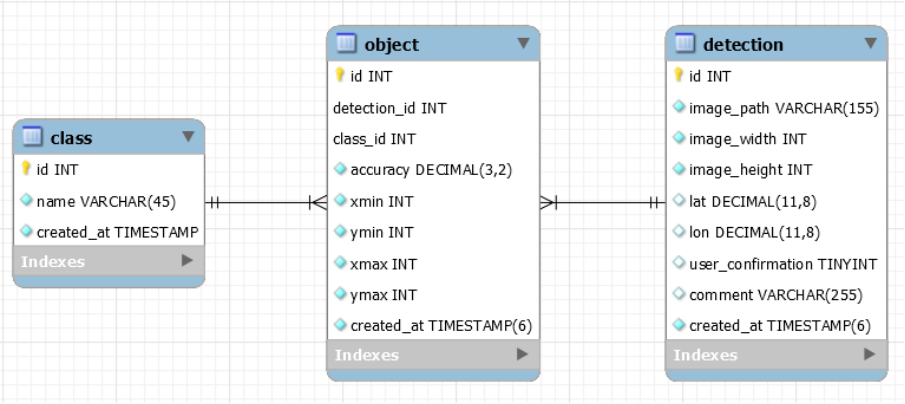


Fig. 29. Database schema.

Firstly, all tables have a primary key "id" which is necessary to distinguish each record in a table and a "created_at" field storing a timestamp of the exact time a record was created.

The "class" table represents each type of object that the machine learning model was trained for, in this case, the five marine megafauna species Mysticeti, Odontoceti, Pinniped, Turtle, and Bird. If a new model is introduced in the system, the new classes will be introduced and the operation will remain the same.

To store information pertaining to the imagery the "detection" table is used. Besides the common fields between all tables mentioned previously, this table contains the following information:

- **image_path:** a file path pointing to the location of the image saved in the internal storage of the CDB;
- **image_width and image_height:** stores the resolution of the imagery, allowing for several cameras with different resolutions to be used;
- **lat and lon:** latitude and longitude representing the geographic coordinates during the detection;
- **user_confirmation:** represents the user feedback about a detection, if the value is NULL then the user has not reviewed this detection else it can be 1 or 0 meaning the user agrees or disagrees with the system's automatic classification;
- **comment:** an optional field where the user can write more in-depth feedback about a certain image.

The previous table only gathers information about the image and user feedback related to detection but the "object" table is the one that relates it to the class, they are associated through

the use of foreign keys[34]. The data stored by this object table is the most relevant for machine learning, being the coordinates of the bounding box which represents the location of the object in the image. The images and object coordinates can be imported in the future to increase the dataset size in the training of a model. The following fields are present in the "object" table:

- **detection_id:** foreign key pointing to the detection in which this object is present;
- **class_id:** foreign key pointing to the class of this object;
- **accuracy:** the score given by TensorFlow, used to filter false positives when the value is below a certain threshold;
- **xmin, ymin, xmax, ymax:** coordinates of the bounding box drawn around the object, these values can be seen in the following figure 30 (note that when working with the OpenCV library and images the top left corner is the point 0,0).

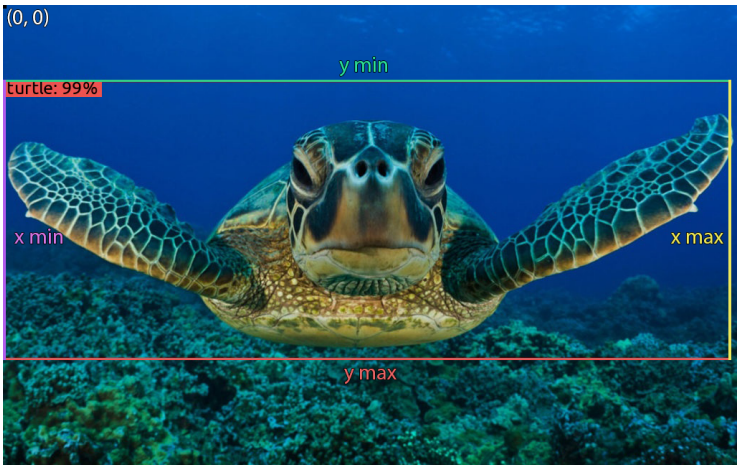


Fig. 30. Coordinates of the bounding box.

The relation between the tables are both one-to-many, in this kind of relationship one record in a table can be associated with one or more records in another table. Meaning, each detection can have one or more objects (e.g. multiple species or several objects of the same species) and a class can be associated with multiple objects.

Storing all this data will be a major contribution to the training of a more robust model. This imagery that is already annotated and reviewed by a user can be used in future machine learning training increasing the dataset size and the performance of the model.

[34] A foreign key is a key used to link two tables together, it is a field (or collection of fields) in one table that refers to the primary key in another table.

4.2.3.1 API

An API was created to enable communication between the multiple software components and the local database in the apparatus. For this purpose, the Python language was used alongside Flask and Flask-RESTful libraries.

Flask is a WSGI (Web Server Gateway Interface)[35] web application framework, its goal is to provide a quick and simple setup to get an application running in a few minutes, with the flexibility to scale up to complex applications. It uses Werkzeug to handle the details of WSGI, which provides the following features:

- a full-featured request object with objects to interact with headers, query arguments, form data, files, and cookies;
- a response object that can wrap other WSGI applications and handle streaming data;
- a routing system for matching URLs to endpoints and generating URLs for endpoints, with an extensible system for capturing variables from URLs;
- HTTP utilities to handle entity tags, cache control, dates, user agents, cookies, files, and more;
- a threaded WSGI server for use while developing applications locally;
- Also, an interactive debugger that allows inspecting stack traces and source code in the browser, allowing the developer to easily identify the cause of errors.

Moreover, the Flask-RESTful is an extension that provides a lightweight abstraction for Flask that works with existing Object-Relational mapping and libraries. It also encourages best practices with minimal setup. The structure of this framework consists mainly of resources and endpoints. Resources are defined similarly to a Python class but with "Resource" as a parameter, moreover they provide easy access to multiple HTTP (Hypertext Transfer Protocol) methods simply by defining them on any resource. For example, there can be a person resource with the get and post methods defined, the get method will return a list of persons from the database while the post method enables the creation of a new entry in the table person. Finally, the endpoints are used to set one or more URLs (Uniform Resource Locator) for each resource, allowing any app to access the data of any resource through its URL. The endpoints can also contain path parameters in the URL allowing data to be sent without a request body. The API was developed with these practices using resources and endpoints to create, read, and update the data in the database with the methods post, get and put respectively.

In total there are twelve resources in the API and they implement the following methods:

[35] Web Server Gateway Interface is a specification that describes how a web server communicates with web applications, and how web applications can be chained together to process one request.

- **Index (/):** this is the root endpoint, its the resource displayed when we access the IP of the API without adding anything ahead;
  - **get:** displays the message "API is running." for troubleshooting purposes.
- **ClassList (/class):** this endpoint is used to manage the class table;
  - **get:** returns a list with all classes present in the database;
  - **post:** inserts a new class in the database by receiving the new class name.
- **ClassByID (/class/<int:id>):** this endpoint is used to fetch the information of a single class;
  - **get:** returns a single class present in the database given its id.
- **DetectionList (/detection):** this endpoint is used to manage the detection table;
  - **get:** returns a list with all detections present in the database;
  - **post:** inserts a new detection in the database when receiving the values image_path, image_width, image_height, latitude and longitude;
  - **put:** used to insert the user feedback (1 if agrees or 0 if not) and their comment in a certain detection.
- **DetectionListUnreviewed (/detection/unreviewed):** this endpoint is used to display detections that still need user feedback;
  - **get:** returns a list with the detections that have NULL value on the user_feedback field.
- **DetectionByID (/detection/<int:id>/<int:raw>):** this endpoint is used to display the images associated with a detection;
  - **get:** displays the detection image given its id and if the parameter "raw" is 0 the original picture is display but for any other number the annotated image is shown with the bounding boxes and labels.
- **DetectionThumbnailByID (/detection/<int:id>/thumbnail):** this endpoint is used to generate and display a thumbnail for a certain detection;
  - **get:** displays a 100x100 pixel detection image given its id.
- **ObjectList (/object):** this endpoint is used to manage the object table;
  - **get:** returns a list with all objects present in the database;

- **post:** inserts a new object in the database when receiving the values detection_id, class_name (this is the value TensorFlow returns and is then used to fetch the class_id when inserting the data), accuracy, xmin, xmax, ymin and ymax.

– **ObjectByID (/object/<int:id>):** this endpoint is used to fetch the information of a single object;

- **get:** returns a single object present in the database given its id.

– **ObjectByDetectionID (/object/detection/<int:id>):** this endpoint is used to gather all the objects contained in a certain detection;

- **get:** returns a list with all objects present in a specific detection given its id.

– **ObjectByClassID (/object/class/<int:id>):** this endpoint is used to gather all the objects of a certain class;

- **get:** returns a list with all objects of a certain class present in the database.

– **NumberOfDetectionsAndAverage (/statistics/numberofDetectionsAndAverage/<int:timeFrame>):** this endpoint is used to calculate the statistics about the number of objects detected and their average accuracy.

- **get:** returns a list with all classes, their number of objects, and the average accuracy, the parameter "timeFrame" can be 0 to fetch all-time stats or any other number representing the past x hours.

Every endpoint created was used in some way throughout the several software components lifecycle, whether it was for testing or production described in the following sections 4.2.4 and 4.2.5.

4.2.4 Scripts

This section describes all the back-end components running in the apparatus and how they interact with each other. There are mainly two components: the object detection program (**Section 4.2.4.1**) and the database (**Section 4.2.3**). Furthermore, since the apparatus is not connected to a screen there is an API that streams the video feed processes by the machine learning through a browser (**Section 4.2.4.2**).

4.2.4.1 Real time object detection

To perform the object detection in real-time a python program was used that is constantly running until it is stopped manually or the apparatus is turned off. There are two main libraries that allow this program to perform object detection for our purpose, which is TensorFlow and OpenCV. The

OpenCV library contains the VideoCapture object providing an easy way to get the connected video camera feed. And TensorFlow is the machine learning framework that was used through the entire process of training the model, therefore using their library to run our model was the most adequate solution. However, this library merely loads a model and runs the inference, hence the output needs to be transformed in a way that is useful for our purpose [63]. In the case of our custom model, it returns only a list of probabilities containing the coordinates, score, and name of the class that was detected thus OpenCV was also used to draw the bounding box and label around the objects using the data resulting from the inference since it has several functions that allow for image manipulation.

A Python class called ObjectDetection was created to handle all the image processing and machine learning inference related to TensorFlow. When initializing an instance of this class the following parameters are required:

- **trained model file path:** a data structure that contains the logic and knowledge of the trained machine learning network;
- **label map file path:** a text file listing all the classes present in the model;
- **threshold:** the minimum accuracy that detection must reach to be saved;
- **useTPU flag:** indicates if the device running this program has a TPU and will use it to achieve a faster inference time.

A model folder can be used as a parameter instead of specifying the model and label map files separately, this method assumes that those files will have the default names detect.tflite and labelmap.txt respectively.

Regarding the image inference, a function called imageInferece was created that receives an OpenCV image and after processing it will return the annotated picture and a JSON (JavaScript Object Notation)[36] object containing the data about it and the objects present in the detection. Therefore to perform a simple image inference an instance of the ObjectDetection class needs to be created and the function imageInference needs to be called and fed an OpenCV image.

As expected this class can only perform object detection on images (see figure 31), meaning we need to adapt this functionality to also work on videos. Since videos are just frames in sequence, we can intercept the video feed and perform the image inference for every frame before displaying it, effectively executing the object detection in any video file or stream.

[36] JSON is a lightweight data-interchange format, easy for machines to parse and generate. It is also easy for humans to read and write.

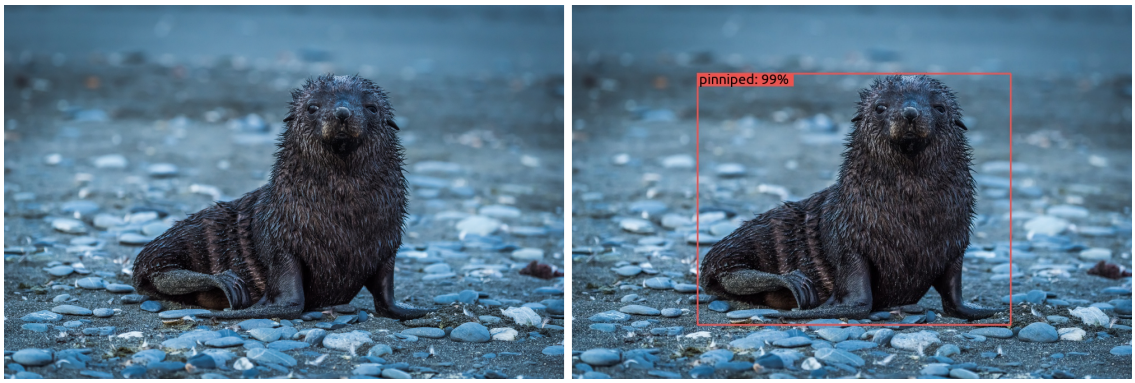


Fig. 31. Original image on the left and annotated on the right

However, this inference requires a lot of processing power which has different negative results for video files or streams. In case of a video file that is 30 frames per second, lets say the device executing the inference is under powered managing to process only 10 frames per second it would take three times the length of the video to finish annotating the entire footage. However this is not a big problem because the length of the video is finite and the resulting annotated video would be the same duration as the original. Although when working with video streams the result needs to be obtained in real time, meaning that for a 30 frames per second the device needs to be capable of processing 30 frames every second, in other words a frame every 33.33 milliseconds. If the device cannot maintain this performance frames will be lost which could contain objects of interest. These factors were considered when choosing a model architecture and device to ensure stable performance. In the following table 12 a benchmark of the inference time between several model architectures and devices.

This benchmark's inference time was calculated using C++, in the case of Python there may be some performance hit due to its overhead because it is more demanding to run. Also, the supporting operations like image loading, saving, or resizing were not taken into account. However, the selected device represented in the last column of the table 12 (CDB) manages to get adequate performance for real-time inference with several model architectures including the one used in the custom trained model which is MobileNet v2 SSD. Seeing as it can perform the inference in an average time of 14 milliseconds, leaving a buffer of almost 20 milliseconds for other operations, in our practical conditions the resulting annotated video stream was able to maintain 30 frames per second. Although this performance varied depending on the temperature of the device, which will throttle the processor's frequency and turn on the cooling fan to avoid overheating, that being said, most of the time it is stable.

Table 12. Inference time (in milliseconds) benchmark between different model architectures and devices. (Source: Edge TPU performance benchmarks)

Model architecture	Desktop CPU[1]	Desktop CPU[1] + Edge TPU	Embedded CPU[2]	Embedded CPU + Edge TPU [3]
DenseNet (224x224)	380	20	1032	25
Inception v4 (299x299)	700	85	3157	102
MobileNet v1 (224x224)	53	2.4	164	2.4
MobileNet v2 (224x224)	51	2.6	122	2.6
MobileNet v1 SSD (224x224)	109	6.5	353	11
MobileNet v2 SSD (224x224)	106	7.2	282	14
ResNet-50 V1 (299x299)	484	49	1763	56
ResNet-50 V2 (299x299)	557	50	1875	59
ResNet-152 V2 (299x299)	1823	128	5499	151
SqueezeNet (224x224)	55	2.1	232	2
VGG16 (224x224)	867	296	4595	343
VGG19 (224x224)	1060	308	5538	357

[1] Desktop CPU: 64-bit Intel(R) Xeon(R) Gold 6154 CPU @ 3.00GHz
[2] Embedded CPU: Quad-core Cortex-A53 @ 1.5GHz
[3] Coral Dev Board

4.2.4.2 Video streaming

Seeing as the apparatus was configured as a headless computer[37] a way to monitor the video feed is helpful. However, connecting a monitor to the apparatus would not be the optimal solution as it would increase the system's power consumption, it would be harder to deploy depending on the size of the monitor, and to check the video feed a user would need to be in close proximity of the device. All these would be inconvenient considering the proposed attachment solution of the device to the mast of a vessel. To solve this problem a remote video streaming feed was implemented, allowing its monitoring through the network either using the mobile app or a web browser to access it.

[37] A headless computer is a computer system or device that has been configured to operate without a monitor (the missing "head"), keyboard, and mouse. A headless system is typically controlled over a network connection.

The implemented solution was based on the **flask-video-streaming**[38] repository developed by Miguel Grinberg which utilizes a Flask API to transmit the video, making it easily accessible remotely through an URL. This repository is easy to deploy seeing as it is consistent alongside the rest of the developed scripts due to it using Flask which is the same framework, written in Python, used by the database API and they share libraries like OpenCV, therefore most of its dependencies are already installed. The OpenCV library is used to handle the video camera drivers and to fetch the frames captured by it. They provide two ways to stream video to the browser, the first is to fetch each frame from the camera and directly display it whilst the other fetches the images from the memory of the device. Originally the streaming was deployed with the first option, meaning it would connect to the camera directly, perform the image inference, and finally display it. However there was a problem with this deployment seeing as there are optimizations developed to ensure efficient operation, the video feed is only fetched when there is an active connection asking to watch it, resulting in no image inference being performed without this stream active. To solve this problem, the second option for streaming was used where it would stream images stored in the memory of the device, and the machine learning script (section 4.2.4.1) was modified to save each frame in the memory overwriting the previous one avoiding excessive usage. With this solution the system works as intended, the object detection is constantly being performed independently, and when there is a request to watch the video feed it will fetch the images stored in the memory and display them as seen in figure 32.

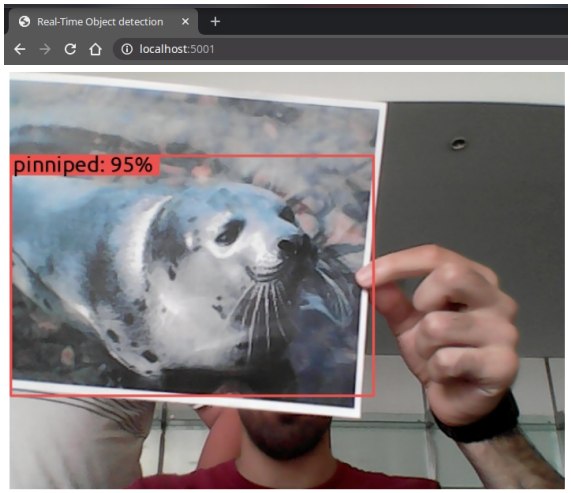


Fig. 32. Real time object detection streaming to a browser.

4.2.4.3 Supervisor

As is evident from the current section 4.2.4 there are three Python scripts to manage, namely the database API, the real-time object detection, and the video streaming. To more easily manage

[38] Accessible at `https://github.com/miguelgrinberg/flask-video-streaming`

them Supervisor was used, which is a client/server system that allows the monitoring and control of processes on UNIX based operating systems. It provides several features that are essential to ensure that all the system's components are operating properly. The structure is simple, any program or command can be set up to be managed by Supervisor with a configuration file where several properties can be defined, most notably:

- **command:** the console command to be executed, in this case, the python commands pointing to each script;
- **autostart:** if set to true, this script will start automatically when Supervisor is started, which is usually when the device is booting up;
- **autorestart:** if set to true, Supervisor will try to restart the script in case of any errors or crashes;
- **stdout_logfile:** a path to a log file in which the output of the program will be written;
- **stderr_logfile:** a path to a log file in where the errors of the program will be written to if any occur.

Hence, Supervisor basically watches these scripts, restart them if they fail, and ensure they start on system boot. Furthermore, by specifying the log files we can quickly find out what is happening at any time during their execution or what errors occur if any. Also, it provides an event notification protocol and a remote procedure call protocol enabling external programs to monitor the current status or even remotely issue commands to control the scripts, for example from a mobile app.

4.2.4.4 WiFi Access Point

During the development of this system both the apparatus and the mobile phone communicated with each other through the network. This was possible seeing as both were connected to the same router's access point, however, when deploying this apparatus in the field as a headless computer(without monitor or peripherals) connecting it to a network would be difficult. Furthermore, a WiFi access point may not be available near the deployment location, meaning the device would function as expected because it does not require an internet connection except there would be no way for the mobile app to access any data at the location.

To solve this issue a utility called RaspAP was installed, enabling the apparatus to also function as a WiFi access point enabling the mobile phone to connect directly to it without any external network equipment being required. The main reason for the selection of this tool is that it provides a quick installer that automatically installs all the Linux packages and configuration files required, and after a quick reboot the access point is operational. Furthermore, it sets-up a dashboard

accessible to any device connected to the access point that gives us control over the relevant services and networking options. Particularly, WiFi's network name and password configuration, network traffic monitoring, and the list of connected clients. This dashboard's main page is represented in the following figure 33.

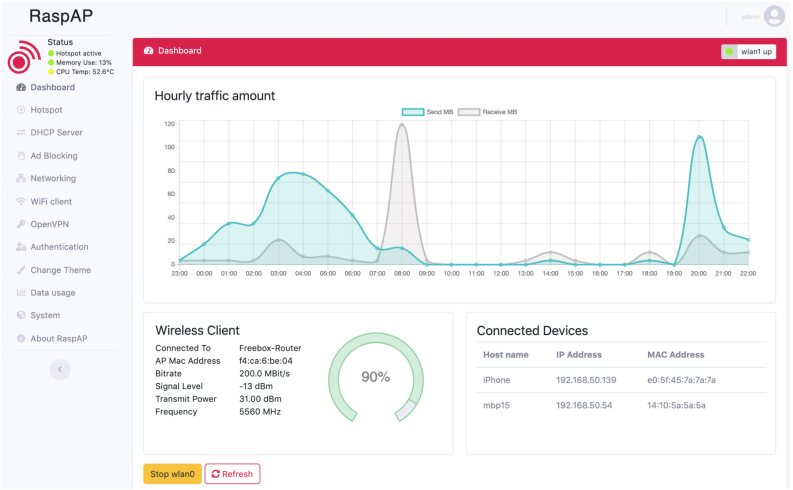


Fig. 33. RaspAP dashboard's homepage displaying information about the hourly traffic and connected devices.

4.2.5 Mobile App

The goal of this mobile app is to provide the user with a way to interact with the data collected and the object detection being performed. Parallel to the object detection, the IoT will store and display the data gathered and images containing objects of interest (in this case marine megafauna) which can be analyzed by the user. This section describes the main three pages of the app: Detections (**Section 4.2.5.1**), Video Feed (**Section 4.2.5.2**) and Statistics (**Section 4.2.5.3**). The app was developed using Flutter for the reasons that it enables a faster development compared to native apps and is cross-platform. The programming language used by flutter is Dart, it is object-oriented and resembles the C language. Besides mobile apps, it can also be used to create web applications, in which case it is transpiled[39] to JavaScript meaning it runs on all web browsers [64].

Before delving into the app's layout, the most important component developed was the data service which is a file containing all the classes related to fetching and writing the data to the database. Every other component of the app uses some class from this file be it for fetching the data about the detection statistics or inputting the user feedback into the database. When an instance of this class is initialized the first methods to be executed take care of setting the IPs and ports for the data video streaming APIs. Besides this, the service's main workload is to get

[39]Transpile, short for transcompile, means to compile (source code) by translating from one source programming language to another, producing translated source code in the other language.

the data from certain API calls and translate them into custom Dart objects seeing as working with the raw data returned by the Future object is much harder [65]. Moreover, three classes were created in this data service to represent the data fetched by the app. These classes are: **(i)** *UnreviewdDetections*, represents the detections for the user to review; **(ii)** *Object*, stores in-depth information (species, number of objects, and average accuracy) about a selected detection; **(iii)** *NumberOfDetectionsAndAverage*, stores the statistics regarding the number of detections of each specie and average accuracy.

4.2.5.1 Detections

When launching the app, the detection page is the first to be displayed. This page starts by trying to connect to the database, this is indicated by showing a loading animation and a message saying "Fetching detections". To achieve this animation a CircularProgressIndicator class was used alongside a Future class, both provided included in Flutter. The future class is used to represent a potential value, or error, that will be available at some time in the future. Receivers of a Future can register callbacks that handle the value or error once it is available, meaning that while the data is being fetched a loading spinner animation is shown provided by the CircularProgressIndicator. If the connection to the database fails an error message is displayed and it will be retried when the page is reloaded (by switching pages or reopening the app). Both of the cases mentioned can be seen in the following figure 34.

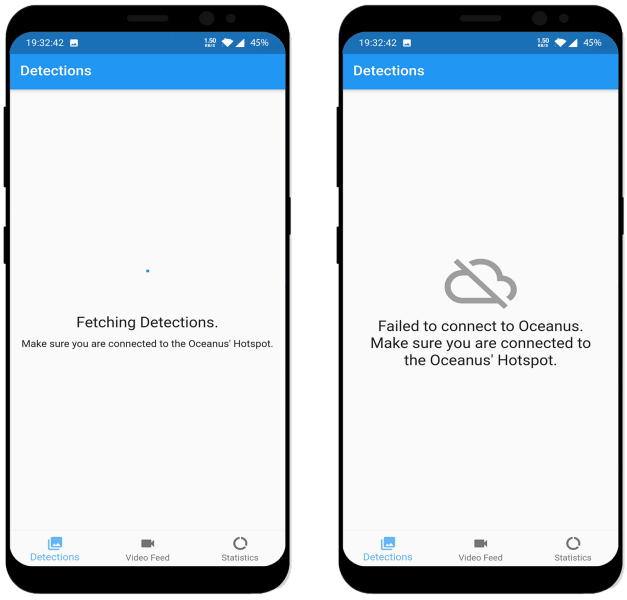


Fig. 34. Detections page trying to fetch data on the left and failed connection on the right.

An image grid containing all the unreviewed detections will be displayed if the data is successfully fetched, this list will then be used to fill an image grid view. To fetch this data two API calls are used, it starts by loading the unreviewed detections data followed by the detections thumbnail

for each of them. Thumbnails (100 by 100 pixels) of the saved images are used in this grid view otherwise the performance would deteriorate by loading all the full-sized imagery. Each individual image will be loaded using the same strategy involving futures and loading animations allowing us to visualize the progress of the download. In the following figure 35 we can see this grid view being loaded.

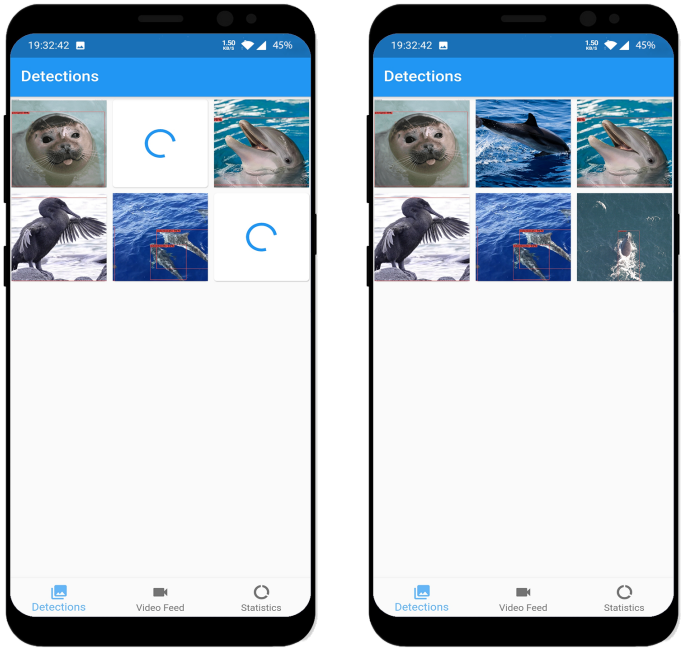


Fig. 35. Detections page loading some images on the left and fully loaded on the right.

By clicking any of the images in the detections a new page will pop up displaying more detailed info about the selected detection, for instance, the number of objects and their labels and the average accuracy of the detections. This page fetches data from two API calls using the selected detection id, first the objects detected and their accuracy and then the full-sized annotated image of the detection. In case that the bounding box is too small to be properly read, this page uses the Photo View widget to display the picture which has a zoom in/out feature by using a pinch gesture, as seen in figure 36 allowing the user to thoroughly analyze it. With all this information the user can then submit their feedback concerning the validity of the automatic classification. There is also an optional comment that can be written by the user to provide further reasoning for their response. This feedback is written into the database with yet another API call, using the put method of the detection endpoint. The user can input their feedback of the images at any time, they are stored locally and will be available until the data is migrated to a remote server for further analysis or model training.

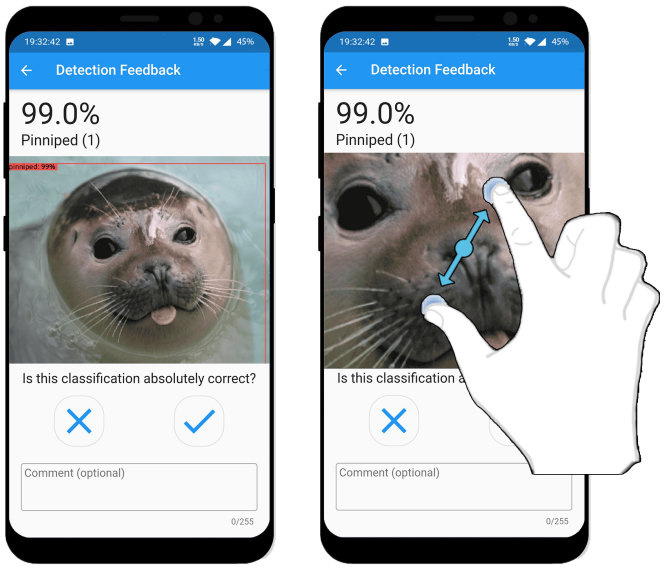


Fig. 36. Detection Feedback page allowing the user to input their feedback and zoom in/out feature shown on the right.

4.2.5.2 Video Feed

This page allows the user to monitor the status of the apparatus' video, by checking if it is running and if the object detection is working properly. The video stream displayed here is provided by the streaming script described in section 4.2.4.2, seeing as it simply sends the video feed to a browser this page consists of a web view with the URL pointing to it. The page's CSS (Cascading Style Sheets) was edited to fit the video to the size of the page while maintaining the aspect ratio of the original resolution.

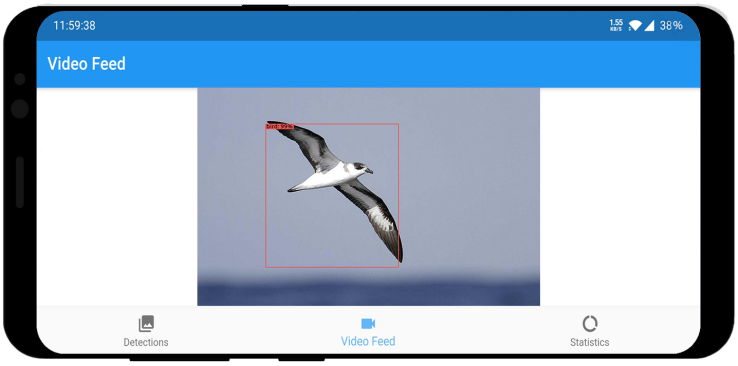


Fig. 37. Video Feed page displaying the apparatus' current video feed performing object detection.

A list of detections of objects of interest displayed here can then be seen on the first page of the app, however, as mentioned previously the user does not need to monitor this stream seeing as this processing is constantly executing in the background.

4.2.5.3 Statistics

The statistics page allows the user to get an overview of the data stored locally on the apparatus and can be filtered by time. The data displayed on this page relates to the number of objects detected and the average accuracy of each class in a certain time frame. This data is fetched through an API call with has a "timeFrame" parameter in the URL which filters it, see section 4.2.3.1.

Displayed in the following figure 4.2.5.3, the data is shown as a list of classes in which each element of the list has the following elements: the class name followed by the number of occurrences in parenthesis, an average of all the occurrences score on the right and a progress bar bellow that helps to visualize this accuracy. Moreover, when there is no data for a specific time frame a message will be displayed as seen on the right of the figure.

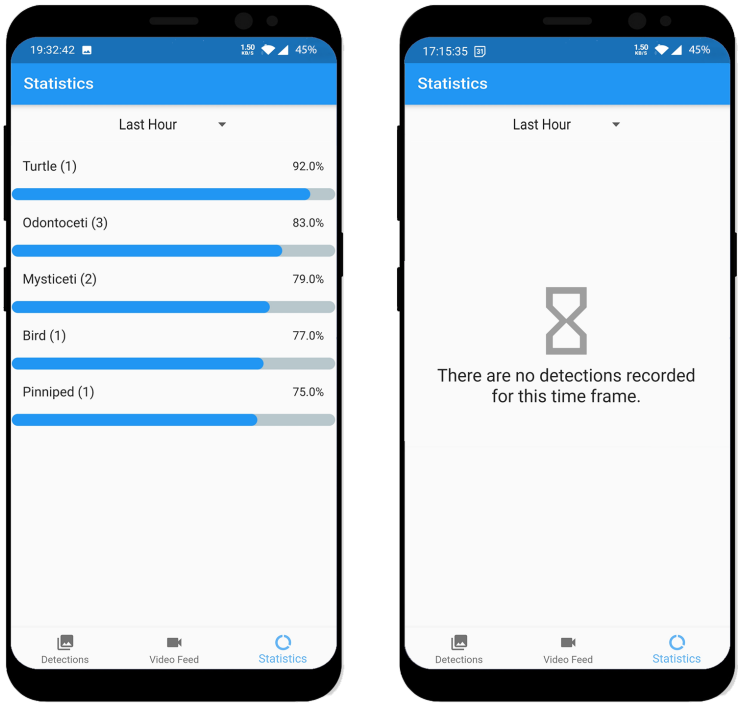


Fig. 38. Page displaying statistics about the detections stored on the apparatus on the left and no data in a certain time frame on the right.

4.3 Summary of the System

This section reported on the components present on the developed system. Firstly, the hardware consisting of four main parts which are the CDB, the camera, the GPS module and a power bank. Then a description of the software's operation, as seen in figure 26, followed by an in depth explanation all the different elements and how they communicate with each other. The main

components are the database, the python scripts and the mobile app. The database was developed using MySQL and is used to store all the data collected by the system. The python scripts are responsible for analysing the camera feed and executing the ML on it using TensorFlow. And the mobile app, developed with Flutter, allows the user to interact remotely with the system seeing as it has no peripherals, namely a keyboard or screen. All these parts work together to identify marine megafauna and gather data about them automatically.

5 Experiments

To tackle the challenge in detecting, classifying, and predicting the marine species, several performed experiments are described in this manuscript in the sections below. Section 5.1 describes the research setting, outlining the location and sample size, while Section 5.2 provides an insight into the study protocol in obtaining the data and testing the research questions.

5.1 Sample

Studies were conducted in Madeira island mostly in a lab setting, however, they aimed to simulate the apparatus' deployment conditions. For some of the experiments the dolphin footage gathered was recorded by mobile phone during a whale watching trip in Madeira, however the remaining footage was obtained online from unknown locations and appears to be gathered from sea vessels and UAVs. Regarding the Human versus Machine experiment (**Section 5.2.2**), a total of 15 master students in computer engineering and interactive media design were participants. Seeing as the specie classification was not a concern in this experiment no marine biology expertise was required for this experiment. Furthermore, regarding the User Validation study (**Section 5.2.1**) a total of 9 testers were invited to participate, 7 of them had no marine biology background and were also master students of computer engineering. However, to gather valuable feedback two of the invited participants are experts in the field, both having finished their PhDs and are currently researchers in the field of marine ecology. One of the experts has participated in more than 15 international projects and is interested in studying the marine ecology of island habitats, in particular in biogeography, population and trophic ecology, telemetry, and habitat use in cetaceans. While the other expert is a team member in 3 international projects and 2 national projects (Portuguese), also a co-founder of the MONICET web-based monitoring platform which aims to gather vast amounts of data gathered in whale watching trips and convert it into valuable information for the public.

5.2 Study Protocol

To answer the aforementioned research questions, several experiments are described related to the user experience (**Section 5.2.1**) and the machine learning's performance (**Section 5.2.2**).

5.2.1 User validation

To understand the practicality of such a system user studies will be performed, mainly on usability and user experience, outlining the pros and cons of using such a system in real-time, on-board the sea vessels. User studies aim to answer "**[RQ2.] To which extent the proposed apparatus can support the marine biologist in counting the species?**" by analyzing the user's interactions

with the system. For this study, users were asked to sit at a desk where the apparatus and mobile app were set up. The camera was pointed at the user and a monitor was displaying the video feed with the annotated objects allowing the user to better understand what is being classified. Test to be performed will give users an understanding of the operation of the system as a whole, how the detections are performed and stored for further inspection.

Invited users have explained the operation of the apparatus and some of its goals, afterwards, they were asked to start the test. The first step of the test is to show the apparatus' camera any photo of the marine megafauna provided to them and watch the machine learning classification occurring in real-time through the video stream, here users tested different photos in different lighting conditions and angles and observed the annotated objects on the screen alongside the accuracy score. User causing a detection is represented in figure 39, meaning this data will be stored in the database. The next step was to browse the object detections saved utilizing the mobile app, at this moment users were asked to select a couple of detections to analyze and report their feedback on the classification (if it was correct or not). Finally, users were asked to explore the statistics page of the app and filter the data displayed.

Fig. 39. User holding imagery of the marine mega fauna in front of the apparatus.

Eventually, after the tests were concluded, users were asked to fill a SUS (System Usability Scale) questionnaire related to the system tested. SUS is a reliable tool for measuring the usability of the system, it consists of 10 statements where the user rates each of them on a scale ranging from strongly disagree to strongly agree. It has become an industry-standard due to being easy to administer to participants and having reliable results even on small sample sizes [66]. To interpret the scores in the SUS questionnaire for each participant we first sum the score for all odd-numbered questions and then subtract 5, and we subtract from 25 the sum of the score for the even-numbered questions. Then we add these two values and multiply them 2.5 resulting in the SUS score from 0 to 100. Although the score ranges from 0 to 100 it is not a percentage, the average SUS score is 68 therefore anything above it is good, furthermore scores above 80,3 are excelent [67].

5.2.2 Human versus Machine

The system developed in this manuscript aims to complement marine biologist's visual surveys, therefore to verify how the object detection may perform during these surveys a comparison test was performed. This test aims to answer "**[RQ3.] How accurate is the software for classification prediction and counting the marine taxa?**" to compare the detection rate of the machine against that of a human and assess if this automatic system can be adequate for surveys. A user test was performed by playing the same collected video footage to human observers and comparing it against and previously collected data by the machine. The human analysis was performed by computer engineering master students who have not watched the gathered video footage previously. Every subject had a stopwatch and counted how many seconds they observe any wildlife in the video footage. Performing this experiment was possible in a classroom where a projector was set up to allow a simultaneous test for every subject where every video was played in succession, while the individual stopwatch times were reported for each one.

Regarding the machine, a video object detection Python script was used to perform the inference for every frame of the video while classifying and counting the number of objects using the custom trained model[40]. Upon the completion of this inference, a list of objects and the number of frames they were detected in was returned. This list could then be used to calculate the number of seconds each specie was detected as well as the false positives seeing as each video only has one specie and multiple could be detected.

Table 13. Sources of the collected test footage and their duration.

#	Marine Megafauna	Testing Footage Source	Duration	Perc.
1	Sea Turtles *(Chelonioidea)*	https://youtu.be/36IquODDQCk (1:01-2:08) https://youtu.be/q8RGB1QDZc0 (0:05-1:06) https://youtu.be/Vx0-rzZQPag (0:05-1:06)	189s	0.08
2	Sea Birds *(Charafdriiformes)*	https://youtu.be/EwPrXOtBoVg https://youtu.be/0POWeKfyN2E https://youtu.be/ah27TO-D5Qk	280s	0.11
3	Toothed Whales, Dolphins *(Odontoceti)*	https://youtu.be/W-7zxhBr7Zo https://youtu.be/MrEB_CxgctE https://youtu.be/PDPLqI818Sc	802s	0.33
4	Baleen Whales *(Mysticeti)*	https://youtu.be/EMybn5NtbtQ https://youtu.be/yCqxJfuthls https://youtu.be/LvF-lPxsi9I	604s	0.25
5	Seals *(Pinnipedia)*	https://youtu.be/vwnXxQHf454 https://youtu.be/KCZg9DSIwX8 https://youtu.be/d8BLbfB2Oas	566s	0.23
TOTAL Testing Footage			**2441s**	

[40]Example resulting annotated videos accessible at `https://youtube.com/playlist?list=PLJX-SO7nrPa5rN0wZfU1s9Khkodg6M7i4`

A total of approximately 41 minutes of video footage was gathered across the five megafaunas (Odontoceti, Mysticeti, Pinniped, Sea birds, Sea turtles), split into three videos for each marine megafauna adding up to fifteen as mentioned in table 13. Each video contains only one of them allowing easier detections for the testers, focusing only on wildlife presence on the screen instead of correctly classifying them.

Most the footage aims to somewhat simulate deployment conditions, except for the turtle footage which could only be found from an underwater perspective, therefore why it is video collected from a sea vessel as is displayed in figure 40.

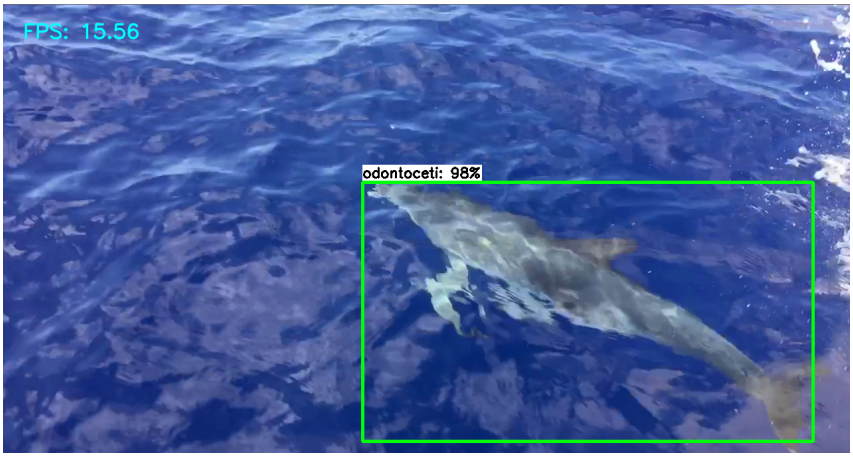


Fig. 40. Footage gathered from a sea vessel displaying a dolphin swimming automatically annotated.

5.3 Summary of the Experiments

This section reported on the background and experience of the experiments' participants, as well as the two types of experiments performed to assess the performance and usability of the system. The first experiment will focus solely on the ML's performance compared to the humans. This is achieved by having humans and the ML analyzing the same videos and essentially counting how many seconds any marine megafauna is present on screen. And the final experiment will use the SUS to evaluate the user experience and the system's usability.

6 Results

In this section, the results of the two studies performed will be presented. Firstly, results of the Human versus Machine video object detection (**Section 6.1**) and subsequently the user validation through system usability scale study of the system (**Section 6.2**).

6.1 Human versus Machine

The following figure 41 displays how many seconds each marine megafauna was detected in the videos by the humans and the machine. This data was gathered from the experiment described in section 5.2.2 where afterward the standard deviation and average total time of marine mega fauna detection by the subjects was computed. Subsequently, results are compared to the machine.

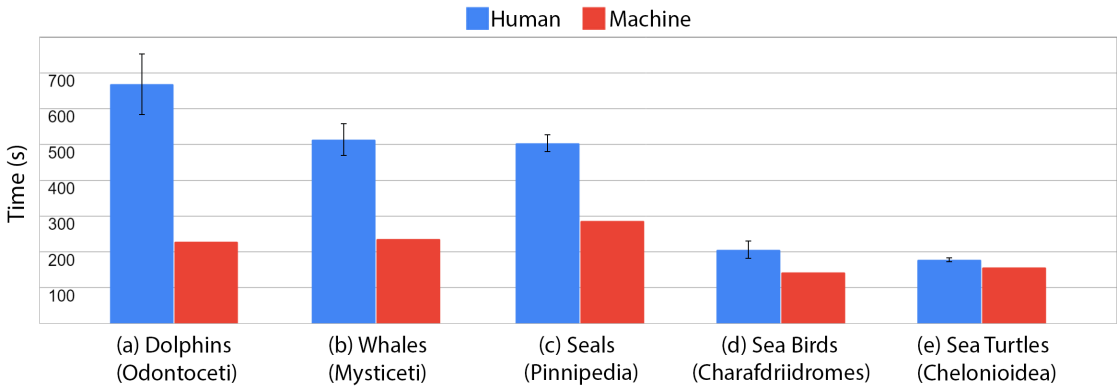


Fig. 41. Human versus Machine experiment: Detection time comparison of the marine mega fauna in seconds.

As is represented in the graph, the ML could detect the marine megafauna averaging about 51% of the time, compared to the humans. The biggest difference is present in the classes Dolphins, Whales and Seals which is due to the animals being submersed during most of the videos making it difficult for the machine to classify their distorted silhouette. This issue is diminished when looking at the other classes, the Birds which were mostly seen in the sky and the Sea Turtles being recorded only underwater avoiding the aforementioned distortion.

6.2 User Validation

Besides the first two questions, this questionnaire consists of a series of statements where the user will rate each one using a five-level Likert Scale ranging from **Strongly disagree (1)** to **Strongly agree (5)**. The first three questions aim to gather data about the user demographics

(age, gender) and their familiarity with technology. The following figures present the results of the questionnaire. Firstly, the questionnaire started with three questions about the users, which are the following:

– **[UQ1.] Age.**

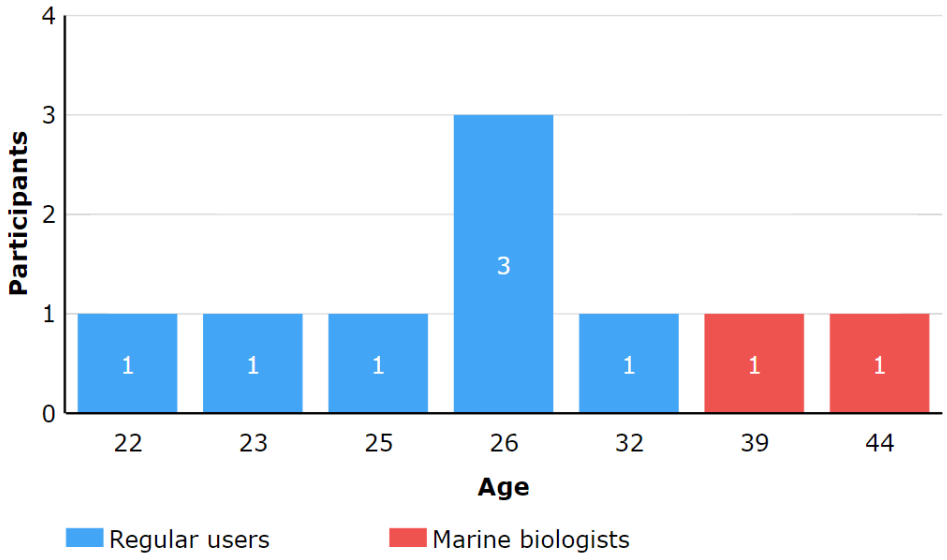


Fig. 42. Age distribution of the participants.

– **[UQ2.] Gender.**

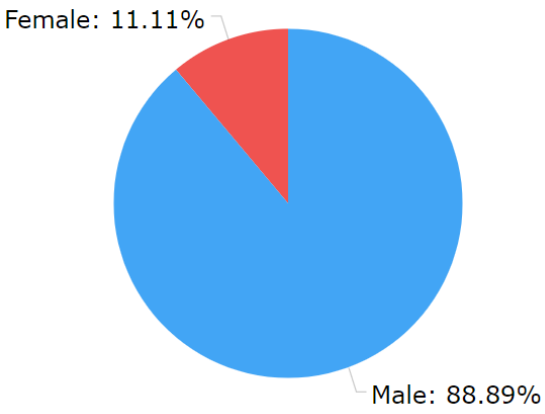


Fig. 43. Gender distribution of the participants.

– **[UQ3.] I am a tech-savvy person.**

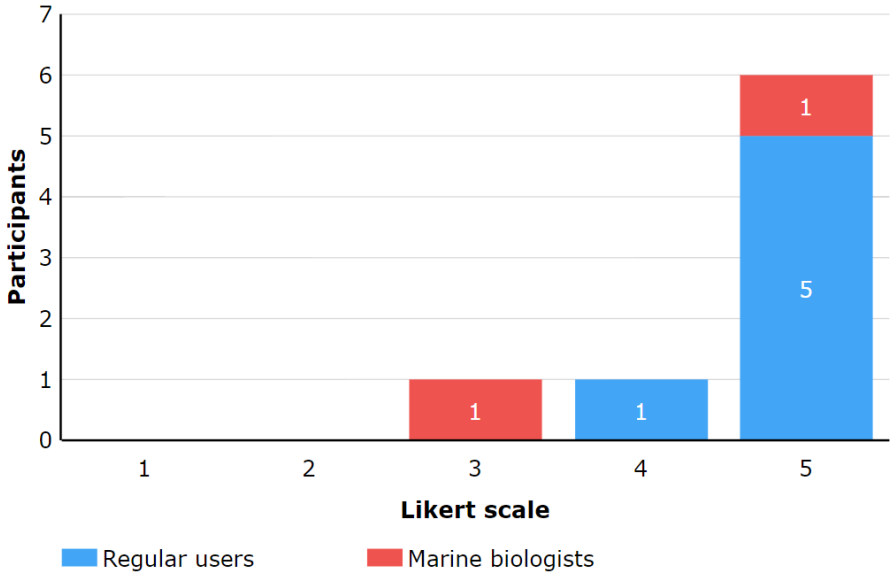


Fig. 44. Technological proficiency of the participants.

Following ten questions are related to the system usability scale, namely:

– **[SUSQ1.] I would like to use this system frequently**

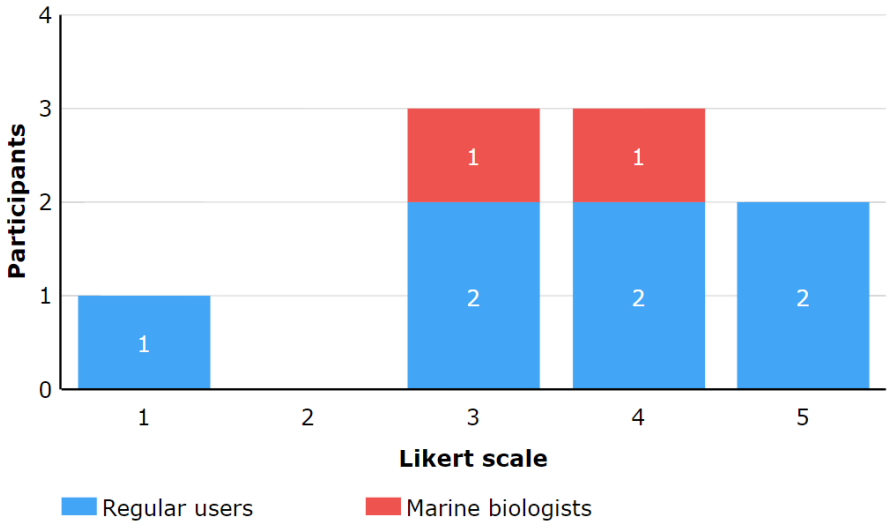


Fig. 45. System usability scale question 1 results.

– **[SUSQ2.] I found the system unnecessarily complex**

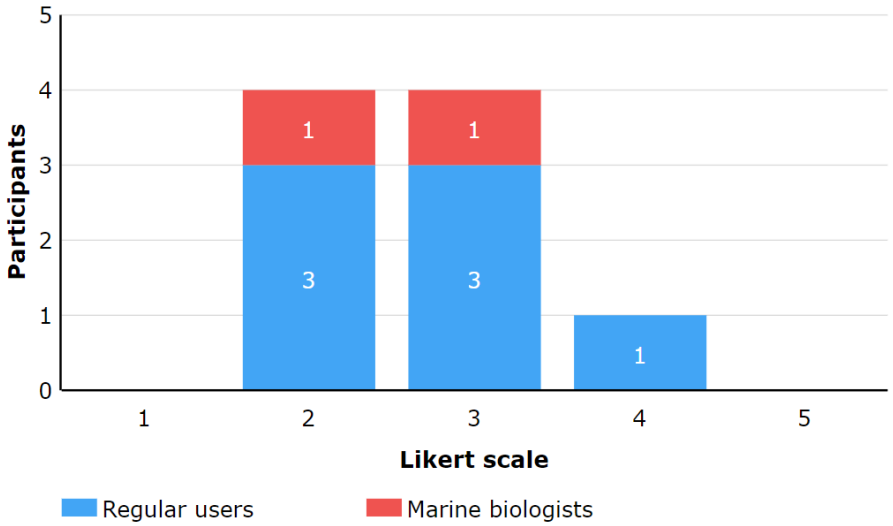


Fig. 46. System usability scale question 2 results.

– **[SUSQ3.] I thought the system was easy to use**

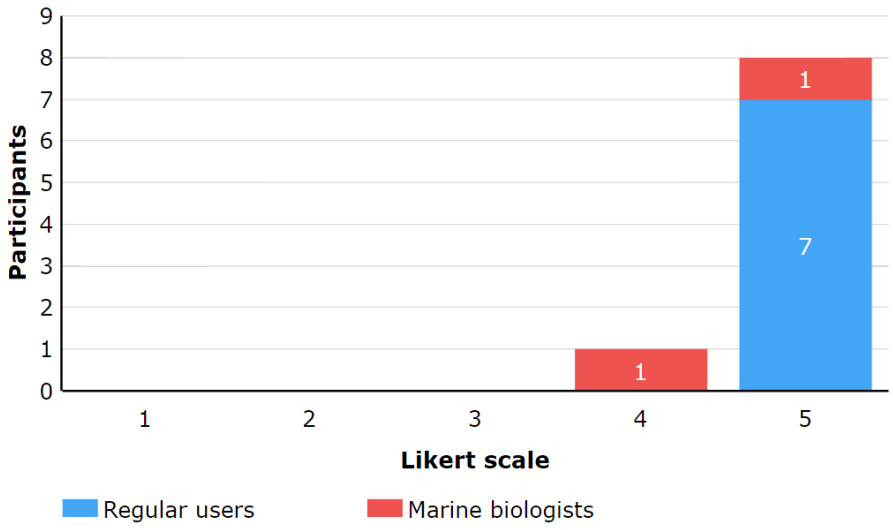


Fig. 47. System usability scale question 3 results.

– **[SUSQ4.] I think that I would need the support of a technical person to be able to use this system**

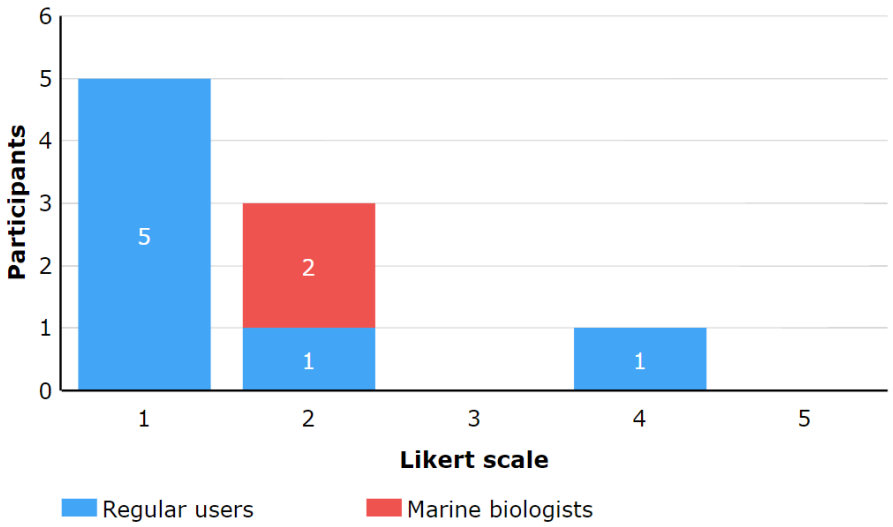


Fig. 48. System usability scale question 4 results.

– **[SUSQ5.] I found the various functions in this system were well integrated**

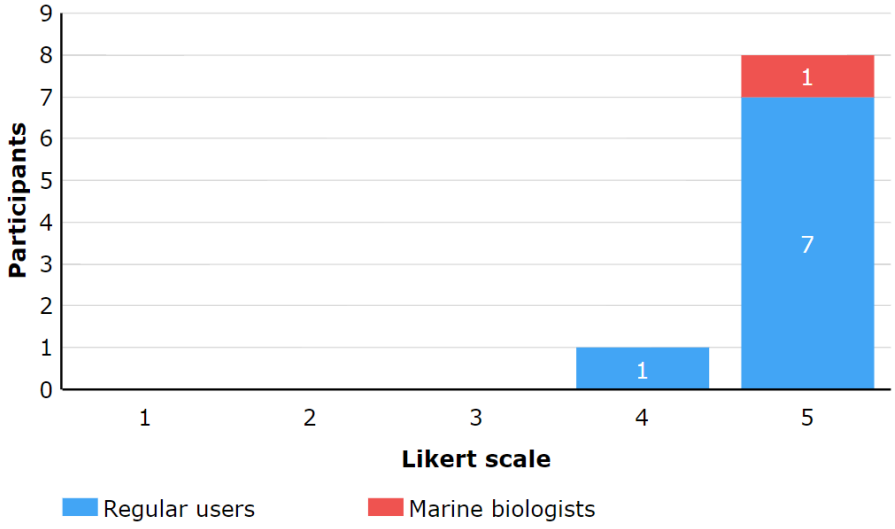


Fig. 49. System usability scale question 5 results.

– **[SUSQ6.] I believe there was too much inconsistency in this system**

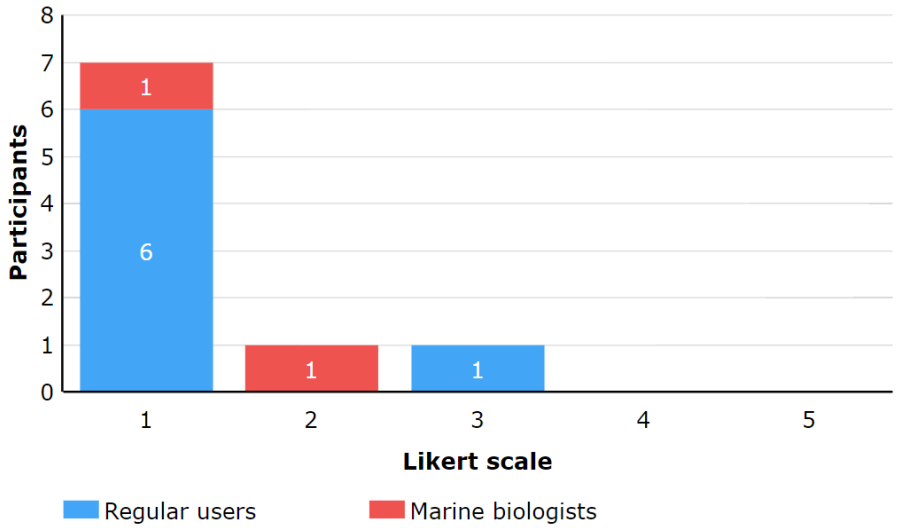


Fig. 50. System usability scale question 6 results.

– **[SUSQ7.] I think that most people would learn to use this system quickly**

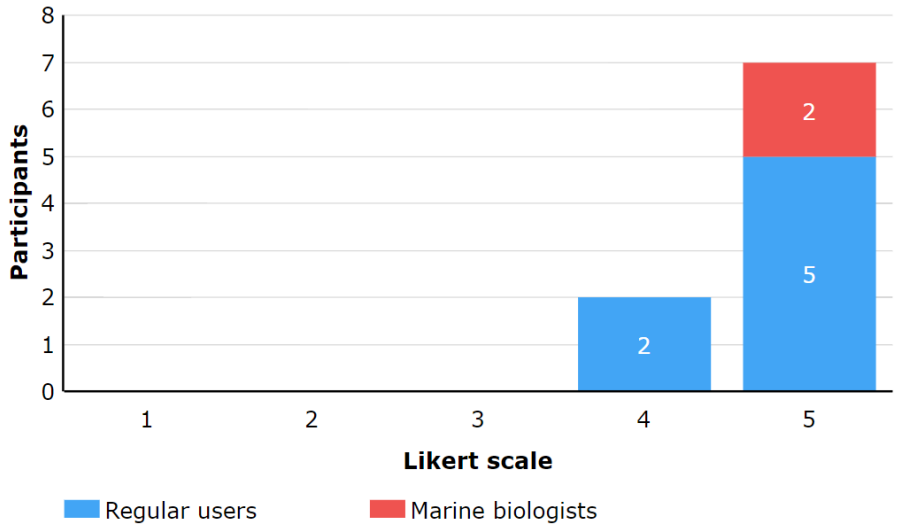


Fig. 51. System usability scale question 7 results.

– **[SUSQ8.] I found the system very inconvenient to use**

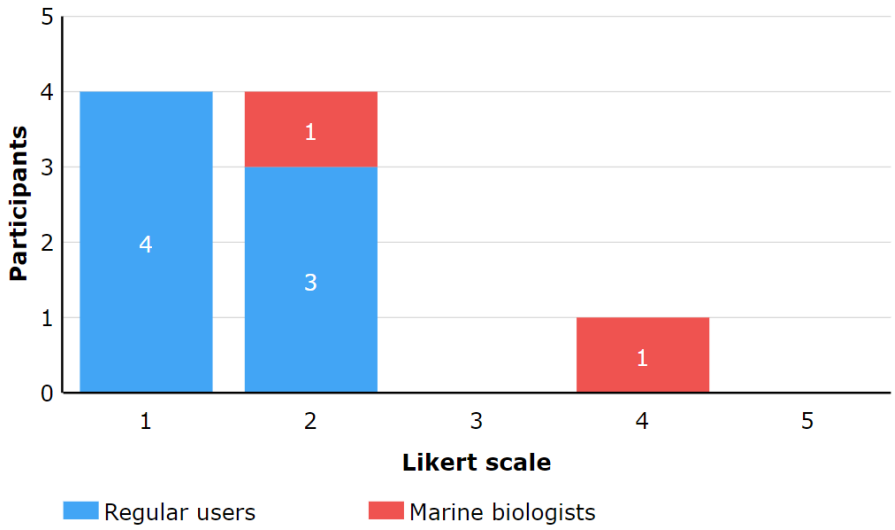


Fig. 52. System usability scale question 8 results.

– **[SUSQ9.] I felt very confident operating the system**

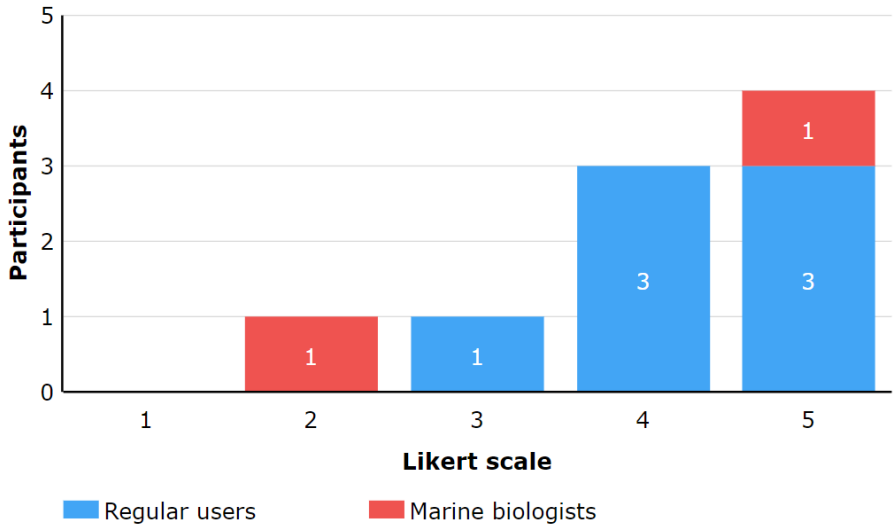


Fig. 53. System usability scale question 9 results.

– **[SUSQ10.] I needed to learn a lot of things before I could use this system**

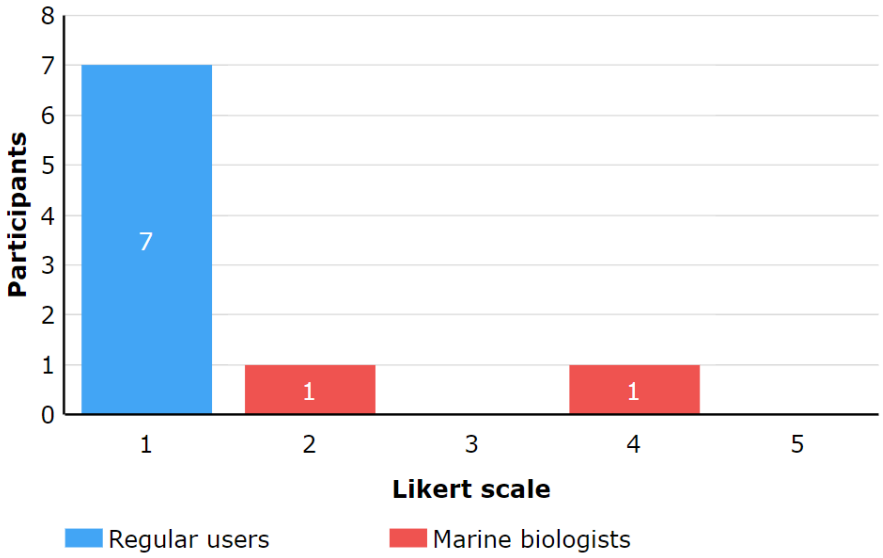


Fig. 54. System usability scale question 10 results.

Considering the results from the questionnaire, the SUS score was calculated for all participants and averaged 83,33. According to SUS guidelines, a score between 68 and 80,3 is good and anything above that is excellent, therefore this system is very promising in terms of user experience. However, considering only the marine biologists questionnaires the resulting score is 72,5.

6.3 Summary of the Results

The results reported in this section provide valuable information concerning the performance and usability of the developed system. Regarding the Human vs Machine experiment, as it can be assessed in the figure 41 the ML could not perform at the same level as humans in all situations, however, it can detect and classify the marine megafauna automatically without any user input. Furthermore, the results from the SUS questionnaire show that the developed system can be used by people of different areas without much problems.

7 Discussion

This section will provide insight into the results gathered throughout the manuscript in findings (**Section 7.1**), along with the limitations of the studies (**Section 7.2**), in addition to future work (**Section 7.3**) and conclusions (**Section 8**).

7.1 Finding Analysis

Regarding the research questions posed in this manuscript, the following findings were obtained when analyzing the results and methodology.

(i) **[RQ1.] How to create low-cost aquatic IoT apparatus for assessing marine biodiversity using aerial imagery?**

As mentioned in the state of art, there are a plethora of solutions available for aerial assessment and multispectral imagery gathering however they mostly remain proprietary and at a high cost. The solution proposed in this manuscript is in essence a modular low-cost system capable of performing visual surveys automatically. Although it was developed with a CDB and a Coral Camera costing around 140 €, these components can be exchanged with a RPi 3 and any compatible camera module which costs around 70 €. The performance would be lower, however, for surveys where there is no sudden movement this solution is adequate. However, as described in this manuscript, for marine megafauna surveys a higher performance was required due to the rapid movements of the species in the video, to achieve adequate performance for this a CDB was necessary to achieving faster inference times. Sometimes the megafauna was only present in the video for a few hundreds of milliseconds which would be impossible to detect with the performance of the former solution. That being said, the costs mentioned only cover the MCU and the camera module seeing as a custom case was developed with medium-density fibreboard and a laser cutter which were available to us. Nonetheless, most of the MCU have mounting holes allowing for easy development of such custom cases. Furthermore, instead of relying on costly aerial survey solutions (e.g. UAVs), the proposed solution can be cheaply mounted in a high vantage point (the sea vessel's mast).

(ii) **[RQ2.] To which extent the proposed apparatus can support the marine biologist in counting the species?**

Given the results presented in section 6.2, the calculated SUS score of 83,3 indicates that the system usability is excellent in general and is very promising in terms of user experience. The system was very simple to use, as reported by the testers and allowed them to feel more confident in the predictions when classifying the marine megafauna. However, when considering only the marine biologists' questionnaires the resulting score was 72,5. Although

it is still above average, they provided some insight into what would make the system even more practical. Firstly, it can only classify the 5 marine megafauna as of now which is not very detailed and during surveys they observers try to classify the species as in-depth as possible. One even suggested that there is a need for an individual identification system, in which the apparatus could identify each individual through their dorsal fin's unique pattern. This would allow for better abundance surveys seeing as the system can omit duplicate counts for the same individual (e.g. in 600 photos there were 10 individuals identified), whereas the current system will count every detection as an individual. Also, the current solution has a stationary camera that can be used to record only one side of the sea vessel, another suggestion was to develop a rotating attachment that could capture imagery in a 360° view.

(iii) **[RQ3.] How accurate is the software for classification prediction and counting the marine taxa?**

Regarding the system's classification performance in section 6.1, in normal conditions, there are still improvements to be made as it could only detect the marine megafauna about 51% of the time compared to that of humans, as seen in figure 41. However, seeing as this is an independent system it is a great complement to human visual surveys due to it continuously assessing the marine megafauna automatically. It can also provide extra data when the observers are not available (e.g. taking notes) as suggested by one of the participants in the tests, additionally, the system helps maintain consistency between trips with different observers. Moreover, in regular whale watching trips where there are no expert observers, the system can be deployed by any user without technical knowledge and automatically collect vast amounts of data.

7.2 Limitations

Regarding machine learning there are a lot of improvements that can be achieved, firstly the dataset size is comparatively small (around 20 000 images after augmentation) looking at other models available online (the Common Objects in Context dataset contains more than 200 000 annotated images). Increasing the dataset would be the first in achieving a more accurate classification, which the system itself is working towards seeing as the images are being stored for future training of the model. Additionally, due to the training hardware accessibility constraints, the model could only be trained for a maximum of 12 hours, by increasing the training time can greatly increase the classification accuracy. Another problem with the trained model is that it can only identify the five marine megafauna. Seeing as it should be recording only the ocean it is adequate, however during testing humans were sometimes classified as Pinnipeds or Birds resulting in some false positives. Also, the field of view of the system is much narrower than that of a human observer and is of a low resolution making it power efficient at the cost of the classification's accuracy. Furthermore, even though the deployment of the apparatus is very simple (simply securing the apparatus and

providing power is enough) installing all the necessary software components will be difficult if the user does not have some experience with software development.

Concerning the performed experiments, although the human versus machine experiment used videos to simulate an expected deployment view it does not provide insight into problems that might occur only in real deployment related to the environment. The video footage gathered was captured in good weather conditions which facilitates the classification, considering other possible weather conditions the performance of the system might drastically worsen.

7.3 Future work

As mention in the previous sections 7.1 and 7.2, some suggestions were provided by experts that participated in the user test. The main goal of future works should be to achieve specific species classification instead of just the five megafaunas proposed in this manuscript. An even more difficult task would be to identify not only the species but also the individuals, allowing for an easier abundance survey seeing as the system would not have duplicate counts of the same individual, although this task might be too hard to perform from an aerial view since the way to identify such individuals is through their dorsal fin's unique pattern. However, a new component could be added to the mobile app where the user could take a photo of any dorsal fin and the system would catalog it. Furthermore, with access to training hardware without time constraints, an effort should be put into training a more robust model until reaching a point where further training would only provide negligible accuracy improvements. Incidentally, increasing the dataset size would greatly benefit the classification's performance yielding higher accuracy scores and reducing the number of false positives, one way this can be achieved is by using the current solution to gather more imagery. Regarding the hardware of the system, as suggested by the experts, a way to improve the field of view should be looked at whether it be a rotation mechanism, multiple camera angles, or a 360º camera.

Concerning the mobile app, a dashboard would be a great addition to allow for a higher degree of control over the system. The current implementation only allows the users to interact with the data stored and watch the video feed, more controls can be implemented such as a power on/off allowing the user to remotely shut down the device temporarily to save power. Also, for more advanced users, a page displaying the system usage of the CPU, memory and disk could be created.

8 Conclusions

Proposed IoT device throughout this dissertation is a first step in applying the real-time classification of the spotted marine species. Conversely, proposed system is primarily designed to aid marine biologists (instead of exchanging them) in collecting the data about marine megafauna during their visual surveys. It achieves this goal by providing the assisting tools which contribute to the data gathering and validation based on real time imagery. Furthermore, this system contributes to future works allowing the automatic imagery collection, allowing for the creation of big data. Besides experts, customers, and service providers of marine leisure activities, for instance, whale watching, diving, or boat rentals, can also use such system as a passive image sensor, as they do not require an expertise in the contribution of the collected data. Indeed, proposed system serves for versatile applications and can be considered as an aiding factor into the data collection using the platforms of opportunities.

Proposed solution allows the storing, cataloging and autonomous alerting of the spotted species. Consequently, system is envisioned to support research efforts both during the field expeditions as well as on land during data analysis. Moreover, the simplicity and flexibility of the system ensure (as seen in the usability tests) that it can be adapted and deployed mostly anywhere (e.g. on UAV, ROV, AUV, USV, sea-vessel mast, Stand-Up Paddling, etc.). Several other devices can be developed with this system as a base, such as, a submerged alternative that achieves underwater wildlife. Also, by replacing the custom trained model there are new possibilities of objects to be monitored, this flexibility allows the same system to monitor any kind of fauna or flora or even marine litter.

Finishing the development of this apparatus required uniting technologies and techniques from diverse areas. The knowledge acquired during my Bachelor's and Master's degrees helped with the its development, however much research and learning was required to complete it. Specifically in the areas of machine learning, model training, microcontrollers, software containers and flutter for mobile app development.

Furthermore, working with experts in several fields of marine biology provided much insight into the state of marine research. There are a lot of challenges regarding marine research that we need to overcome when researching the seas, namely that it is an environment not easily access by humans, it requires a lot of tools to perform studies (especially for underwater) and much remains to be learned from fully exploring the oceans. Even so, further studying of marine environments can provide knowledge essential to better understand our planet.

Altogether, the proposed system for marine megafauna assessments provides several advantages for the future field studies and is also a major contribution to future machine learning efforts in bridging with the marine ecology.

References

[1] Richard Sears and William F Perrin. Blue whale: Balaenoptera musculus. In *Encyclopedia of marine mammals*, pages 120–124. Elsevier, 2009.

[2] Joe Roman, James A Estes, Lyne Morissette, Craig Smith, Daniel Costa, James McCarthy, JB Nation, Stephen Nicol, Andrew Pershing, and Victor Smetacek. Whales as marine ecosystem engineers. *Frontiers in Ecology and the Environment*, 12(7):377–385, 2014.

[3] Marten Scheffer, Steve Carpenter, and Brad de Young. Cascading effects of overfishing marine systems. *Trends in Ecology & Evolution*, 20(11):579–581, 2005.

[4] Filipe Alves, Rita Ferreira, Miguel Fernandes, Zofia Halicka, Luís Dias, and Ana Dinis. Analysis of occurrence patterns and biological factors of cetaceans based on long-term and fine-scale data from platforms of opportunity: Madeira island as a case study. *Marine ecology*, 39(2):e12499, 2018.

[5] Ana Dinis, A Carvalho, Filipe Alves, Cátia Nicolau, Cláudia Ribeiro, Manfred Kaufmann, A Cañadas, and Luís Freitas. Spatial and temporal distribution of bottlenose dolphins, tursiops truncatus, in the madeira archipelago, ne atlantic. *Arquipélago-Life and Marine Sciences*, 33:45–54, 2016.

[6] Raspberry Pi Foundation. What is a Raspberry Pi? (Accessed on 10/10/2019).

[7] NVIDIA. NVIDIA Jetson Nano Developer Kit | NVIDIA Developer. (Accessed on 10/10/2019).

[8] Dev board | coral. `https://coral.ai/products/dev-board/#description`. (Accessed on 24/11/2019).

[9] Oceans - united nations sustainable development. `https://www.un.org/sustainabledevelopment/oceans/`. (Accessed on 14/11/2019).

[10] Goal 14 .:. sustainable development knowledge platform. `https://sustainabledevelopment.un.org/sdg14`. (Accessed on 14/11/2019).

[11] Clean ocean | project aware. `https://projectaware.org/issue/clean-ocean`. (Accessed on 14/11/2019).

[12] Annalisa Sambolino, Filipe Alves, Ana Mafalda Correia, Rita Ferreira, Patrícia Carvalho, Gustavo Silva, and Ana Dinis. Monitoring cetaceans in the madeira archipelago from a ferry along a fixed transect. *journal*, 2017.

[13] Lian Pin Koh and Serge A Wich. Dawn of drone ecology: low-cost autonomous aerial vehicles for conservation. *Tropical Conservation Science*, 5(2):121–132, 2012.

[14] Peter T Fretwell, Iain J Staniland, and Jaume Forcada. Whales from space: counting southern right whales by satellite. *PLoS One*, 9(2):e88655, 2014.

[15] ESA. SPACEWHALE - MAPPING WHALES FROM SPACE | ESA Business Applications, 2018. (Accessed on 11/10/2019).

[16] Hidef environmental consultancy - aerial survey hidef and bioconsult sh. https://www.hidefsurveying.co.uk/. (Accessed on 11/10/2019).

[17] Public Lab Store. Balloon aerial mapping kit – public lab store. https://store.publiclab.org/products/balloon-mapping-kit?variant=7028822724. (Accessed on 16/11/2019).

[18] DJI. Buy dji p4 multispectral agriculture drone today at dronenerds cp.ag.00000206.01. https://www.dronenerds.com/products/drones/enterprise-drones/phantom-4-enterprise/p4-multispectral.html. (Accessed on 18/11/2019).

[19] Rededge-mx sensor kit. https://micasense.com/shop/RedEdge-MX-Sensor-Kit-p121781377. (Accessed on 18/11/2019).

[20] heliguy. Second hand matrice 200 v1 body only - no accessories - heliguy. https://www.heliguy.com/approved-used-dji-matrice-200-v1-body-only-p6578. (Accessed on 18/11/2019).

[21] AEROKATS. Aerokats (kite-based) - globe.gov. https://www.globe.gov/web/aren-project/overview/aerokats. (Accessed on 23/11/2019).

[22] quad. Waterproof drones – the hexh2o and quadh2o professional waterproof drones. https://www.quadh2o.com/hexh2o/hexh2o-kit/. (Accessed on 25/11/2019).

[23] Mohamed Mirza, Yasser Asrul Ahmad, and Teddy Surya Gunawan. Low altitude balloon iot tracker. In *2017 IEEE 4th International Conference on Smart Instrumentation, Measurement and Application (ICSIMA)*, pages 1–6. IEEE, 2017.

[24] The Things Network. Lorawan® distance world record broken, twice. 766 km (476 miles) using 25mw transmission power. https://www.thethingsnetwork.org/article/lorawan-distance-world-record. (Accessed on 17/11/2019).

[25] James A Armstrong, Philip A Russell, Leslie E Sparks, and Dennis C Drehmel. Tethered balloon sampling systems for monitoring air pollution. *Journal of the Air Pollution Control Association*, 31(7):735–743, 1981.

[26] MicaSense. Rededge sensor. https://www.dronenerds.com/products/cameras-sensors/multispectral/micasense/micasense-rededge-sensor-rededge-dronenerds.html. (Accessed on 17/11/2019).

[27] ML. 9 applications of machine learning from day-to-day life. https://medium.com/app-affairs/9-applications-of-machine-learning-from-day-to-day-life-112a47a429d0. (Accessed on 24/11/2019).

[28] Tensorflow. https://www.tensorflow.org/. (Accessed on 24/11/2019).

[29] Opencv: Introduction. https://docs.opencv.org/4.5.0/d1/dfb/intro.html. (Accessed on 24/11/2019).

[30] Simplecv. http://simplecv.org/. (Accessed on 24/11/2019).

[31] Deep learning toolbox - matlab. https://www.mathworks.com/products/deep-learning.html. (Accessed on 24/11/2019).

[32] Pytorch. https://pytorch.org/. (Accessed on 24/11/2019).

[33] Keras: the python deep learning api. https://keras.io/. (Accessed on 24/11/2019).

[34] Accord.net machine learning framework. http://accord-framework.net/. (Accessed on 06/12/2020).

[35] Chainer: A flexible framework for neural networks. https://chainer.org/. (Accessed on 06/12/2020).

[36] Apache mxnet | a flexible and efficient library for deep learning. https://mxnet.apache.org/versions/1.7.0/. (Accessed on 06/12/2020).

[37] Caffe | deep learning framework. https://caffe.berkeleyvision.org/. (Accessed on 06/12/2020).

[38] Deeplearning4j. https://deeplearning4j.org/. (Accessed on 06/12/2020).

[39] Bigdl project. https://bigdl-project.github.io/master/index.html. (Accessed on 06/12/2020).

[40] ESA. Newcomers earth observation guide | esa business applications. https://business.esa.int/newcomers-earth-observation-guide. (Accessed on 11/10/2019).

[41] Rededge-mx - micasense. https://micasense.com/rededge-mx/. (Accessed on 17/11/2019).

[42] Maia the multispectral camera. Instruments for your multispectral surveys. https://www.spectralcam.com/. (Accessed on 17/11/2019).

[43] DJI. P4 multispectral, for precision agriculture and land management. https://dronedj.com/2019/09/24/dji-phantom-4-multispectral/. (Accessed on 17/11/2019).

[44] Parrot. Sequoia+. https://www.parrot.com/business-solutions-us/parrot-professional/parrot-sequoia. (Accessed on 18/11/2019).

[45] SlantRange. 4p sensor. https://www.dronenerds.com/products/cameras-sensors/enterprise-cameras/multispectral/slantrange/slantrange-4p-sensor-10400-slantrange.html. (Accessed on 18/11/2019).

[46] SlantRange. 4p+ sensor. https://www.dronenerds.com/products/cameras-sensors/enterprise-cameras/multispectral/slantrange/slantrange-4p-sensor-10402-slantrange.html. (Accessed on 18/11/2019).

[47] Sentera. Agriculture mapping software and crop management solutions. https://sentera.com/. (Accessed on 18/11/2019).

[48] Sharon L Nieukirk, Leila Lemos, Todd E Chandler, and Leigh G Torres. Drone up! quantifying whale behavior from a new perspective improves observational capacity. *Frontiers in Marine Science*, 5, 2018.

[49] Alex Borowicz, Hieu Le, Grant Humphries, Georg Nehls, Caroline Höschle, Vladislav Kosarev, and Heather J Lynch. Aerial-trained deep learning networks for surveying cetaceans from satellite imagery. *PloS one*, 14(10):e0212532, 2019.

[50] Prateek Murgai. Development of an automatic classification system for the cetaceans using their vocalizations. *Sensors*, 2015.

[51] J. Lopez, J. Schoonmaker, and S. Saggese. Automated detection of marine animals using multispectral imaging. In *2014 Oceans - St. John's*, pages 1–6, Sep. 2014.

[52] Marko Radeta, Miguel Ribeiro, Dinarte Vasconcelos, Jorge Lopes, Michael Sousa, João Monteiro, and Nuno Jardim Nunes. Seamote-interactive remotely operated apparatus for aquatic expeditions. In *IFIP Conference on Human-Computer Interaction*, pages 237–248. Springer, 2019.

[53] Jon Schoonmaker, Tami Wells, Gary Gilbert, Yuliya Podobna, Irina Petrosyuk, and Joseph Dirbas. Spectral detection and monitoring of marine mammals. In *Airborne Intelligence, Surveillance, Reconnaissance (ISR) Systems and Applications V*, volume 6946, page 694606. International Society for Optics and Photonics, 2008.

[54] Stacey Kuznetsov, George Noel Davis, Eric Paulos, Mark D Gross, and Jian Chiu Cheung. Red balloon, green balloon, sensors in the sky. In *Proceedings of the 13th international conference on Ubiquitous computing*, pages 237–246. ACM, 2011.

[55] T Jensen, A Apan, F Young, and L Zeller. Detecting the attributes of a wheat crop using digital imagery acquired from a low-altitude platform. *Computers and electronics in agriculture*, 59(1-2):66–77, 2007.

[56] John Saghri, Dustin Blackwell, John Hupton, Robert Hursig, Jessica Kiefer, Matt Schlutz, and Scott Seims. balloonsat: design, implementation, and application of a low-cost tethered weather balloon remote sensing station. *Honors Undergraduate Research Journal*, 2(1):6, 2009.

[57] Marko Radeta, Nuno J Nunes, Dinarte Vasconcelos, and Valentina Nisi. Poseidon-passive-acoustic ocean sensor for entertainment and interactive data-gathering in opportunistic nautical-activities. In *Proceedings of the 2018 Designing Interactive Systems Conference*, pages 999–1011. ACM, 2018.

[58] Guobao Xu, Yanjun Shi, Xueyan Sun, and Weiming Shen. Internet of things in marine environment monitoring: A review. *Sensors*, 19(7):1711, 2019.

[59] Jon Schoonmaker, Joseph Dirbas, Yuliya Podobna, Tami Wells, Cynthia Boucher, and Daniel Oakley. Multispectral observations of marine mammals. *Proceedings of SPIE - The International Society for Optical Engineering*, 10 2008.

[60] Raspberry. Pi noir camera v2. `https://www.raspberrypi.org/products/pi-noir-camera-v2/?resellerType=home`. (Accessed on 07/12/2020).

[61] Cetus: Cetacean monitoring surveys in the eastern north atlantic. `http://www.vliz.be/en/imis?dasid=6202&doiid=350`.

[62] Jetson nano brings ai computing to everyone | nvidia developer blog. `https://developer.nvidia.com/blog/jetson-nano-ai-computing/`. (Accessed on 09/12/2020).

[63] Tensorflow lite inference. `https://www.tensorflow.org/lite/guide/inference`. (Accessed on 24/11/2019).

[64] A simplified introduction to dart and flutter. `https://www.freecodecamp.org/news/https-medium-com-rahman-sameeha-whats-flutter-an-intro-to-dart-6fc42ba7c4a3/`. (Accessed on 11/29/2020).

[65] Fetch data from the internet - flutter. `https://flutter.dev/docs/cookbook/networking/fetch-data`. (Accessed on 11/29/2020).

[66] System usability scale (sus). https://www.usability.gov/how-to-and-tools/methods/system-usability-scale.html. (Accessed on 08/12/2020).

[67] Measuring and interpreting system usability scale (sus) - uiux trend. https://uiuxtrend.com/measuring-system-usability-scale-sus/#interpretation. (Accessed on 12/14/2020).