



# **Smartrawl 4.0**

## **Final Report**

---

**A REPORT COMMISSIONED BY  
FIS AND PREPARED BY:**

**Paul Fernandes, Dewei Yi,  
Shaun Fraser, Stewart Chalmers  
University of Aberdeen**

**Published by: Fisheries Innovation & Sustainability (FIS)**

This report is available at: <https://www.fisorg.uk>

### **Dissemination Statement**

This publication may be re-used free of charge in any format or medium. It may only be reused accurately and not in a misleading context. All material must be acknowledged as FIS copyright and use of it must give the title of the source publication. Where third party copyright material has been identified, further use of that material requires permission from the copyright holders concerned.

### **Disclaimer**

The opinions expressed in this report do not necessarily reflect the views of FIS and FIS is not liable for the accuracy of the information provided or responsible for any use of the content.

### **Suggested Citation:**

Fernandes, P., Yi, D., Fraser, S., Chalmers, S. (2024) Smartrawl 4.0 Final Report.  
A study commissioned by Fisheries Innovation & Sustainability (FIS) <https://fisorg.uk>.

**Title:** Smartrawl 4.0 Final Report

**First published:** March 2024

© FIS





# SMARTRAWL 4.0

## Final Report

14 APRIL 2022

---



by: Paul Fernandes, Dewei Yi, Shaun Fraser and Stewart Chalmers

**Front cover.** *An image taken in March 2022 of a cod inside the extension of a trawl deployed from the Atlantia research vessel, off the Shetland Islands.*

## Executive Summary

Smartrawl 4.0 was part of a series of phased projects, to develop a selective device to operate in a demersal trawler allowing for fish to be released in-situ underwater, thus allowing the skipper of the vessel to comply with the Landings Obligation.

There were 3 objectives of Smartrawl 4.0. i) To integrate Smartrawl sub-systems, by developing software and adapting electronics of the Smartrawl stereo camera system. ii) To develop artificial intelligence algorithms to differentiate images of fish from prawns, and preliminary fish species algorithms; and iii) To conduct sea trials in support of AI development for species identification, testing of the integrated system, and prototype gates.

System integration was partially complete. A new computer was added to the stereo camera system which has the power to enable AI algorithms to operate. This required adaptation to the power supply and software. AI algorithms were demonstrated to operate on the same type of computer but have not yet been integrated into the stereo camera system.

Extensive development of AI was carried out to prepare the data required to train AI algorithms. Significant progress was made and a process to detect fish and size them was demonstrated on the type of computer that was integrated into the stereo camera. This process took 2 seconds. Validation of the species identification was not possible in the time available. This report provides a detailed account of the extensive AI work done so that it may be replicated by others.

Extensive sea trials were carried with 27 trawl deployments during which the camera was operational. Incrementally significant changes were made to improve image quality, including: a new blue background to provide greater image contrast (see front cover); adjustment of white balance to obtain true colours underwater; and trials of a sediment suppression system. This will provide large numbers of images for future AI training exercises.

It was not possible to manufacture the final component of the gate (latch to control opening and closing) in time for it to be tested. It was however completed and is ready for future phases.

Future work should focus on testing the opening and closing mechanism of the gate and integrating the AI systems to the mechanism with simple logic based on fish sizing. This should be possible with the systems built here in Smartrawl 4.0.

# Contents

|   |    |
|---|----|
| Executive Summary.....  | 3  |
| Introduction .....  | 6  |
| Smartrawl Phase 4.0 objectives .....                            | 6  |
| Objective 1: System integration .....                           | 6  |
| Objective 2: Artificial intelligence algorithms.....            | 8  |
| Raw Datasets.....   | 8  |
| Image Extraction and Filtering .....                            | 10 |
| Image Labelling.....  | 12 |
| Coarse Labelling .....  | 15 |
| Fine Labelling .....  | 18 |
| Statistics of labelled images .....                             | 22 |
| Data Analysis Tools.....  | 24 |
| Recommendations for future labelling work.....                  | 25 |
| Data Conversion for Deep Learning Model.....                    | 26 |
| Biometric Identification.....                                   | 28 |
| Deep Learning Model for Fish Detection and Recognition .....    | 31 |
| Instance Segmentation and MaskRCNN .....                        | 31 |
| Training MaskRCNN Model.....                                    | 31 |
| Training .....  | 33 |
| Testing and Inspection .....                                    | 33 |
| Initial assessment.....   | 34 |
| Model Improvement.....  | 36 |
| Fish Size Detection .....                                       | 37 |
| Size Estimation .....   | 41 |
| Deployment of Deep Learning Model on Lightweight AI Device..... | 46 |
| Future work.....  | 49 |
| Objective 3: Shetland sea trials.....                           | 52 |
| Introduction.....   | 52 |
| Initial camera sea trials .....                                 | 52 |
| D1 (30/11/2021).....  | 52 |
| D2 (02/12/2021).....  | 53 |
| D3 and D4 (10/12/2021) .....                                    | 54 |
| D5 and D6 (25/01/2022) .....                                    | 56 |

|                                       |    |
|---------------------------------------|----|
| D7 (02/02/2022).....                  | 58 |
| Initial tank tests.....               | 59 |
| Subsequent bench tests.....           | 60 |
| Subsequent tank tests.....            | 61 |
| Subsequent camera sea trials.....     | 61 |
| D8 (04/03/2022).....                  | 62 |
| D9 (07/03/2022).....                  | 62 |
| D10, D11, and D12 (14/03/2022).....   | 63 |
| Sediment suppression experiments..... | 65 |
| D13, D14, and D15 (21/03/2022).....   | 66 |
| D16 and D17 (23/04/2022).....         | 68 |
| D18 (24/03/2022).....                 | 70 |
| D19 and D20 (29/03/2022).....         | 71 |
| D21 (30/03/2022).....                 | 73 |
| D22 (31/03/22).....                   | 75 |
| D23, D24, and D25 (31/03/2022).....   | 76 |
| D26 and D27 (01/04/2022).....         | 78 |
| Gate sea trials.....                  | 78 |
| References.....                       | 78 |
| Acknowledgements.....                 | 78 |

## Introduction

The Smartrawl is a technological development designed to avoid discards and bycatch in demersal fishing trawls, ensuring that only fish and shellfish that are intended to be landed are caught at sea. The system consists of a stereo camera with lighting in the trawl extension to obtain high quality images of fish traversing into the cod-end through the trawl net. These images will be analysed by an onboard computer to determine the size and species of the fish. A signal is then sent to a 'gate' located in the trawl extension to catch or release the fish: if an unwanted fish is identified, a signal is sent to open the gate and the fish is released back into the wild; if the fish is wanted, a signal is sent to close the gate and the fish passes into the cod end to be captured. This was originally designed to be self-contained without information being passed to the bridge of the vessel. However, an interim modification involves the transfer of information from the stereo camera to the bridge via an underwater communications device (modem).

This report describes Phase 4.0 of the project funded under FIS039 which lasted from October April 2021 to March 2022. This phase had the following objectives:

### Smartrawl Phase 4.0 objectives

1. To integrate Smartrawl sub-systems, by developing software and adapting electronics of the Smartrawl stereo camera system.
2. To develop artificial intelligence algorithms to differentiate images of fish from prawns, and preliminary fish species algorithms.
3. To conduct sea trials in support of AI development for species identification, testing of the integrated system, and prototype gates.

### Objective 1: System integration

Previous phases of the Smartrawl project procured new components for the Smartrawl system to enable communication of what was being detected by the net camera back to the bridge of the fishing trawler. However, the pandemic prevented full integration and testing of these new components. In this phase, we aimed to complete system integration by adapting the existing stereo camera components to include a computer powerful enough to run the fish detection and sizing AI algorithms developed in previous phases. This requires some software adaptation and development, as well as minor electronic hardware modifications.

An update to the Odroid XU4 single board Linux processor was to be investigated with the aim of replacing this with an Nvidia Nano multicore processor. This processor was chosen principally due to its superior speed in the numeric-intensive video recognition routines. To incorporate an Nvidia Nano, it was necessary to consider certain hardware and software modifications to the existing system. This was however planned with the aim of keeping as much as possible of the existing hardware both mechanical and electronics unaltered. There would inevitably need to be software changes made to implement the integration of the new system since timing would be different and there would be certain unavoidable electronics hardware modifications.

The basic system layout modified to incorporate the Nvidia Nano and updated



batteries is shown in Figure 1. The current system comprises of a power distribution board which allows two 12V 40Ah NiMH batteries to be connected either in series to produce a +24V main supply to power the Linux board, cameras and strobes. This board also allows re-configuration of the battery connections thereby allowing external access for charging each separately.

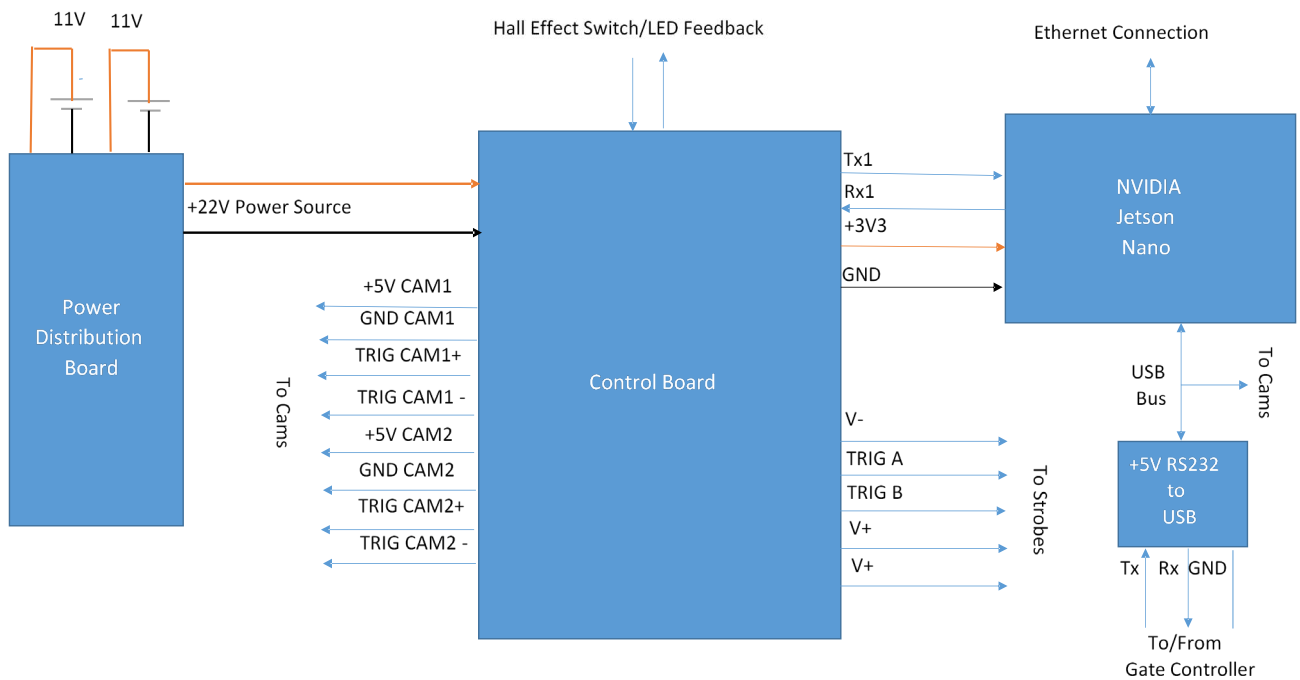


Figure 1. Stereo Camera Electronics board interconnections

Principally due to its heatsink, the Nvidia Nano is larger and deeper than the Odroid and therefore requires some of the area in the housing given over to the batteries to be sacrificed to accommodate the Nano. This can be achieved by replacing the NiMH batteries by Li-ion ones which are more power and volume dense. The batteries which met the requirement were of 11 V 35 Ah rating requiring only 33% of the volume of the NiMH ones. These are also much more rapid charging.

The controller board is central to the system and is based around a 32-bit general-purpose Arduino microcontroller which manages power to the cameras and strobes and controls powering on and off the Nano board in a controlled fashion. When the main program is running it operates in one of two modes. Maintenance mode allows each sub-system, via the controller board, to be controlled, tested and, if appropriate, calibrated under user control. Program updates can also be downloaded and captured photographs uploaded while in this mode. In acquisition mode the controller is responsible for activating the cameras and strobes with very high precision timing and synchronisation.

The controller board had been designed to plug into the GPIO port of the Odroid XU4 to interconnect ground and a 3V3 asynchronous RS232 type serial communications link. The majority of pins on the connector however were not connected, being present only to provide mechanical support between the two boards. The existing controller board could be provided with a wiring harness to re-route the appropriate signals to the Nano 3V3 GPIO port. A future update would entail re-laying out the control board to enable a neater, more reliable and

more permanent interconnect.

The current controller board did not have the facility for communications to/from the gate controller enabled, although asynchronous serial RS232 signals to allow this were routed from The Odriod XU4 to the power/charging connector. Activation of this link from the Nano via a USB to 5V logic level converter (shown in the bottom right of Figure1) would allow the Nano in the camera housing to communicate directly with the gate controller as a virtual COM port. Because there is an independent microcontroller within the release latch, commands regarding the position and timing of the gate release could then be sent directly from the Nano program, based on the results from the fish recognition routines.

All hardware modifications and re-wiring required to update the camera system were identified and implemented. The system is at the point where the software is being modified to reflect these changes and thus enable the overall system. This will allow the high-level software running on the Nano processor to interact with the control processor and hence as an integrated system capable of acquiring and storing stereo images at a rate of at least 2 frames per second.

## Objective 2: Artificial intelligence algorithms

The smartrawl system is based on the acquisition and processing of images in the extension of the trawl such that the gate ahead of the cod end can be opened or closed to catch or release species and sizes of fish of interest. For this to operate effectively, the images need to be analysed automatically to determine the species and size. This consists of (1) detecting a fish; (2) determining its size; and (3) determining the species. This requires artificial intelligence algorithms to achieve this. Although differentiating fish species is likely to be a significant task, requiring more time than would be allowed for in the current round of funding, we aimed to start the process here, and deliver on a preliminary objective of identifying fish and discriminating them from other objects, particularly prawns. This would be a major step in dealing with some of the issues of discards in prawn fisheries.

In this section, the structure and origin of the raw data, data pre-processing, data extraction, image labelling, and the final statistical results are described.

### Raw Datasets

Five different raw datasets were included in the Smartrawl 4.0 project. They were the dataset from the Sparkling Star deployment collected on 19 July 2019, and four different datasets collected in Shetland (during Smartrawl 4.0) on 30 November 2021, 2 December 2021 and 10 December 2021. These datasets contain different identifiers such as the number of each species captured, date of collection, time of collection, image number, and species name. Table 1 provides details of these data collected including the number of images collected from the left and right cameras, which came to a total of 44064.

Table 1. Details of the raw datasets used in Smartrawl 4.0

| Raw Datasets                                     | Folder name      | Number of Images |
|--|------------------|------------------|
| <b>Sparkling Star Deployment</b><br>19 July 2019 | Left             | 1000             |
|  | Right            | 1000             |
| <b>Deployment 1</b><br>30 November 2021          | 1.1-Right Camera | 5679             |
|  | 1.2-Left Camera  | 5678             |
| <b>Deployment 2</b><br>2 December 2021           | 2.1-Right Camera | 5701             |
|  | 2.2-Left Camera  | 5700             |
| <b>Deployment 3</b><br>10 December 2021          | 3.1-Right Camera | 4786             |
|  | 3.2-Left Camera  | 4785             |
| <b>Deployment 4</b><br>10 December 2021          | 4.1-Right Camera | 4868             |
|  | 4.2-Left Camera  | 4867             |
| <b>Total</b>                                     |                  | 44064            |

For a clearer and more specific details of the structure of the data, Figure 2 shows the nature of the dataset from the Sparkling Star deployment collected on 19 July 2019. The dataset has been annotated by marine biology students and includes spreadsheets containing information on the date, time, number, number of organisms, and type of organism for each image collected.

|     | A        | B        | C     | D         | E         | F              | G                  | H          | I   | J       | K       |
|-----|----------|----------|-------|-----------|-----------|----------------|--------------------|------------|-----|---------|---------|
| 1   | Date     | Time     | Image | imecheck. | imecheck. | iber of organi | Description        | omning Tot | sum | Haddock | Dogfish |
| 326 | 19.07.19 | 10:54:19 | 3522  | 10:54:19  | 0         | 1              | <i>haddock</i>     | 85         | 1   | 1       |         |
| 327 | 19.07.19 | 10:54:20 | 3524  | 10:54:20  | 0         | 1              | <i>haddock</i>     | 86         | 1   | 1       |         |
| 328 | 19.07.19 | 10:54:21 | 3525  | 10:54:20  | 0         | 1              | <i>haddock</i>     | 86         | 0   |         |         |
| 329 | 19.07.19 | 10:54:21 | 3526  | 10:54:22  | 0         | 1              | <i>haddock</i>     | 87         | 1   | 1       |         |
| 330 | 19.07.19 | 10:54:35 | 3553  | 10:54:35  | 0         | 1              | <i>haddock</i>     | 88         | 1   | 1       |         |
| 331 | 19.07.19 | 10:54:37 | 3557  | 10:54:37  | 0         | 1              | <i>whiting</i>     | 89         | 1   |         |         |
| 332 | 19.07.19 | 10:54:40 | 3564  | 10:54:40  | 0         | 1              | <i>unidentifie</i> | 90         | 1   |         |         |
| 333 | 19.07.19 | 10:54:45 | 3573  | 10:54:45  | 0         | 1              | <i>dab</i>         | 91         | 1   |         |         |
| 334 | 19.07.19 | 10:54:47 | 3577  | 10:54:47  | 0         | 1              | <i>haddock</i>     | 92         | 1   | 1       |         |
| 335 | 19.07.19 | 10:54:48 | 3579  | 10:54:48  | 0         | 1              | <i>haddock</i>     | 92         | 0   |         |         |
| 336 | 19.07.19 | 10:54:58 | 3600  | 10:54:59  | 0         | 1              | <i>haddock</i>     | 93         | 1   | 1       |         |
| 337 | 19.07.19 | 10:54:59 | 3601  | 10:54:58  | 0         | 1              | <i>haddock</i>     | 93         | 0   |         |         |
| 338 | 19.07.19 | 10:54:59 | 3602  | 10:55:00  | 0         | 1              | <i>haddock</i>     | 93         | 0   |         |         |
| 339 | 19.07.19 | 10:55:00 | 3604  | 10:55:00  | 0         | 1              | <i>haddock</i>     | 93         | 0   |         |         |
| 340 | 19.07.19 | 10:55:07 | 3617  | 10:55:07  | 0         | 1              | <i>haddock</i>     | 94         | 1   | 1       |         |
| 341 | 19.07.19 | 10:55:12 | 3628  | 10:55:13  | 0         | 1              | <i>haddock</i>     | 95         | 1   | 1       |         |
| 342 | 19.07.19 | 10:55:16 | 3635  | 10:55:16  | 0         | 1              | <i>haddock</i>     | 96         | 1   | 1       |         |
| 343 | 19.07.19 | 10:55:18 | 3640  | 10:55:19  | 0         | 2              | <i>haddock</i>     | 98         | 2   | 2       |         |
| 344 | 19.07.19 | 10:55:19 | 3641  | 10:55:19  | 0         | 1              | <i>haddock</i>     | 98         | 0   |         |         |
| 345 | 19.07.19 | 10:55:19 | 3642  | 10:55:19  | 0         | 1              | <i>haddock</i>     | 98         | 0   |         |         |
| 346 | 19.07.19 | 10:55:21 | 3645  | 10:55:21  | 0         | 1              | <i>haddock</i>     | 99         | 1   | 1       |         |
| 347 | 19.07.19 | 10:55:21 | 3646  | 10:55:21  | 0         | 1              | <i>haddock</i>     | 99         | 0   |         |         |
| 348 | 19.07.19 | 10:55:22 | 3647  | 10:55:21  | 0         | 1              | <i>haddock</i>     | 99         | 0   |         |         |
| 349 | 19.07.19 | 10:55:22 | 3648  | 10:55:23  | 0         | 1              | <i>haddock</i>     | 99         | 0   |         |         |
| 350 | 19.07.19 | 10:55:31 | 3665  | 10:55:30  | 0         | 1              | <i>haddock</i>     | 100        | 1   | 1       |         |

Figure 2. Details of the dataset from Sparkling Star's deployment collected on 19 July 2019

## Image Extraction and Filtering

The four raw datasets from the 2021 deployments were relatively large and not easy to process. Therefore, we need to perform pre-processing steps such as extraction and filtering of the images, as some of the images do not contain organisms.

Feature extraction will be used as a major step in dimensionality reduction, using machine learning and deep learning algorithms to filter out images that may contain fish species while retaining information from the raw datasets. This process is done using MATLAB (Matrix Laboratory) software. Specifically, MATLAB is an interactive environment for algorithm development, data visualisation, data analysis and numerical computation for applications in different fields such as image processing, deep learning, signal processing and analysis to name a few, and it can also be used by calling programs written in languages such as Python.

For the four datasets collected in 2021, the left camera file and the right camera file were each divided into folders for every 1000 images during the initial processing. Figure 3 shows an example of the 2.1-Right Camera from deployment 2.

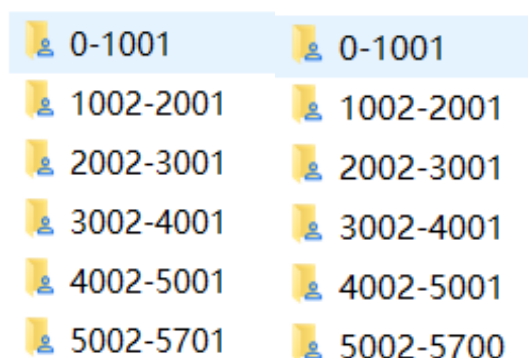


Figure 3. Example of a preliminary division (2.1-Right Camera from deployment 2).

The image data collected are in ".jpg" format. It was necessary to first select the corresponding folder for image filtering via MATLAB. We refer to resnet50 for loading the pre-training network, specifically, ResNet-50 is a convolutional neural network with 50 layers of depth. The pre-training network can classify images into 1000 object classes, such as various objects and many animals (Figure 4).

```
% Load pretrained network
net = resnet50();

imageFiles = dir('*.jpg');
numberOfFiles = length(imageFiles)

for i = 1:numberOfFiles
    imageName = imageFiles(i).name;
    img = imread(imageName);
    imageFeatureSet{i} = DeepFeature(img, net);
end
```

Figure 4. Load pre-trained network.

A pre-trained network is then used as the feature extractor by using layer activation as the feature, where the main support vector machine (SVM) algorithm is used. The predictor variables are first normalised using a radial basis and a training SVM classifier (Figure 5).

```
y = ones(size(X,1),1);

rng(1);
SVMModel = fitcsvm(X,y,'KernelScale','auto','Standardize',true,...
    'OutlierFraction',0.07);
```

Figure 5. Training SVM classifier.

Next, cross-validation of the SVM classifier is performed, plotting the observations and decision boundaries as shown below.

```
CVSVMModel = crossval(SVMModel);
[~,scorePred] = kfoldPredict(CVSVMModel);
outlierRate = mean(scorePred<0)
```

Figure 6. Cross-validation SVM classifier.

To keep the files from being repetitive and clearly filtered, we then created a new folder "filteredImages". More importantly, the net can only handle 224\*224 RGB images, so we used an augmentedImageDatastore created from the training and test sets to resize the images and convert them to RGB on the fly, and then used the CNN to extract the training features so that each layer of the CNN would correspond to the input image. Yet only a few layers in the CNN are suitable for image feature extraction, so to extract features more effectively, we visualise the data by obtaining the weights of the convolutional layers and scaling and adjusting them (Figure 7).

```
imageSize = net.Layers(1).InputSize;
augmentedImage = augmentedImageDatastore(imageSize, image, 'ColorPreprocessing', 'gray2rgb');

% Get the network weights for the second convolutional layer
w1 = net.Layers(2).Weights;

% Scale and resize the weights for visualization
w1 = mat2gray(w1);
w1 = imresize(w1,5);
```

Figure 7. Features extraction by deep learning.

The results show that the algorithm performs well in the process of image extraction and image filtering, and it filtered out most of the images that contained fish, which helped us

save a lot of time. The final number of files produced in "filteredImages" is shown in Table 2.

Table 2. Number of Filtered images.

| Raw Datasets                            | Folder name      | Number of Filtered images |
|---|------------------|---------------------------|
| <b>Deployment 1</b><br>30 November 2021 | 1.1-Right Camera | 781                       |
|   | 1.2-Left Camera  | 823                       |
| <b>Deployment 2</b><br>2 December 2021  | 2.1-Right Camera | 552                       |
|   | 2.2-Left Camera  | 554                       |
| <b>Deployment 3</b><br>10 December 2021 | 3.1-Right Camera | 338                       |
|   | 3.2-Left Camera  | 415                       |
| <b>Deployment 4</b><br>10 December 2021 | 4.1-Right Camera | 272                       |
|   | 4.2-Left Camera  | 781                       |
| <b>Total</b>                            |                  | 4516                      |

## Image Labelling

The images from the left camera and the right camera are paired with each other according to image extraction and image filtering. We first labelled the images for the dataset from Sparkling Star's deployment collected on 19 July 2019, as they had been annotated with organisms by marine biologists.

For the 4.0 project, we used the LabelMe annotation tool for labelling images. LabelMe is an annotation tool written in python that can be used to annotate online images by polygons, rectangles, circles, lines and points without the need to install large datasets on the computer. The following computer environments are required (including the command in Figure 8):

Python3

Anaconda

Ubuntu / macOS / Windows

```
(base) C:\Users\Administrator>labelme
[INFO ] __init__:get_config:70 - Loading config file from: C:\Users\Administrator\.labelmerc
```

Figure 8. Open LabelMe tool via anaconda.

Figures 9 - 11 show some nice, labelled images from the Sparkling Star deployment collected on 19 July 2019.

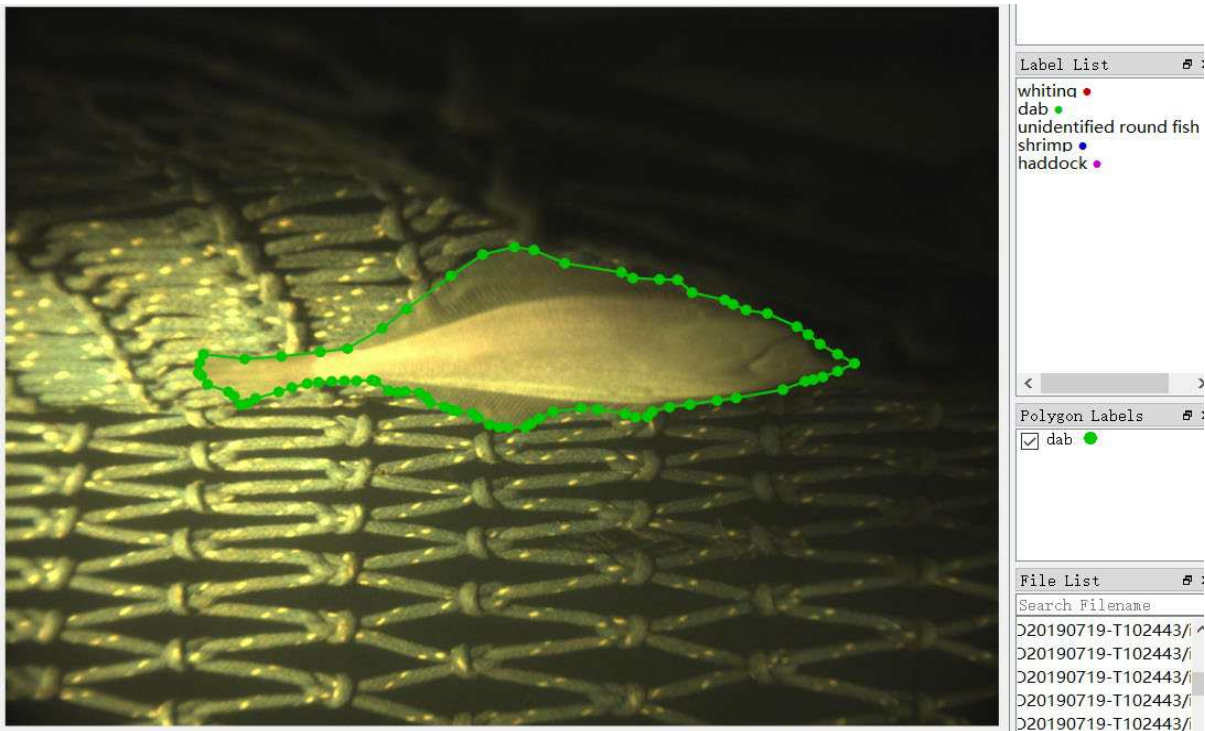


Figure 9. Labelled image of a flatfish from the Sparkling Star deployment.

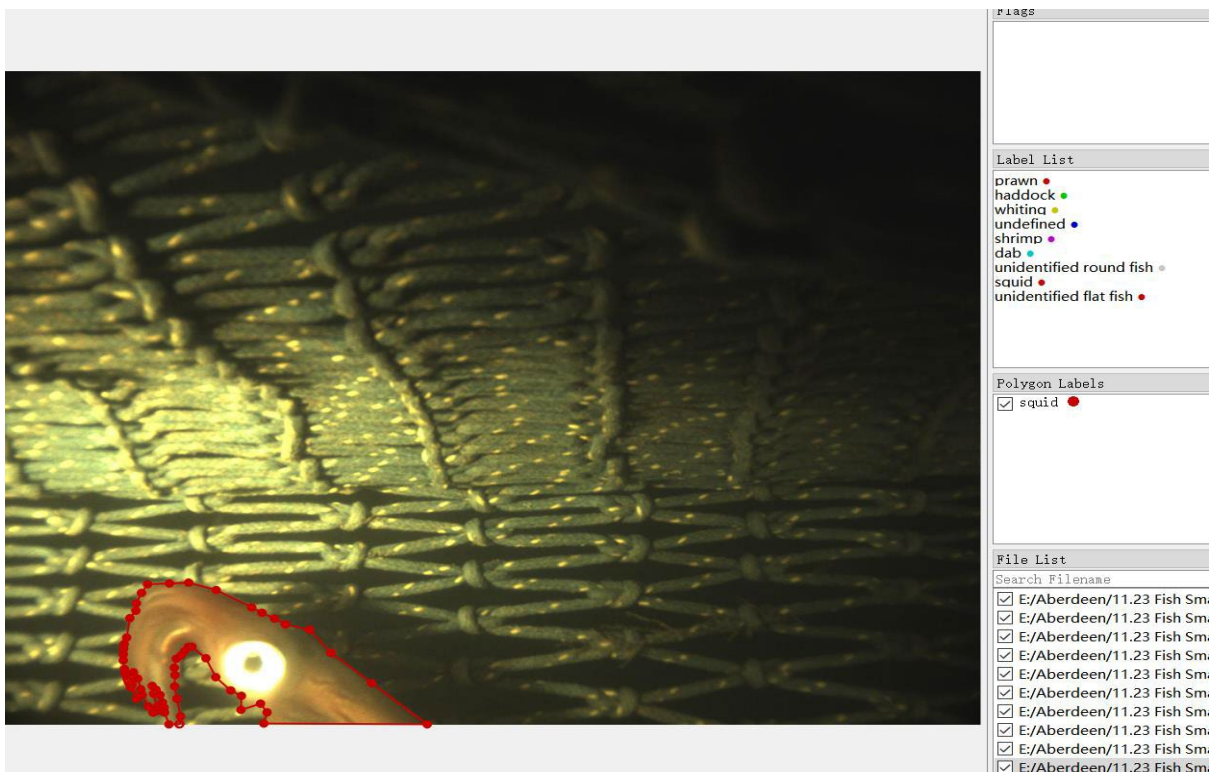


Figure 10. Labelled image of part of a squid from the Sparkling Star deployment.

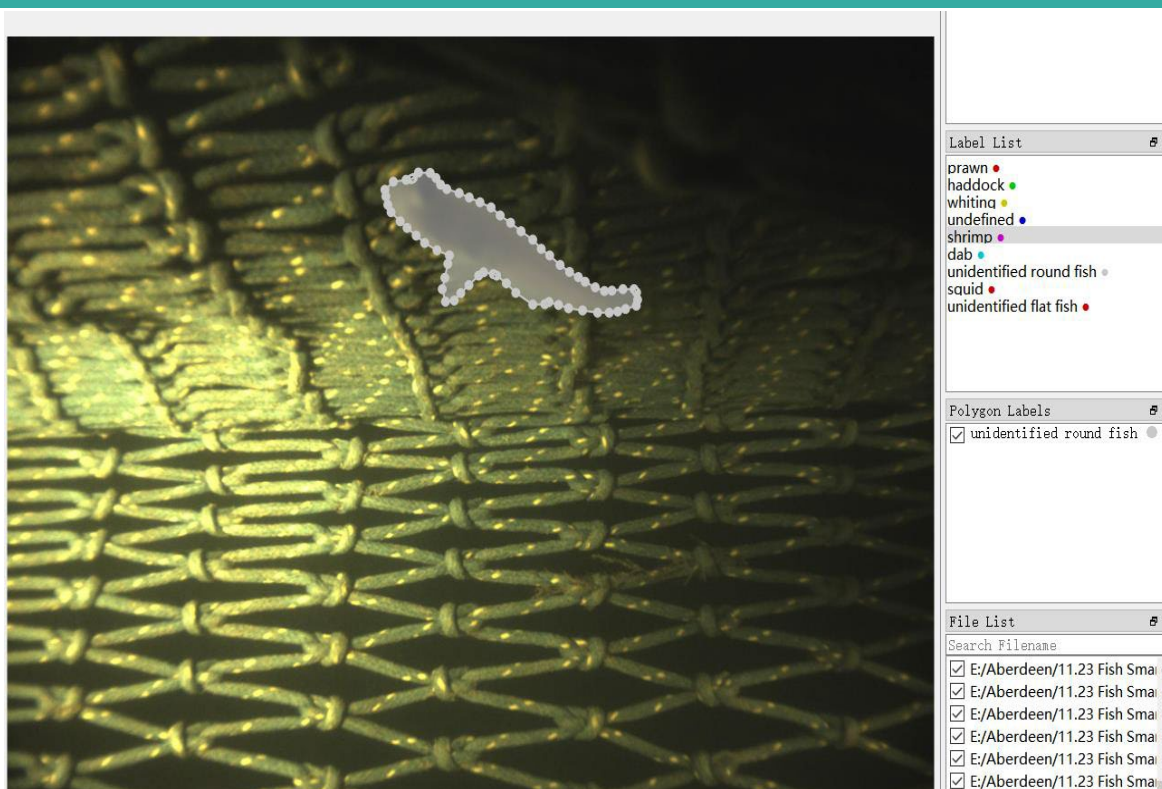


Figure 11. Labelled image of an unidentified roundfish from the Sparkling Star deployment.

To provide a clearer representation of the quality of each image, a new template was created based on the information in the table previously annotated by the marine biologist. The cameras from the left or right in this table each contain 5 identical attributes, namely the number of complete fish, LabelMe, quality, head and tail.

More specifically, a tick is used to indicate a labelled image. If the description contains two or more species of fish, "fish name: X (e.g., dab: x or dab, haddock: x)" is used for unlabelled fish. Also, for the question of how to determine the quality of fish with "\*", "\*\*\*" and "√\*\*\*", "\*" is defined as half-bodied, headless or tailless, with inconspicuous fins and particularly faint edges, and extremely reflective. "\*\*\*" is defined as showing more than 50% of the apparent organism and being slightly reflective and slightly blurred. "√\*\*\*" is defined as showing more than 80% of a clear organism, or a complete organism (even in the case of multiple fish, 1 fish has a complete organism). This is shown in Figure 12.



| Image | Number of organisms | Description  | Left |         |         |      | Right |         |         |      |
|-------|---------------------|--------------|------|---------|---------|------|-------|---------|---------|------|
|       |                     |              | 0    | Labelme | Quality | Head | Tail  | Labelme | Quality | Head |
| 2699  | 1                   | dab          | ✓    | **      |         |      | x     |         |         |      |
| 2702  | 1                   | dab          | ✓    | ✓***    | ✓       | ✓    | ✓     | *       |         |      |
| 2711  | 1                   | unidentified | ✓    | *       |         |      | x     |         |         |      |
| 2712  | 1                   | whiting      | ✓    | *       |         |      | x     |         |         |      |
| 2729  | 1                   | whiting      | ✓    | **      |         |      | x     |         |         |      |
| 2745  | 1                   | haddock      | ✓    | ✓***    | ✓       | ✓    | x     |         |         |      |
| 2746  | 1                   | haddock      | ✓    | **      |         |      | x     |         |         |      |
| 2756  | 1                   | haddock      | ✓    | *       |         |      | x     |         |         |      |
| 2757  | 1                   | haddock      | ✓    | **      |         |      | x     |         |         |      |
| 2758  | 1                   | haddock      | ✓    | **      |         |      | x     |         |         |      |
| 2775  | 2                   | haddock      | ✓    | ✓***    | ✓       |      | ✓     | **      |         |      |
| 2788  | 1                   | haddock      | ✓    | *       |         |      | x     |         |         |      |
| 2806  | 1                   | unidentified | ✓    | *       |         |      | ✓     | *       |         |      |
| 2815  | 1                   | haddock      | ✓    | **      |         |      | x     |         |         |      |
| 2816  | 1                   | haddock      | ✓    | ✓***    | ✓       | ✓    | x     |         |         |      |
| 2878  | 1                   | dab          | ✓    | ✓***    | ✓       | ✓    | ✓     | **      |         |      |
| 2879  | 1                   | dab          | ✓    | ✓***    | ✓       | ✓    | ✓     | *       |         |      |
| 2881  | 1                   | haddock      | ✓    | ✓***    | ✓       | ✓    | ✓     | ✓***    |         | ✓    |
| 2882  | 1                   | haddock      | ✓    | **      |         |      | x     |         |         |      |
| 2885  | 1                   | haddock      | ✓    | *       |         |      | x     |         |         |      |
| 2886  | 1                   | haddock      | ✓    | **      |         |      | x     |         |         |      |
| 2939  | 1                   | haddock      | ✓    | *       |         |      | x     |         |         |      |
| 2942  | 1                   | shrimp       | ✓    | *       |         |      | x     |         |         |      |
| 2943  | 1                   | shrimp       | ✓    | *       |         |      | x     |         |         |      |
| 2983  | 1                   | haddock      | ✓    | **      |         |      | x     |         |         |      |
| 3059  | 1                   | haddock      | ✓    | **      |         |      | ✓     | ✓***    | ✓       | ✓    |
| 3060  | 1                   | haddock      | ✓    | ✓***    | ✓       | ✓    | ✓     | **      |         |      |
| 3061  | 1                   | haddock      | ✓    | **      |         |      | x     |         |         |      |

Figure 12. New template form on Sparkling Star deployment.

However, for the four datasets collected in 2021, there was no such detailed sheet, so we have divided the labels into two parts, mainly coarse and fine labelling. For the coarse labelling, there will be three-point image annotation by marine biology experts to help identify the species.

### Coarse Labelling

Coarse labelling involves identifying the organisms detected in the stereo camera images. When possible, animals were detected down to genus or species level. When identification was more challenging the identification would move up the taxonomic hierarchy until a positive identification was certain; for simplicity common names were used. When an organism was identified a rough polygon was added around the animal, so it was clear which animal had been given the classification. This was completed using the software LabelMe.

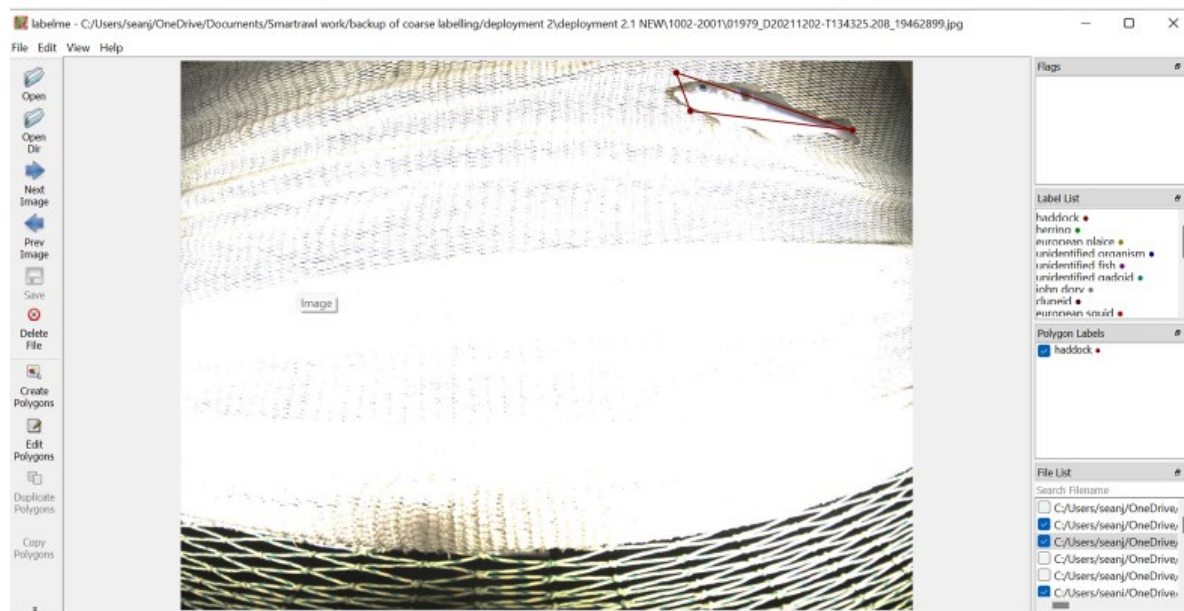


Figure 13. Screen capture from the LabelMe software showing an image of a labelled haddock.

Figure 13 shows a screengrab of the Labelme window with an image of a labelled haddock. On the far left is a column of actions that can be performed with LabelMe. Once a data set is downloaded the ‘open dir’ function is used to open a directory to photos that are being analysed. In images with organisms, the ‘create polygon’ function is used to manually draw a shape round an organism. When this has been completed a table appears where a label can be selected, or a new label added (Figure 14).

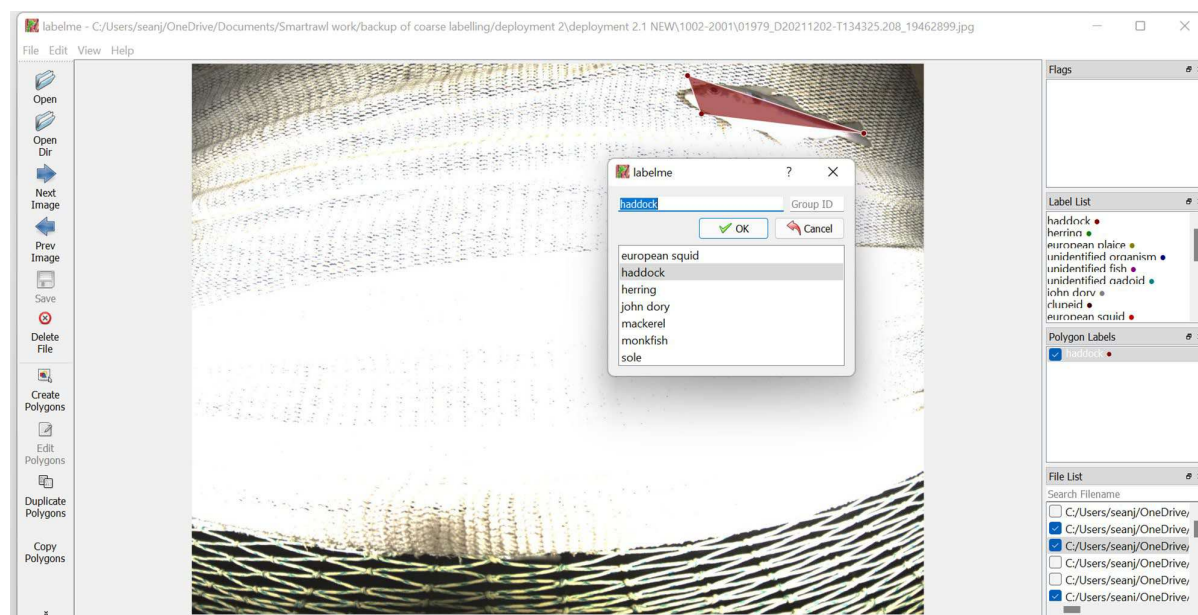


Figure 14. Screen capture from the LabelMe software showing the labelling of a polygon by selection from a drop-down list.

In these images (Figure 13 and Figure 14) the organism has been assigned the label haddock as there is sufficient information to give it this classification. The animal has a morphology consistent with gadoids and there are additional features (shape of pelvic fin, black lateral line, St Peter’s thumbprint, colour) that are characteristic of haddock.

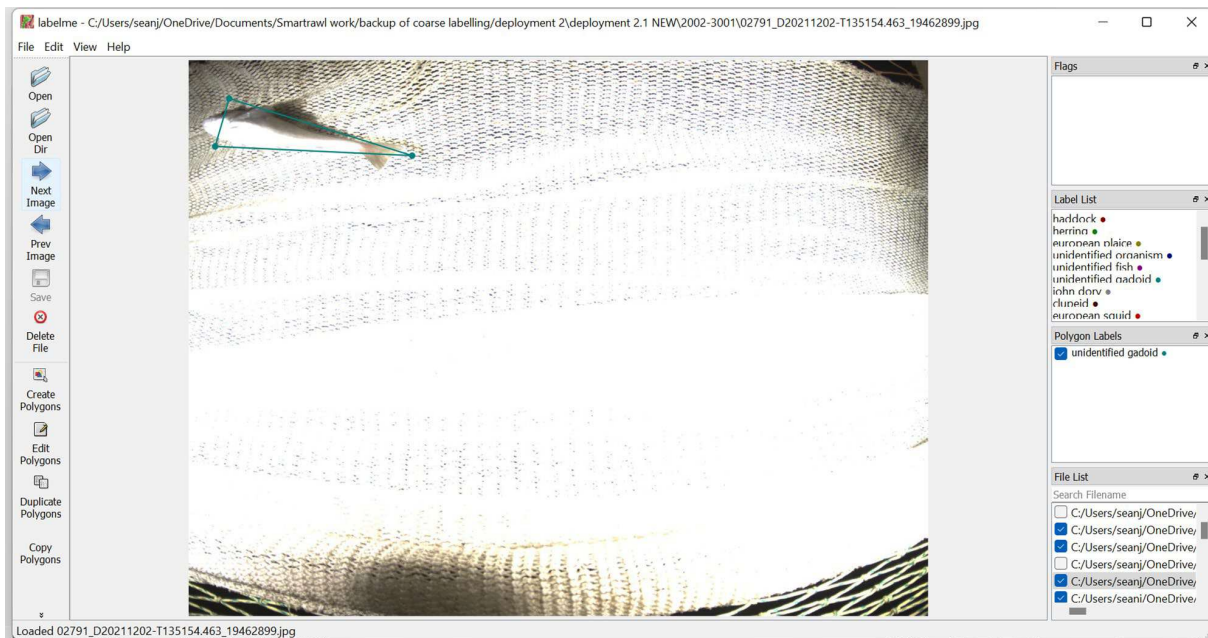


Figure 15. Screen capture from the LabelMe software showing a labelled gadoid.

In the case of Figure 15 we have also seen a fish with very similar morphology to the haddock; there is sufficient information to classify this animal as a gadoid. However, we can't see those other features that we used to classify the animals as haddock features, therefore, the organism was identified as a gadoid.

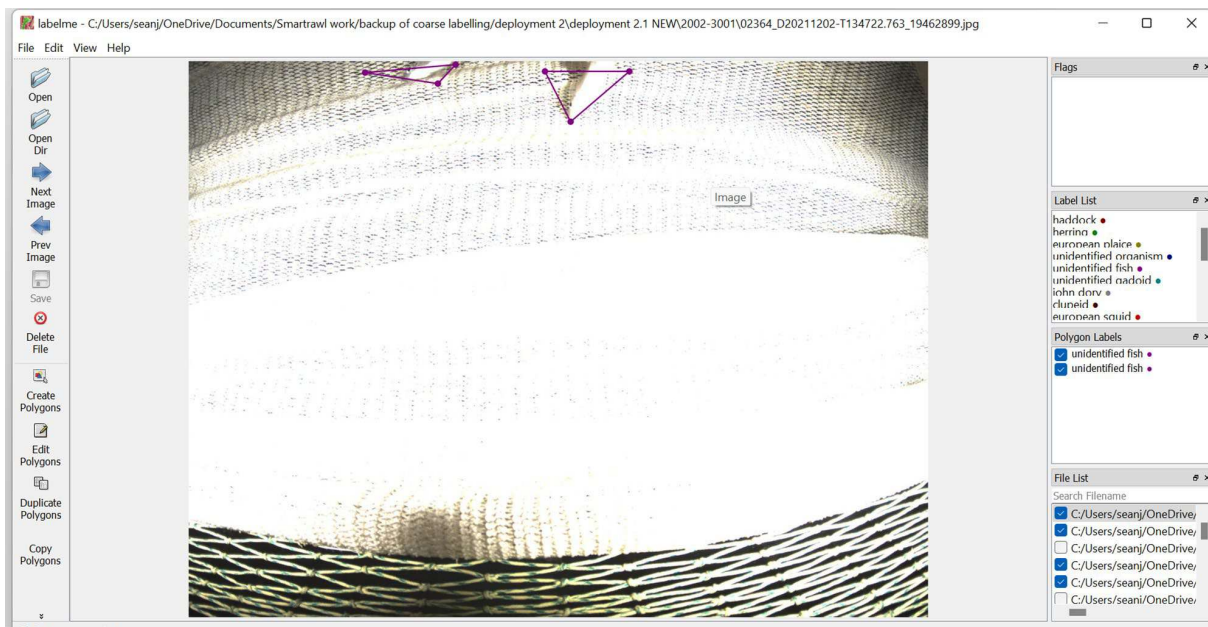


Figure 16. Screen capture from the LabelMe software showing an unidentified organism.

On rare occasions we can see the animal is a fish but cannot give it any further classification. (Figure 16). Rarer still were occasions where some animals could not be confidently classified as a fish or another group due to the lack of information in the images. These were simply labelled 'unidentified organism'.

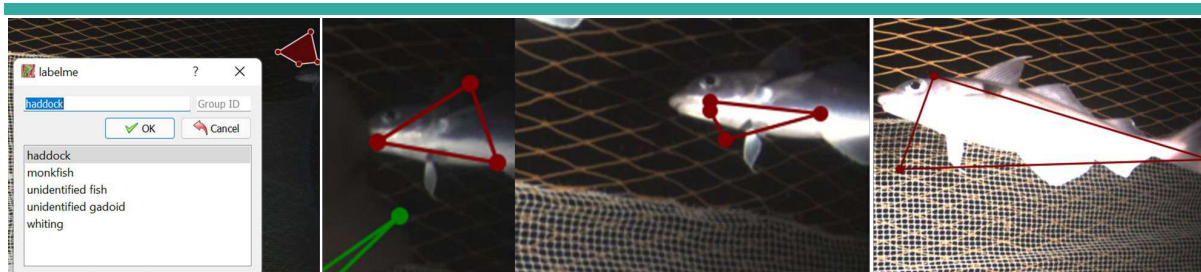


Figure 17. Classify fish by motion trajectories.

The sequence of images from Figure 17 shows how fish can be tracked in photos to make identification easier. The first image in the sequence is the head of a fish. The animal was identified as a haddock. Though there was not sufficient information in this image to show that this animal was a haddock, in subsequent images the animal stays in the frame long enough to show its full morphology and details (identification criteria previously outlined). This individual, therefore, was positively classified as a haddock in all images.



Figure 18. Issues classifying fish by motion trajectories.

Figure 18 shows an image with several animals. Many of them were classified as 'whiting' as a school of whiting had been captured by the trawl. Some of the positively identified whiting (in green) could be tracked across photos in the same way as Figure 17, however, with many fast-moving fish it was difficult to track individuals across multiple images.

## Fine Labelling

The annotation of the images from the marine biologist provided the species names of the organisms in the images. However, training deep learning algorithms requires bounding boxes or pixels to indicate the true location of the object you wish to detect. Therefore, it was necessary to use the LabelMe tool to draw more detailed and accurate polygon bounding boxes around the images in the dataset, and then create the "JSON" file format required for training the AI model.

Currently, we have prepared some labelled images for reference. Since labelling images is very time-consuming, it takes about 4-6 hours per 100 images, and may take longer if the number of fish in the image exceeds three or more. Figures 19 - 21 contain some good examples that will show the difference between coarse and fine labels.

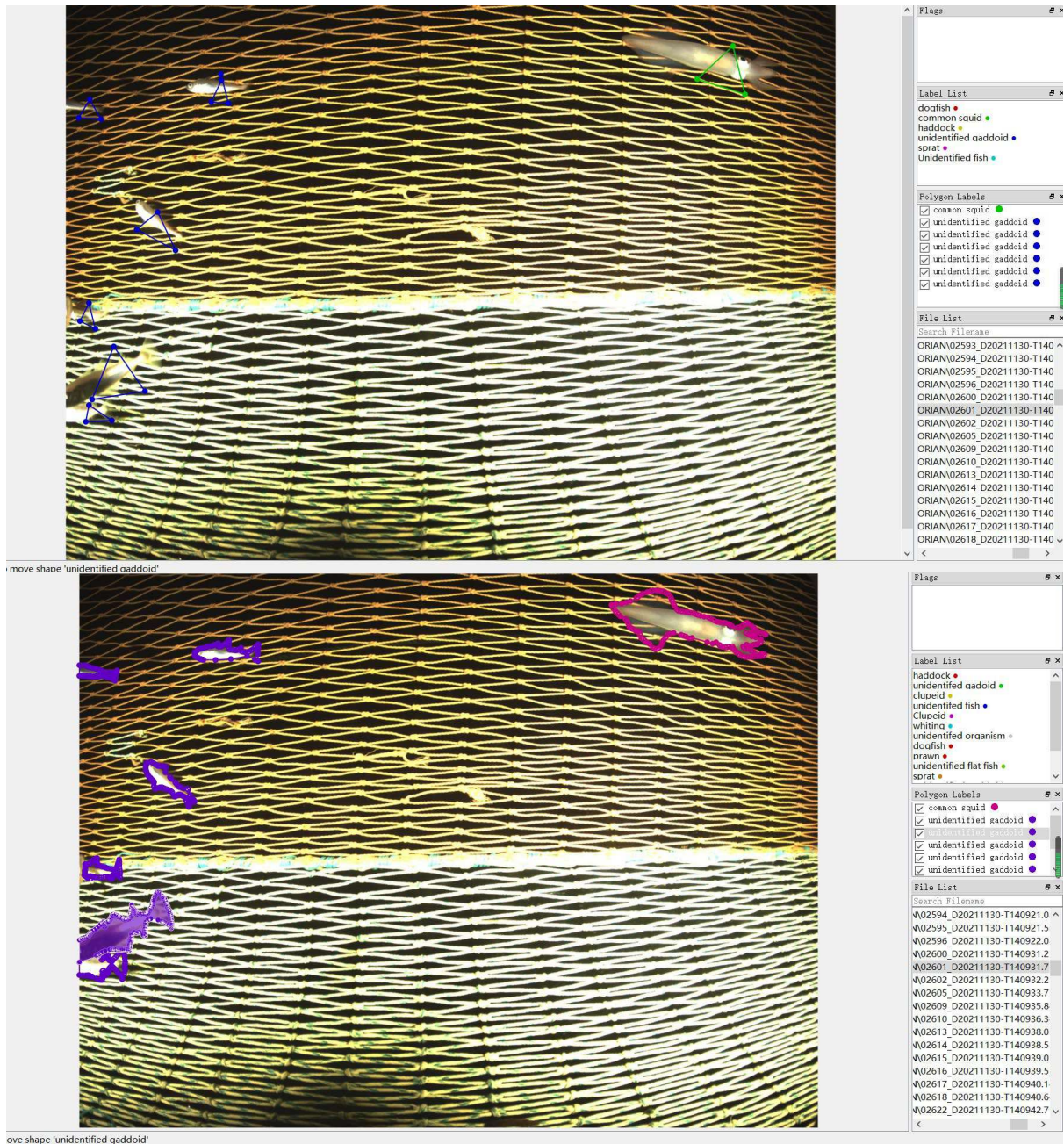


Figure 19. Example from Deployment 1 image number 02601. The images contain two different species, unidentified gadoid and common squid, annotated with polygons to better depict the body, head and tail of the fish.

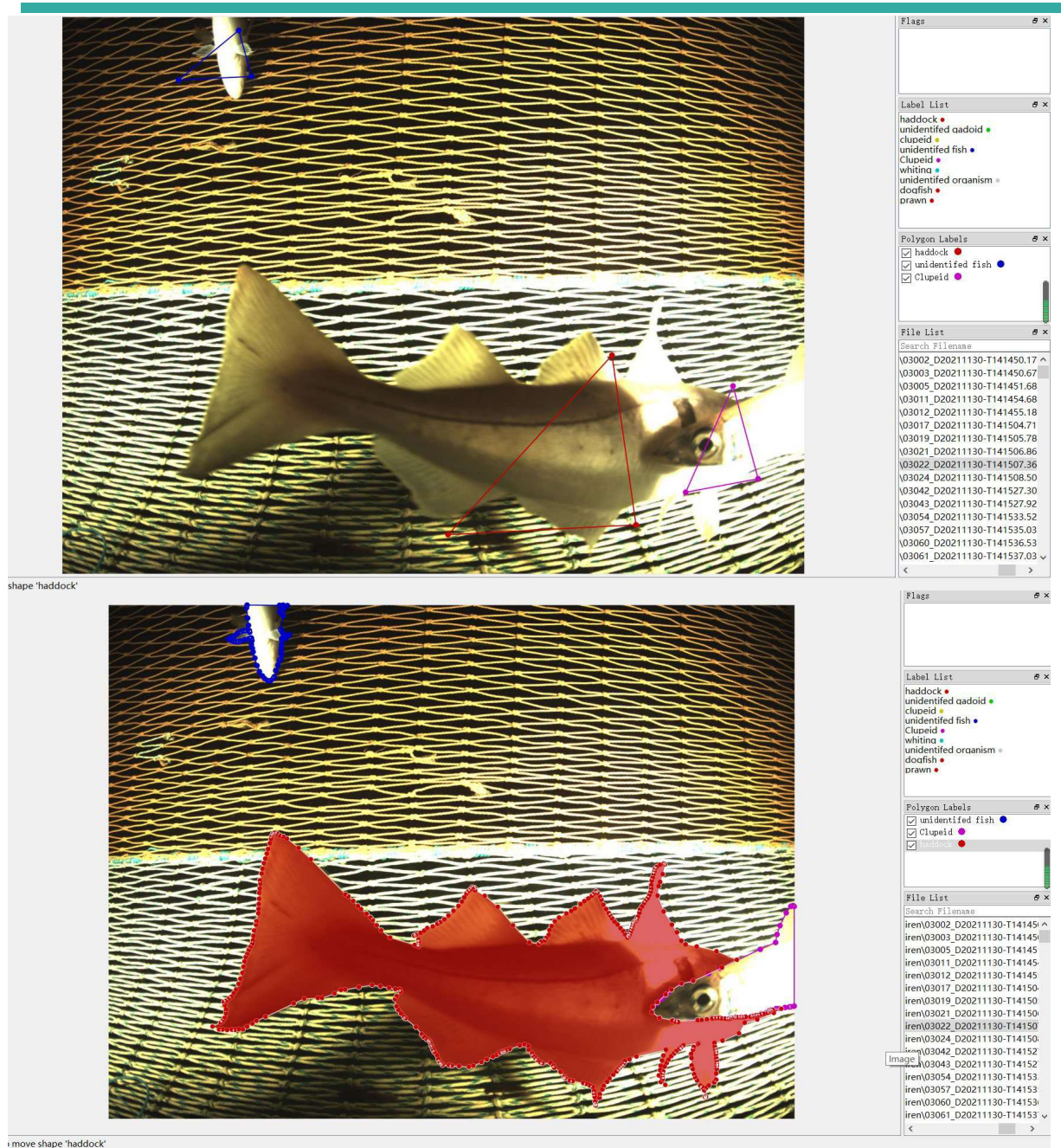


Figure 20. Example from Deployment1 image number 03022. The image contains three different species, unidentified fish, clupeid and haddock. Although their bodies are partially overlapping, they can still be clearly annotated with polygons using LabelMe.

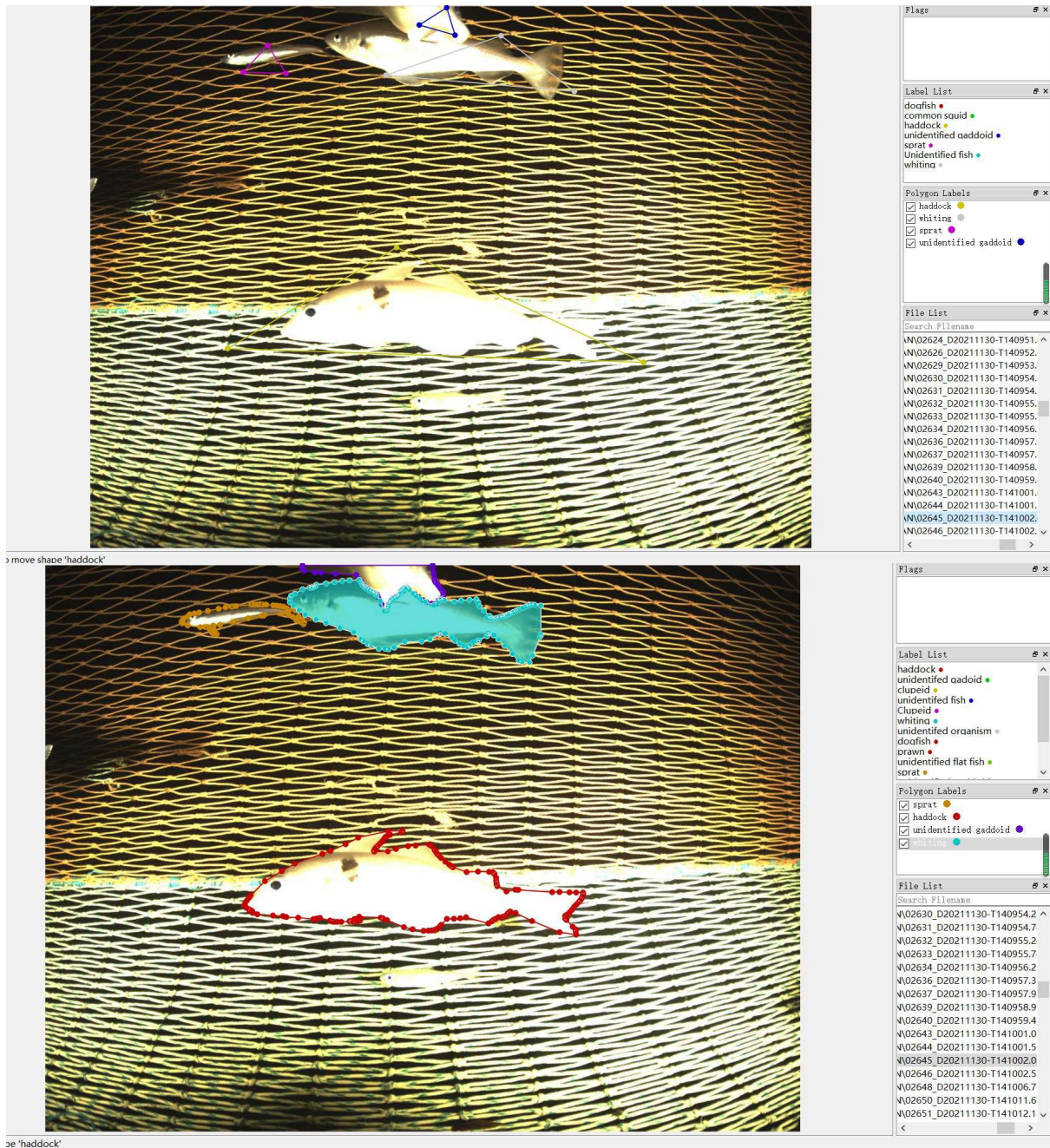


Figure 21. Example from Deployment1 image number 02645.

Different kinds of organisms are represented in these figures, they are haddock, whiting, sprat and unidentified gadoid. Obviously, they are all reflective to some extent, which means that precise labelling is not easy. A corresponding degree of magnification and adjustment of contrast and brightness are therefore performed. For example, the image numbered 02645 will be enlarged to 327% or more to facilitate clear identification of the boundaries (Figure 22).

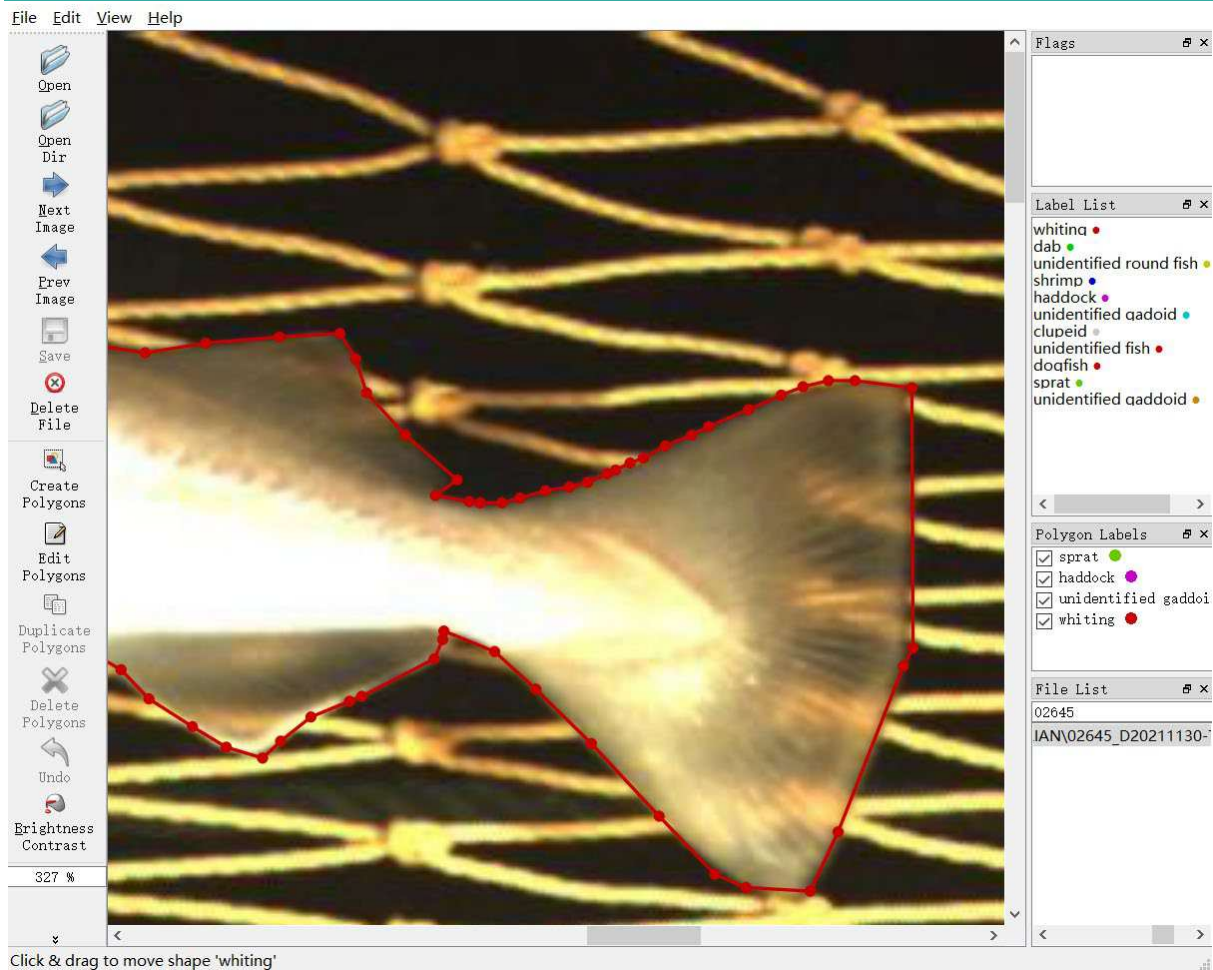


Figure 22. Partial enlargement image of number 02645.

## Statistics of labelled images

After these pre-processing steps such as image extraction, image filtering and image labelling, the images containing fish were arranged in different folders. Each deployment contained the folders: Coarse labelling, Fine labelling, Raw data and a table of statistics. Figure 23 shows the file classification in deployment1.

| Name                     | ↑ |
|--------------------------|---|
| Course Labelling         |   |
| Fine Labelling           |   |
| Raw Data                 |   |
| D1-NewDataset-1.1-R.xlsx | 👤 |
| D1-NewDataset-1.2-L.xlsx | 👤 |

Figure 23. File classification in Deployment1.



Table 3 shows the number of fish images classified by coarse labelling, for four different datasets collected on 30 November 2021, 2 December 2021 and 10 December 2021.

*Table 3. Number of Coarse labelled images.*

| <b>Coarse labelled images</b>           | <b>Deployment</b> | <b>Number of Coarse labelled images</b> |
|---|-------------------|---|
| <b>Deployment 1</b><br>30 November 2021 | 1.1-Right Camera  | 781                                     |
|   | 1.2-Left Camera   | 823                                     |
| <b>Deployment 2</b><br>2 December 2021  | 2.1-Right Camera  | 552                                     |
|   | 2.2-Left Camera   | 554                                     |
| <b>Deployment 3</b><br>10 December 2021 | 3.1-Right Camera  | 338                                     |
|   | 3.2-Left Camera   | 415                                     |
| <b>Deployment 4</b><br>10 December 2021 | 4.1-Right Camera  | 272                                     |
|   | 4.2-Left Camera   | 372                                     |
| <b>Total</b>                            |                   | <b>4107</b>                             |

Table 4 shows the number of fish images classified by fine labelling, including data from the Sparkling Star deployment collected on 19 July 2019 and data from deployments 1 to 4 collected in 2021.

*Table 4. Number of Fine labelled images.*

| <b>Fine labelled images</b>                      | <b>Deployment</b> | <b>Number of Fine labelled images</b> |
|--|-------------------|---------------------------------------|
| <b>Sparkling Star Deployment</b><br>19 July 2019 | Left              | 1000                                  |
|  | Right             | 1000                                  |
| <b>Deployment 1</b><br>30 November 2021          | 1.1-Right Camera  | 641                                   |
|  | 1.2-Left Camera   | 30                                    |
| <b>Deployment 2</b><br>2 December 2021           | 2.1-Right Camera  | 147                                   |
|  | 2.2-Left Camera   | 0                                     |
| <b>Total</b>                                     |                   | <b>2818</b>                           |

## Data Analysis Tools

Python tools were created to allow easy analysis of a dataset of stereo images and to enable filtering and correction of already labelled data (Figure 24). Stereo datasets are fundamentally different from other image datasets as each image must be linked with its pair. For this application, we needed to employ triangulation to accurately size fish. In the algorithm testing phase, we therefore needed to be able to quickly extract those images for which we do not have that data, or at least be able to immediately reject images for which are highly likely going to give poor results.

```
Total Dataset Size: 997
Missing Right Images: 0
Missing Left Images: 2
Mismatched Object Count: 46
Missing Left Labels: 0
Missing Right Labels: 0
Mismatched Label Set: 54
No Fish on One Side: 452
Full Stereo Fish: 268

Process finished with exit code 0
```

Figure 24. Sample Python Output after calling `.analyse()` on `StereoData` for `SparklingStar` dataset.

The `StereoData` class was created in Python. This can be instantiated by providing the paths to the left and right image directories. Image annotations relating to both left and right are included in these directories and take the form of the JSON files created by the `LabelMe` tool. A full description of properties and methods relating to this class is given in the documentation.

Once created, it can be used to identify pairs of images where the number or nature of the labelled objects is different on left and right sides (Figure 25). This can be used to quickly find and correct some labelling errors. As more deployments are labelled, this will help speed up the process and improve the data quality. It also offers ways of viewing paired images in one graphic within the Python environment and creating a directory of joined images for easy comparison.

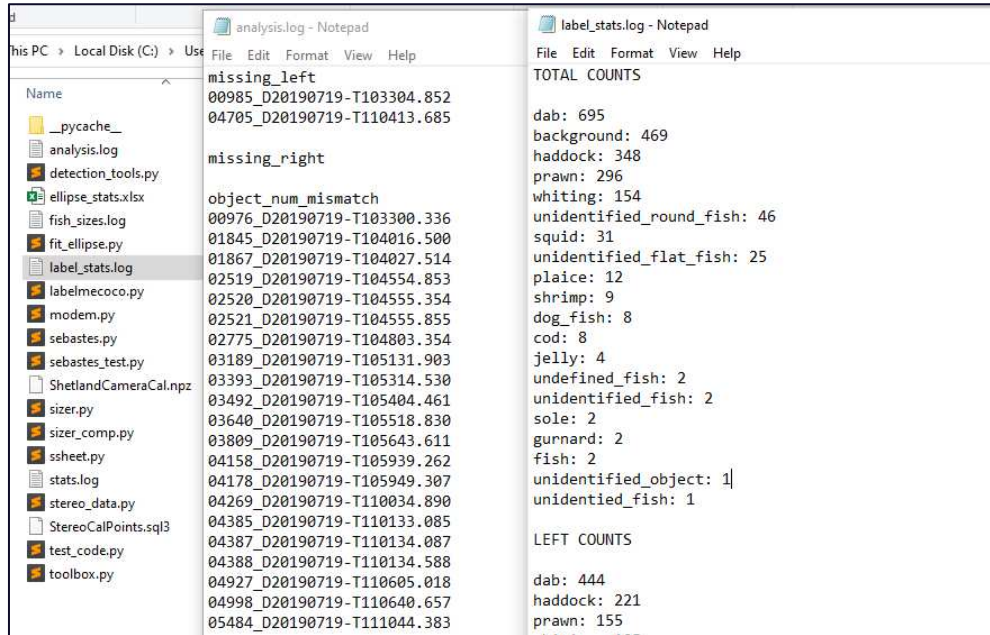


Figure 25. analysis.log and label\_stats.log show lists of images that may need revision, or which are unsuitable for the sizing algorithm and the class and object count summaries for both left and right datasets.

The label\_stats method also gives the ability to browse the statistics of the classes labelled so far in the dataset and to\_excel will export both the left and right images into an excel file for each with column totals for the classes (). This also will enable a ready-made spreadsheet filled with all the images to be labelled when a new deployment is started, making the process more efficient.

| Date       | Time         | Image | Number of organisms | Description | Reason for unidentified | running_total | sum | haddock | dogfish | unidentified | unidentified_flat_fish | unidentified_round_fish | jelly | prawn | shrimp | cod | thornback |
|------------|--------------|-------|---------------------|-------------|-------------------------|---------------|-----|---------|---------|--------------|------------------------|-------------------------|-------|-------|--------|-----|-----------|
| 19.07.2019 | 10:32:59.835 | 975   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:00.336 | 976   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:00.846 | 977   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:01.348 | 978   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:01.847 | 979   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:02.349 | 980   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:03.350 | 982   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:03.851 | 983   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:04.351 | 984   | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:05.353 | 986   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:05.854 | 987   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:06.354 | 988   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:06.854 | 989   | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:07.355 | 990   | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:08.370 | 992   | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:08.871 | 993   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:09.873 | 995   | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:10.374 | 996   | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:10.874 | 997   | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:11.376 | 998   | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:12.876 | 1001  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:13.377 | 1002  | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:13.877 | 1003  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:14.377 | 1004  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:14.878 | 1005  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:15.378 | 1006  | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:15.893 | 1007  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:16.392 | 1008  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:16.893 | 1009  | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:17.394 | 1010  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:17.895 | 1011  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:18.396 | 1012  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:18.897 | 1013  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:19.398 | 1014  | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:19.899 | 1015  | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:20.400 | 1016  | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:20.901 | 1017  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:21.402 | 1018  | 0                   |             |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:21.902 | 1019  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:22.403 | 1020  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |
| 19.07.2019 | 10:33:23.421 | 1022  | 1                   | prawn:1     |                         |               | 0   | 0       | 0       | 0            | 0                      | 0                       | 0     | 0     | 0      | 0   | 0         |

Figure 26. Generated Excel spreadsheet from StereoData. A blank version can be created when a new dataset is labelled.

## Recommendations for future labelling work

From the data analysis and observations in later sections, we have the following comments to make about how to refine the labelling process in future to improve the sizing and

detection processes:


- Use of the analysis tools to standardise and correct labelling as the dataset is labelled is recommended e.g., the use of multiple classes such as “fish”, “undefined fish”, “unidentified fish” etc
- Contours of fish at the edge of the scene should be drawn with only two vertices and right up against the edge of the image. Failure to do this can affect the detection accuracy.
- Use of the analysis.log file to see all images labelled as “faulty\_contours” (this is flagged when a contour consists of less than 6 points. In some cases, this means that the image may have missed the fine detail labelling. Such cases will affect both the detection algorithm and the testing of the sizing algorithm.
- Fine details outside of the main body shape of the object (e.g., legs on prawns) can adversely affect size estimation and it may be worth considering outline the most salient features using a less detailed outline.

## Data Conversion for Deep Learning Model

While all care was made to be consistent in labelling, some typos, labelling errors and inconsistencies will occur in the long term. Furthermore, the desired classes may change with time. The StereoData merge\_classes and ignore\_classes tools allow all images with a given label to be combined into another class, renamed or removed from the dataset as it is represented in the Python environment without making permanent changes to the original .JSON files.

The end goal of the data preparation, however, is to provide a .JSON file that can be used as an input to the training process of Mask RCNN [1]. This is a single file, as opposed to the individual JSONs created by LabelMe. merged\_jsons will create two JSON files for the left and right image sets. During object detection training, the dataset directories and matching annotation files must be added to the path\_catalog used by the Python script. If the labels need adjusted from those created in the initial LabelMe process, then the merge and ignore methods above should be called first before generating the annotation files. The merge allows more than one label to be applied.

While the StereoData class recognises images with no corresponding .JSON files as being labelled as \_\_background\_\_, this is not currently used in training. The algorithm does, however, expect the existence of such a class internally. For this reason, the class ids start at 1 rather than 0, with 0 being reserved for the \_\_background\_\_ class.



```

left_data.json - Notepad
File Edit Format View Help
  "file_name": "06725_D20190719-T112107.744_18563406.jpg"
},
"categories": [
  {
    "supercategory": "cod",
    "id": 1,
    "name": "cod"
  },
  {
    "supercategory": "dab",
    "id": 2,
    "name": "dab"
  },
  {
    "supercategory": "dog fish",
    "id": 3,
    "name": "dog fish"
  },
  {
    "supercategory": "fish",
    "id": 4,
    "name": "fish"
  }
]

```

Figure 27. Sample .JSON file used in the training process.

The structure of the combined .JSON files (Figure 27) is a dictionary structure containing the following:

- **“Images”**: A list of information for all labelled images in the dataset detailing the height, width, id and filename of each.
- **“Categories”**: A list of all classes found in the dataset detailing the supercategory, id (for the class) and name for each. Name and supercategory are the same in this project.
- **“Annotations”**: A list of annotations for each image, giving information on the segmentation (a flattened list of the (x,y) coordinated of each point in the contour), image\_id (that links with a specific file in “images”, bbox (bounding box of the contour using the coco convention), category\_id (numerical label assigned in “categories”), id (the unique id for each user annotation), “iscrowd” (0 in this project) and “area” (area of the contour used in IoU [2] calculations).

Subsequent changes to these files may be needed to expand, contract and/or renumber the categories to fit with later datasets or try different training settings. This can be done with some custom tools found in `json_toolbox.py`. `unify_labels` takes a list of .JSON files and gives them consistent “categories” and “annotations” entries. `remove_categories` and `remove_from_all` takes a list of classes we are not interested in and removes them from the training process by removing all annotations and images from the file or list of files respectively. It will not overwrite the previous file, but create a new JSON with a similar or user-defined name. `rename_categories` and `rename_all` perform similar functions, but allow for renaming or merging of classes. Note that these changes involve also making changes to the `predictor.py` and `.yaml` files used in training described in the Section on Training MaskRCNN Model.

## Biometric Identification

In this section, we explain the principles used to recognise different species of fishes. There are hundreds even thousands of species of fishes. In the current case, we only need to focus on few of them. Here, we focus on two main categories: Gadoids and other fish. For Gadoids, it can be further divided into whiting, cod, and haddock. For other fish, it can be further divided into monkfish, dog fish, clupeids, John dory, and flat fish. More specific details are provided in .



Figure 28. Image of the gadoid whiting distinguished by: silver, long and thin (more so than haddock and cod); white lateral line; elongate pelvic fin.

### Cod



Figure 29. Image of the gadoid cod distinguished by: larger and broad (more than haddock and whiting); Brown in colour and speckled; white lateral line; Spike like extension to the third dorsal fin sometimes seen; barbel under the mouth.

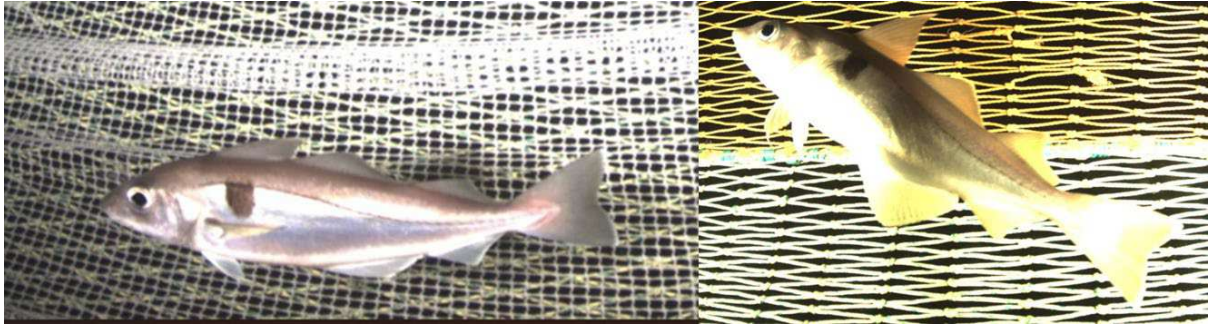


Figure 30. Image of the gadoid haddock, by far the most common gadoid, distinguished by: Silver in colour; Black spot (devil's thumb print); black lateral line; Tall and pointed first dorsal fin

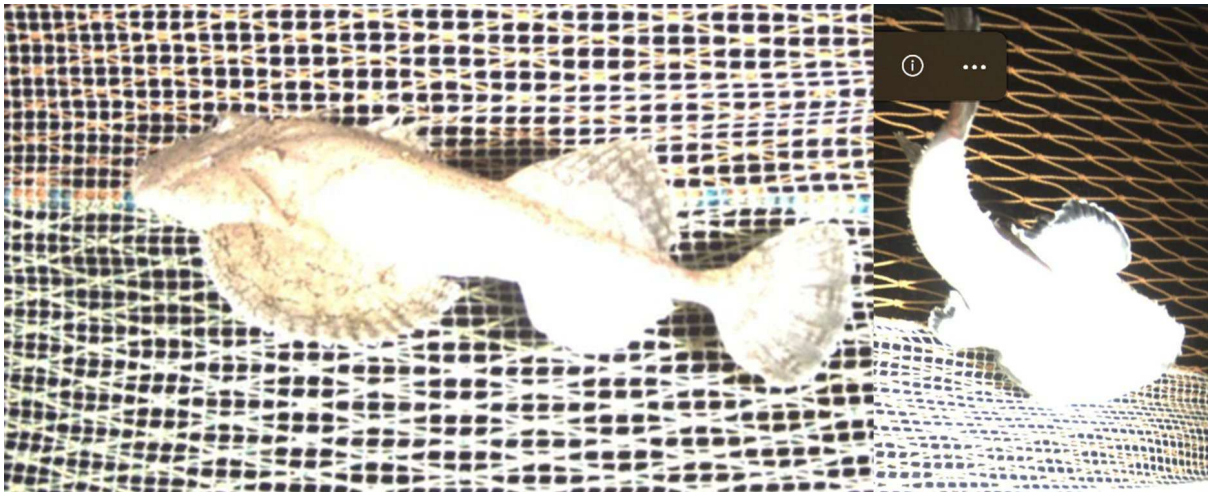


Figure 31. Images of the anglerfish or monkfish, distinguished by: Large and round; Dark brown in colour- white underside; Large fan shaped pectorals which are black tipped from the underside; broad triangular shaped body; spike-like protrusions from first dorsal



Figure 32. Image of the dogfish, the most common cartilaginous fish seen, distinguished by: Long Cream coloured and speckled body- almost eel-like; Broad pectorals; Primary and secondary dorsal fins towards the posterior end of the animals

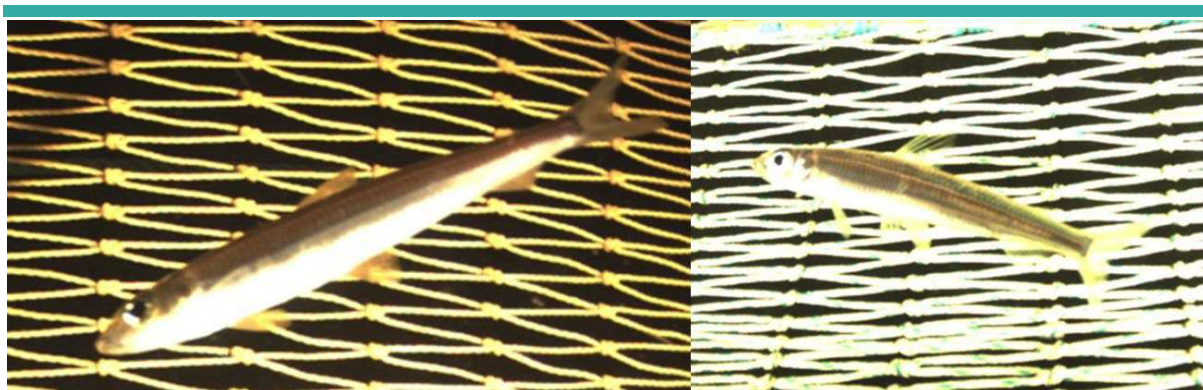


Figure 33. Image of clupeids (herring and sprat), distinguished by: Lunate- caudal tail fin; Small translucent fins -often not visible other than dorsal rays; Often oil-slick colouration. Difficult to identify down to species level in these images (possibly herring on the left and sprat on the right though this is speculative)



Figure 34. Image of John Dory, distinguished by: Pink and white with a round shape; Primary dorsal fin with string like spines; Large black spot (St peter's thumb print); Spines (5) on anal fin .



Figure 35- Images of flatfish. The flatfish seen in these images are all brown in colour. The shape varies for each species but can be very difficult to see unless orientated perfectly. This makes identification challenging. When seen from the side often appear worm/ lamprey-like. Dab (A) are round and often speckled with small black spots, but this is often not enough to identify them. European Plaice (B) are diamond shaped the easiest to identify with large orange spots. Turbot (C) are similar in colour to dab but are much broader. They have an asymmetrical diamond shape.



## Deep Learning Model for Fish Detection and Recognition

### Instance Segmentation and MaskRCNN

The current network investigated so far in the project is Mask-RCNN (<https://arxiv.org/abs/1703.06870>) with code adapted from Facebook Research (<https://github.com/facebookresearch/maskrcnn-benchmark>). This is a semantic segmentation network, which means that it takes an image and attempts to label the objects that appear within it while generating an accurate boundary outline around them. Note that this process involves simultaneously classifying and generating a contour for every detected object and that the two are intertwined during training.

The output of the inference stage of the trained network will be in the form of a list of identified objects restricted to the recognised classes and a binary mask for each of these in the form of an array the size of the original image with 1 indicating that pixel is part of the object and zero otherwise.

The essential components of this network are as follows (Figure 36):

- **CNN Backbone:** Feature extraction network that encodes a given image or region as a vector. Our default choice is FNet, but other options are available. E.g., ResNet-50
- **RPN (Region proposal network):** Which generates a selection of bounding boxes according to predefined ratios and sizes that can be used as a shortlist for classification of possible objects.

**ROI (Region of Interest) Head:** Which simultaneously performs the regression and classification of the bounding boxes from the RPN and predicts a binary mask for each ROI bounding box.

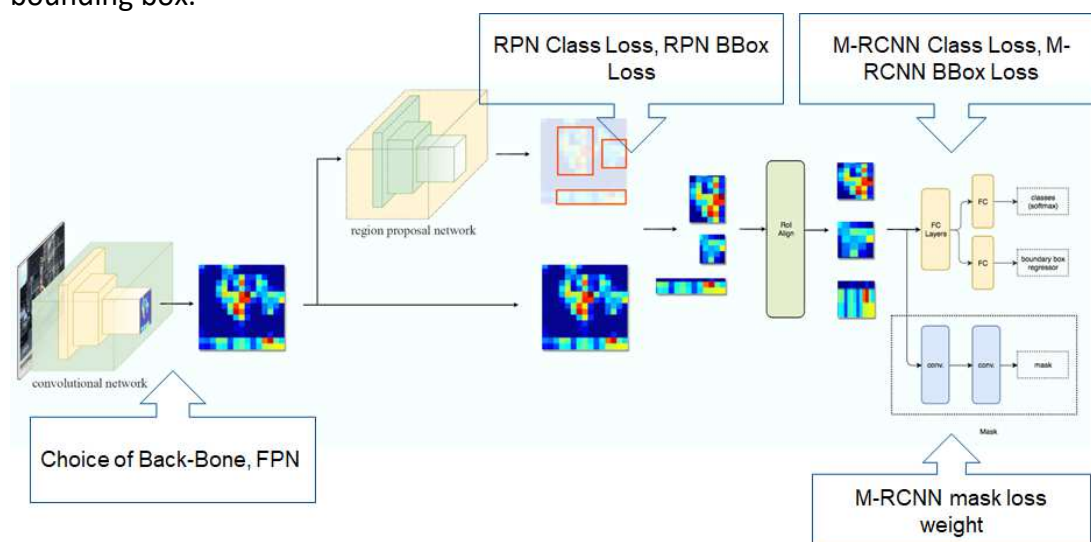


Figure 36. Architecture of MaskRCNN: Training involves optimising a loss function comprised of the four loss functions above.

### Training MaskRCNN Model

#### Initial Set-up

To train multiple datasets under a variety of different settings, we need to set up a file structure recognisable to our training algorithm (Figure 37).

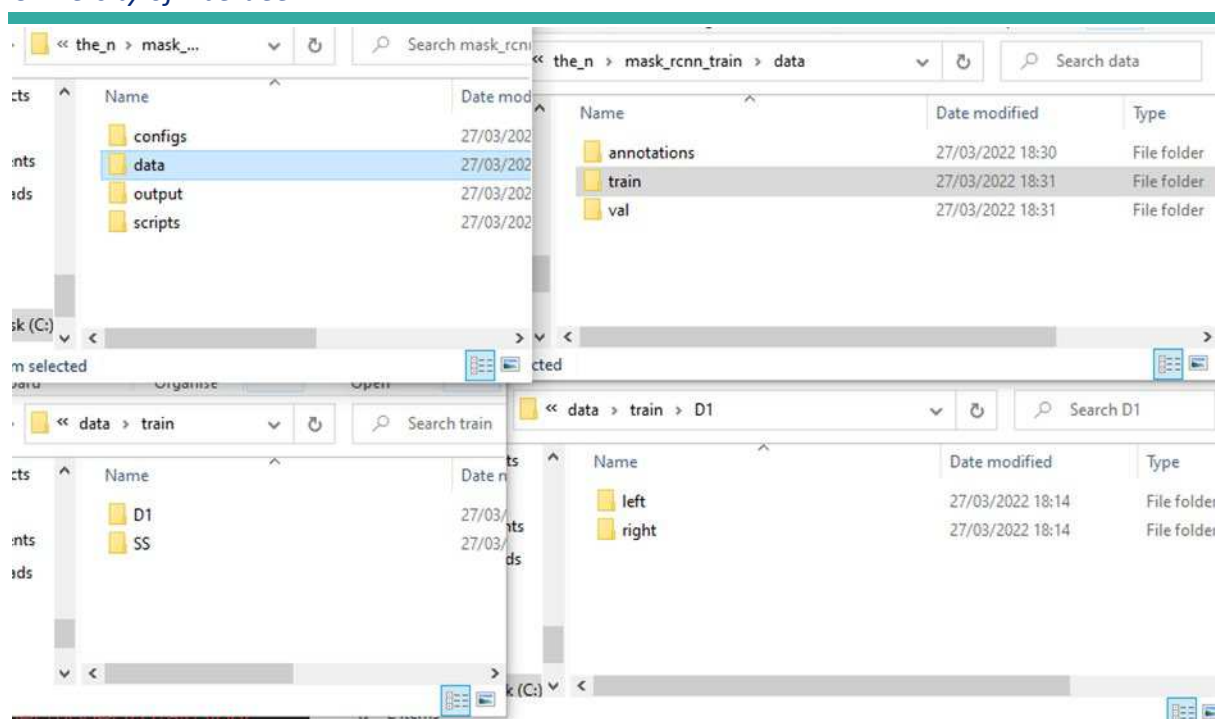


Figure 37. File structure of dataset and annotations

In the base directory, we call it `mask_rcnn_train`, the following directories must be created:

**Configs:** Where we store the configurations of the network structure and training, namely the architectures used for each component mentioned above and their training parameters (where applicable), the learning rate and related solver parameters, the designated directory for training output (including periodic saves of the network at various points in training) and a pointer to the `path_catalog` which holds paths to all the data and associated annotations (Figure 38). Each config file is in `.yaml` format. This is also the location of the `paths_catalog.py` referred to above.

**Data:** Which stores all the training and test data for the project. It is subdivided into `train`, `val` and `annotations`. `Train` contains a directory for each deployment collected which each themselves contain subdirectories labelled `left` and `right` for each side of the stereo camera recording. These contain all the relevant images in JPEG format. The ground truth class labels and segmentation data for each are stored in the JSON format generated from StereoData as described in the Section on Data Conversion for Deep Learning Model. The paths of both the directory containing the images and the corresponding annotations file must be contained within the `path_catalog` so that the configuration files can gain access to the data, so these must be expanded if and when new data is labelled. Note also that we should use the tools in `json_toolbox.py` to ensure consistency in labelling across all datasets.

**Output:** This will contain a series of subdirectories generated automatically when we undergo one training session.

**Scripts:** Contains python code essential in training and testing the output of our networks (namely `train.py`, `test.py`, `fit_test.py` and `predictor.py`).

```

class DatasetCatalog(object):
    DATA_DIR = "data"
    DATASETS = {
        "SS_left": {
            "img_dir": "train/SS/left",
            "ann_file": "annotations/SS_left.json"
        },
        "SS_right": {
            "img_dir": "train/SS/right",
            "ann_file": "annotations/SS_right.json"
        },
        "D1_left": {
            "img_dir": "train/D1/left",
            "ann_file": "annotations/D1_left.json"
        },
        "D1_right": {
            "img_dir": "train/D1/right",
            "ann_file": "annotations/D1_right.json"
        },
        "D2_left": {
            "img_dir": "train/D2/left",
            "ann_file": "annotations/D1_left.json"
        },
        "D2_right": {
            "img_dir": "train/D2/right",
            "ann_file": "annotations/D1_right.json"
        }
    }
}

```

Figure 38. `paths_catalog.py` in the `configs` directory must be amended with the locations of the images and associated combined annotations file.

## Training

While operating in a virtual environment with the correct packages installed (see section 7.2), the command-line instructions to train a network from scratch is as follows. This is run from the directory shown above.

```
python scripts/train_net.py --config-file "configs/default_train.yaml"
```

This will train according to the training parameters in the config file. For a test run involving training on a single GPU for a single dataset of 500 labelled images, this took around 72 mins.

### Model Training Hyperparameters

A relatively good initial starting point for training had the following setup. It should be noted that several other iterations before this had effectively zero accuracy with only rare examples of finding any objects at all (Figure 39).

## Testing and Inspection

An evaluation will be saved in the outputs directory at the end of the training process, giving accuracy statistics in terms of Average Precision [3] at various scales (see <https://cocodataset.org/#detection-eval>). For a visual inspection of what masks are being generated, run the following line in command prompt from the main directory.

```
python fit_test.py --img_dir=IMGS_PATH --save-dir=SAVE_DEST
```

This will take all images in the directory with path `IMGS_PATH`, process them with the model indicated by `default_val.yaml`, and save them in a directory given by path `SAVE_DEST`. By default, this will use the `.yaml` file stated, but another can be chosen using the option `--config-file=ALT_YAML`. Visually inspection for modes of failure can help diagnose which parameters need changing.

```

MODEL:
  META_ARCHITECTURE: "GeneralizedRCNN"
  WEIGHT: ""
  BACKBONE:
    CONV_BODY: FBNet
  FBNET:
    ARCH: "default"
    BN_TYPE: "bn"
    WIDTH_DIVISOR: 8
    DW_CONV_SKIP_BN: True
    DW_CONV_SKIP_RELU: True
    DET_HEAD_LAST_SCALE: 0.0
  RPN:
    ANCHOR_SIZES: (16, 32, 64, 128, 256)
    ANCHOR_STRIDE: (16, )
    BATCH_SIZE_PER_IMAGE: 256
    PRE_NMS_TOP_N_TRAIN: 6000
    PRE_NMS_TOP_N_TEST: 6000
    POST_NMS_TOP_N_TRAIN: 2000
    POST_NMS_TOP_N_TEST: 100
    RPN_HEAD: FBNet.rpn_head
  ROI_HEADS:
    BATCH_SIZE_PER_IMAGE: 256
  ROI_BOX_HEAD:
    POOLER_RESOLUTION: 6
    FEATURE_EXTRACTOR: FBNet.roi_head
    NUM_CLASSES: 17
  ROI_MASK_HEAD:
    POOLER_RESOLUTION: 6
    ROI_BOX_HEAD:
      POOLER_RESOLUTION: 6
      FEATURE_EXTRACTOR: FBNet.roi_head
      NUM_CLASSES: 17
    ROI_MASK_HEAD:
      POOLER_RESOLUTION: 6
      FEATURE_EXTRACTOR: FBNet.roi_head_mask
      PREDICTOR: "MaskRCNNConv1x1Predictor"
      RESOLUTION: 12
      SHARE_BOX_FEATURE_EXTRACTOR: False
    MASK_ON: True
  DATASETS:
    TRAIN: ("coco_2014_train",)
    TEST: ("coco_2014_val",)
  SOLVER:
    BASE_LR: 0.06
    WARMUP_FACTOR: 0.1
    WEIGHT_DECAY: 0.0001
    STEPS: (60000, 80000)
    MAX_ITER: 100000
    IMS_PER_BATCH: 1
  TEST:
    IMS_PER_BATCH: 1
  INPUT:
    MIN_SIZE_TRAIN: (320, )
    MAX_SIZE_TRAIN: 640
    MIN_SIZE_TEST: 320
    MAX_SIZE_TEST: 640
    PIXEL_MEAN: [103.53, 116.28, 123.675]
    PIXEL_STD: [57.375, 57.12, 58.395]
  OUTPUT_DIR: "./dyoutput"
  PATHS_CATALOG: "dyconfigs/paths_catalog.py"

```

Figure 39. Contents of a sample configuration file with initial settings.

## Initial assessment

Figure 40 shows some generated contours on the training set. Figure 40 c, d, e, f and i show, although the classifications of d and f are incorrect. The initial training shows a propensity to over-label objects as “fish” for which we only have 2 examples in the training set. Unifying some of the labels may help with this. Figure 40 j, g and b either don’t identify that there is a fish present or only identify part of the fish. Where this occurs, it seems to be the case that the fish is very close to the camera. Adjusting some of the bounding box sizes in the RPN would possibly help with this. The Section on Fish Size Detection will also show that these close-up fish are also difficult to give an accurate size measurement for. Figure 40 a, f and h show the most common problem appearing in the initial training - that of contours that are good but appear in multiple instances (often labelled differently). This may be a result of overfitting.

Figure 41 shows the Average Precision figures for this benchmark. These are relatively poor in terms of where we want to be but shows promise as we have not yet explored many variations in hyperparameters or further techniques such as data augmentation. We also are using a fairly small dataset so far with labellings that are not yet standardised. In the next section we will show some routes we have taken, but not yet fully explored due to time constraints on the project’s current phase.

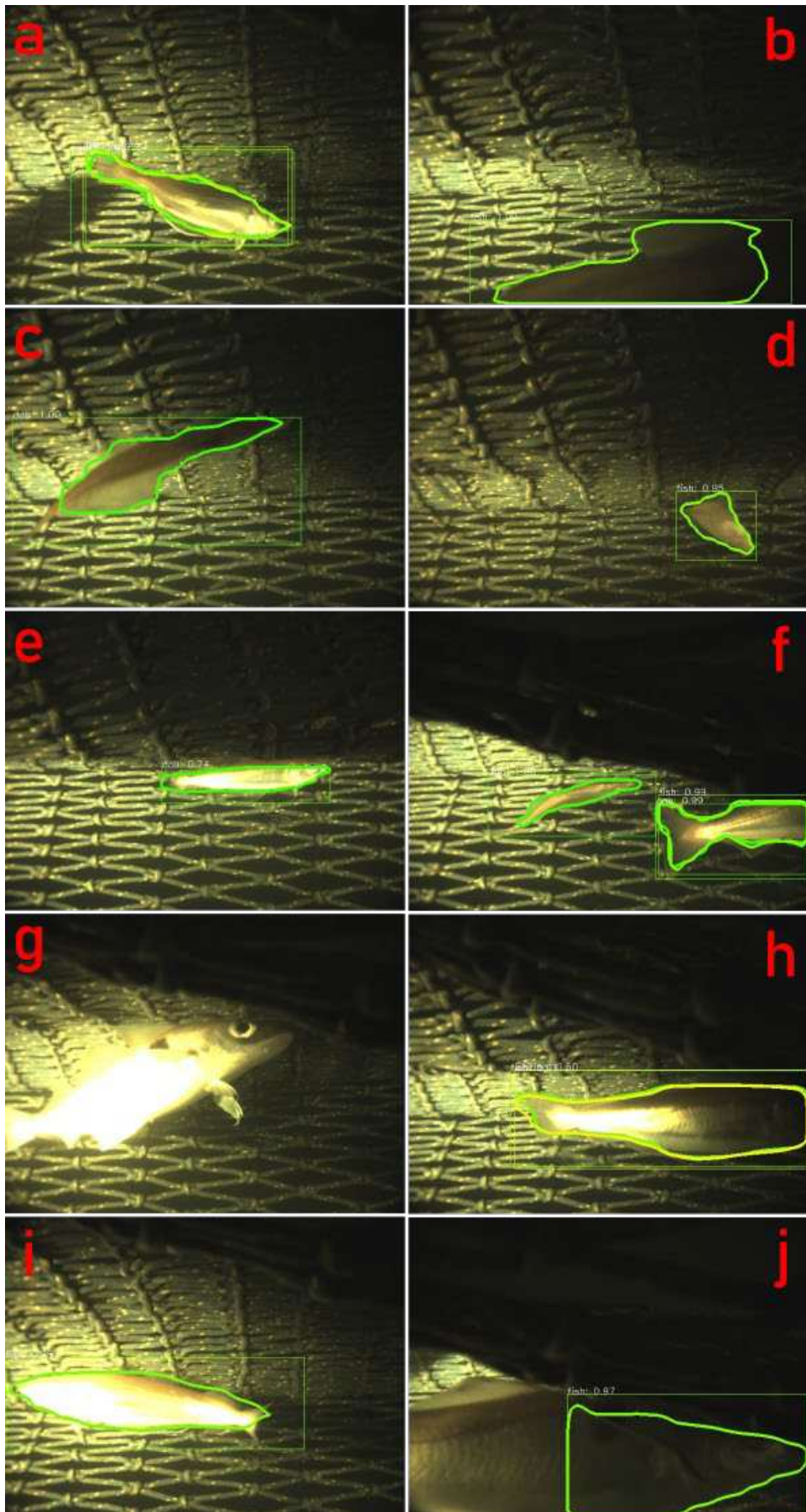


Figure 40. Good and Bad Detection Examples.

```

0
1
Anaconda Powershell Prompt (anaconda3)
Loading and preparing results...
DONE (t=0.03s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *segm*
DONE (t=0.37s).
Accumulating evaluation results...
DONE (t=0.08s).
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.131
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.286
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.109
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.007
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.149
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.213
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.220
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.220
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.014
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.251
2022-03-27 18:54:08,442 maskrcnn_benchmark.inference INFO:
Task: bbox
AP, AP50, AP75, APs, APm, APl
0.1760, 0.2986, 0.1743, -1.0000, 0.0253, 0.1989
Task: segm
AP, AP50, AP75, APs, APm, APl
0.1307, 0.2857, 0.1093, -1.0000, 0.0066, 0.1489

```

Figure 41. Benchmark for Precision and Recall.

## Model Improvement

With some minor changes, namely increasing batch size to 8 and increasing our training images to 1000, the above was greatly improved (Figure 42). A greater batch size was tried, but it required too much memory (9 Gb+). The improved version had a max memory allocation of 2864 Mb and there is some room for exploration here. The running time was 88 minutes with 20000 iterations. This is, nevertheless, only on the training data and we are not testing on data the training algorithm has not seen before. At this stage, however, we are getting ballpark figures for what training settings to use to train a more robust system. This will need to be fine-tuned.

```

Accumulating evaluation results...
DONE (t=0.14s).
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.755
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.997
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.905
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.611
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.759
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.771
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.778
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.778
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.641
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.784
Loading and preparing results...
DONE (t=0.03s)
creating index...
index created!
Running per image evaluation...
Evaluate annotation type *segm*
DONE (t=0.66s).
Accumulating evaluation results...
DONE (t=0.14s).
Average Precision (AP) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.663
Average Precision (AP) @[ IoU=0.50 | area= all | maxDets=100 ] = 0.978
Average Precision (AP) @[ IoU=0.75 | area= all | maxDets=100 ] = 0.749
Average Precision (AP) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Precision (AP) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.504
Average Precision (AP) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.680
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 1 ] = 0.678
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets= 10 ] = 0.685
Average Recall (AR) @[ IoU=0.50:0.95 | area= all | maxDets=100 ] = 0.685
Average Recall (AR) @[ IoU=0.50:0.95 | area= small | maxDets=100 ] = -1.000
Average Recall (AR) @[ IoU=0.50:0.95 | area=medium | maxDets=100 ] = 0.554
Average Recall (AR) @[ IoU=0.50:0.95 | area= large | maxDets=100 ] = 0.702
2022-04-01 10:30:00,261 maskrcnn_benchmark.inference INFO:
Task: bbox
AP, AP50, AP75, APs, APm, AP1
0.7554, 0.9968, 0.9053, -1.0000, 0.6112, 0.7591
Task: segm
AP, AP50, AP75, APs, APm, AP1
0.6626, 0.9778, 0.7490, -1.0000, 0.5044, 0.6803

```

Figure 42. Improved results on training data.

## Fish Size Detection

The size detection portion is dependent upon the accuracy of the contours obtained from Mask RCNN, but the methods used need to be tested separately under the assumption that the contours are well-defined. The process involves two stages: ellipse fitting and stereographic projection.

### Ellipse Fitting

Sizing of a fish will be estimated by extracting a best-fit ellipse from the contour of the identified object. An elliptical template was chosen because bounding boxes don't give us an accurate idea of maximum length and finding the maximum distance between any two points in the contour does not allow us to estimate the size of fish partially out of shot and will, in other cases, overestimate the size of the fish.

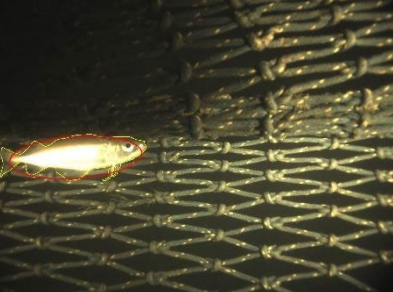
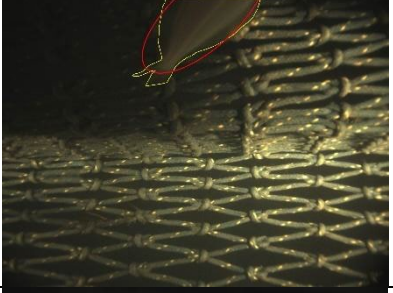

We test using the contours labelled as ground truth in section 3. In the final process, however, we will require that the shape extracted from the segmentation algorithm is sufficiently accurate. This shape will be in the form of a binary mask over the pixels indicating 1 if it is part

of the identified object and 0 otherwise. We have provided code to extract the ellipse directly from this binary mask.

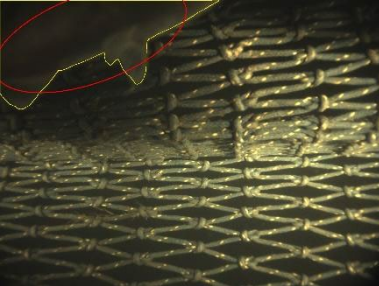
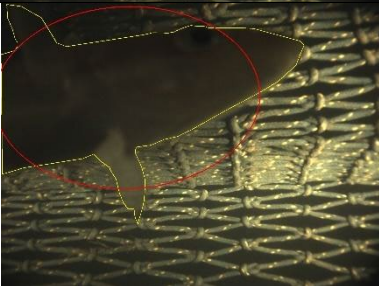

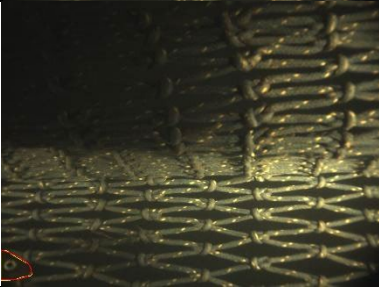


Several methods were investigated under two different point selection methods. The first selection method uses the given points of the contour as input, whereas the second takes a uniform sample of points along the perimeter of the contour. Ellipse fitting was performed using a least squares method and a smallest enclosing ellipse method, but the former proved to be more useful in simulating the real-life method of measuring fish size i.e., they are usually estimated by body length not including the tail.

The method `extract_ellipse_fits` can be called on the `StereoData` object to inspect how ellipses were fit to ground truth examples. An additional function `process_dir` takes a single directory containing images and associated .JSON files, calculates fitted ellipses and saves the original images with the fitted ellipse drawn round the object. This serves for an initial visual inspection of how appropriate the algorithm is. The table below shows some examples of good and bad examples with suggestions of how to cope with possible problems. The same phenomena were observed across all methods described earlier.

Table 5. Good and bad ellipse fits based on ground truth contours with solutions and analysis.

| Image Example   | Notes  |
|---|--|
|   | <p><i>Ideal:</i> Good placement of fish. Ellipse estimates approx. practical size of fish. i.e., body portion not counting tail.</p>   |
|  | <p><i>Off-Shot:</i> Fish not ideally placed (partially off shot). Good size estimate nonetheless. This generally occurs when we have a sizable portion of the fish visible in the image and it is cut off by only one side.</p>  |
|  | <p><i>Irregularly shaped contour:</i> Least square method of fit fails as the appendage skews the least squared method by moving the centre. Providing ground truth contours limited to the fish body may help here but would require a different labelling strategy. An alternative is to use the image processing trick of <b>erosion</b> to remove extremities, although there would be some additional modifications needed.</p> |



|  |   |   |
|--|---|---|
|  |    | <p><i>Corner case:</i> Where the fish is found cut off on two edges, the size estimation is usually very bad as it will try to fit the ellipse within the image. Luckily, these cases can be filtered out when doing the size estimation by sensing that the bounding box of the object is at the extreme edges of the image.</p> |
|  |    | <p><i>Close-up:</i> These generally give bad size estimates. Since both left and right images are likely to be close shots, it may be best to ignore detections that take up too much of the field of view. This can be filtered by rejecting bounding boxes above a given threshold.</p>   |
|  |   | <p><i>Tail end:</i> Shape is not close to an ellipse and so fits poorly. We may be able to assume this fish would appear in a previous shot, but movement of the camera and water current could cause this not to be the case.</p>  |
|  |  | <p><i>Peeking Fish:</i> It may be the case that only the tip is identified, but it's not clear that the fish is correctly sized. Since it's highly likely this fish will pass into shot and we are not interested in very small fish, it makes sense to reject these instances for a more favourable image.</p>                   |
|  |  | <p><i>Obstruction:</i> Several images in the SparklingStar dataset suffer from not having a clear view because of the net. This can be easily solved by a recording set-up that prevents or minimises this.</p>   |
|  |  | <p><i>Fish-on-Fish:</i> Where a fish is undersized because it is hidden behind another fish. Unlike the above, this is not easily fixable. Without a specific method to counter this error, we may need to take this as an accepted failure case.</p>   |

## *Statistical Analysis of Ellipse Fitting*

To test the fitting algorithms more thoroughly, sets of 1000 ellipses of a range of scales, eccentricities and rotation were tested by extracting a series of points and adding noise to each ground truth ellipse and attempting to reconstruct the ellipse parameters using the chosen method. The ellipse parameters are the location of the centre pixel in the image ( $x_c$ ,  $y_c$ ), the maximum and minimum distances from this centre to the perimeter ( $a$  and  $b$ ), and the angle between the maximum diameter and the horizontal ( $\Theta$ ).

The ranges used in the simulation were lengths from 100 to 1000 pixels, eccentricities from 0.95 to 0.995 (approximately length to width ratios of between 3:1 to 10:1) and  $\Theta$  from -30 to 30 degrees clockwise. The centres were fixed at (0, 0) as accuracy is independent of the centre. The noise added to each generated point was uniformly distributed with magnitude between 10 and 30% of the size of the ideal ellipse. A sampling process was also employed to induce further irregularity in the points to be fitted while still maintaining the same number of points in the contour.

The errors between the fitted ellipse and the ground truth parameters were measured in terms of absolute relative error for  $a$  and  $b$ , distance from the true centre divided by the average radius and the absolute value of the error for the angle. Different circumstances were modelled:

- Whole fish entirely within the image.
- Fish truncated by the far left-hand side.
- “Corner Fish”: Fish truncated from both left and bottom sides
- “Close-up”: Fish truncated from the far left and right-hand sides.

The number of points in the contour was also varied between 10 and 30, which is a reasonable estimation of the manually labelled contours. Note also that there is a uniform sampling method to increase the number of points if needed.

The algorithm for fitting ellipses requires at least 5 points to work. A list of images with contours constructed using less than 5 points can be found in `analysis.log` generated by the StereoData analysis. Again, it is possible to extract the required number of points using the “`fix_contours`” argument when calling `extract_lengths` on the dataset (see sizing section).

## *Results of Ellipse Fitting*

### **a and b parameters**

With contour size of 30 points and 10% noise, the mean error for  $a$  and  $b$  is approx. 1% with std of 1%. This rises to approx. 2% mean error and 1.5% std with 20% noise and 3.3% mean error and 2.5% std at 30% noise. Reducing the number of points to 20 increases the worst case to 4.5% mean error and 3.5% std and reducing still to 10 points gives a mean error of 8.5% with 10% std. With some additional computation cost, the contour sampling method can be used to increase accuracy.

For the left-truncated fish, 30-point contours lead to a mean error of 5% and std of 5% for 10% noise and up to 15% mean error and 11% std. This becomes worse with decreasing number of contour points.

For the corner fish case, even the best-case scenario has 17% mean error and 19% std, rising

as high as 35% mean error 200% std for 10 points with the highest level of noise. This can be expected when the ellipse is contained within the image and is only a portion of the whole fish.

Surprisingly, the close-up case where the fish is truncated from both left and right sides had a high error with 17% mean error and std of 25%, even at 10% noise and 30 points. This can be explained by the fact that the curvature of the shape is low in the central portion and so small errors in this can project to large changes in where the end points of the ellipse exist out of shot.

### Centre position

The location of the centre for 30-point contours is roughly similar to the error in a and b, with a slightly increased error in the 30% noise case with 4% mean and 3% std. A slightly higher error is acceptable in this case as a larger error in the centre can be compensated by a small change in rotation.

### Rotation

In general, the angle of the body of the fish is very accurate, only failing in the most sparsely sampled, irregularly shaped and truncated versions of test ellipses. The mean error for 30 points is 0.6 degrees with std of 0.5 degrees at 30% noise.

### Conclusion

We expect most real fish of interest to us to have a regular shape more consistent with low levels of noise and hence the ellipse fitting will work well. The exploration of edge cases gives some cause for concern, even if we have a very nicely defined contour, although this can be counteracted by using the sampling method a by detecting and rejecting these cases in the live system in favour of accuracy. Applying noise does, however, give us some indication on what to expect if the contours from the object detection are not good as we can view them as the true contour with noise added.

### Size Estimation

The sizing algorithm uses code sourced from the Sebastes software to convert 2D locations on each image to points in 3D space and then use the projected points of the extrema of the fitted ellipses along the maximum diameter to give a size.

A FishFitter class was created to perform all the necessary calculations. It is instantiated with a file containing the calibration details of the stereo camera set-up for each deployment. This can take the form of a numpy .npz file generated using calibration on OpenCV, a .mat file generated using camera calibration on MATLAB, or a .JSON file containing a dictionary with following parameters:

**Camera Matrices ("LM", "RM"):** For the left and right cameras which contain information about the focal length of the camera and the location of the centre of the image.

**Distortion coefficients ("ldist", "rdist"):** Which describe the radial and tangential distortion of the optical field.

**Rotation Matrix ("R"):** Describing the relative position of the right camera to the left camera.

**Translation Matrix ("T"):** Describing the relative position of the right camera to the left camera.

These are components of the pinhole camera model.<sup>1</sup> Two things should be noted here. Firstly, the .mat and .npz files give a different number of parameters for the distortion vector (5 vs 6). The difference is only related to the degree of the polynomial used in the radial distortion calculation.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f_x x'' + c_x \\ f_y y'' + c_y \end{bmatrix}$$

$$\begin{bmatrix} x'' \\ y'' \end{bmatrix} = \begin{bmatrix} x' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + 2p_1 x' y' + p_2 (r^2 + 2x'^2) + s_1 r^2 + s_2 r^4 \\ y' \frac{1+k_1 r^2+k_2 r^4+k_3 r^6}{1+k_4 r^2+k_5 r^4+k_6 r^6} + p_1 (r^2 + 2y'^2) + 2p_2 x' y' + s_3 r^2 + s_4 r^4 \end{bmatrix}$$

$$r^2 = x'^2 + y'^2$$

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} X_c/Z_c \\ Y_c/Z_c \end{bmatrix},$$

$r$  is the radius of a point from the focal centre of the lens and  $x'$  and  $y'$  are homogeneous coordinates of the 2D image.  $f_x$  and  $f_y$  are focal properties of the camera lens and are normally the same value.  $C_x$  and  $C_y$  are the centre of the image.  $k_1$  to  $k_6$  are radial distortion coefficients and  $p_1$  and  $p_2$  are tangential distortion coefficients.  $s_1$  to  $s_4$  are the thin prism distortion coefficients and are not used in this application. The  $ldist$  and  $rdist$  parameters obtained in the calibration process give variables in the following order:

$$\text{Distortion coefficients} = (k_1 \quad k_2 \quad p_1 \quad p_2 \quad k_3)$$

Where  $k_3$  is set as 0 for the .mat calibrations. These parameters should also be monotonic and, if they are not, the documentation says to consider the calibration "failed". A non-monotonic calibration can lead to two points in 2D space being mapped to the same coordinate in 3D space. If the calibration is incorrect, then the projections are mostly accurate in the centre of the image, but increase in inaccuracy towards the edges. This would be a problem as this case happens often in our application. A further investigation into the magnitude of the errors generated here and a deeper understanding of calibration in underwater environments to correct any possible assumptions would be useful.

---

<sup>1</sup> [https://docs.opencv.org/3.4/d9/d0c/group\\_\\_calib3d.html](https://docs.opencv.org/3.4/d9/d0c/group__calib3d.html)

## Testing

The sizing algorithm was run on both the Sparkling Star and Shetland Deployment 1 datasets labelled so far, although there are unresolved issues with both. We do not have precise calibration data for the former and, although we have calibrations for the second, we do not have many images from the left-hand side of this dataset so far. Furthermore, the Shetland deployment generally has a higher number of fish per image and the problems outlined in the next section restrict what we can do with the few images we have. We only currently have 40 labelled images for the left-hand side and many of these are not suitable for automatic sizing. The `extract_lengths` method on `StereoData` operates via the `extract_ellipse_fits` method and accepts the same arguments, namely:

- *“single”* : Where we only size for images that have a single object in both left and right images.
- *“matching”*: Where we only size for images that have a matching non-zero number of objects in both left and right images.
- *“full\_only”*: Where we only select images with all objects entirely contained within both left and right images i.e., none of the edge cases mentioned in the previous section.

Note that this process does not currently look at the object labels. The extra *“catalog”* argument saves all filtered results to *“fish\_sizes.log”*. If the labels are the same for the contours as they appear in the annotation, then the line for that image in the log file will be a name for each object followed by its size in mm. If the labels differ, both labels will be given, followed by the size.

For the SparklingStar dataset, we see prawns of a reasonable size, but other fish seem undersized. The calibration is not correct here though. The following gives a list of example size measurements (in mm):

|  |   |
|--|---|
| 00975_D20190719-T103259.835 prawn 33.185 | 05789_D20190719-T111317.441 dab 89.5 dab 52.041<br>dab 59.554     |
| 00977_D20190719-T103300.846 prawn 35.958 | 05806_D20190719-T111325.970 dab 91.436                            |
| 00978_D20190719-T103301.348 prawn 36.643 | 05808_D20190719-T111326.971 dab 108.086                           |
| 00979_D20190719-T103301.847 prawn 37.149 | 05883_D20190719-T111404.757 dab 60.823                            |
| 00980_D20190719-T103302.349 prawn 36.359 | 05924_D20190719-T111425.334 dab 56.754                            |
| 00982_D20190719-T103303.350 prawn 18.373 | 06077_D20190719-T111542.162<br>unidentified_round_fish 21.248     |
| 00983_D20190719-T103303.851 prawn 12.654 | 06114_D20190719-T111600.726 dab 108.269                           |
| 00988_D20190719-T103306.354 prawn 26.124 | 06313_D20190719-T111741.147 dab 78.256                            |
| 00993_D20190719-T103308.871 prawn 24.371 | 06331_D20190719-T111750.160 whiting 39.78                         |
| 00996_D20190719-T103310.374 prawn 10.084 | 06426_D20190719-T111837.786 haddock 52.677                        |
| 01003_D20190719-T103313.877 prawn 30.864 | 06459_D20190719-T111854.342 whiting 129.008                       |
| 01004_D20190719-T103314.377 prawn 23.718 | 06478_D20190719-T111903.898 dab 93.369                            |
| 01005_D20190719-T103314.878 prawn 18.589 | 06652_D20190719-T112031.144 haddock 108.036<br>whiting dab 94.774 |
| 05651_D20190719-T111208.242 dab 78.065   |   |
| 05653_D20190719-T111209.243 dab 85.397   |   |

The small prawn sizes make more sense when we look at the image using the `show_image` method from `StereoData`. The 10 mm prawn estimate image derives from Figure 43.



Figure 43. The hidden prawn.

Whereas the final measurements (above) derive from Figure 44

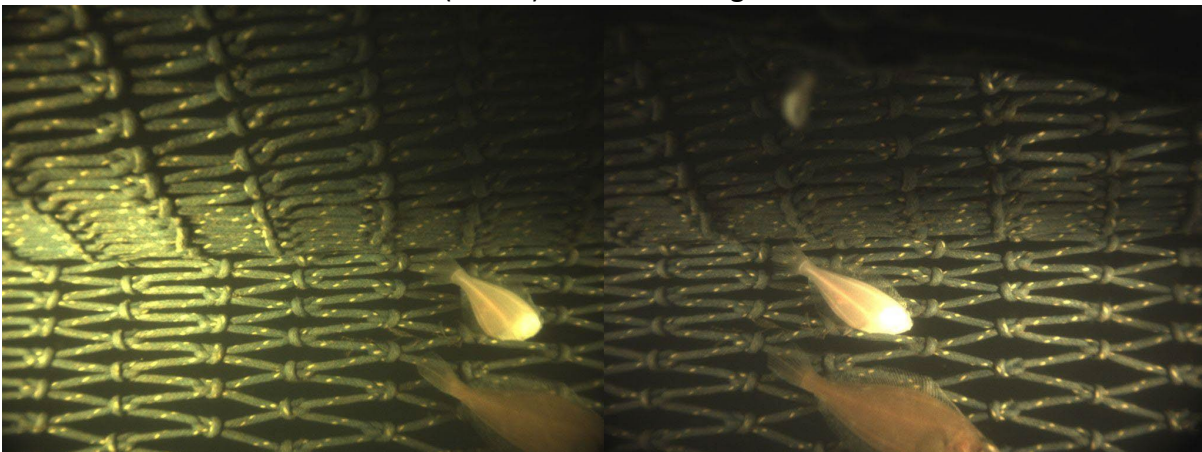


Figure 44. Small fish image.

For the Shetland dataset, the sizes seem reasonable general, but it can't assign the correct ellipse on both sides and there may also be inconsistency in labelling.

02266\_D20211130-T140548.163 unidentified\_gadoid haddock 785.274

02360\_D20211130-T140644.906 unidentified\_gadoid haddock 691.154

02409\_D20211130-T140714.273 unidentified\_gadoid haddock 340.919

02494\_D20211130-T140803.277 haddock unidentified\_gaddoid 331.682

02511\_D20211130-T140817.528 unidentified\_gadoid haddock 235.855

02594\_D20211130-T140921.084 european\_squid haddock 370.85 cluepid common\_squid 265.286

02651\_D20211130-T141012.180 unidentified\_gadoid haddock 412.676

02692\_D20211130-T141040.340 haddock 375.66

02697\_D20211130-T141043.054 cluepid sprat 582.238

02701\_D20211130-T141051.235 haddock 448.843

02714\_D20211130-T141058.039 haddock 696.622

02752\_D20211130-T141132.517 cluepid sprat 680.678

02787\_D20211130-T141157.072 cluepid sprat 678.555

02812\_D20211130-T141216.131 unidentified\_flat\_fish Unidentified\_fish 332.843

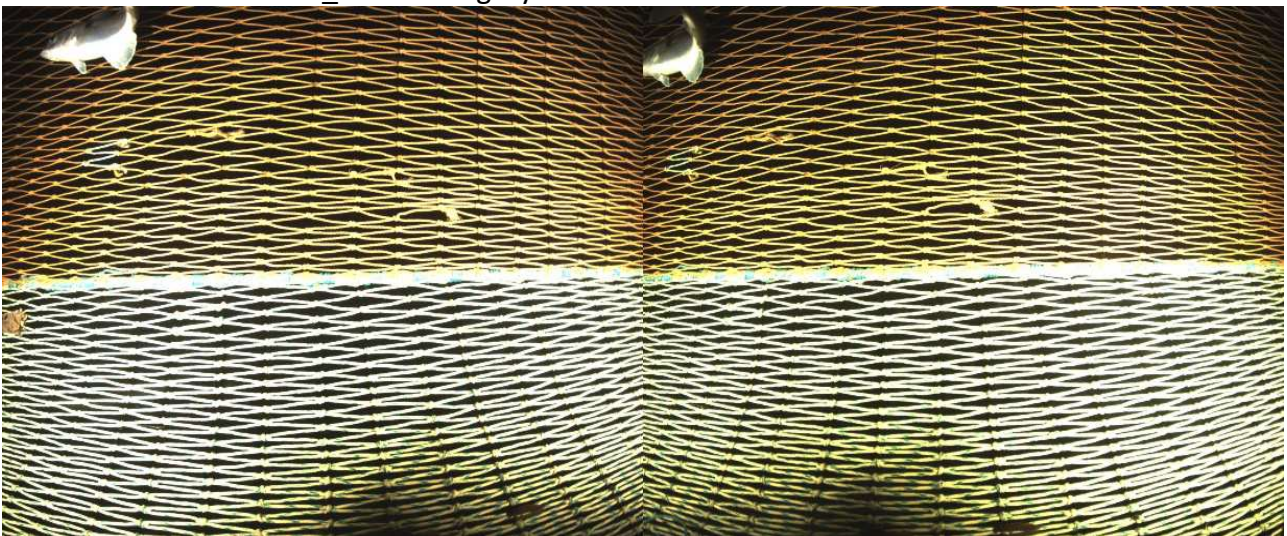
02818\_D20211130-T141219.166 haddock herring 319.148 cluepid herring 228.127 cluepid haddock 310.247

02831\_D20211130-T141232.616 unidentified\_gadoid Unidentified\_fish 229.794 unidentified\_fish haddock 141.13

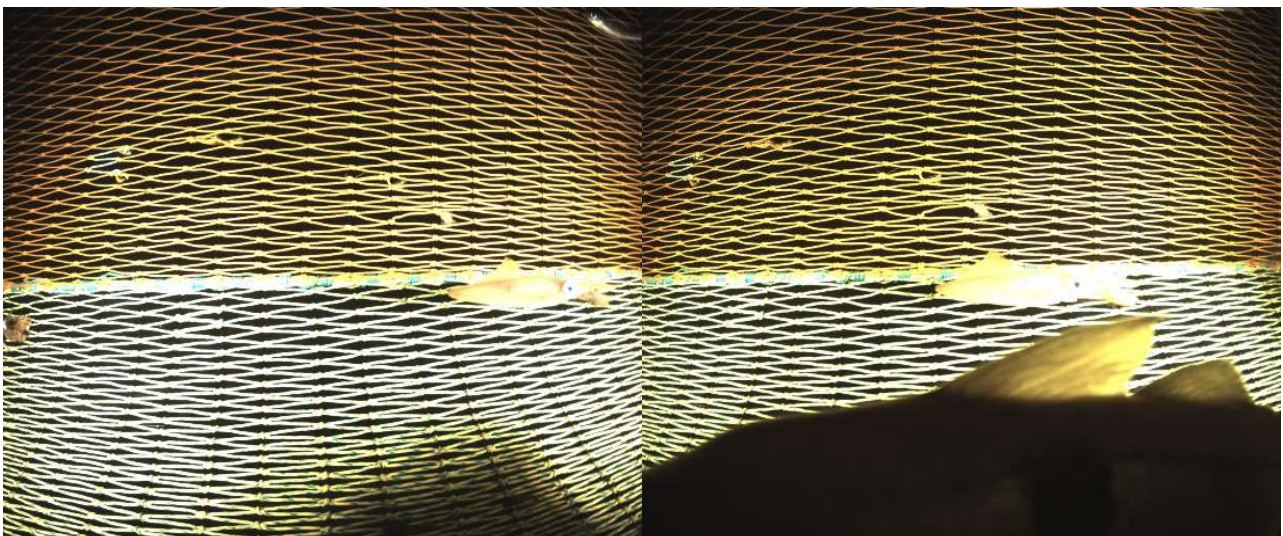
02885\_D20211130-T141314.036 haddock 600.218

02982\_D20211130-T141432.462 unidentified\_organism unidentified\_gaddoid 317.991

Possible mismatched labelling can be remedied by using the analysis.log file and looking at the images listed in the mismatched\_labels category.



*Figure 45. 02701\_D20211130-T141051.235 haddock 448.843.*



*Figure 46. 02594\_D20211130-T140921.084 European squid haddock 370.85 cluepid common\_squid 265.286 (mismatched fish between images)*

## Deployment of Deep Learning Model on Lightweight AI Device

Due to the limited space in the housing, our developed deep learning model needs to be deployed on a small and lightweight AI device. After comparing different AI devices, we find that Jetson Nano is a promising choice for this project. The size of the Jetson Nano Developer Kit is only 100 mm x 80 mm x 29 mm. In this section, we first explain how to set up Jetson Nano and flash the required Operation System (OS) and power supply. Then, to deploy our fish detection model, the installation of specific software and required dependencies are discussed. Lastly, some demos are provided to show the performance on detecting fishes and recognising their species along with running time.

### Setting up on Jetson Nano Developer Kit

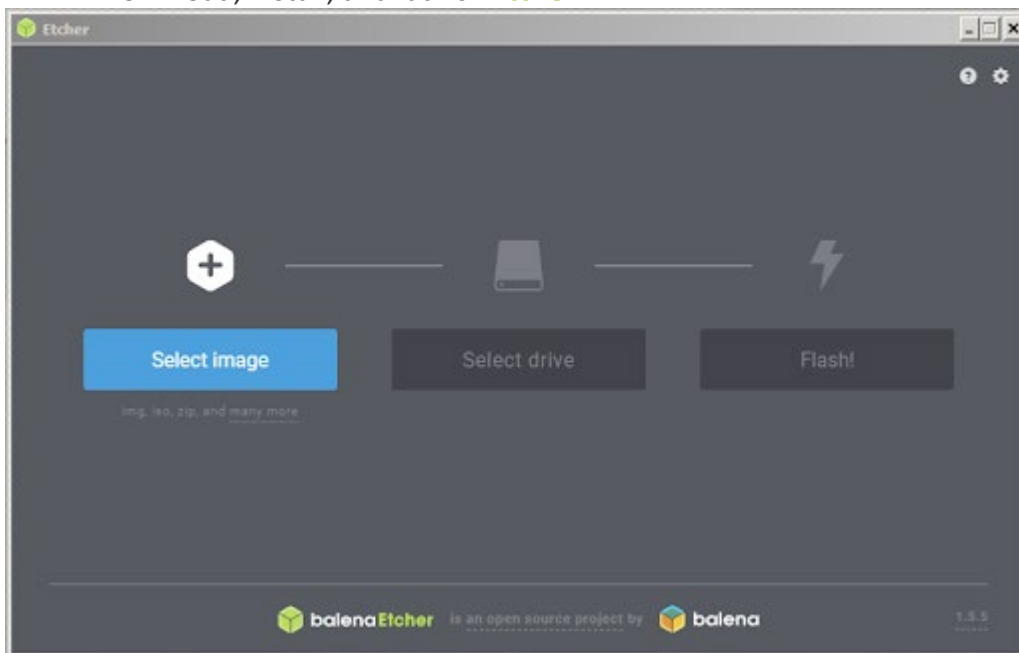
#### microSD Card and Power Supply

The Jetson Nano is driven by the OS of Ubuntu 18.04.6 LTS, where the OS is flashed in a microSD card with 64 GB. The Jetson Nano Developer Kit uses a microSD card as a boot device and for main storage. It's important to have a card that's fast and large enough for your projects; the minimum recommended is a 32 GB UHS-1 card. For Micro-USB Power Supply, it needs to be powered with a power supply that can deliver 5V=2A at the developer kit's Micro-USB port.

#### Flash OS Image to the microSD Card

To write an OS image to the microSD Card, it needs a computer with Internet connection and the ability to read and write SD cards, either via a built-in SD card slot or adapter. The OS image of [Jetson Nano Developer Kit SD Card Image](#) needs to be downloaded and then it is written to the microSD card by following the instructions below according to the computer's operating system: Windows, macOS, or Linux. In this project, we use Windows and use Etcher to write the Jetson Nano Developer Kit SD Card Image to your microSD card. The corresponding instructions are provided as follows.

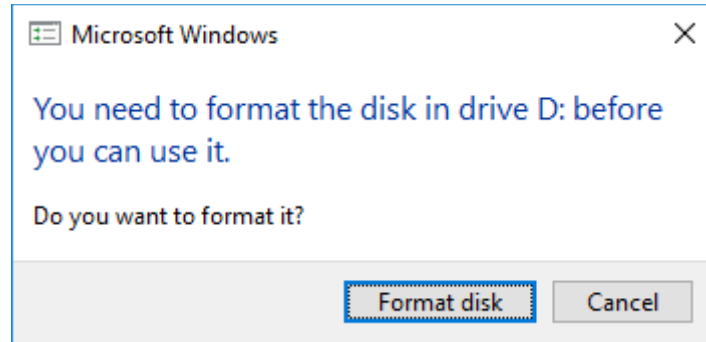
1. Download, install, and launch **Etcher**.



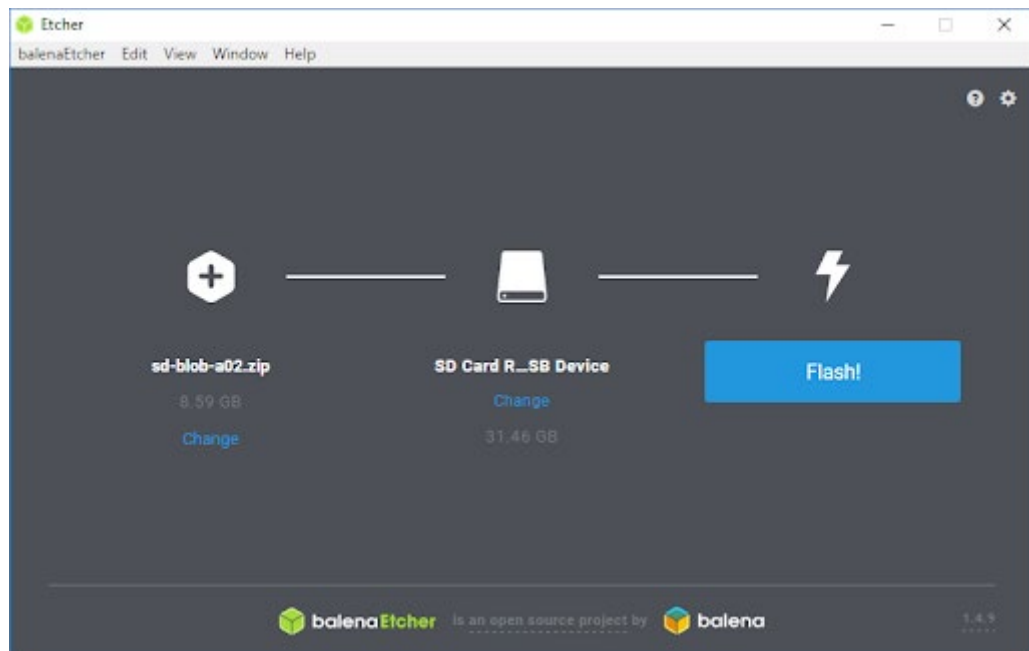
1. Click "Select image" and choose the zipped image file downloaded earlier.



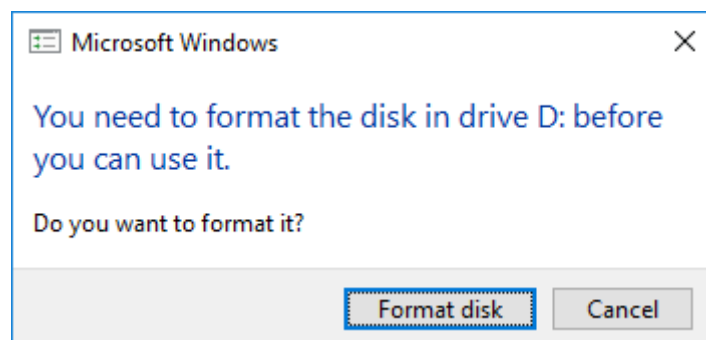
2. Insert the microSD card if not already inserted.  
Click *Cancel* (per [this explanation](#)) if Windows prompts you with a dialog like this:



3. Click "Select drive" and choose the correct device.



4. Click "Flash!" It will take Etcher about 10 minutes to write and validate the image if your microSD card is connected via USB3.
5. After Etcher finishes, Windows may let you know it doesn't know how to read the SD Card. Just click *Cancel* and remove the microSD card.



## *Installation of Required Dependencies and Mask R-CNN*

### Required Dependencies

After we successfully flash an OS image on Jetson Nano, the next step is to install Pytorch, a deep learning framework, and the required dependencies so that our developed deep learning model can run on Jetson Nano to detect and recognise fishes. To install required dependencies, we can follow instructions on this [link](#). The required dependencies are listed as follows.

- Python 3
- CUDA
- ZED SDK (Python API)
- Pytorch
- OpenCV
- Apex

### Mask R-CNN Installation

The original installation of Mask R-CNN is not easy to follow. Here, we provide an easy to install version, which only needs three steps to complete the installation of Mask R-CNN by using the command line.

- git clone <https://github.com/facebookresearch/maskrcnn-benchmark.git>
- cd maskrcnn-benchmark
- python setup.py install
- 

### *Demos and Running Time*

In this subsection, we demonstrate the performance of fish detection and recognition. They are two tasks. For fish detection, it targets on finding out whether there is fish showing in the current image and locating the position of fish. For fish recognition, it recognises the species of fish if there is a fish detected. An example of running Mask R-CNN model is provided in Figure 47. The location and species of fish are both provided by Jetson Nano. In addition, it is important to know the running speed of processing a pair of images from the left camera and right camera. According to our testing, the average speed to process per image is approximately one seconds on Jetson Nano. Therefore, it takes around two seconds to process a paired image of stereo camera.

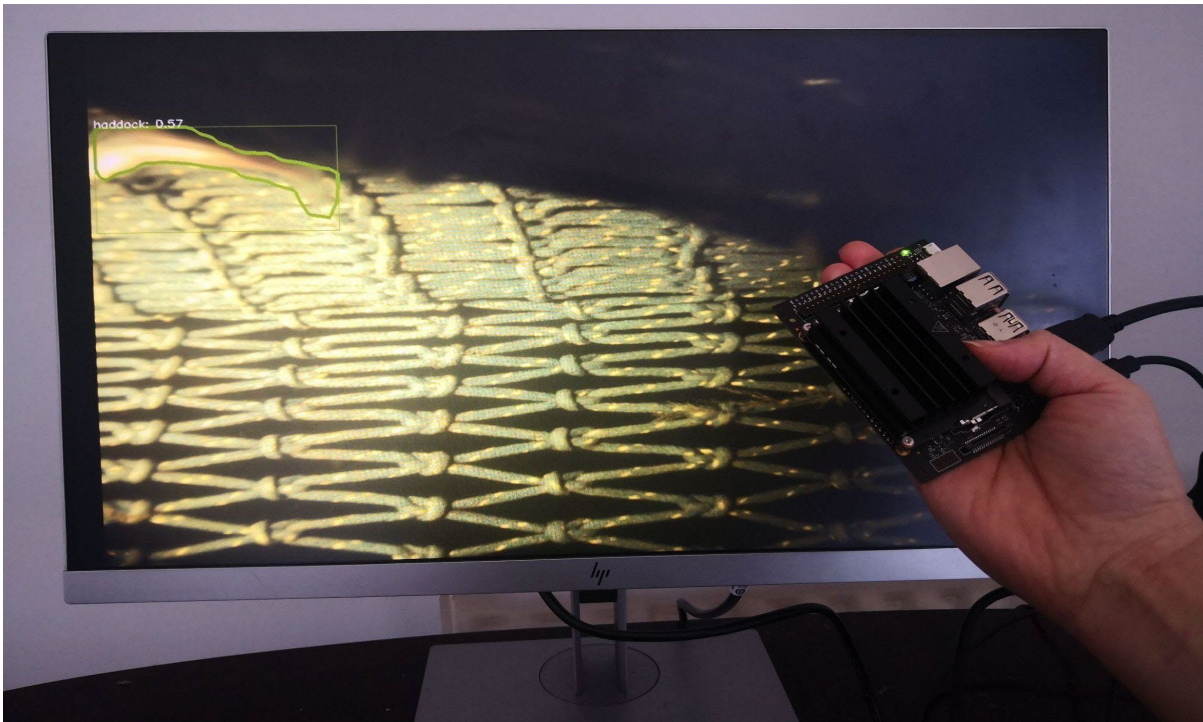


Figure 47. Mask R-CNN Running on Jetson Nano

### *Future work*

For now, we found that it needs to take 2 seconds to detect, recognise, and size the fish. To accelerate the speed of sizing the fish, we can attempt two solutions. The first solution is to reduce time consumed on the unnecessary images. In fish catching, we found that there are no fish for most situations. Therefore, to reduce the workload, we can train a lightweight image classification model to identify whether a fish exists. If so, we will pass the image to the pipeline of detection, recognition, and sizing. The second solution is to design a lightweight model to detect, recognise, and size fish. We can learn from lightweight models, e.g., MobileNet. For now, MobileNet is only used for image classification. Inspired by its design, we will transfer to our instance segmentation application.

### **Future work**

The future work can be conducted from three aspects. First, the performance of fish detection and recognition can be enhanced by conducting hyperparameter search, utilising more advanced networks, and enriching data. Second, existing size detection method works for single target (fish) while such a method is still challenging to detect sizes of multiple targets (fishes). There are several potential solutions to tackle the problem. The most straight forward and promising solution is to project all pixels from a paired stereo images to 3D space and then carry out fish detection and species recognition. Third, inference speed on embedded device needs to be accelerated so that the operation of latch can be carried out without any delay. A tailored lightweight deep learning model will be developed to realise real-time fish detection, specie recognition, and size estimation. More detailed discussions for each aspect are provided separately as shown below.

### *Improvement of fish detection and recognition*

Due to time constraints, it was not possible to fully explore improvements, although we are well-

positioned to make a much-improved fish detection network. We make some recommendations on how to do so here.

### Hyperparameter search

We created a python script that could generate config files by grid or random search. We also generated a python script to generate a .bat file that could run a sequence of command line instructions where the results could be saved individually into subdirectories of the output folder. With these tools, it is possible to enter a range of values for several training parameters and run dozens of simulations in succession. Although this would take a few days of computation time, it is a feasible route. Some mechanism of stopping poorly performing settings would save on wasted computation, but this has not yet been implemented.

### Alternative networks

We may also wish to try other feature extractors in the backbone portion of the network. It is possible that it may improve both accuracy and inference speed, but further research is needed. We may also change the detection framework to mmdetection or detectron2, successors of maskrcnn (which is no longer being maintained by its creators). This may, however, present problems as we don't yet have a tested way of loading these networks onto the Jetson Nano device.

### More Data

As we get more labelled data, there will be a greater variety of environments and instances of classes for which we have too few examples. Furthermore, the main dataset we have been training with (SparklingStar), generally has fewer instances of fish per image and more cases where fish are obscured by the net on the right-hand side. The new Shetland datasets have more groups of fish, a greater variety of species and are generally higher quality because of suggestions to improve the recording setup to provide the most useful training data. In hypothetical further stages of this project, this data is extremely promising.

### Data Augmentation

We can perform data augmentation to increase the variety in the training data. The maskrcnn training process involves instantiating a COCODataset object (found in maskrcnn/data/datasets/coco.py) which has an optional transforms keyword argument. Maskrcnn has some inbuilt facility for applying transforms via the .YAML calibration file, but they are very limited (only random horizontal and vertical flips and ColorJitter). These are found in maskrcnn/data/transforms/transforms.py. To extend these, they would have to be imported from torchvision or a similar package. The training and maskrcnn code would have to be expanded to allow for this.

### *Size Detection of Multiple Targets (Fishes)*

At the present time, we have no way of accurately mapping multiple targets between two images in an accurate way. For this reason, we were restricted to testing cases where we could obtain accurate data. To avoid errors in badly fit ellipses, only fish which are fully contained in both left and right images were considered. For SparklingStar, there is a recurring problem of the net obscuring the right-hand image. Some of those remaining images where the fish is recognised on both sides suffer from

There are, however, several solutions to this problem. The simplest approach is to apply ordering based on the relative locations of objects to one another, although this contains some assumptions

that may not be generally valid. It's possible that fish may exchange positions between left and right viewpoints. Another option is to use more sophisticated techniques well-known in computer vision such as block matching and instance identification, where we generate a unique identifier using feature extraction for each instance and pair them between images. A third way is to use 3D projection to localise each object in space and pair them by mapping to the 3D environment. Which is the most suitable for our application depends on the computational overheads of existing solutions, although most computation will still be at the detection inference stage.

To account for variation in the sizing of the same object, we can combine the solution of the previous problem to maintain an average sizing for the fish and thereby increase accuracy.

In general, the sizing seems to be the most difficult aspect of the process so far as we don't yet have a consistent way to automatically pair objects and it is hard to estimate ground truth. Lacking this and a larger set of paired images, it is difficult to assess properly at this point.

### *Inference Acceleration on Embedded AI Device*

For now, we found that it needs to take 2 seconds to detect, recognise, and size the fish. To accelerate the speed of sizing the fish, we can attempt two solutions. The first solution is to reduce time consumed on the unnecessary images. In fish catching, we found that there are no fish for most situations. Therefore, to reduce the workload, we can train a lightweight image classification model to identify whether a fish exists. If so, we will pass the image to the pipeline of detection, recognition, and sizing. The second solution is to design a lightweight model to detect, recognise, and size fish. We can learn from lightweight models, e.g., MobileNet. For now, MobileNet is only used for image classification. Inspired by its design, we will transfer to our instance segmentation application.

### *References*

- [1] He, K., Gkioxari, G., Dollár, P., & Girshick, R. (2017). Mask r-cnn. In Proceedings of the IEEE international conference on computer vision (pp. 2961-2969).
- [2] Yi, D., Fang, H., Hua, Y., Su, J., Quddus, M., & Han, J. (2021). Improving Synthetic to Realistic Semantic Segmentation with Parallel Generative Ensembles for Autonomous Urban Driving. IEEE Transactions on Cognitive and Developmental Systems.
- [3] Papandreou, G., Zhu, T., Chen, L. C., Gidaris, S., Tompson, J., & Murphy, K. (2018). Personlab: Person pose estimation and instance segmentation with a bottom-up, part-based, geometric embedding model. In Proceedings of the European conference on computer vision (ECCV) (pp. 269-286).

## Objective 3: Shetland sea trials

### Introduction

It was an exceptionally poor winter for fieldwork at sea. January and February were characterised by almost endless westerly gales. Consequently, much of the fieldwork in Shetland was undertaken in borderline and unfavourable conditions characterised by westerly swell. In the last two weeks of March the weather finally eased and consecutive days at sea were possible.

Gear adjustments and rigging were guided by the intuition of the Shetland UHI Skipper and Fisheries Technician who have many years of experience at sea previously as fishermen. Sea trials were conducted using the 12m *Atlantia II* using a rock hopper net based on a Jackson 352 trawl. This was the same net used in previous trials of the Smartrawl gate device prototype. A Notus net monitoring system was used to assess door spread and headline height during specific tows.

### Initial camera sea trials

Activity in Shetland for Smartrawl Phase 4 commenced on 18/11/2022 with the delivery of the stereo camera system to Shetland UHI and basic training delivered to Shaun Fraser by John Polanski in the operation of the camera. Initial sea trials commenced subsequently during the first available weather window. The following text provides narrative with supporting images of each trawl deployment (designated by D and a sequential number) on respective days.

### D1 (30/11/2021)

The centre of the camera was fixed 2m forward of the forward edge of the escape windows previously cut (and subsequently repaired) into the net extension for the Smartrawl gate device, and thus about 11m from the end of the cod-end. The camera was tied onto the port-side selvedge throughout all sea trials so that the shorting plug was facing aft to reduce the opportunity for snagging. The camera was secured by its stainless-steel structural frame at each corner using braided twine which was tied to the net at a distance of four masks from the selvedge. The carbon fibre poles and plastic bolts of the camera's supporting structure were prone to snagging which was mainly an issue when fastening and removing the camera from the net.



Figure 48. Attaching stereo camera system to the rock hopper net aboard *Atlantia II*.

Deployment and recovery of the camera from the vessel was found to be straightforward. Adding the camera did not present any major operational issues during shooting and hauling of the gear. The main limitation was that the camera could not be taken up onto the net drum, which effectively left the remaining net to the cod-end to be hauled aboard by other means. The plugs and magnetic switch that controlled the camera were found to be easy to use and very practical at sea.

The images indicated that the camera was well balanced in the extension with the opposite selvedge horizontal in the image, however at some points the camera frame seemed to have a tendency to pitch down or sink relative to the net, especially towards end during haul back, perhaps relating to drag on the frame.

It was noted at this stage that the stereo camera images were generally overexposed.



Figure 49. Example stereo camera images from D1. Note these and subsequent images are illustrative examples taken from different times during the tow and are not simultaneous photos from the two stereo cameras.

## D2 (02/12/2021)

White netting was provided by colleagues in Aberdeen to provide a background that might be better suited to the development of algorithms for fish discrimination. This was fixed into the net extension in the area of the camera field of view. The staff fitting the white netting found it to be difficult and awkward to incorporate into the extension due to the small mesh size and its relative inflexibility.

The addition of the material did not present any additional difficulties during shooting and hauling. It was noted that sometimes there were apparent breaks in the strobe frequency.



Figure 50. Rigging white netting into net extension aboard *Atlantia II*.

On inspection of the images, the size of the white netting was insufficient to entirely fill the camera field of view. Further, the images were even more exposed than before. It was also noted that there was some shadowing of fish close to the camera lenses and that the images indicated that some fish were passing below and particularly above the field of view.



Figure 51. Example stereo camera images from D2.

### D3 and D4 (10/12/2021)

It was decided to try a different type of white material that would hopefully fill the field of view better and be easier to rig to the extension material. New material was sourced and tied in a T90 configuration.

Here and in all subsequent tows the centre of the camera had been shifted to 2.5m forward of the forward edge of the escape windows cut (and subsequently repaired) for the Smartrawl gate device, and thus 11.5m from the end of the cod-end. This adjustment was to provide more room for background materials and potential rigid components.

The new material was first tried unsupported (D3) and then with rigid pipes attached (D4). Medium Density Polyethylene (MDPE) piping was added to the extension fore and aft of the camera at the leading and trailing edges of the white material. The pipes had been previously cut from straight sections and formed into hoops by securing the ends to a wooden plug with screws. By undoing the



screws, it was possible to open the hoop, and thus to insert the pipe into the netting at sea before closing the hoop and securing with the screws. However, this was an awkward process in the limited room available at the stern. It would not be possible to accommodate both the gate and the rigid hoops at the stern, and neither could be taken onto the net drum.



Figure 52. Attaching and deploying MDPE piping in the net extension.

The position of the new white material had been over-compensated from the previous tows and adjusted too far downwards so that unfortunately a large section of extension netting was visible. Further, the lighter material and T90 rigging did a comparatively poor job of blocking the netting behind. This effect did however lead to less over-exposure of the images. As the exposure seemed to be related to the choice of background material, it was felt useful to finalise the background material before consideration was given to adjusting exposure levels.



Figure 53. Example stereo camera images from D3.



Figure 54. Example stereo camera images from D4.

There was a modest improvement in the shape and tension of the background material due to the piping. The main advantage to the net shape due to the pipes was potentially during the shooting and hauling where the net shape in the camera field of view was more consistent than in the unsupported case where it tends to collapse.

## D5 and D6 (25/01/2022)

Due to the limitations identified of the white netting already tried, other potential background materials were sought. The use of green tarpaulin-type material was found to be the best background in recent relevant work (Sokolova et al., 2021). Consequently, a supplier was identified (Allplas.co.uk Limited) and green 610gsm reinforced PVC tarpaulin material was ordered with brass eyes fitted around the perimeter at 250mm intervals. A 2.5m x 3m square sheet was trialed as the camera background, with the longest side oriented along the length of the net and the perimeter laced to the extension netting using 8mm bungee cord material.

The green sheet was rigged and first deployed with the pipes in place (D5), and then subsequently trialed with the pipes removed (D6).



Figure 55. Attaching and deploying (D5) green tarpaulin sheet as an alternative camera background.



Figure 56. Deploying and recovering the stereo camera with pipes removed during D6.

Due to a heavy swell on nearby fishing ground the green sheet was first trialled during D5 and D6 in a small, sheltered area among the Islands near Scalloway which, somewhat predictably, had few fish. The area was also unfortunately found to contain various dumped coils of rope which was caught by the gear in both tows and likely caused the substantial amounts of sediment suspended which obscured many of the images.



Figure 57. Example stereo camera images from D5.



Figure 58. Example stereo camera images from D6.

As before, it was felt that the use of the pipes only provides a modest improvement to image background and that, overall, the use of pipes or other rigid components to stabilise the camera field of view did not seem like a practical option in combination with the gate. Consequently, in subsequent tows no pipes or other components were used to support the camera field of view area. It was noted that there seemed to be a red colouration in the images from both D6 and D7, but it was felt that a longer tow without the complications of gear entanglement would be beneficial before considering further.

## D7 (02/02/2022)

A longer tow in the Scalloway Deeps area provided higher quality images to consider the performance of the green sheet in more detail.

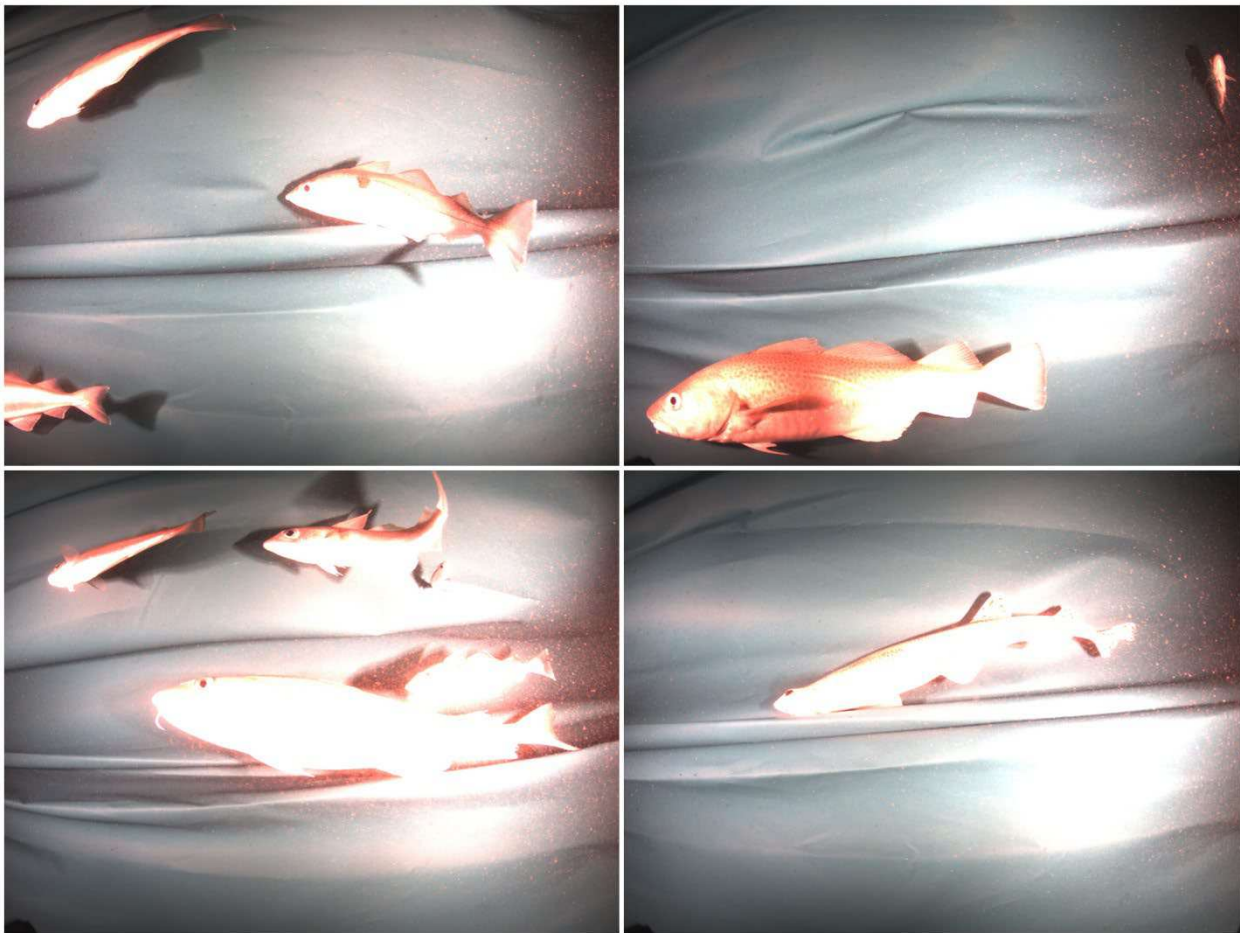


Figure 59. Example stereo camera images from D7.

Although the visibility was good, the red colouration was still clearly present, and the images were still overexposed. It also seemed desirable to reduce the lateral folding observed in the unsupported green sheet. However, the green tarpaulin material seemed durable and presented a good contrast to the fish species observed and prevented any fish from being meshed or snagged in the camera field of view.

It was decided at this point to proceed with the green sheet but to first improve the camera exposure

and colour balance before further sea trials.

Adjustment of camera settings

Initial bench test

Adjustments to gains settings and colour balance were possible with using the FlyCap2 application installed on the camera. A process of iterative adjustments was undertaken, first on the bench to find approximate settings that would serve as a starting point for tank tests. The impact of adjustments was assessed by reference to specialist colour cards which show various specific colours as well as a row of grey shades. After each adjustment, the grey shades were systematically analysed and further adjusted with the assumption that the red, green, and blue component colour values should be approximately equal. After some adjustment, it was clear that the default settings used up until then were far from optimal.

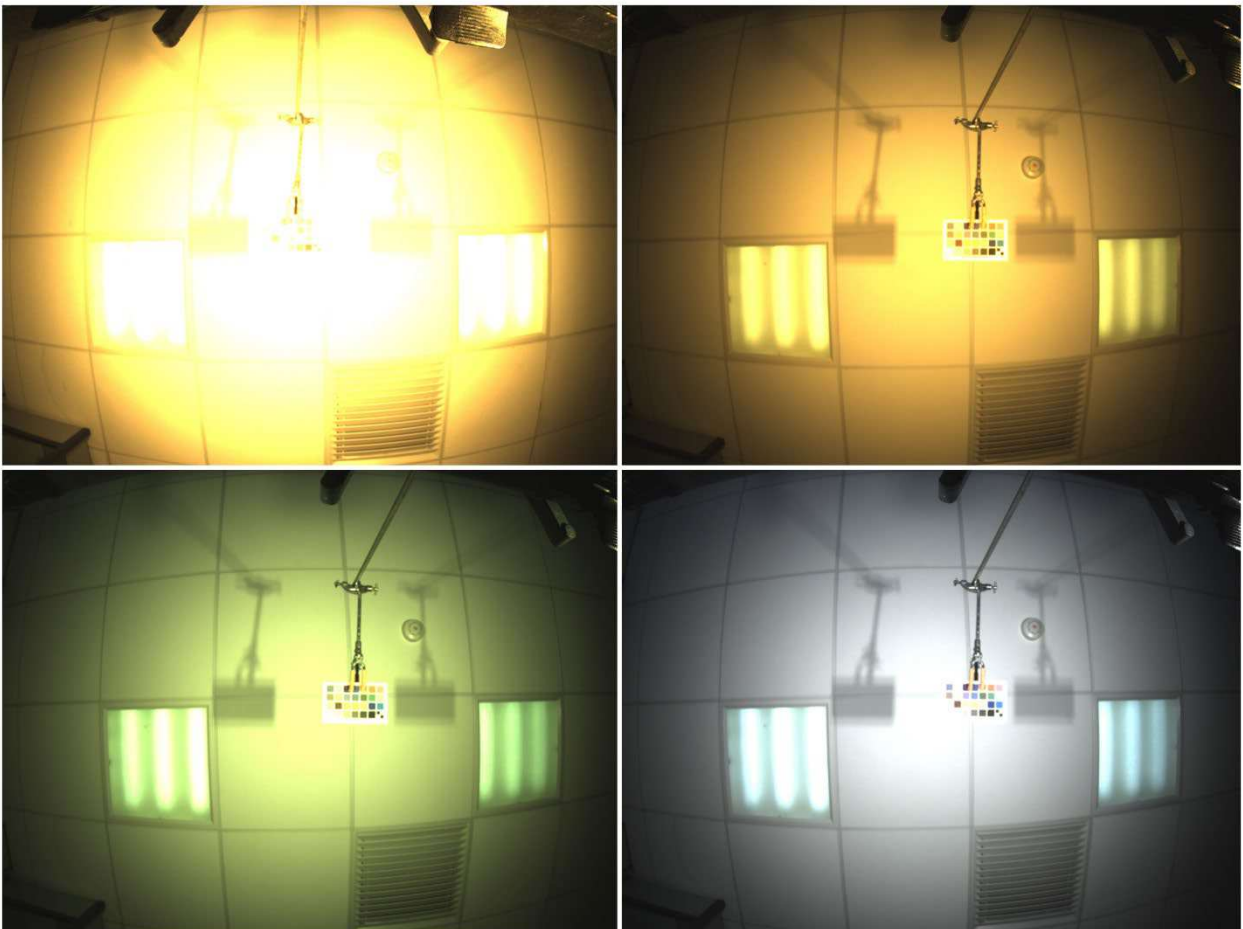


Figure 60. Examples of bench test images, starting with default settings (top left) and progressing through various adjustments to gain and colour balance settings. Note the colour card in the centre of the image.

## Initial tank tests

A suitable tank was identified and test rig constructed using a new tarpaulin sheet made from the same material as the background sheet fitted in the trawl. This setup enabled the controlled underwater adjustment of the camera in a situation as similar to the sea trials as possible. Final tank test adjustments were made in the absence of any ambient lighting.



Figure 61. The tank and test rig used for underwater adjustments prior to being filled with water (left) and later during adjustments (right).

Only adjustments to gain, red, white-balance, and blue, white-balance were found to have any impact in the FlyCap2 application. The best settings identified in the initial tank tests involved reducing the gain all the way to zero, some increase of red, white balance, and increasing blue, white balance all the way to maximum. These drastic adjustments provided a much-improved colour balance to visible squares on the colour card; however, the image was still somewhat overexposed, and some lighter squares were still not visible at all.



Figure 62. Colour card images during tank tests with default settings (left) compared to initial adjusted settings (right).

## Subsequent bench tests

Following correspondence with Rick Towler the limitations of FlyCap2 were identified and further adjustment to the camera software settings had to be made by editing the settings .ini file. Consequently, the bench tests and tank tests were repeated using various exposure settings and further iterative adjustment of gain and colour balance settings. Subsequent bench tests using lower exposure settings provided good results with all colour squares visible and consistent results at different distances.



Figure 63. Example stereo camera images from subsequent bench tests using lower exposure settings.

## Subsequent tank tests

Further tank tests at lower exposure settings provided improved image quality with all colour squares visible and colour components balanced. The final settings arrived at here and used in subsequent sea trials were:

Exposure = 600 us; Gain = 0; WB red = 565; WB blue = 1023.



Figure 64. Example stereo camera images from each camera of stereo camera system using final settings during tank tests.

## Subsequent camera sea trials

Further sea trials were now undertaken to test the updated settings. The green sheet was adjusted to increase tension and to reduce the lateral folding observed in previous tows by lacing the long sides to netting closer to the camera.

## D8 (04/03/2022)

The first deployment with the updated settings showed fish in natural looking colour and without the overexposure seen in previous deployments. The shape of the green sheet was also improved in comparison to previous tows, although it was still somewhat dynamic with occasional folds forming. It was thought likely that the sheet would appear flatter still if there was a substantial catch in the cod-end to stabilise that area of the net. It was noted that while fish near the opposite side of the extension to the camera were in focus, many fish closer to the camera were slightly out of focus. Although it was thought that it may be possible to slightly improve image focus, this was likely to be an involved process that there was insufficient remaining time to undertake without a significant impact on field trials.

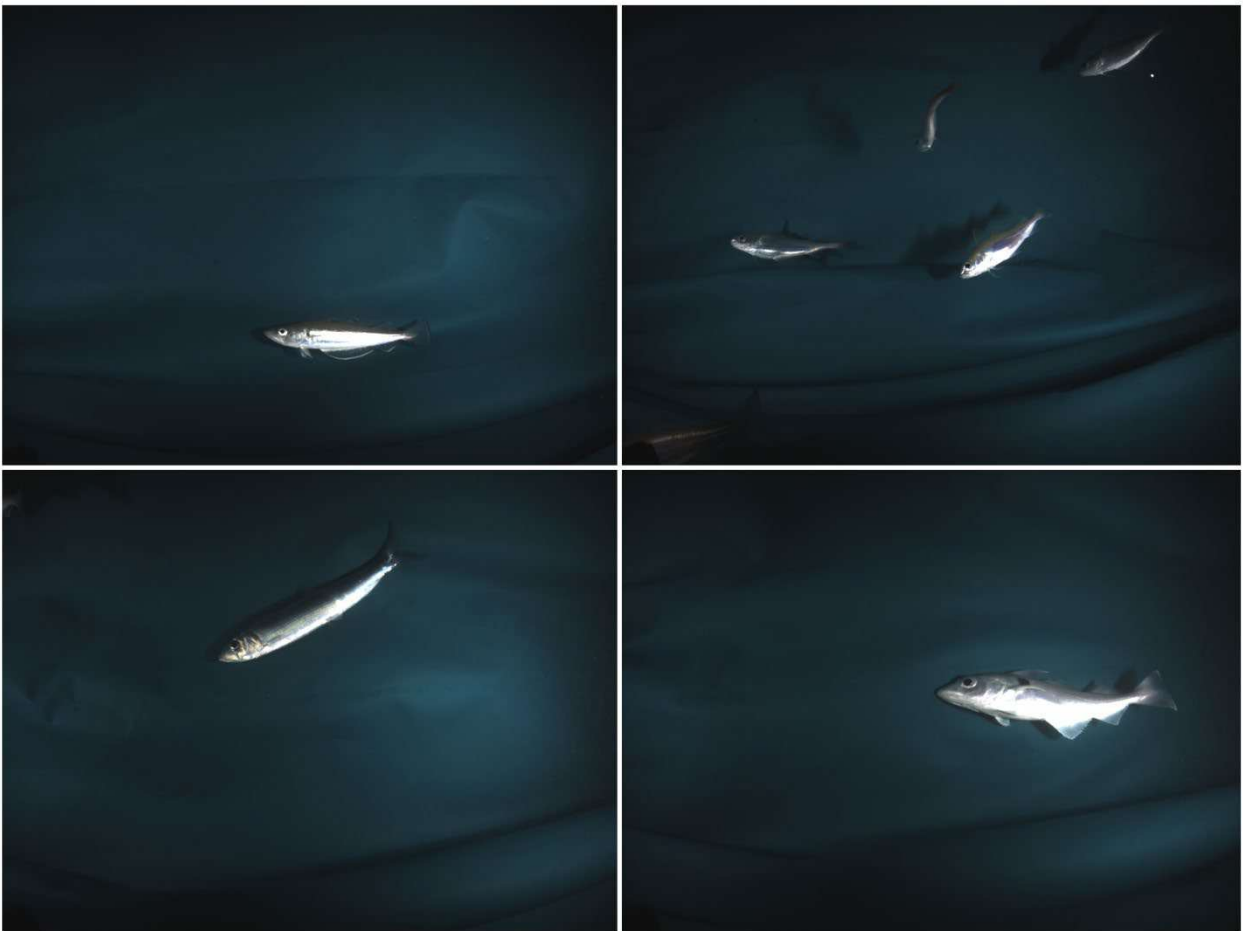


Figure 65. Example stereo camera images from D8.

## D9 (07/03/2022)

A further tow with the same setup was undertaken during rough conditions. The motion caused gear components to fail and entangle which likely substantially compromised the net geometry during the entire tow and led to it being classed as a “foul haul”. Nonetheless, some fish were observed, and image quality looked encouraging.





Figure 66. Example stereo camera images from D9 which was a foul haul due to gear entanglement.

### **D10, D11, and D12 (14/03/2022)**

Three further tows were undertaken across a range of depths to further assess image quality with different levels of ambient light. The first tow (D10) was in the Scalloway Deeps (mean tow depth approx. 91m), the second (D11) was in the Burra Haaf (mean tow depth approx. 100m), and the third in a nearby shallow in the vicinity of South Havra (mean tow depth approx. 37m). Image quality was high throughout, and wide variety of species were observed and could be manually identified with ease.

It was found after D12 that a stainless-steel bar that had been incorporated into the stereo camera frame to protect the lenses and adjust the buoyancy was missing. A new slightly larger bar was machined and fitted before the next available weather window. There was some indication in subsequent images that the new bar increased the tendency of the camera frame to pitch down, particularly during haul back.

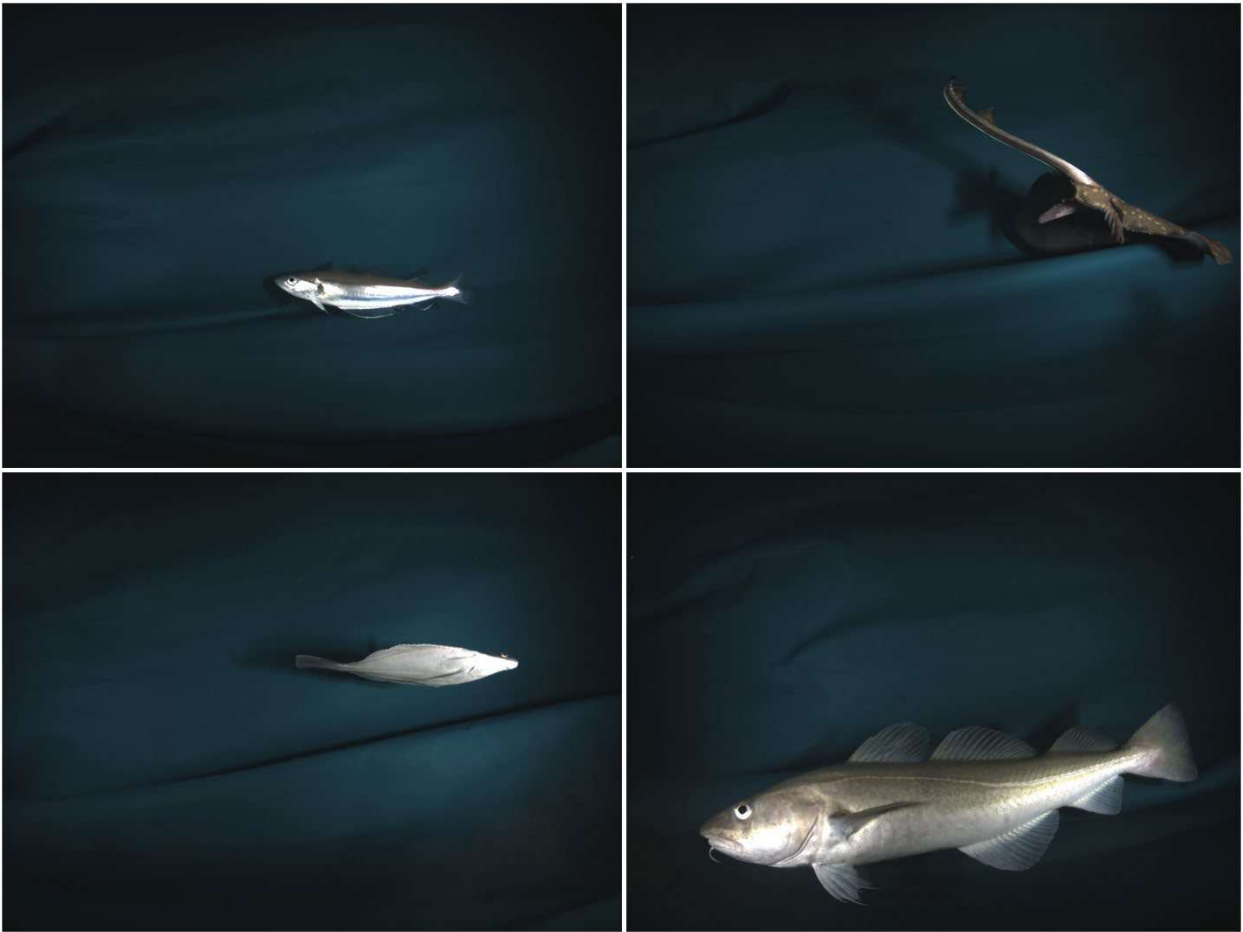


Figure 67. Example stereo camera images from D10.

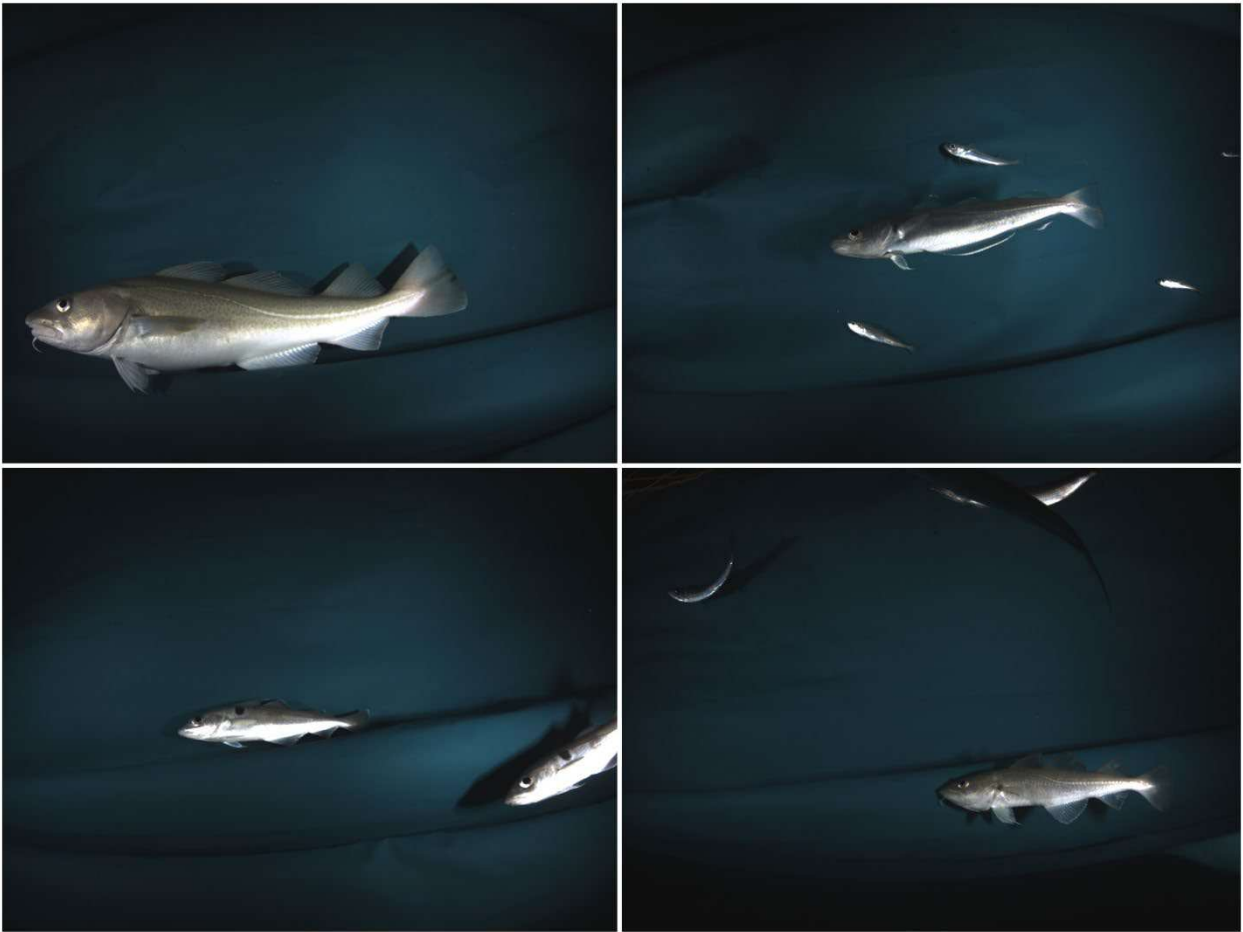


Figure 68. Example stereo camera images from D11.



Figure 69. Example stereo camera images from D12.

## Sediment suppression experiments

Previous experience had suggested that suspended sediment would be unlikely to cause major problems to image quality in nearby fishing grounds. However, in anticipation of potential problems for application to other areas and fisheries, e.g., *Nephrops* fisheries, areas with fine sediment were sought to test a sediment suppression system (SSS) based on recent work by Sokolova et al., (2022).

## D13, D14, and D15 (21/03/2022)

Three tows (D13-15) were undertaken in areas thought to be characterised by fine sediment based on local knowledge. From an initial inspection of the images from D13-D15 it seemed that D13 had a substantially higher proportion of images obscured by sediment. Consequently, D13 was used as the baseline for further tests which aimed to replicate this tow with gear modifications intended to reduce the amount of sediment in the camera field of view.



Figure 70. Example stereo camera images from D13.



Figure 71. Example stereo camera images from D14.



Figure 72. Example stereo camera images from D15.

All images from a 30-minute section of the data from D13 were analysed by Shetland UHI staff to make some initial quantification of sediment levels. If more than 50% of an image was judged to be affected by sediment, it was classed as “obscured”. By assessing all images in the 30-minute section the overall proportion of obscured images could be estimated. Recognising that this approach depended on a somewhat subjective assessment, the process was undertaken by two different analysts. The first analyst estimated that 15.5% of images were obscured in D13. The second analyst estimated that 14.4% of images were obscured in D13.

## D16 and D17 (23/04/2022)

Following insights from Sokolova et al. (2022), a custom made 8.4m x 5m 610gsm reinforced PVC tarpaulin sheet with brass eyes at 500mm intervals was ordered to make a sediment suppression system (SSS). The material was the same as used in the camera background sheet, as this had by now been proven as practical at sea multiple times. The dimensions of the sheet were scaled up from the dimensions given in Sokolova et al. (2022) in proportion to the relative theoretical fishing circles. On arrival the SSS sheet was found to be awkwardly large and more difficult to fit than indicated in Sokolova et al., (2022). The leading edge was lined up to the footrope and the required offset was found to greatly exceed the 0.1 m defined in Sokolova et al., (2022) which led to substantial folds forming if the sheet was secured along the entire leading edge to the footrope. The options considered at this point were either to either cut the leading edge of the sheet to match the shape of the footrope, or to tie the sheet to flat to the fishing circle. Another consideration was the

durability of the sheet, as cutting a curved profile on the leading edge would be more vulnerable to tearing than the original flat folded seam. Consequently, it was decided to secure the leading edge of the sheet to the fishing circle in the first instance by braided twine and to let further modifications to the sheet to be guided by the following results. In this arrangement, only the central section of the sheet was fitted directly to the footrope.



Figure 73. Initial fitting of the SSS to the gear.

Unfortunately, during D16 after hauling the gear the stereo camera displayed a battery fault warning and had stopped recording soon after being submerged. Subsequent examination suggested that the ingress of fine sediment into electrical components had caused a connection issue with the plugs. The plugs were cleaned and lubricated, and after a bench test the sea trials were reassumed. D17 successfully replicated D13 by fishing over the same tow track. Initial inspection of the images indicated that there was still a substantial number of images obscured by sediment. Subsequent analysis by the same staff that analysed D13 indicated that the images in D17 were in fact more obscured than in the baseline. The first analyst estimated that 18.9% of images were obscured in D17 (compared to 15.5% in D13). Similarly, the second analyst estimated that 16.6% of images were obscured in D17 (compared to 14.4% in D13).



Figure 74. Example stereo camera images from D17.

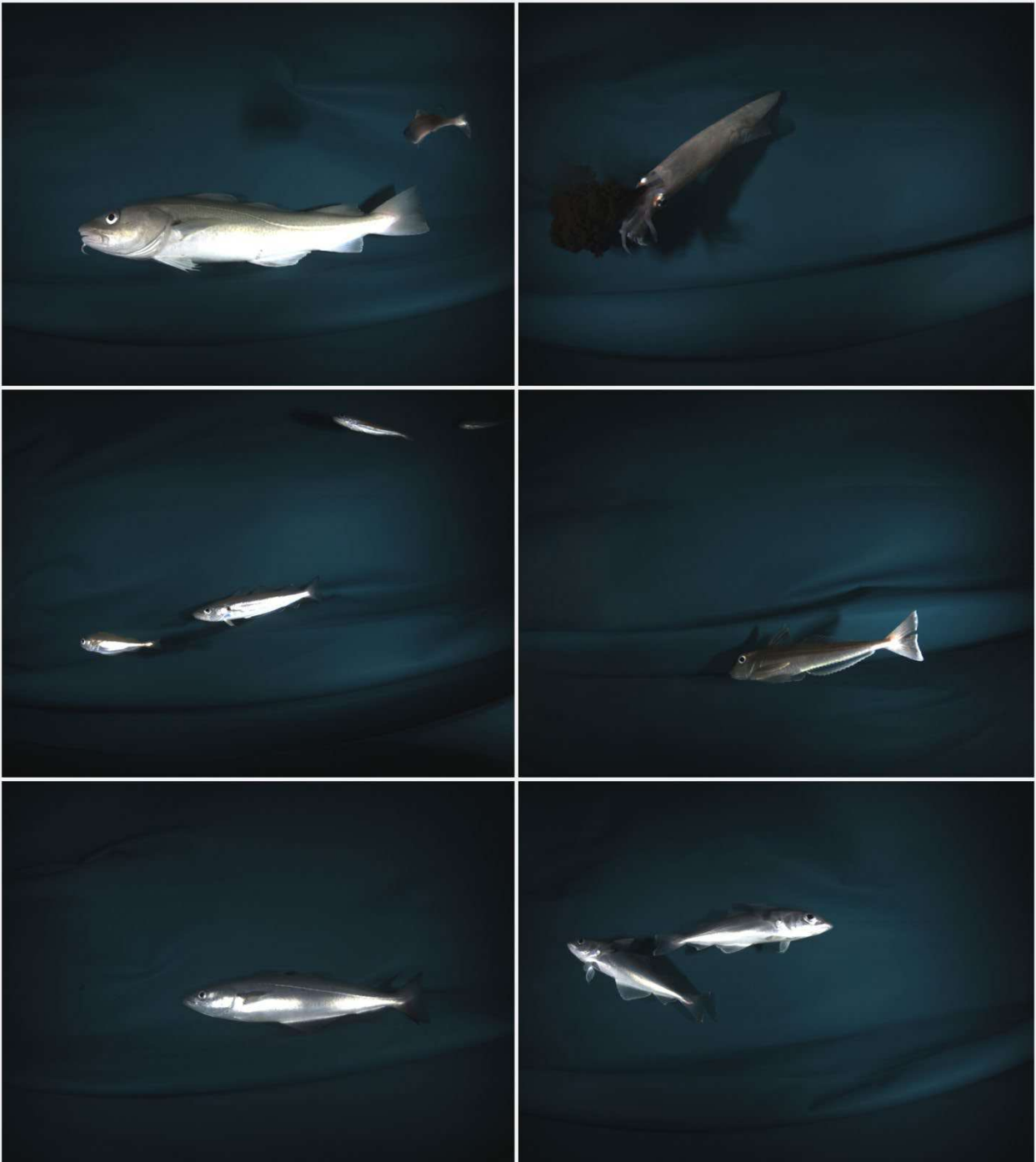
Monitoring of door spread and headline height in both D16 and D17 with comparison to related baseline tows indicated that the incorporation of the SSS did not have a detectable impact on these fishing characteristics.

It was noted that when hauling the SSS onto the net drum that it had tendency to trap water which may make shooting and hauling more difficulty due to the additional weight pulling the net down between the net drum and the stern.

## **D18 (24/03/2022)**

While preliminary analysis of results from D13 and D17 was ongoing, a longer tow (4 hours) was undertaken during D18 to provide a larger dataset of images from a tow more representative of commercial practices. The tow was undertaken in the Burra Haaf fishing ground and a wide variety of species were observed with large cod the most prevalent.





## D19 and D20 (29/03/2022)

D19 and D20 involved the use of GoPro camera gear to attempt to understand how the SSS behaved and interacted with sediment with the aim to inform the design of improvement to the SSS. Two short tows were undertaken in a relatively shallow area where good quality videos had been recorded using the exact same approach in previous years. The stereo camera was also fitted and recorded as normal. In the first tow (D19), a GoPro was fitted in a protective housing with lights to near the headline of the net pointing towards the footrope to observe the SSS at the mouth of the net. In the second tow (D20), the same GoPro setup was fitted on the top panel of the net pointing aft towards the trailing edge of the SSS. Unfortunately, in both deployments the visibility in the video footage

was found to be very poor at depth with high concentrations of plankton causing high levels of backscatter from the lights which prevented detailed analysis of the SSS. However, it was briefly apparent during D20 that the SSS was tending to fold and wave in a dynamic manner.



Figure 75. Rigging of GoPro camera housing during D19 to inspect mouth of net (top figures) and example screenshot from resulting video (bottom figure) showing poor visibility and high levels of backscatter with headline and ground gear faintly visible.



Figure 76. Example stereo camera images from D19.

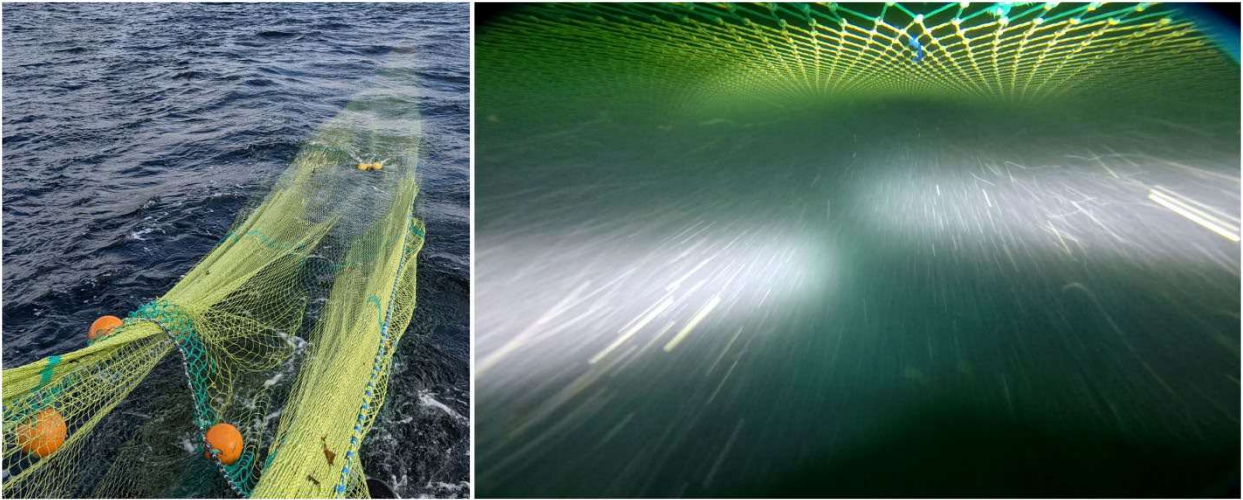


Figure 77. Rigging of GoPro camera housing during D20 to inspect mouth of net (left) and example screenshot from resulting video (right) showing poor visibility and high levels of backscatter with SSS faintly visible in centre of image.



Figure 78. Example stereo camera images from D20.

## D21 (30/03/2022)

Following the unfavourable %observed results and limited insights from the GoPro videos, it was felt that some alterations were required to try and improve the performance of the SSS. Consequently, the leading edge of the SSS was cut to more closely match the shape of the footrope during towing. This involved cutting 1.25m of material away at the centre of sheet, far in excess of the 0.1m offset described in Sokolova et al., (2022). The resulting curved edge had new brass eyes fitted and was tied with braided twine to the footrope. The trailing edge of the SSS was laced to the bottom panel and between the selvages using 8mm bungee cord.

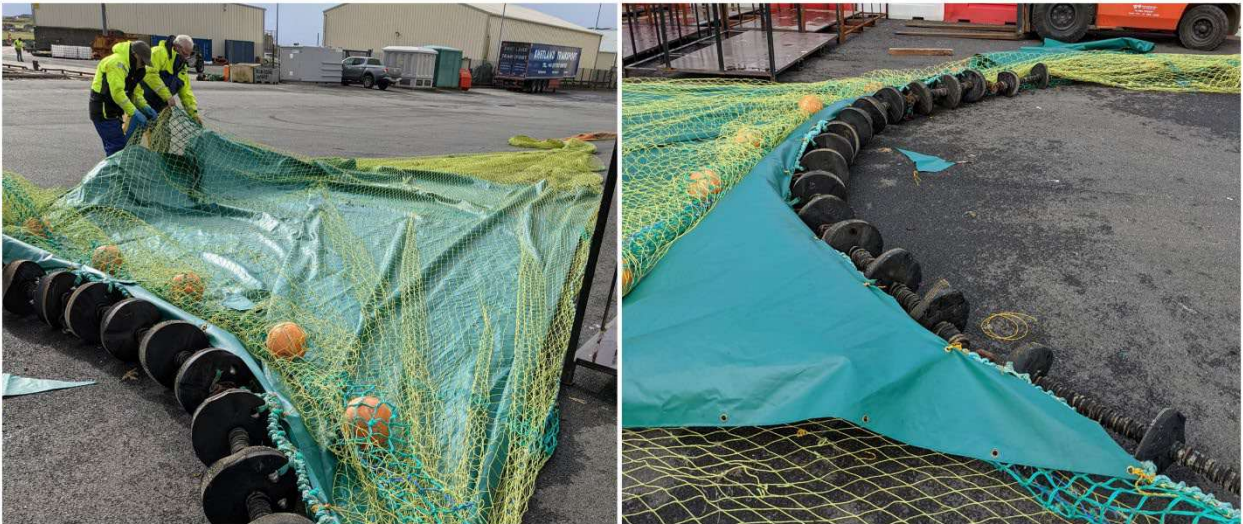


Figure 79. Rigging of altered SSS prior to D21.

Note, that normally when hauling the gear back aboard onto the net drum the friction between the ground and the rubber hoppers causes the ground gear to rotate and drag along the pier, generally without consequence. However, with the sheet fitted close to the footrope then as the hoppers rotated the sheet came into contact with the pier and would be worn through very quickly if this had not been monitored closely and manually corrected.

D21 was undertaken in the exact same area of muddy sediment as D17 and D13 to assess the performance of the altered SSS. During the tow it seemed that approximately 2m of door spread had been lost in comparison to previous tows. On hauling, it was immediately clear that the SSS had ripped away from the footrope and was left connected only at the opposite end. Upon hauling the gear out onto the pier and inspecting more closely it became apparent the SSS had also torn along its entire length during the tow. The reason for the dramatic failure of the SSS isn't obvious. It's very unlikely that a boulder caused the damage since that exact tow track had already been followed multiple times in the preceding days. The damage could perhaps be related to the large size of the SSS and it overlapping and being fastened to strong structural components of the net (i.e., the footrope and selvages) with insufficient slack between them.

A quick inspection of the stereo camera images indicate that high levels of sediment were present in D21.



Figure 80. Example stereo camera images from D21.

## D22 (31/03/22)

The largest surviving rectangular section of the SSS was salvaged after D21 and measured 3m x 6.7m. This was then refitted to the net. The leading edge was again cut, but this time with a proportionally reduced offset (150mm) required to match the shape of the foot rope due to the reduced width of the SSS. The leading and trailing edges were both fastened this time with 8mm bungee cord. This time the brass eyes at the curved leading edge were punched through a folded seam for extra strength. The trailing edge was fixed along a width defined down a side knot from the footrope using a zig-zag pattern to provide more slack than in D21.



Figure 81. Detail of SSS rigging to footrope (left) and at trailing edge (right) following further alterations prior to D22.

D22 successfully replicated D13 and D17 by fishing over the same tow track. Analysis by the same staff that analysed D13 and D17 indicated that the images in D22 were substantially less obscured than in the previous comparison tows. The first analyst estimated that 8.9% of images were obscured in D22 (compared to 15.5% in D13 and 18.9% in D17). Similarly, the second analyst estimated that 8.5% of images were obscured in D22 (compared to 14.4% in D13 and 16.6% in D17). These results indicate that smaller refitted SSS has led to a substantial improvement in image quality.

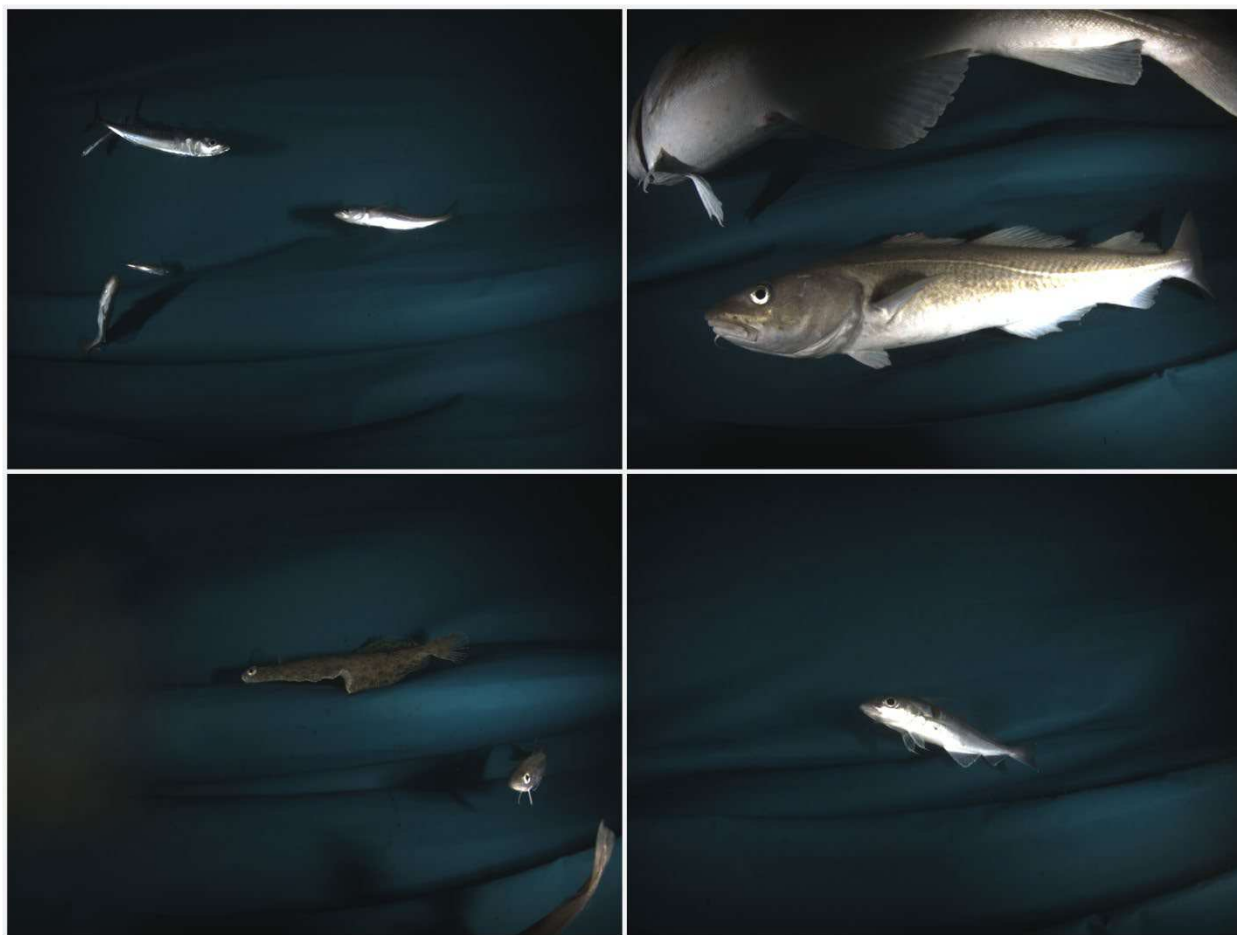


Figure 82. Example stereo camera images from D22.

## **D23, D24, and D25 (31/03/2022)**

Further tows were undertaken to inspect the interaction of sediment and the SSS using GoPro gear in very shallow waters and without artificial light.

During D23 the GoPro camera was mounted as in D19 and clearly showed the suspension of sediment by the ground gear and the leading edge of the SSS.



Figure 83. Example screenshot from D23 GoPro video.

During D24 the GoPro camera was mounted as in D20 and clearly showed the trailing edge of the SSS and the turbulent interactions with suspended sediment.

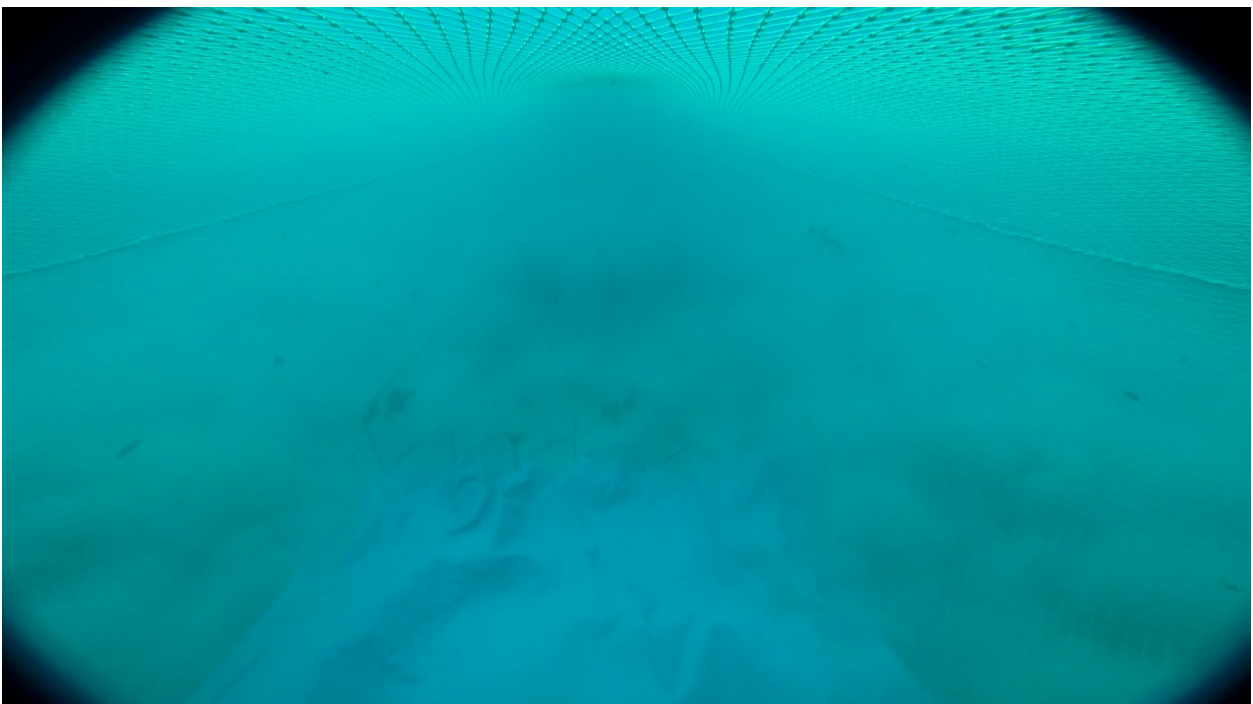


Figure 84. Example screenshot from D24 GoPro video.

In D25 the GoPro camera was mounted in the same position as in D24 but facing forward rather than aft. The footage captured the ground gear and the leading edge of the SSS at the mouth of the net from the opposite direction to D23.



Figure 85. Example screenshot from D25 GoPro video.

Considered altogether, the GoPro videos from D23 to D25 indicate that the smaller refitted SSS performs consistently along its length as hoped and prevents a substantial proportion of suspended sediment from entering the net.

## **D26 and D27 (01/04/2022)**

During the last scheduled day of the sea trials two longer tows were undertaken on fishing grounds to increase the available stereo camera dataset. Images yet to be inspected.

## **Gate sea trials**

Gate arrived to SUHI on afternoon of 06/04/22. On inspection on the morning of 08/04/22 the latch was found to not prevent the rotating mechanisms from spinning freely. Sea trials are awaiting further instruction from Aberdeen colleagues and vessel availability.

## **References**

Sokolova, M., O'Neill, F. G., Savina, E., and Krag, L. A. 2022. Test and development of a sediment suppressing system for catch monitoring in demersal trawls. *Fisheries Research*, 251: 106323.

## **Acknowledgements**

Thanks to Kenneth Pottinger (Skipper), Davie Riley (Fisheries Technician and Crew), Gary Fullerton (Crew), Sarah Ayres (Fisheries Research Assistant), and Angharad Powell (Fisheries Research Assistant).



Fisheries Innovation & Sustainability is a coalition of experts driving strategic innovation for a prosperous and sustainable UK seafood industry. Our remit is to facilitate, coordinate and leverage investment for innovation in UK seafood.

Our Member Organisations include:



Scottish Charity No: SC045119 | Company No: SC477579

