



SMARTRAWL 5.0 Final Report

FIS045

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

Rosie Ashworth
Paul Fernandes
Dewei Yi
David Morrison
Shaun Fraser

FEBRUARY 2025

SMARTRAWL 5.0

FINAL REPORT

By: Rosie Ashworth,
Paul Fernandes,
Dewei Yi,
David Morrison,
Shaun Fraser

Front cover. *Images taken in November 2023 gate trials aboard the Atlantia II research vessel, off the Shetland Islands.*

Executive summary

Smartrawl 5.0 was a project funded by the Seafood Innovation Fund of the United Kingdom's Department for Environment, Food & Rural Affairs, to develop the Smartrawl system. Smartrawl is a selective device retrofitted inside the net of a demersal trawl allowing for fish to be either caught or released in-situ underwater in order to mitigate against discards and bycatch.

There were 7 milestones of Smartrawl 5.0:

1. The construction of the latch to control the rotation of the gate, allowing the system to catch and release species.
2. The integration of an AI-capable Single Board Computer (SBC), the Nvidia Jetson Nano. This has the AI algorithms incorporated and logic to communicate with the latch.
3. To further improve the initial artificial intelligence algorithm, through training of labelled images from previous camera trials.
4. To conduct six days of sea trials in Shetland to test the gate rotation and latch functionality.
5. The completion of the final artificial intelligence algorithm, documented to successfully detect, identify and size programmed species.
- 6 To conduct six days of sea trials in Shetland testing the fully integrated system (all three components of Smartrawl).
7. To conduct two days of sea trials on a small offshore mixed demersal commercial trawler.

Milestone 1: Latch Construction for Gate Control

The first milestone focused on the development and construction of a latch mechanism designed to control the rotation of the Smartrawl gate. This latch system was crucial for enabling the selective catch-and-release functionality, allowing the gate to either retain target species or release non-target species back into the marine environment. The design and engineering phase involved ensuring the latch could engage and disengage reliably under real-world fishing conditions. After several iterations the final latch was designed, constructed and tested at sea. Video evidence was obtained showing that this worked effectively.

Milestone 2: SBC Integration with the Smartrawl System

The second milestone aimed to integrate an AI-capable **Single Board Computer (SBC)**—the **Nvidia Jetson Nano**—to process real-time data from the stereo camera. This SBC was programmed with AI algorithms capable of identifying and sizing eight species of fish, making instant classification decisions. Additionally, communication logic was developed to enable interaction between the AI system and the latch mechanism, ensuring that the gate responded appropriately to classification signals. This SBC also replaced the original computer which controlled the camera image acquisition. The new computer and associated control boards were successfully integrated.

Milestone 3: AI Algorithm Enhancement

The third milestone focused on improving the artificial intelligence (AI) algorithm responsible for species identification and size classification within the Smartrawl system. This was achieved by training the algorithm with a large dataset of labelled images collected from previous camera trials. The training and development process massively enhanced the AI's ability to accurately identify target species, ensuring more precise catch-and-release decisions during live operations.

Milestone 4: Shetland Sea Trials for Gate and Latch Testing

The fourth milestone involved a six-day field trial in Shetland aboard the vessel *Atlantia II* to test the functionality of the Smartrawl gate's rotation and latch mechanism at sea. The trials aimed to capture video footage of the gate in both catch and release modes, assess its rotational performance, and evaluate the effectiveness of the latch system. Despite early structural challenges, successful footage was obtained, demonstrating the gate's ability to selectively retain or release species. However, a malfunction in an early version of the latch prevented full testing at depth. Engineers later identified loose cabling as the issue, which was corrected for subsequent trials. The fieldwork provided valuable insights into the robustness of the gate design and highlighted necessary modifications for improved durability and performance in subsequent deployments.

Milestone 5: Smartrawl 5.0 – AI Component Final Report

The fifth milestone focused on finalising the AI component of the Smartrawl system, with a comprehensive study on fish species identification and sizing using advanced AI algorithms. This phase involved refining the dataset, optimizing the model's performance, and preparing practical installation and usage documentation for deployment.

A key achievement was the creation of a robust dataset comprising 2,918 labelled images of nine commercial fish species. The dataset captures fish at various angles, positions, and lighting conditions to improve the AI's ability to generalise across real-world scenarios. Detailed installation and usage instructions were prepared, guiding users through different configurations based on operating system (Windows vs. Linux), hardware (Jetson Nano Orin vs. workstation), and processing mode (inference vs. training). The algorithm is now implemented on the SBC and can also be run on archived data in the laboratory.

Milestone 6: Shetland Field Trials – Full Smartrawl System Integration

The Shetland field trials marked the first full-scale deployment of the Smartrawl system aboard *Atlantia II*, testing its AI-driven stereo camera, gate, and latch mechanism. Despite weather and equipment delays, the system was successfully deployed and recovered from a small fishing vessel, demonstrating its ease of use. Initial gate tests revealed rotation inconsistencies due to internal obstructions and bearing issues, requiring reengineering. The first successful at-depth gate rotation was recorded on Day 4, though occasional obstructions, affected performance. The fully integrated system was tested on Day 5, but the latch and image overexposure required camera setting adjustments. Subsequent hauls revealed inconsistencies in system boot-up, latch engagement failures, and data acquisition issues, with stereo camera and strobe malfunctions preventing image validation. However, final hauls (Days 9-10) demonstrated the latch successfully engaging and triggering gate rotation which was a first. While the trials showcased Smartrawl's potential, they also exposed some technical challenges, including gate rotation issues, camera exposure problems, and latch system inconsistencies. This will guide further refinements for future system refinements and testing.

As a result of delays in equipment and weather restraints trials aboard a commercial fishing vessel could not occur. So, milestone 7 was not completed, however Smartrawl phase 6 plans to test the system on commercial vessels.

Table of Contents

Executive summary.....	3
Introduction.....	8
Phase 5 Objectives	8
Milestone 1: Latch construction& Milestone 2: SBC Integration.....	9
Introduction	9
Components	10
1. Cameras	10
Information.....	10
Connector / Pinout	11
Synchronisation with Strobes	11
2. Strobes	11
Information.....	11
Internal Connection	12
Strobe Maintenance and Access	13
3. Magnetic Switch	14
Information.....	14
Schematic Diagram.....	16
4. Batteries	17
Information.....	17
5. Sensors and Real-Time Clock (RTC).....	18
Information.....	18
6. Latch	19
Information:.....	20
7. HardwareIntegration	20
ESP32 Data acquisition.....	20
Architecture.....	20
I2C address conflict:	22
Incompatible GPIO library:.....	22
8. Jetson Nano.....	23
Architecture.....	23
P-FET Switch:	23
Synchronisation of Strobe and Camera	24
9. Latch architecture	25
a.Power management:	25
b. Latch control:	26
10. SoftwareIntegration	28
Camera Driver Node.....	28

Latch Solenoid Controller Node	28
Strobe Trigger Node.....	29
AI Module ROS Wrapper.....	29
AI Module.....	29
11. Filestorage	30
Sambashare	30
On Windows 11	30
On Windows 10	31
After setting the static IP	33
Parameter files.....	34
Milestone 3: Development and completion of final AI Algorithm.....	36
Abstract	36
Milestone 4: Six days of field trials in Shetland.....	58
Introduction	58
Aim and pre-trial objectives	59
Aim	59
Specific objectives.....	59
Day 1 – 14 th November 2023	60
Day 2 – 15 th November 2023	61
Day 3 – 16 th November 2023	64
Results	65
Conclusion	66
Milestone 5: Smartrawl 5.0: AI Component Final Report	67
1. Overview	67
2. Building Dataset for Commercial Fish Species	67
3. Installation and Usage Instructions	68
3.1 Inference Only	69
Create conda environment.....	70
Download MMDetection.....	70
3.1.2 Installation (GPU)	72
3.2 Usage.....	73
3.2.1 Note on CSV log.....	76
3.3 Training	77
3.4 Extracting Calibration File	78
4. Improved Performance statistics to Identify Different Species on Jetson Orin Nano	78
4.1 Jetson Orin Nano	78
4.1.1 CUDA.....	79
4.1.2 Jetson Orin Nano Use Cases.....	79
4.1.3 Software and Hardware Requirements.....	80

4.2 Improved performance statistics to identify commercial fish species	80
4.2.1 Quantitative Evaluation on Jetson Orin Nano.....	80
4.2.2 Qualitative Evaluation	81
5. Fish Sizing Improvements	83
5.1 Background	83
5.1.1 Stereo projection	83
5.1.2 Notes	84
5.2 Fish Sizing Results	85
5.2.1 Stereoscopy	85
5.2.2 Demos of fish Sizing.....	86
References.....	88
Milestone 6 : Shetland field trials of the full Smartrawl system integration	90
Day 1 - 21/11/2024:	91
Day 2 - 22/11/2024:	92
Day 3 - 28/01/2025:	92
Day 4 - 29/01/2025:	92
Day 5 - 31/01/2025:	93
Day 6 – 05-07/02/2025:	94
Day 8 (10/02/2025):	94
Day 9 – 12/-2/2025:.....	95
Day 10 - Feb 14/02/25	97

Introduction

Smartrawl is an in-water sorting device which is retrofitted inside the extension of a commercial fishing trawl net. The system has three components: a stereo camera, taking images of animals in the trawl; a computer, with artificial intelligence to detect, identify and size animals; and a gate, controlled by the computer to open or close via the latch, catching or releasing animals. Smartrawl is designed with, and for, UK fishers, and their vessels to mitigate against discards and bycatch: the entire system needs no cables from the vessel and can be pre programmed dependent on the fishermen's desired catch.

The stereo camera system, takes paired images of animals as they pass by. The system consists of two cameras, 15cm apart, and two flash units, further apart (70cm), all controlled to trigger simultaneously and continuously (2 Hz) by a Single Board Computer (SBC). The camera, SBC, and batteries are housed in single underwater aluminium housing, as is each flash unit, all rated to a depth of 500m. The housing and flash units are mounted in a neutrally buoyant frame designed to protect the glass domes of the camera housing and flash units. This frame has been demonstrated to be robust, and is easily retrofitted to the inside of the extension.

Images acquired by the stereo camera system are then processed by the AI algorithm where species are detected identified and measured. If specified species and sizes are detected the controller sends a signal to trigger the latch, enabling the gate to rotate into the catch position.

A patented gate system rotates using the force of water passing through the trawl to operate it. It does this by incorporating a cylindrical design with rotating conical sections which open and close off the cod-end, and adjacent doors to side panels in the extension, to catch or release animals. The force of water acts on vanes in the cylindrical gate to drive the rotation between the catch and release states. The rotation is restrained by a latch which is released under control of a computer. The cylindrical gate fits into the existing extension by virtue of its diameter being equal to that of the extension during trawling.

Phase 5 Objectives

1. The construction of the latch to control the rotation of the gate, allowing the system to catch and release species.
2. The integration of an AI-capable Single Board Computer (SBC), the Nvidia Jetson Nano. This will have the AI algorithms incorporated and logic to communicate with the latch.
3. To further improve the initial artificial intelligence algorithm, through training of labelled images from previous camera trials.
4. To conduct six days of sea trials in Shetland to test the gate component and rotation.
5. The completion of the final artificial intelligence algorithm, documented to successfully detect, identify and size programmed species.
- 6 To conduct six days of sea trials in Shetland testing the fully integrated system (all three components of Smartrawl).
7. To conduct two days of sea trials on a small offshore mixed demersal commercial trawler.

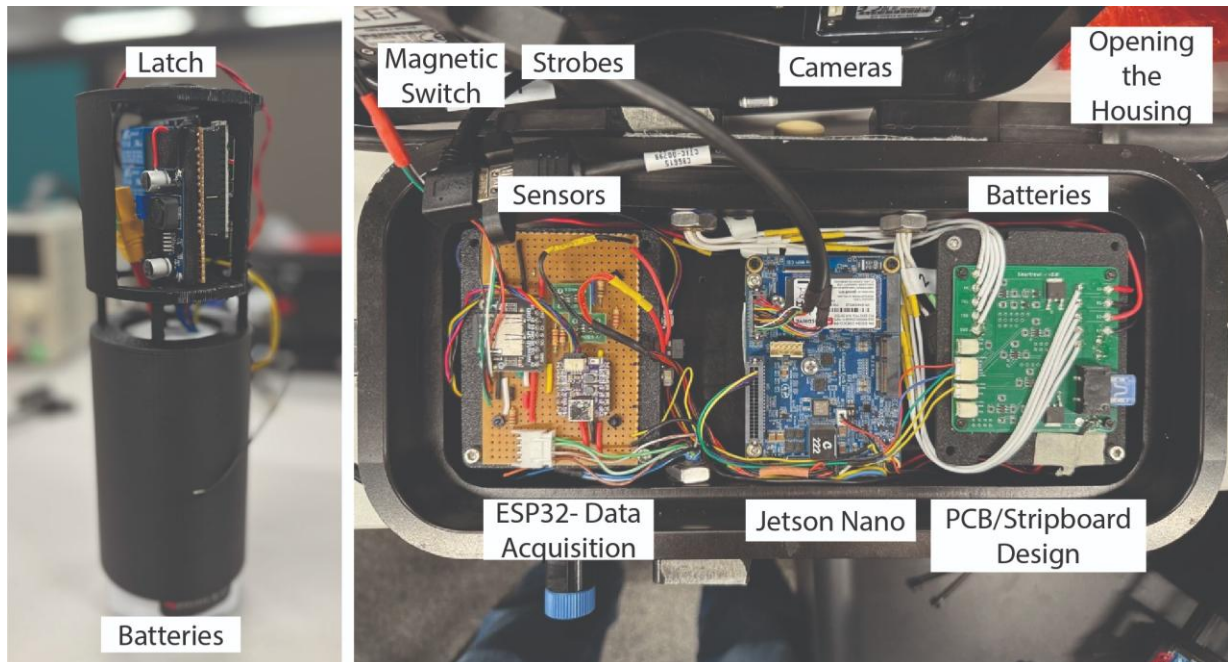
Milestone 1: Latch construction& Milestone 2: SBC Integration

Introduction

The Smartrawl system is specifically designed to enhance selectivity in fishing by identifying and capturing target fish species while releasing non-target species unharmed. This capability is achieved through the integration of advanced artificial intelligence (AI) algorithms, which allow the system to analyse visual data in real time. By recognising the distinct physical characteristics of the desired species, the system automates decision-making, ensuring that only fish meeting specific criteria are retained.

The system is built around a camera suite, micro-controller (**QT PY ESP32 PICO board**), sensors, a single-board computer (**Jetson Orin Nano**) with a carrier board (**Hadron**), and a pair of strobes and cameras. Upon activation, the micro-controller enters a deep sleep mode immediately to reduce power consumption and can be woken by a magnetic switch. Once activated, sensors begin collecting data (orientation, depth, temperature, and power consumption) and store it on the onboard SD card. When the system reaches the desired depth or set timer, the micro-controller powers up the single-board computer, which then triggers the cameras to capture images when the strobes flash. These images are processed by the computer, and upon detecting the target species, a signal is sent to trigger the latch, securing the catch. This process continues until the system is powered off.

This document provides a comprehensive overview of the Smartrawl system, detailing the components within the camera housing and latch container.



Components

1. Cameras

Information

i **Camera Model:** CM3-U3-31S4CS

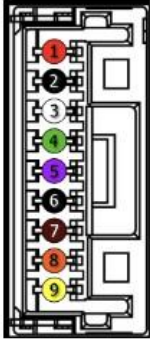
Resolution: 2048 x 1536

Website: <https://www.teledynevisionsolutions.com/products/chameleon3-usb3/?model=CM3-U3-31S4C-CS>

Spinnaker SDK <https://www.teledynevisionsolutions.com/support/support-center/software-firmware-downloads/iis/spinnaker-sdk-download/spinnaker-sdk-download-files/>

Connector / Pinout

Table 6.1: GPIO pin assignments (as shown looking at rear of camera)

Diagram	Color	Pin	Function	Description
	Red	1	V_{EXT}	Allows the camera to be powered externally 5 - 24 VDC
	Black	2	GND	Ground for Input/Output, V_{EXT} , +3.3 V pins
	White	3	+3.3 V	Power external circuitry fused at 150 mA maximum
	Green	4	GPIO3 / Line3	Input/Output
	Purple	5	GPIO2 / Line2	Input/Output
	Black	6	GND	Ground for Input/Output, V_{EXT} , +3.3 V pins
	Brown	7	OPTO_GND	Ground for opto-isolated IO pins
	Orange	8	OPTO_OUT / Line1	Opto-isolated output
	Yellow	9	OPTO_IN / Line0	Opto-isolated input

JST connector female: BM09B-NSHSS-TBT JST

connector male: NSHR-09V-S

JST contact: SSSL-003T-P0.2

Synchronisation with Strobes

- Connect pin 4 (GPIO3) from both cameras to strobe trigger signal from Jetson •

Connect pin 2 (GND for input/output) from both cameras to ground

2. Strobes

Information

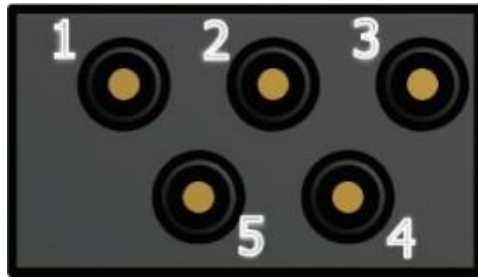
i Model Cree XLamp CXB3590 LED

ATTENTION

Keep the exposure short – 1~2 milliseconds

Place the strobe face down on a rag to protect the glass

Connector on Camera Housing



Face view (male)

SubConn Male	To PCB
1	GND
2	Trigger B
3	Trigger A
4	Power B
5	Power A

The trigger pins in Strobe A and Strobe B are internally tied to the same pin to SubConn male connector (the white connector in below image), allowing them to be triggered by a single signal, Trigger A. The Trigger B is just reserved for further use.

Internal Connection



SubConn Bulkhead	Strobe A	Strobe B	Voltage
1	GND	GND	GND
2	X	X	X
3	Trigger	Trigger	5V
4	Power B	X	24V
5	X	Power A	24V



Note:

The circuitry inside the strobe is designed by Rick.

The connections may varies in different strobes.

The trigger pins for both Strobe A and Strobe B are connected to the same pin on the SubConn male connector, allowing them to be triggered by a single signal.

There are additional electronics in the strobe. There are two high current led drivers and supporting circuitry. The trigger input is nominal 5V. The trigger is thru an Attiny which controls the PWM input to the drivers. The Attiny just provides a safety where if the input trigger is held high for >50 ms, the driver PWM is dropped to 50%. These are designed as strobes and as such aren't meant to be on very long. 1000-2000 us is the nominal range of exposure.

Strobe Maintenance and Access

Disassemble

Place the strobe face down on a rag

Remove rear plug: Removing the bottom knob by rotating it anti-clockwise using spanner



Pressurise chamber: Hold the compressed air supply to the air hole and start pressuring the chamber **Very Slowly** until the front port pops out



Open up the case



Assemble

Arrange Cables Ensure all cables are neatly placed within the cavity.

Place Metal Plate Position the metal plate (where the LEDs and PCB sit) securely on the rim, making sure no cables are trapped underneath.

Attach Glass Secure the glass by screwing it into place.

Reinstall Bottom Knob Reattach the bottom knob, ensuring the o-ring is properly positioned, and tighten by rotating clockwise with a spanner.

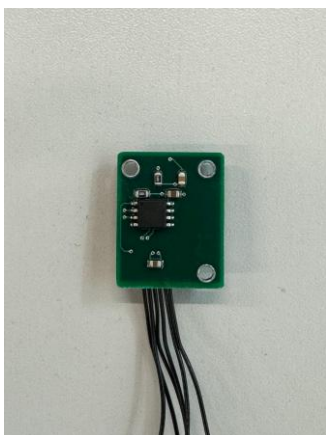
3. Magnetic Switch



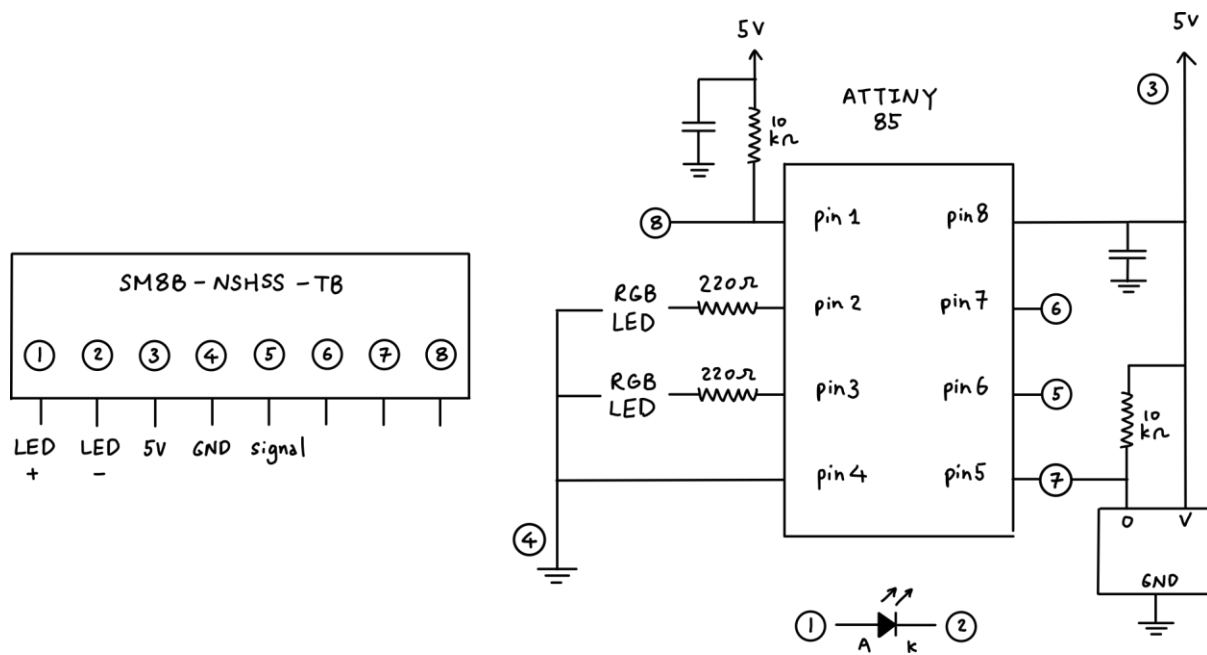
This is designed by Rick. The following is from our best understanding

Information

This magnetic switch board is mainly composed of an Attiny85 microcontroller and a Hall effect switch, which together detect the presence of a magnetic field to trigger the system's activation.



Schematic Diagram



⚠ There is no ballast resistor connected to the LED on this PCB, so a 1k resistor is required if the LED is to be safely powered and to limit the current flowing through it.

Operation Table

Place the magnetic stylus on top of the circular area on the PCB

	pin ⑤	pin ⑥
beginning	5V	5V
orange	0V	5V
orange → blue	5V	5V
orange → white	5V	0V
white → blue	5V	5V

4. Batteries

Housekeeping - very **IMPORTANT**

Never leave it unattended while charging

Always charge the battery in a fire-safe and well-ventilated location Do not exceed the recommended charge rate of 10A

♦
If the battery becomes hot or shows any signs of damage or changes in appearance during charging, discontinue use immediately.

♦
To turn off the battery, simply press the "**Stop**" button to disconnect the charging connection, then proceed to unplug the cables.

Information

There are four batteries includes in the system.

Rechargeable batteries in camera housing

Rechargeable batteries in latch container

Coin battery for real time clock(RTC) connected to ESP32 PICO

Rechargeable battery for real time clock(RTC) connected to the Hadron carrier board



Name	Ansmann Standard Li-ion 3S1P Battery Pack	BlueRobotics Lithium-ion Battery	Vanadium Pentoxide Lithium Rechargeable Coin Battery	CR1220 Lithium Manganese Dioxide Coin Battery
Nominal Voltage	10.8V	14.8V	3V	3V
Capacity	3.5 Ah	15.6 Ah	7 mAh	36 mAh
Purpose	There are the two batteries sitting each side of the camera housing, located beneath the electrical circuit. When the short power plug is inserted into the 7-pin Subconn connector, the two batteries are connected in series, delivering a 24V unregulated output to power the strobes.	This is a Lithium- ion battery located at the bottom of the latch container. This battery powers the Arduino circuit in the latch container, as well as the ESP32 PICO board along with sensors and Jetson Orin Nano in the camera housing.	This maintains timekeeping for the RTC on the Jetson Orin Nano when the main power supply is disconnected. When external power is available, it charges the battery.	This maintains timekeeping for the DS3231 RTC connected to the ESP32 PICO when the main power supply is disconnected. When external power is available, the DS3231 operates normally without drawing power from the battery.
RS stock no.	144-5695		669-0505	866-0653


5. Sensors and Real-Time Clock (RTC)


Information

The following 4 components are daisy-chained together to ESP32 PICO through STEMMA QT/QWIIC (JST SH-4) cables.

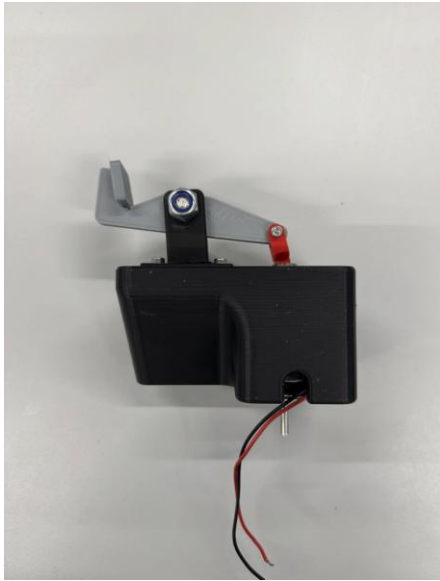


Module Name	Pressure Sensor	Power Monitor Module	IMU Orientation Sensor	Real Time Clock (RTC)
Chip Name	Keller 4LD	INA219	BNO055	DS3231
I2C Address	0x40	0x44	0x28	0x68
Digikey no.		1528-904-ND	1528-4646-ND	1528-5188-ND
Libraries	<KellerLD.h>	<Adafruit_INA219.h>	<Adafruit_BNO055.h>	<RTCLib.h>
Purpose	High-precision digital pressure sensor to determine the camera's depth underwater.	High-side voltage and current sensor for monitoring the power supplied to the Jetson Orin Nano.	9-axis IMU for determining the orientation of the camera housing.	High-accuracy real-time clock with battery backup, providing precise timestamps.

 Note: The 0x40 I2C address is already reserved on the Hadron carrier board, which has only one I2C bus. This is one reason why the pressure sensor is connected to the ESP32 instead. Although its I2C address could be changed using a USB (K-404-T) device, this would incur an additional cost of £125.

 Bridge the A1 solder jumper behind the module to change the I2C address from 0x40 to 0x44

6. Latch



Information:

i Model BLM5/08-12VDC-80W P/T

Coil Voltage 12 VDC

Stroke 25 mm

The magnetically latching solenoid is chosen for its ability to maintain its position without requiring a constant power supply.

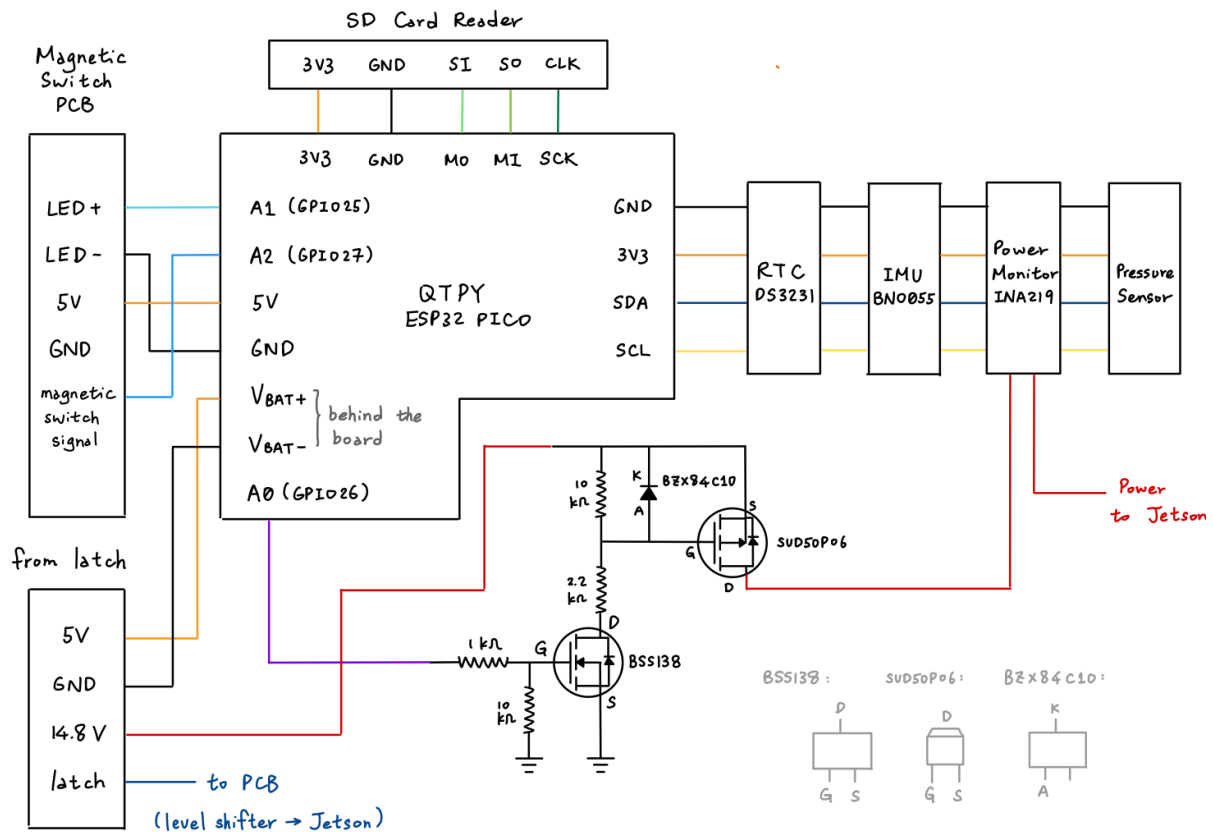
7. Hardware Integration

ESP32 Data acquisition

Architecture

The schematic below depicts the hardware integration for an ESP32-based data acquisition system. Upon power-up, the ESP32 PICO microcontroller enters deep sleep mode to conserve power. An interrupt pin connected to the magnetic switch, which is pulled low when a magnet is detected, allows the microcontroller to wake up. Once activated, the microcontroller interfaces with a real-time clock (DS3231), an inertial measurement unit (BNO055), a power monitor module (INA219), and a pressure sensor (Keller 4LD), all connected in a daisy-chain configuration via STEMMA connectors. When a specified depth or time threshold is reached, the microcontroller triggers a signal to a P-FET switch circuitry (BSS138 + SUD50P06), which powers on the Jetson board. All sensory data is timestamped and stored on an SD card for later

retrieval and analysis. The components are currently integrated onto a stripboard for compact assembly, with the potential for future integration into a customised PCB design.



The ESP32 PICO microcontroller is dedicated to sensor data acquiring. Ideally, this shall be integrated with Jetson Orin Nano along with Hadron carrier board. However, the two following issues preventing this integration, which is why the ESP32 PICO is being used in this setup instead.

I2C address conflict:

It appears that the I2C address 0x40, which is used by the Keller pressure sensor on I2C bus 1 of the carrier board, is already reserved for an existing device on the NVIDIA module.

Running

`i2cdetect -y -r 1` will display "UU" in this case. To resolve this, you will need the K-404-T to change the I2C address or use a multiplexer, which will take up additional space.

Incompatible GPIO library:

The GPIO library used by the development kit is not supported by the carrier board for JetPack 5, meaning the libraries for pressure sensor may not be available.

8. Jetson Nano

Architecture

The Jetson Orin Nano single-board computer, paired with the Hadron carrier board, manages strobe triggering, camera image capture, AI computation and latch command. The schematic below details the interface between the Jetson Orin Nano and other system components - strobes, cameras and latch. Three separate control signals from the Jetson (pins PN.01, PQ.06, and PH.00) manage the latch trigger, strobes and cameras trigger, and strobe power enable functions respectively. This design ensures clean signal conversion and reliable power switching for the strobe lights while maintaining proper isolation between the control and power circuits.

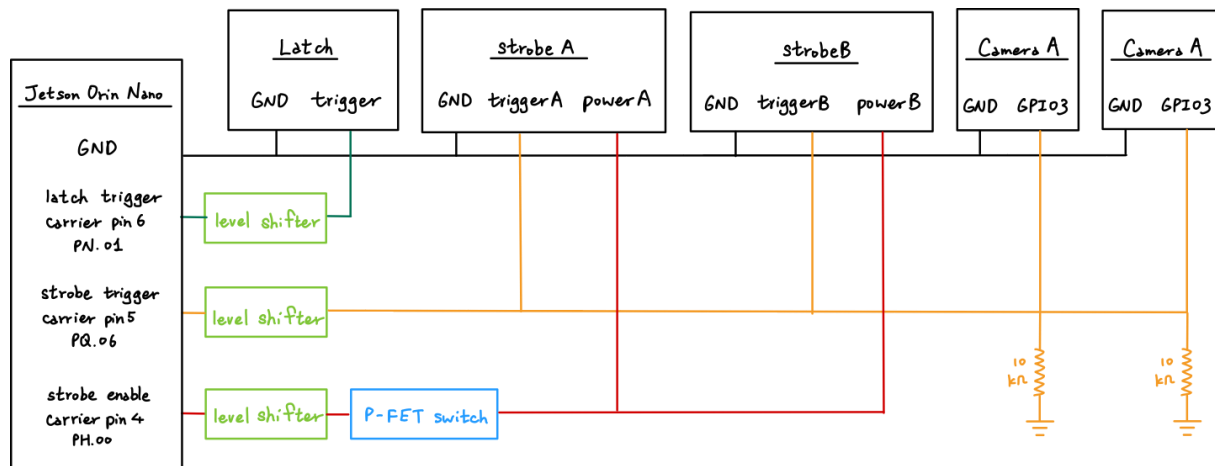
The system's circuitry is primarily composed of the following two circuits: **Level**

Shifter:

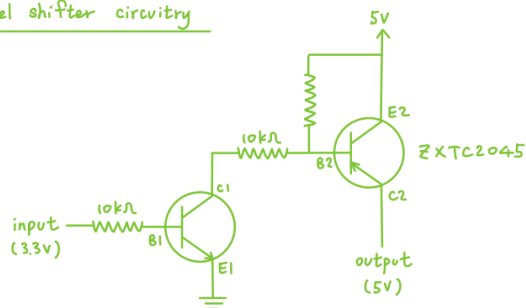
The level shifter circuitry employs a Sziklai pair (also known as a complementary compound pair) configuration using ZXTC2045 transistors, which combines an NPN (Q1) and PNP (Q2) transistor to achieve superior voltage level translation. This configuration is used to convert the Jetson's 3.3V signals to 5V for compatibility with other system components - strobes, cameras and latch. While a single transistor could perform basic switching, the Sziklai pair offers several advantages. When the input signal goes high, Q1 conducts, which then drives Q2 into saturation, providing a more robust and cleaner output signal. The compound arrangement provides higher current gain compared to a single transistor, ensuring sharp switching transitions and better load driving capability. Additionally, this configuration maintains consistent performance across temperature variations and offers improved linearity in the transition region, making it ideal for reliable signal level conversion in marine environments where stable operation is crucial

P-FET Switch:

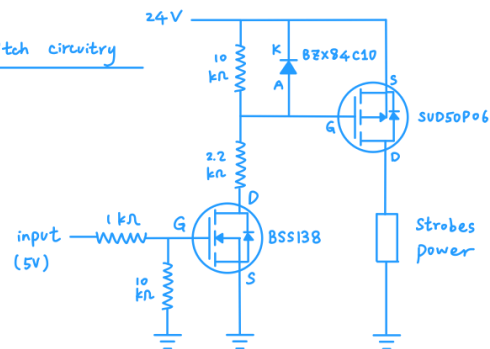
A P-FET switch circuitry utilises an N-Channel enhancement MOSFET (BSS138) and P Channel enhancement MOSFET (SUD50P06) to create a high-side switch, enabling control of high-voltage loads using a low-voltage control signal. This circuitry is implemented for the strobe enable functionality. When activated, this P-FET switch circuitry allows 24V to flow to the strobe, providing the required power. This dual-MOSFET configuration is necessary because P-channel MOSFETs require their gate voltage to be lower than their source voltage to conduct. The N-channel MOSFET acts as a driver, effectively pulling down the P-channel MOSFET's gate voltage to turn it on while maintaining proper voltage levels for both MOSFETs. This arrangement also ensures robust isolation between the low-voltage control circuitry and the high-voltage load circuit.



level shifter circuitry



P-FET Switch circuitry



Synchronisation of Strobe and Camera

The camera and strobe synchronisation system employs a carefully coordinated timing mechanism to ensure precise image capture while maintaining power safety and efficiency. The strobe trigger signal from Jetson Orin Nano is directly connected to strobe trigger pins and the camera's GPIO pins, enabling the camera to synchronise its image capture with the strobe's flash. This synchronisation is crucial as it ensures the camera's exposure coincides with the peak illumination from the strobe. Before triggering the flash, the system first manages the strobe's power charging cycle through its enable pin. This pin is initially pulled high for 1ms to allow the strobe's internal capacitors to charge, then pulled low. This controlled charging approach prevents excessive current draw that could rapidly drain the batteries. Once the capacitors are charged, the trigger pin initiates a 2ms flash command, during which the stored energy in the capacitors is discharged to provide illumination for image capture, rather than drawing directly from the batteries. This sophisticated power management and timing strategy ensures optimal image quality while protecting battery life by utilising stored capacitor energy rather than direct battery power for the high-current flash event.

It is worth noting that although the flash time is set to 2ms, oscilloscope measurements reveal an actual flash duration of 9ms. The primary reason for the timing discrepancy between the commanded flash duration and the measured flash duration likely stems from software execution and system latency. First, Python's `time.sleep()` is not designed for microsecond-precise timing. When a 2ms sleep is requested, the actual sleep time might be longer because Python needs to wait for the next available CPU cycle after the sleep period, and the scheduler might not immediately return control to the process. Secondly, using subprocess module to control GPIO adds another layer of system calls and potential delays. Each subprocess call involves creating a new process, executing the command, and cleaning up, which introduces additional overhead. This process creation and management time adds to the overall timing inconsistency.

9. Latch architecture

Architecture

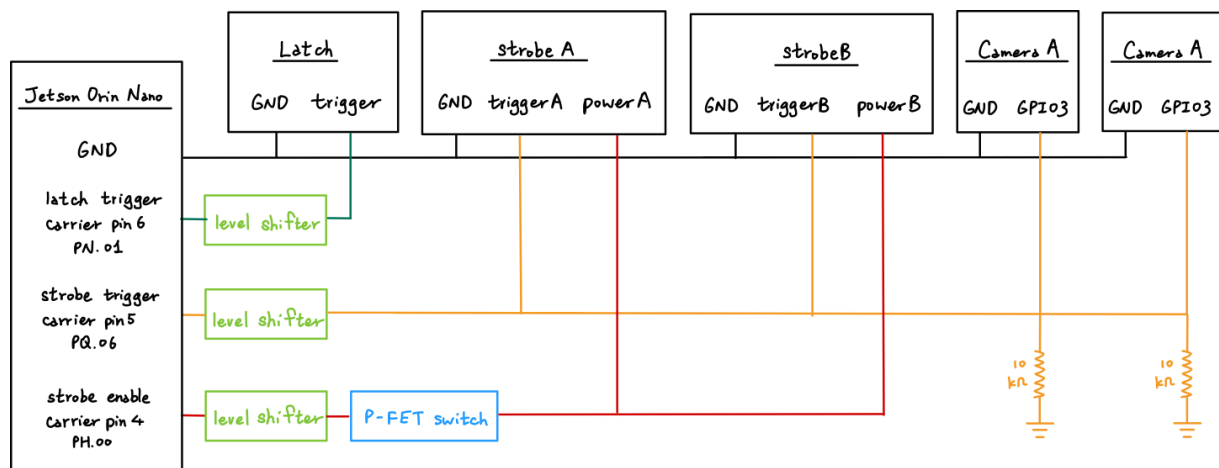
This schematic below illustrates the power distribution and control system for components locating inside the latch container. This latch system allows the attaching gate to be set into 'catch' or 'release' position.

a. Power management:

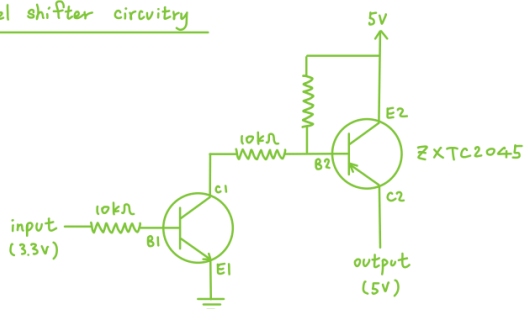
The system is powered by a 14.8V Lithium-ion battery connected through a switch and protected by a 10A fuse. The power path is fed to a low voltage (XH-M609), which serves as a critical battery protection component in the system. It monitors the 14.8V battery voltage and automatically cuts off power when the voltage drops below a preset threshold, which prevents over-discharge that could permanently damage the lithium battery. This is followed by a buck converter (step-down) switching regulator (HW411) that converts 14.8V input to a stable 9V output for powering the Arduino Nano microcontroller. Switching regulators are preferred over linear regulators in this application because of their higher efficiency, typically exceeding 90%, which is crucial for battery-powered systems.

b. Latch control:

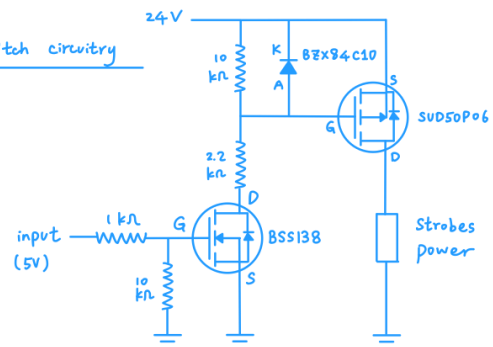
The latch is attached to the gate to set the get into either 'catch' or 'release' position. Here, a latch solenoid (BLM5 8-12V P/T) is deployed as it is a specialised type of solenoid designed to hold a position without continuous power. When energised, the latch solenoid moves a plunger to a specific position and then "latches" in place, maintaining that position even after the power is removed. To return to the initial position, a reversed power is needed to unlatch it. Therefore, only a brief surge signal is required to latch or unlatch the solenoid. To provide bidirectional pulses, the system employs a 2-channel relay configuration, which interfaces with the Arduino Nano. When Channel 1 is activated and Channel 2 is deactivated, current flows in one direction to release the lock. Conversely, when Channel 1 is deactivated and Channel 2 is activated, current flows in the opposite direction to engage the locking mechanism. A 1k Ω pull-down resistor is connected to latch trigger pin, which receives the latch control signal from the Jetson Orin Nano.

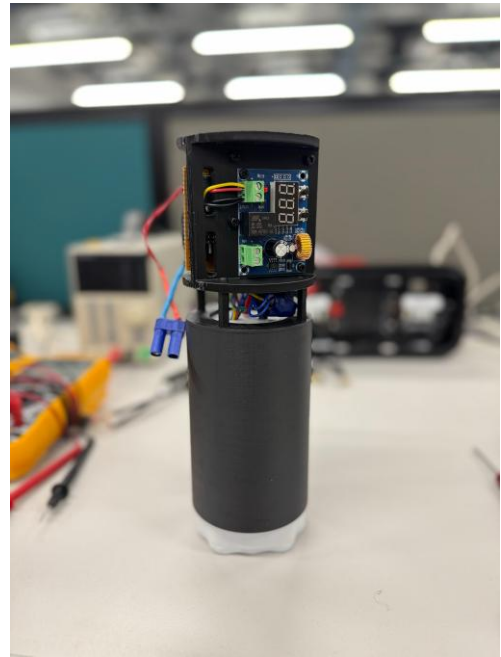


level shifter circuitry



P-FET Switch circuitry

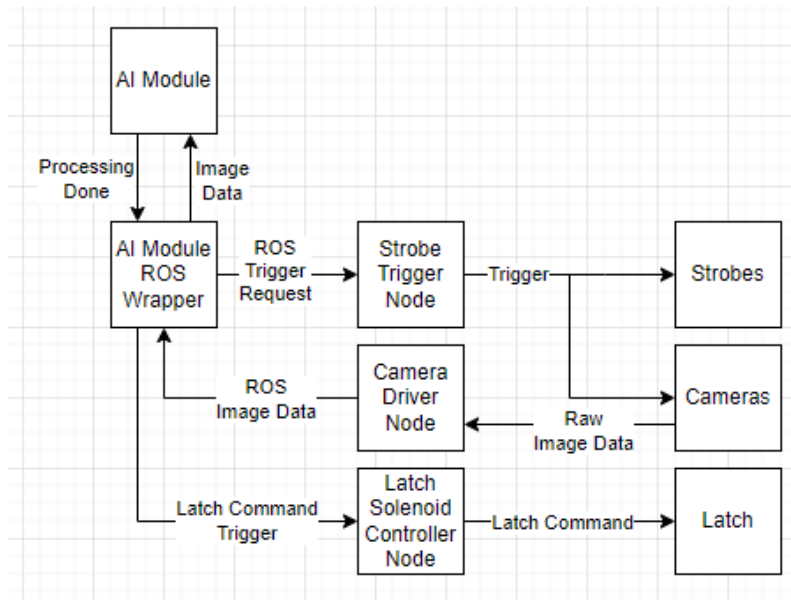




10. Software Integration

Node Structure

Overview



The overall node structure can be seen in the above diagram.

Camera Driver Node

This node is responsible for retrieving the raw image data produced by the cameras to convert to a ROS Image message. This image message is published to the ROS network, where the AI Module ROS Wrapper can subscribe from to retrieve the image data. This node is also responsible for setting camera parameters on boot.



The configuration of the cameras can only occur on boot, meaning the cameras will not accept new parameters being set by the Camera Driver Node being restarted with a different configuration without a full power cycle.

Latch Solenoid Controller Node

This node is responsible for processing Latch Command Trigger requests and changing the state of the latch according to the request. It makes use of a ROS service server to allow other nodes on the ROS network to trigger this behaviour.

Strobe Trigger Node

This node is responsible for the triggering of the strobe/camera pair using a PLC interface. It makes use of a ROS service server to allow other nodes in the ROS network to trigger the strobe/camera pair. The system is configured to turn on the strobe for 2ms while the camera takes the image in the same timing. The triggering of the camera results in a new image being captured, which is processed by the Camera Driver Node.

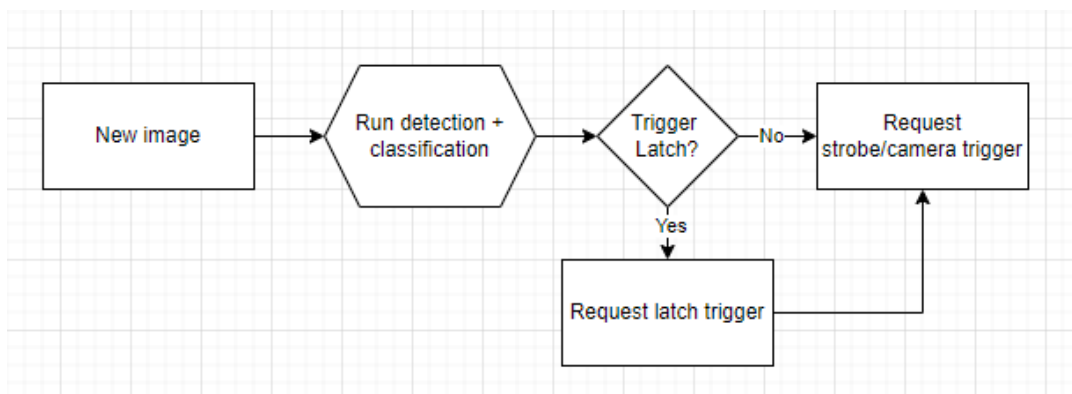
AI Module ROS Wrapper

This node is a basic interface between the AI Module python code and the ROS network. This node creates functions that allows the AI Module to make service requests in the ROS network (mainly triggering the strobe/camera pair & triggering the latch) as well as converting the image data published by the Camera Driver node to a data format easily accessible by the AI Module python code.

AI Module

This is the main processing module for running classification. The node is also responsible for triggering the strobe/camera pair, as well as triggering the latch to switch states.

The high-level overview of the AI module action flowchart is as follows:



New image is received by the AI Module (from the AI Module ROS Wrapper), triggering the processing of the image

Object detection and classification is ran on the image to detect any fish that may be in frame

Once classification is done, whether the latch is triggered is decided based upon configuration parameters. If yes, the latch trigger request function is called for the Latch Solenoid Controller Node to handle. If no, the latch trigger step is skipped. The processed image is saved to disk at this stage.

Regardless of the latch being triggered, once processing is complete the strobe/camera pair trigger request function is called for the Strobe Trigger node to handle.

As described under the Strobe Trigger Node, the triggering of the strobe/camera pair in step 4. results in the cameras capturing a new image for the Camera Driver Node to acquire and publish to the ROS network. This image is then subscribed to by the AI Module ROS wrapper, which is sent to the AI Module to start the process from step 1 again.

The above architecture results in the AI Module's processing speed become the bottleneck for the camera framerate. (The current configuration results in 1 image every 2 seconds)

11. File storage

Sambashare

The system makes use of Sambashare to make it possible for devices connected on the same network via ethernet as the system to access the parameter files as well as the saved images themselves.

The static IP of the system is 192.168.88.245, and the system must be powered to be able to access via ethernet connector.

Power settings are 15V 2A if you have a power supply, otherwise connecting the positive and negative terminals of a car battery to the respective power cables should suffice to power the system, don't forget to use the magnetic switch to power on once the system is being powered by an external source)



If you are on Windows 10, extra steps are required. Refer to guide here for how to set up an extra network location:

<https://www.techrepublic.com/article/how-to-connect-to-linux-samba-shares-from-windows-10/>

On Windows 11

While connected via ethernet with the system, you will need to edit their ethernet connection settings to the following or similar with a static IP



On Windows 10

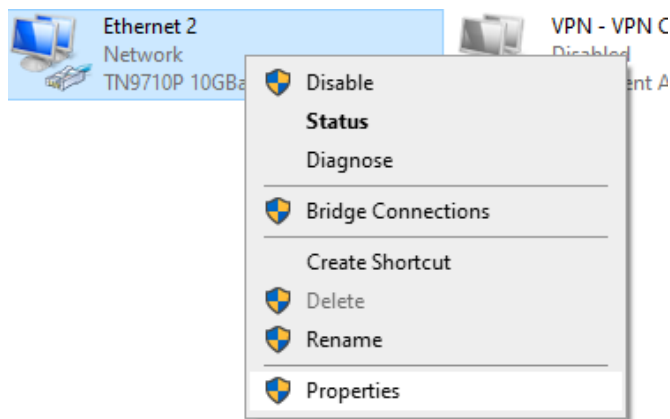
Open the Windows search bar and enter the following and press enter to open Network Connections:

then press enter.

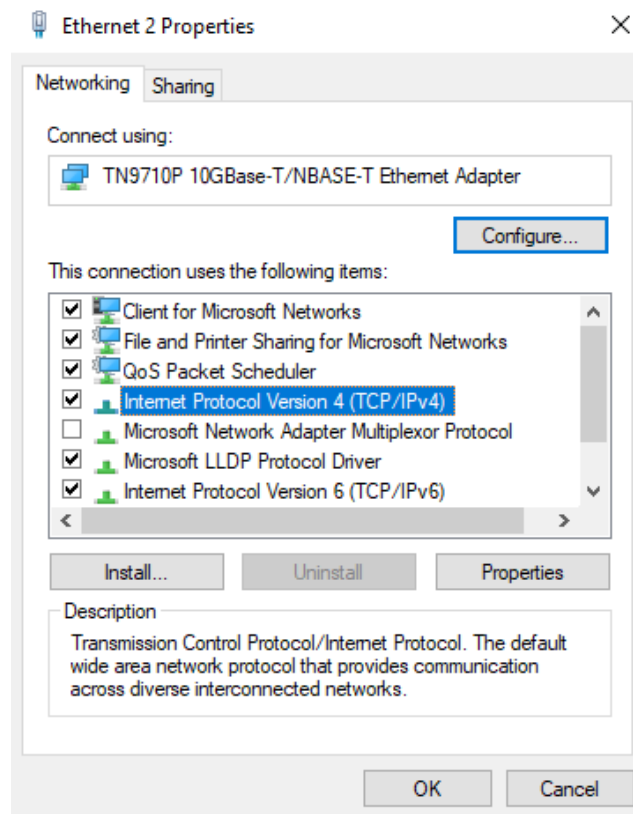


This will lead you to a window that looks like below:

While connected to the system via ethernet, right click on the Ethernet icon that shows as connected and select “Properties”:



Double-click on “Internet Protocol Version 4 (TCP/IPv4)”:



In the following window, enter the details shown below:

General

You can get IP settings assigned automatically if your network supports this capability. Otherwise, you need to ask your network administrator for the appropriate IP settings.

☐ Obtain an IP address automatically

☒ Use the following IP address:

IP address: 192 . 168 . 88 . 10

Subnet mask: 255 . 255 . 255 . 0

Default gateway: . . .

☐ Obtain DNS server address automatically

☒ Use the following DNS server addresses:

Preferred DNS server: . . .

Alternate DNS server: . . .

☐ Validate settings upon exit

Advanced...

OK Cancel

Click the OK button on “Internet Protocol Version 4 (TCP/IPv4) Properties” window, and also click the OK button on “Ethernet Properties” window to save the settings.

After setting the static IP

Once connected via ethernet you can access the config files and images by going into file explorer, selecting the address bar in the file explorer and entering the following:

\\192.168.88.245\sambashar

When you first connect you'll be prompted to give credentials:

Windows Security

Enter network credentials

Enter your credentials to connect to:

User name

Password

Domain:

☐ Remember my credentials



You might see it come up with your current account as a default selection, you can get to above screen or similar by clicking the option for signing in on a new/different account

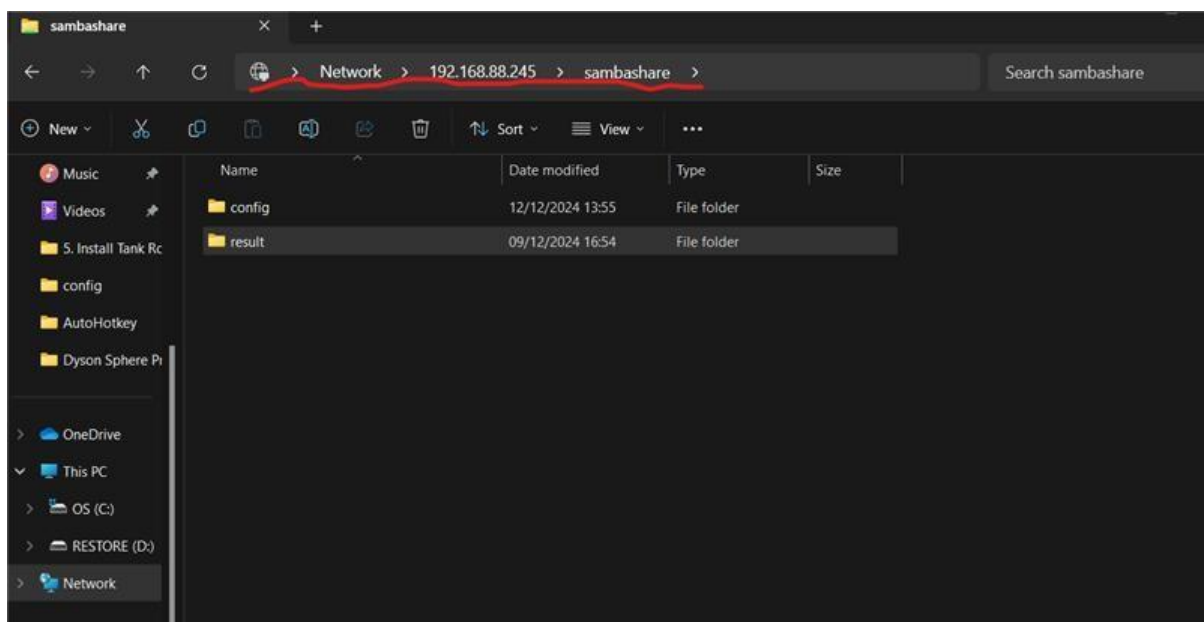
The username and password are:

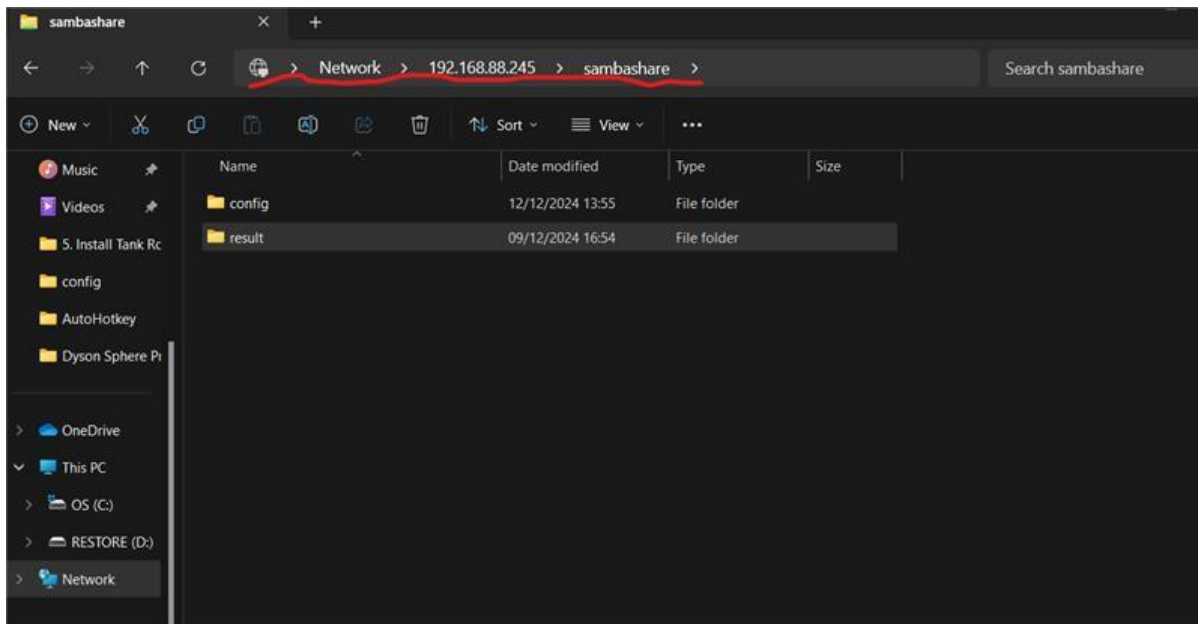
username: root

password: smartrawl

Parameter files

Config files are found in the "config" subfolder once connected to the sambashare directory:





the red lined section is the address bar of the file explorer in windows Camera parameters can be found under

`config/camera_params.csv`

The params that can be adjusted are:

gain_auto	"Continuous" will set it to automatic gain adjustment regardless of value set for "gain". When set to "Off", it will take and use the value in "gain" as a static value.
gain	A static value used for the gain assuming "gain_auto" is set to "Off"
exposure_auto	"Continuous" will set it to automatic exposure adjustment regardless of value set for "exposure_time". When set to "Off", it will take and use the value in "exposure_time" as a static value.
exposure_time	A static value used for the exposure time assuming "exposure_auto" is set to "Off"



NOTE As described in the Camera Driver Node section, a full reboot is required for the changes to take effect on the cameras

Milestone 3: Development and completion of final AI Algorithm

Smartrawl 5.0 Report for AI Package: An Underwater Aquatic Animal Instance Segmentation Benchmark Dataset for Sustainable Fishing and Beyond

Dewei Yi, Chris Moorhead, Yiren Li, Yining Hua, and Paul G. Fernandes

Abstract

Sustainable fishing would greatly benefit from improved precision in the capture process, requiring fine-grained underwater aquatic animal instance segmentation (UAAIS). While deep learning has shown promise in instance segmentation, deep-based UAAIS is hindered by limited labeled datasets for object detection and semantic segmentation. To address this, we constructed UAAIS2K, a dataset with 2,431 labeled images encompassing various objects encountered in fishing. It includes common marine animals, camouflaged species, and underwater scenes without marine life. Each image from UAAIS2K dataset has rich annotations, including an object-level mask, a category name, a bounding box, and attributes. Furthermore, a novel instance segmentation approach, Cascaded Boost Seesaw Contrastive Network (CBSC-Net), is proposed. By employing contrastive learning and integrating cascaded instance boosting and seesaw loss modules, our proposed method enhances instance segmentation performance for fish detection and species identification in underwater trawl capture scenarios. Through extensive experiments comparing with 10 cutting-edge models, our method demonstrates superior qualitative and quantitative performance, positioning it as an effective solution for fish detection and species identification in underwater trawl capture scenarios.

Index Terms

Instance segmentation; convolutional neural network (CNN); underwater dataset; contrastive learning;

I. INTRODUCTION

Sustainable fishing necessitates addressing challenges like discarding and bycatch, where unwanted animals are caught and discarded, often resulting in their death [1]. To achieve greater precision in the fish capture process, image analysis techniques, particularly deep learning, have been employed in various aquaculture tasks, including fish detection, marine animal identification, and underwater image enhancement [2–5]. Among these tasks, the instance segmentation of marine animal plays a vital role since it can provide important information for identifying marine animals from the complex underwater environment. Such information proves invaluable to the fishing industry, enabling the identification of fish prior to capture, and facilitating the use of selective devices to release unwanted animals.

Leveraging artificial intelligence techniques, deep learning has been widely applied in marine animal research, encompassing fish identification [5], marine animal identification [6], marine biology and archaeology [4] and underwater image enhancement [3]. In trawl nets, camera systems have been deployed to enhance marine animal species identification and reduce non-target species catches [7]. Undersea aquatic animal instance segmentation (UAAIS) plays a pivotal role in providing crucial species identification and abundance information, offering the potential for more cost-effective fishery resource monitoring.

However, accurate UAAIS poses a significant challenge due to the complexity of the marine fishing environment. Underwater images are prone to quality issues caused by the intricate nature of underwater settings. Insufficient luminosity results in low image brightness, while underwater turbidity leads to blurriness and color distortion. Several methods have been proposed to address these challenges, including image enhancement techniques that aim to resolve the low-quality issue of underwater images [3]. While these methods primarily focus on enhancing the overall visual quality and are not directly tailored to the UAAIS task. As suggested in [8], instance segmentation approaches can be categorised into one-stage methods and two-stage methods. While two-stage methods entail slightly higher computational costs, they generally offer better accuracy compared to one-stage methods. Given the importance of accuracy in deep-sea vision-based fishing, we adopt a two-stage architecture in our proposed method, recognizing its potential to deliver superior results.

To tackle the unique challenges associated with the underwater environment and enhance the accuracy of UAAIS for effective deep-sea fishing applications, we introduce a novel cascade-boosting seesaw contrastive learning instance segmentation network in this paper, which is designed for fish detection and species identification in undersea fish trawl scenarios. Specifically, our proposed method improves instance segmentation performance through the integration of cascade module, instance boosting module, contrastive learning module, and seesaw loss module. The instance boosting module enhances training data by introducing augmented instances through the insertion of objects in the surrounding area of their initial positions, along with augmented variations of scale and rotation. The cascade module consists of multiple detection and segmentation branches, utilising a cascaded approach for bounding box regression and detection to improve the quality of hypotheses and achieve high-quality object detection. The contrastive learning module facilitates contrast between multiple image views by comparing cluster assignments rather than features, enabling the calculation of contrastive loss. To enhance the training process, the input image undergoes a boosting process, generating additional instances. These boosted samples are subsequently fed into both the Cascade module and the contrastive learning module, from which their respective training losses are obtained. Finally, the acquired training losses from the Cascade module and contrastive learning modules are fed into the seesaw loss module, which calculates the overall loss. Then, the overall loss is utilised to optimise the network parameters, fine-tuning the model for improved performance. To validate the performance of our proposed method and compare it with state-of-the-art techniques, we have curated a newly collected undersea fish trawl image dataset.

In summary, this paper presents several key contributions:

- A novel cascaded Boost Seesaw Contrastive Network is proposed for instance segmentation, enabling fish detection, shape segmentation, and species recognition. The cascaded mechanism combines multiple detection branches in parallel with a segmentation branch.
- An advanced instance-level augmentation technique is introduced across all branches to achieve a synergistic effect, resulting in improved object detection and semantic segmentation performance.
- To address the challenge of long-tailed data, a seesaw loss function is introduced. It mitigates overwhelming gradients from negative samples in tail classes and compensates for misclassified samples, reducing false positives in instance segmentation.
- A unique fish-catching dataset is collected from the North Sea, consisting of extensive underwater fish and trawl images. This dataset serves as a valuable resource for training and evaluation purposes.

Our proposed method is also compared against state-of-the-art (SOTA) methods based on the collected dataset, to demonstrate its superiority.

A. Instance Segmentation

Instance segmentation methods can be broadly classified into two categories: two-stage instance segmentation and one-stage instance segmentation. Two-stage models typically follow a detection-before-segmentation strategy. These models employ a two-stage detector to obtain target bounding boxes, followed by pixel classification and mask acquisition within the bounding boxes. Mask RCNN [9] is a classic two-stage instance segmentation model, where a mask segmentation branch is added on Faster RCNN. Inspired by this, Mask Scoring RCNN [10] is proposed, which is used to re-evaluate the quality of predicted masks by adding a MaskIoU Head module in instance segmentation branch. Taking advantage of the cascade strategy, Cascade Mask RCNN [11] is proposed introducing multiple detection modules with segmentation heads in each module. Instaboost [12] leverages instance boosting to enhance object detection and pixel-level segmentation performance. PointRend [13] improves instance segmentation by iteratively predicting masks at progressively finer resolutions using upsampling and convolutional layers.

In contrast, one-stage instance segmentation models are anchor-free and eliminate the need to set anchor-boxes on feature maps. SOLO [14] simplifies the model structure by categorising instance segmentation into category and mask prediction tasks. SOLO divides the input image into $S \times S$ grids, with the category prediction branch responsible for grid category prediction and the mask prediction branch responsible for predicting masks in the corresponding feature map. On the basis of the SOLO, SOLOv2 [15]

introduces dynamic convolution, enabling dynamic instance segmentation based on location information and significantly improving performance. Inspired by object detection method YOLO, YOLACT [16] is proposed for instance segmentation which break instance segmentation into two concurrent subtasks: (1) producing a collection of prototype masks and (2) estimating mask coefficients for each instance. Unlike YOLACT, QueryInst [17] utilises a Query-Adaptive Module (QAM) to adaptively incorporate object context information from different regions of the input image, enhancing handling of complex instances with occlusions and irregular shapes.

Considering the importance of accuracy in deep sea vision-based fishing, where two-stage models achieve high segmentation accuracy, our proposed method for deep-sea fishing also adopts a two-stage instance segmentation model.

B. Marine Animal Detection

In recent years, there has been a growing research focus on marine animal detection, specifically the localization and identification of marine animals in images. While some studies have focused on scallop and coral reef detection, the majority have centered around fish detection [18][19]. Early efforts in this field employed traditional computer vision methods, such as morphological or histogram techniques [20][21]. However, a comparative study conducted by [22] revealed that deep learning-based approaches outperformed these traditional methods in coral reef fish detection. Another work by [23] utilised ResNet with cross-layer pooling to enhance the discriminative ability of fish category classification, while [19] presented a deep learning-based model for scallop detection based on a variant of YOLOv2.

In the context of sustainable fishing, three primary challenges arise: 1) the presence of rapidly moving targets, 2) the occurrence of multiple objects from different categories within a single image, and 3) the importance of accurately determining the quantities of each category. To address these challenges, [24] employed RetinaNet to analyze multiple frames of video and identify fish specifically in the final frame.

[25] achieved favorable results in fish detection, primarily in well-lit wide-view settings. [26] utilised the Single Shot Multibox Detector (SSD) with a MobileNet backbone and achieved some success with lower-quality images.

However, all of these previous works relied on bounding boxes for localization, in contrast to our segmentation approach. The motivation and advantage of our segmentation approach lie in providing improved size estimation for captured fish. While [21] successfully applied a segmentation technique in a similar study, they employed traditional techniques and did not attempt species identification. On the other hand, [27] performed segmentation and identification tasks but not in a live environment. Their focus was on post-capture fish discards rather than developing an integrated system to prevent discards in real-time.

III. PROPOSED BENCHMARK DATASET

During our comprehensive literature review, we discovered a notable void in the field of underwater object detection—an absence of a large-scale, real-world deep-sea image dataset. In order to fill this gap and contribute to the advancement of underwater object detection methodologies, we present the UAAIS2K dataset in this section. The dataset is accompanied by detailed descriptions covering key aspects, including image collection, data annotation, data pre-processing, data augmentation, and dataset features.

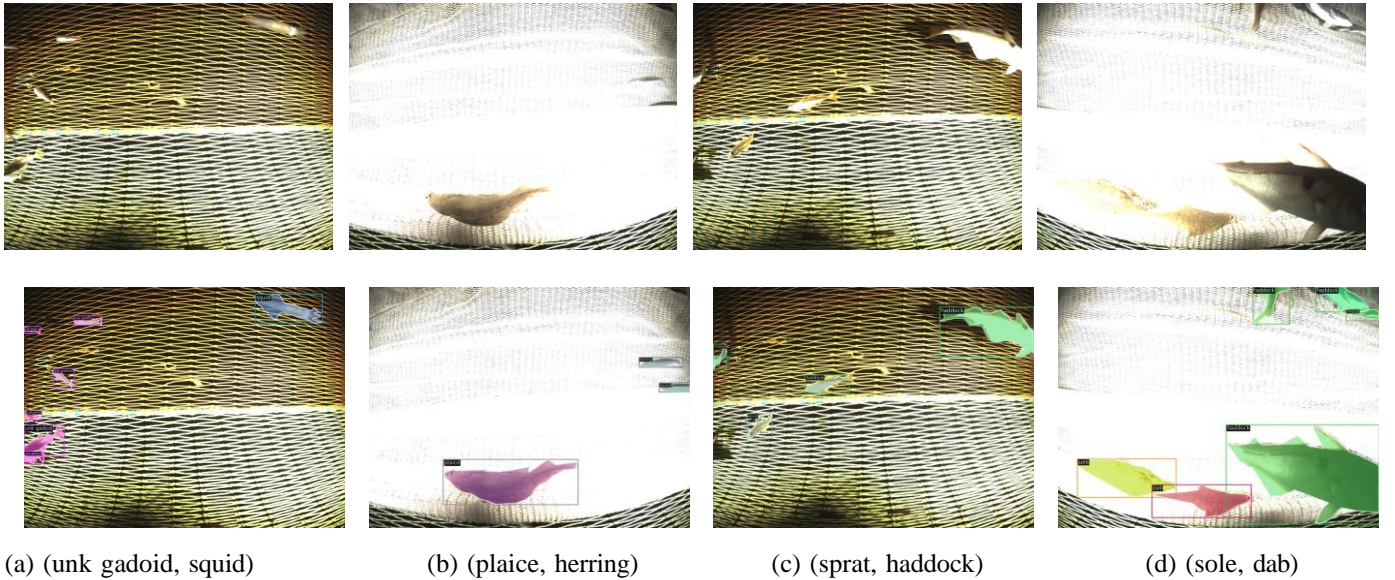


Fig. 1: Examples of raw images and their annotations in UAAIS2K dataset. Top row: raw images collected in diverse undersea trawl scenes; Bottom row: their corresponding annotations of object, pixel, and category levels, where the captions under images identify underwater animals showing in the images.

A. Image Collection

The aquatic images in our dataset were collected from different deployments conducted in the North Sea, specifically from the Sparkling Star and Shetland deployments. The Sparkling Star deployment took place in 2019, while the Shetland deployment consists of more recent images from 2022. Our UAAIS2K dataset encompasses a diverse range of aquatic animals, including 17 categories such as clupeid, haddock, prawn, squid, unknown (unk) gadoid, whiting, cod, dab, herring, sole, unknown fish (unk fish), unknown organism (unk organism), dogfish, plaice, sprat, unknown flatfish (unk flat fish), and unknown round fish (unk round fish). In total, the dataset comprises 2,326 underwater trawl images, with a total of 3,356 instances of aquatic animals annotated within these images.

TABLE I
THE STATISTICS OF UAAIS2K DATASET (UNK: UNKNOWN)

Category	# of Instances	Category	# of Instances
Clupeid	168	Sole	32
Haddock	1029	Unk fish	183
Prawn	331	Unk organism	24
Squid	49	Dogfish	27
Unk gadoid	446	Plaice	19
Whiting	191	Sprat	20
Cod	19	Unk flat fish	43
Dab	700	Unk round fish	46
Herring	29	Total	3356

B. Data Annotation

Human experts meticulously annotated the images collected from three deployments in our dataset. The annotations encompass four types of labels: object-level, pixel-level, category-level, and attributes. To annotate raw image data, LabelMe annotation software tool is used to produce object level, pixel level, and category level annotations. For the attribute annotation, we follow the setting of MS COCO dataset to identify medium size objects and large size objects.

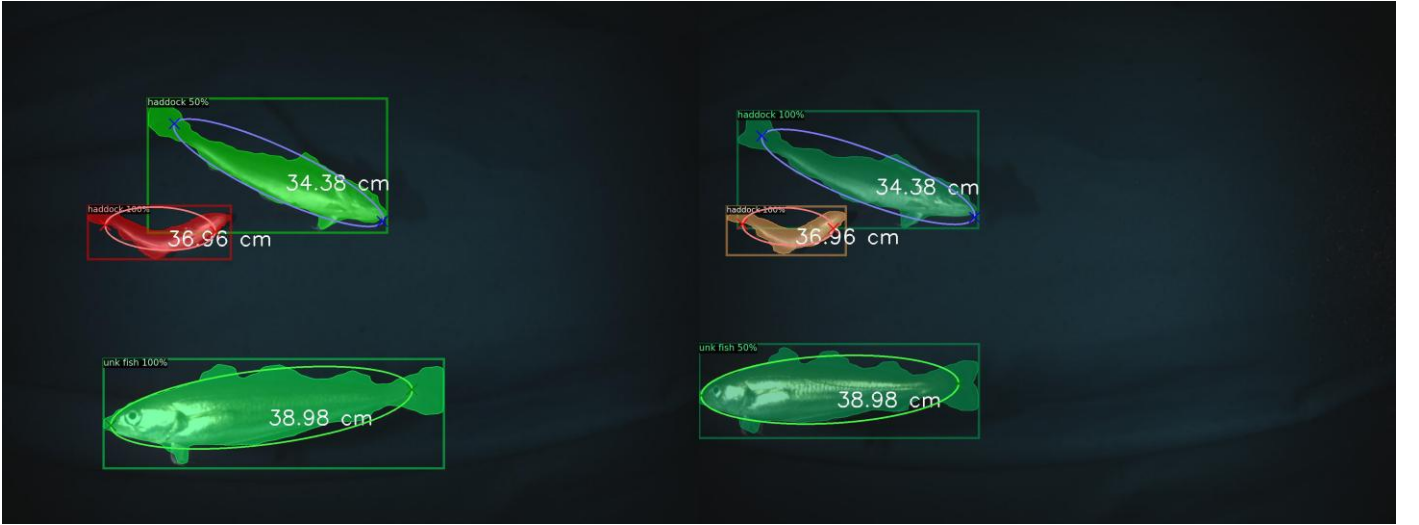


Fig. 2: There are three fish instances with size estimations in paired stereo images (left camera image and right camera image). Ellipse estimators used for stereographic projection of head and tail points.

1) *Object level annotation:* Drawing inspiration from the MS COCO dataset, we employed object-level annotations in our dataset. Each object is localized using rectangular bounding boxes, aligning our label format with MS COCO for better consistency. Object-level annotations are crucial for object detection tasks, and examples of such annotations can be found in Figure 1.

2) *Pixel level annotation:* Pixel-level annotation involves assigning each pixel within an image to a specific category, enabling evaluation for semantic segmentation tasks. Similar to the object-level annotations, our label format adheres to the MS COCO dataset standards. Figure 1 demonstrates the pixel-wise annotations, with distinct masks assigned to different underwater animals.

3) *Category level annotation:* Our dataset includes 17 categories of marine animals, such as clupeid, cod, dab, dogfish, haddock, herring, plaice, prawn, sole, sprat, squid, unknown fish (unk fish), unknown flatfish (unk flatfish), unknown gadoid (unk gadoid), unknown organism (unk organism), unknown round fish (unk round fish), and whiting. Figure 1 exhibits the presence of various fish categories in an underwater image.

4) *Attributes*: To provide informative attribute labels, each image in our dataset is annotated with three attributes: big object, medium object, and multiple objects. We followed the attribute annotation setting of the MS COCO dataset, which enables the identification of medium-sized and large-sized objects. By incorporating these detailed annotations, our dataset offers valuable information for object detection, semantic segmentation, and attribute analysis tasks, enabling researchers to delve into various aspects of underwater marine animal analysis.

C. Data Pre-processing

The original data were captured using a stereo camera system. To filter out images without fish in an unsupervised manner, we developed a CNN-based one-class support vector machine (SVM) filter algorithm. In this approach, a pre-trained CNN is used to extract features from the raw images, which are then passed to the one-class SVM to cluster the images containing fish. This automated filtering process does not require any labels.

D. Data Augmentation

Data augmentation is employed to increase the amount of data available for training and testing, aiming to enhance generalisation and prevent overfitting. For training data augmentation, common techniques such

TABLE II CHARACTERIZATION OF RAW DATA

Raw Datasets	Folder name	Number of Data
Sparkling Star Deployment	Left	1000
19 July 2019	Right	1000
Deployment 26	2899-Right Camera	17391
1 April 2022	2900-Left Camera	17390
Deployment 27	2899-Right Camera	17489
1 April 2022	2900-Left Camera	17488

as zooming, flipping, shifting, etc., are applied to create replicas of images from the training dataset. In testing data augmentation, multiple enhanced copies of each test image are generated, and the model provides predictions for each copy, forming an ensemble of predictions. In this study, both the training and testing data are augmented to improve performance, following the augmentation operations and scale jitter as described in [28].

E. Dataset Features and Statistics

The undersea fishing image collection in our dataset offers several notable features. Firstly, it covers a diverse range of undersea fishing scenes, providing a comprehensive representation of the underwater environment. Secondly, the dataset includes pixel-level annotations for 17 commercial fish species, enabling detailed analysis and species-specific tasks. Thirdly, paired images from the stereo-vision system are included, allowing for fish size estimation. Lastly, the dataset is split into training and testing sets, considering the balance of fish species and numbers. This division facilitates convenient training of models and evaluation of object detection and semantic segmentation performance.

F. Existing data and New progress on Data Annotation

This subsection describes the raw data structure with blue background, data preprocessing, data extraction, image labelling and final statistical results. In particular, the Smartrawl 5.0 project consists of three different raw datasets: the dataset captured on 19 July 2019 from the Sparkling Star deployment and the dataset captured on 1 April 2022 from the North Sea. These datasets contain different identifiers such as the number of each species captured, the date and specific time of collection, image number and species name. The table below shows the structure of the raw data and the total amount of data from the left and right cameras. The

table below shows the raw data collected on the different dates, as well as the total amount of image data from the left and right cameras.

For a clearer and more specific details of the structure of the data, the following is a descriptive table of the dataset from Sparkling Star deployment collected on 19 July 2019. The dataset has been annotated by oceanographers and includes spreadsheets containing information on the date, time, number, number of organisms, and type of organism for each image collected. These image datasets containing different species are accessible and available in real time.

As the original dataset collected for deployment in 2022 is relatively large and not easily processed, we will need to perform pre-processing steps such as extracting and filtering the images, as some images do not contain organisms. Feature extraction will be used as the primary step in downscaling, using machine learning and deep learning algorithms to filter out images that may contain fish species while retaining information from the original dataset. During the initial processing, the left and right camera files were divided into a folder for every 1000 images, and filtering was performed using MATLAB (Matrix Laboratory). The total number of filtered images in the deployed dataset is shown in Table III.

IV. PROPOSED UNDERWATER INSTANCE SEGMENTATION MODEL

A. Architecture of Cascaded Boost Seesaw Contrastive Network (CBSC-Net)

Our network architecture comprises several modules: the cascade module, instance boost module, contrastive learning module, and seesaw loss module. As shown in Figure 4, the input image is first

	A	B	C	D	E	F	G	H	I	J	K
1	Date	Time	Image	imecheck.	imecheck.	iber of organi	Description	running 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	

Fig. 3: Details of the dataset from Sparkling Star's deployment collected on 19 July 2019.

TABLE III
NUMBER OF FILTERED IMAGES

Raw Datasets	Folder name	Number of Filtered images
Deployment 26	2899-Right Camera	1808
1 April 2022	2900-Left Camera	1808
Deployment 27	2899-Right Camera	1507
1 April 2022	2900-Left Camera	1507

boosted to generate additional instances, thereby enhancing the training process. These boosted samples are then passed through the cascade module and contrastive learning module to calculate their respective training losses. The obtained losses from the cascade and contrastive learning modules are subsequently fed into the seesaw loss module, which computes the overall loss to optimize the network parameters.

The instance boosting module augments the training data by inserting objects near their original positions and applying additional jittering in terms of scale and rotation. The cascade module consists of multiple detection and segmentation branches, which work together to improve detection performance. This is achieved through cascaded bounding box regression and detection. The contrastive learning module focuses on contrasting multiple image views based on their cluster assignments rather than their features, enabling the extraction of contrastive loss.

B. Cascaded Boost

In terms of the cascade architecture, the segmentation branch is integrated in parallel with multiple detection branches. To address the challenges of determining the optimal placement and number of segmentation branches, we introduce a single mask prediction head at each stage of the Cascade R- CNN. This approach maximizes the diversity of samples used to learn the mask prediction task. During inference, the model predicts segmentation masks on the patches generated by the final object detection

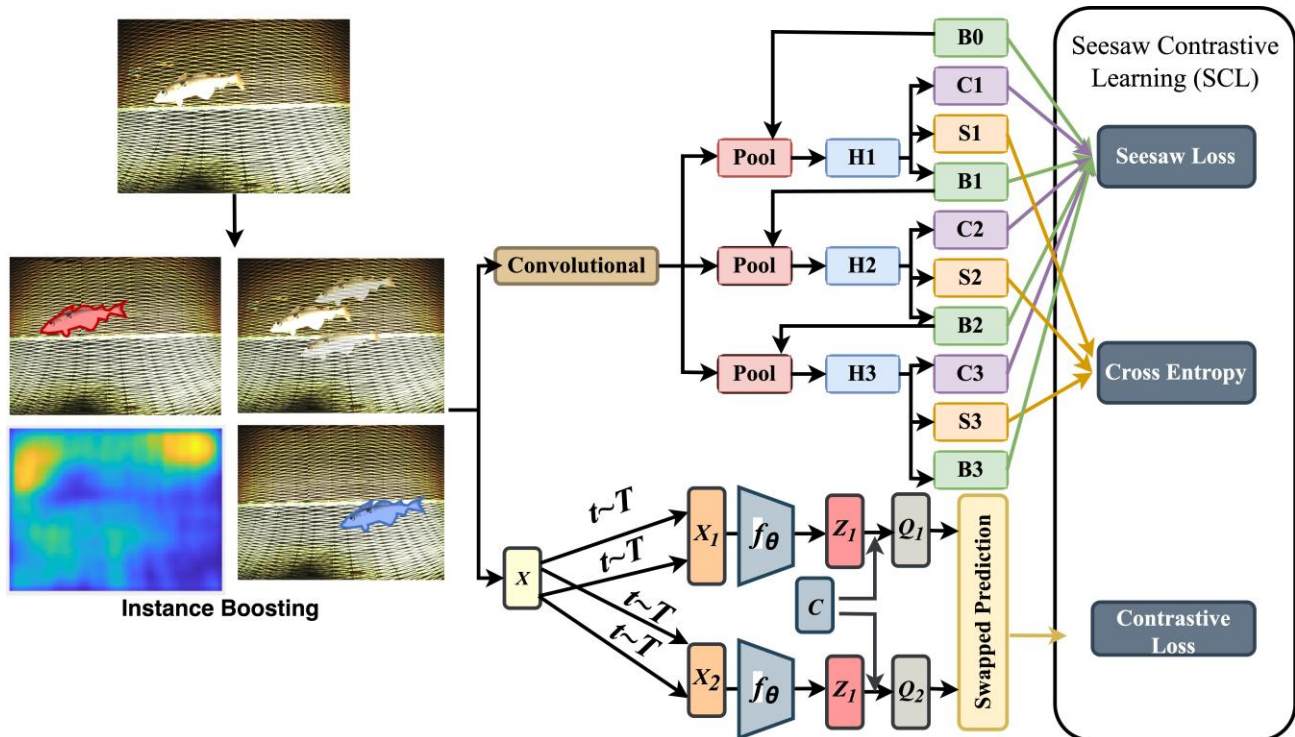


Fig. 4: The overall architecture of the proposed cascaded boost seesaw contrastive network, where three instance segmentation networks (i.e., $H1$, $H2$, and $H3$) are cascaded together. Their corresponding bounding box branches (i.e., $B0$, $B1$, $B2$, $B3$) and Classification branches (i.e., $C1$, $C2$, $C3$) are optimised by Seesaw loss. Segmentation branches (i.e., $S1$, $S2$, and $S3$) are optimised by Cross Entropy. X is a instance boosted fish image and constrastive loss is computed by the feature vectors from swapping assignments between views of the same image.stage, and the final mask prediction is obtained through an ensemble of specifically designed segmentation branches.

For data augmentation, instance-level boost is applied within the cascaded network architecture. The affine transformation matrix \mathbf{A} is utilised to define the placement of a cropped object patch within the original image. This matrix determines the position and orientation of the object when it is placed back onto the original image.

$$\mathbf{A} = \begin{bmatrix} s \cos r & s \sin r & \Delta x \\ -s \sin r & s \cos r & \Delta y \\ 0 & 0 & 1 \end{bmatrix}, \quad \Delta x, \Delta y, s, r \in \mathbb{R} \quad (1)$$

where Δx denotes the horizontal offset and Δy denotes the vertical offset. s and r are the scale variance and rotation in degrees, respectively. Thus, the placement can be determined uniquely.

According to stochastic grammar of images, a probabilistic model is capable of capturing the inherent frequency of object occurrences in a natural setting and then sampled to synthesise an extensive configurations to encompass unseen instances in the test set. With considering this objective, we establish a probability density function $f(\cdot)$ to assess the plausibility of placing object O onto the provided image I using a designated transformation tuple. Considering the original coordinates of the object as (x_0, y_0) and given the image and object conditions (I, O) , a probability map P is formulated as follows.

$$P(x, y, s, r|I, O) = f(\Delta x, \Delta y, s, r|I, O) \quad (2)$$

where the new coordinates are determined as $x = x_0 + \Delta x$ and $y = y_0 + \Delta y$. Notably, the identity transform $(x_0, y_0, 1, 0)$, i.e., the original paste configuration, should exhibit the highest probability.

Due to the inherent continuity and redundancy of pixel-level information in images, the probability map $P(x, y, s, r|I, O)$ is anticipated to have high values within a small neighbouring region around $(x_0, y_0, 1, 0)$. Following [12], we employ object jittering, a technique that involves randomly sampling transformation tuples from the neighbouring space of the identity transform $(x_0, y_0, 1, 0)$ and applying an affine transformation A to paste the cropped object. If the background exhibits a similar pattern over a wide range, the position of (x, y) is possible to be extended beyond the neighbouring area of (x_0, y_0) . Thus, consistency heatmap is used to identify the redundancy in continuous but non-aligned features of background. With the guidance of such heatmap, we can maximise the utility of our object jittering.

C. Seesaw Contrastive Learning

To further enhance performance, we have integrated seesaw loss into our contrastive learning approach. In contrastive learning, the goal is to learn visual features by swapping assignments between multiple views of the same image, as exemplified by the Swapping Assignments between multiple Views (SwAV) method. This approach enables the learning of visual features in an unsupervised manner, without relying on explicit labels. Unlike traditional clustering-based methods, SwAV focuses on enforcing consistent mapping between views of the identical image, rather than solely comparing features.

Inspired by contrastive instance learning, which aims to ensure consistent mapping between views of the identical image without explicitly targeting specific codes, we generate a code by augmenting the image and subsequently predict this code from other augmented versions of the identical image. From diverse augmentations of the identical image, we derive two distinct image features denoted as z_t and z_s . These features are used to produce their respective codes, q_t and q_s by matching them to a collection of K prototypes. To address the "Swapped" prediction problem, we employ the following contrastive loss function:

$$L_{CL}(z_1, z_2) = l(z_1, q_2) + l(z_2, q_1) \quad (3)$$

where the function $l(z_1, q_2)$ serves to measure the alignment between the features z_1 and a given code q_2 which is similar to $l(z_2, q_1)$. Contrastive learning is employed to compare the features z_1 and z_2 through q_1 and q_2 , which are their corresponding intermediate codes. When z_1 captures the same information with z_2 , the corresponding code q_1 of z_1 should be feasible to be predicted by z_2 .

1) *Online clustering*: For each image x_n , it undergoes a transformation t to produce an augmented n view x^t . This augmented view, denoted as x_n^t is then mapped to a vector representation using the function f_θ . Subsequently, the obtained feature is projected onto the unit sphere as follows

$$z_n^t = \frac{f_\theta(x_n^t)}{\|f_\theta(x_n^t)\|_2} \quad (4)$$

where $\|\cdot\|_2$ is L_2 norm, transformation t is sampled from a set of image augmentations T , and f_θ is a non-linear mapping function to map x^t to a latent feature vector. By using the feature z^n , we then derive a code q^n by mapping it to a set of K trainable prototype vectors c_1, \dots, c_K . Such a mapping process allows us to capture the essence of the feature within the code representation. To facilitate this, we construct a matrix C where each column corresponds to one of the prototypes c_1, \dots, c_K . Thus, matrix C is given as follows.

$$C = [c_1, \dots, c_K] \quad (5)$$

For the sake of calculating these codes and dynamically update the prototypes, the contrastive loss can be expressed as two terms for the swapped prediction. The first term involves predicting the code q_1 from the feature z_2 . The second term predicts the code q_2 from z_1 . Each term represents the cross entropy loss computed between the code and the probability, which is obtained by applying the softmax function to the dot products of z_i and all prototypes in the matrix C , i.e.,

$$l(z_1, q_2) = - \sum_{k=1}^K q_2^k \log \frac{\exp(\frac{1}{\tau} z_1^T c_k)}{\sum_{j=1}^K \exp(\frac{1}{\tau} z_1^T c_j)} \quad (6)$$

where τ is a temperature parameter to adjust the spread of the probability distribution obtained from dot product computations [28]. By applying this loss to both images and pairs of data augmentations, the below loss function is derived for the swapped prediction problem.

$$\begin{aligned} l(z_1, q_2) = & -\frac{1}{N} \sum_{n=1}^N \left[\frac{1}{\tau} (z_1^n)^T C q_2^n + \frac{1}{\tau} (z_2^n)^T C q_1^n \right. \\ & \left. - \log \sum_{k=1}^K \exp\left(\frac{(z_1^n)^T c_k}{\tau}\right) - \log \sum_{k=1}^K \exp\left(\frac{(z_2^n)^T c_k}{\tau}\right) \right] \end{aligned} \quad (7)$$

The joint minimization of the derived loss function is performed with respect to both the prototypes C and the parameters θ of the image encoder f_θ responsible for generating the features z^t . During online code computation, we calculate the codes exclusively using the image features within a batch. This approach allows SwAV to cluster multiple instances to the prototypes C . To ensure distinct codes for different images in a batch and avoid a trivial solution where all images have the same code, we employ an equipartition constraint. By using the prototypes C , we compute codes that evenly partition all examples in a batch. The objective is to map B feature vectors $Z = [z_1, \dots, z_B]$ to the prototypes $C = [c_1, \dots, c_K]$, denoted as codes $Q = [q_1, \dots, q_B]$. The optimization goal is to maximize the similarity between the features and the prototypes, which is denoted as follows.

$$Q^* = \max_{Q \in \mathbb{R}_+^{K \times B}} Tr(Q^T C^T Z) - \epsilon \sum_{ij} Q_{ij} \log Q_{ij}$$

(8)

where Q^* is the solution for maximising the similarity of features and prototypes. The parameter ϵ controls the smoothness of the mapping. A high ϵ leads to strong entropy regularization so as to result in a trivial solution. All samples collapse into a single representation and are uniformly assigned to all prototypes. Therefore, in practice, we maintain a low value for ϵ to avoid this issue.

2) *Multi-crop Augmenting views*: To overcome the weakness of random crops whose the memory and compute requirements are increased exponentially with the increase of the number of crops or “views”, a multi-crop strategy is introduced into our proposed method. There are two standard resolutions used in cropping and sampling V additional low resolution crops in order to cover only small parts of the image. With applying low resolution images, it can remain a small increase in the compute cost. More specifically, the contrastive loss is generalised through the following formulation

$$L(z_{t_1}, z_{t_2}, \dots, z_{t_{V+2}}) = \sum_{i=1}^2 \sum_{v=1}^{V+2} 1_{v \neq i} l(z_{t_v}, q_{t_i}) \quad (9)$$

where $i \in \{1, 2\}$ represent two standard resolution crops. $1_{v \neq i}$ stands that it equals to one when $v \neq i$, otherwise it equals to zero. Only codes of the full resolution crops are computed because high computation load for computing codes of all crops. In practice, it observes that using only partial information from small crops, which cover a limited area of the images, can lead to a degradation in the quality of assignments. To solve this issue, multi-crop augmentation strategy [28] is adopted to improve the performance.

3) *Seesaw loss*: To alleviate the problem on long-tailed data, seesaw loss is introduced to address the issue of overwhelming gradients of negative samples on tail classes and to compensate the gradients of misclassified samples to avoid false positives for instance segmentation. In seesaw loss, the impact of negative samples from head classes on a tail class is reduced by decreasing the gradients imposed on the tail class. The Seesaw loss is given as follows.

$$L_{seesaw}(Z) = - \sum_{i=1}^C y_i \log \left(\frac{e^{z_j}}{\sum_{j \neq i}^C B_{ij} e^{z_j} + e^{z_i}} \right)$$

(10)

where $Z = [z_1, z_2, \dots, z_C]$ and $y_i \in \{0, 1\}$, $i \in \{1, \dots, C\}$. B_{ij} is a tunable balancing factor between different classes. By a careful design of B_{ij} , Seesaw loss adjusts the punishments on class j from positive samples of class i . Seesaw loss determines B_{ij} by a mitigation factor and a compensation factor as

$$B_{ij} = M_{ij} \cdot C_{ij} \quad (11)$$

where the mitigation factor M_{ij} reduces the penalty on the tail class j in proportion to the ratio of instance numbers between the tail class j and the head class i . Conversely, the compensation factor C_{ij} amplifies the penalty on class j whenever an instance from the head class i is misclassified as class j .

4) *Overall Optimisation Function:* The overall loss is formed by three parts. The first part is the seesaw loss of bounding box and class branches. The second part is the contrastive loss of swapping assignments between views. The third part is the cross entropy loss of the segmentation branch, which is provided as below.

$$L_{CE}(Z) = - \sum_{i=1}^C y_i \log\left(\frac{e^{z_i}}{\sum_{j=1}^C e^{z_j}}\right) \quad (12)$$

where y_i is the predicted logits of pixel-level classification. For mask loss, there is a prediction mask produced for each class and therefore the mask loss is specified as the average binary cross entropy loss for segmentation.

Therefore, the overall loss combines all seesaw loss, contrastive loss, and cross entropy loss together, which is defined as follows.

$$L_{overall} = L_{seesaw}(Z) + L_{CE}(Z) + L_{CL}(z_1, z_2) \quad (13)$$

D. Implementation Details

The implementation of our method is based on PyTorch, which is a deep learning framework. For the backbone network, the pre-trained ResNet50 [29] on ImageNet [30] is chosen due to its competent performance [30]. Stochastic Gradient Descent (SGD) is used to optimise instance segmentation networks. Following the work in [12], the networks are trained for 48 epochs and batch size is set 2. The initial Learning Rate (LR) is set to 0.02 to avoid exploding gradients and the momentum and weight decay is set to 0.9 and 0.0001, respectively. For the hardware configurations, we run all experiments on a PC with CPU: Intel 2.60GHz i7-10750H, GPU: GeForce RTX 3090, and RAM: 64 GB.

V. EXPERIMENTS AND EVALUATION

A. Experiment Settings

We divide UAAIS2K dataset into training and testing sets. For the image level, in the training set of our UAAIS2K dataset, it contains 1,853 images. In the testing set of our UAAIS2K dataset, it consists of 463 images. For the instance level, there are 2,676 instances in the training set and 680 instances in the testing set.

To test the performance of our CBCS-Net, we compared our CBCS-Net with existing state-of-the-art (SOTA) methods, including ten of Mask R-CNN (MRCNN) [9], Cascade MRCNN [11], Mask Scoring RCNN [10], Instboost [12], Yolact [16], point rend [13], and Queryinst [17], which can both provide predicted bounding boxes and segmentation masks; three of semantic segmentation models, Solo [14], Solov2 [15], and SparInst [31], which solely provide predicted segmentation masks.

B. Evaluation Metrics

To evaluate our method comprehensively, we assess the performance of both instance mask prediction and BBOX detection in underwater environments. We report the standard OCO metrics including Average Precision (AP), AP_{50} , AP_{75} , AP_S , AP_M , and AP_L . AP is measured by averaging over different thresholds of IoU from 0.5 to 0.95 with a step size of 0.05. That is, AP is calculated by averaging the AP s over all object categories, which is six categories in our case and all 10 IoU thresholds from 0.5 to 0.95 with a step size of 0.05. Such averaging over IoUs and categories provides a thorough evaluation rewards models with better performance. For AP_{50} and AP_{75} , the threshold is set to 0.5 and 0.75 for BBOX/segmentation predictions. AP_S , AP_M and AP_L are about results regarding small, medium and large instances. Small instances are defined as being with an area of 32×32 pixels. Medium instances are defined as being between 32×32 and 96×96 pixels in area. Large instances are defined as being an area larger than 96×96 pixels. These definitions make a distinction between the object sizes. This is provided as some datasets, such as COCO have an imbalance in object sizes. The definition of AP is given by

$$AP = \frac{1}{10} \sum_{i=0.5}^{0.95} AP_i, \quad AP_i = \frac{1}{|n|} \sum_{c \in n} \frac{|t_p^c|}{|f_p^c| + |t_p^c|} \quad (14)$$

where n is the number of classes within the dataset. t_p^c and f_p^c are true positives and false positives of class c .

TABLE IV
EVALUATION ON SEGMENTATION AND OBJECT DETECTION

Method	Segm						BBox					
	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L	AP	AP ₅₀	AP ₇₅	AP _S	AP _M	AP _L
MRCNN [9]	0.148	0.260	0.141	0.050	0.036	0.174	0.132	0.258	0.119	0.150	0.091	0.138
Cascade MRCNN [11]	0.208	0.346	0.212	0.000	0.094	0.227	0.229	0.350	0.249	0.000	0.174	0.237
Mask Scoring RCNN [10]	0.138	0.248	0.136	0.117	0.070	0.148	0.126	0.241	0.109	0.300	0.097	0.131
Queryinst [17]	0.183	0.272	0.198	0.250	0.134	0.196	0.185	0.276	0.205	0.200	0.200	0.190
Instaboost [12]	0.223	0.369	0.218	0.125	0.134	0.240	0.226	0.372	0.239	0.300	0.232	0.228
Yolact [16]	0.191	0.296	0.202	0.200	0.127	0.204	0.190	0.302	0.204	0.100	0.236	0.195
Point Rend [13]	0.197	0.324	0.201	0.050	0.104	0.207	0.161	0.313	0.141	0.200	0.185	0.158
SOLO [14]	0.141	0.256	0.127	0.000	0.128	0.154	-	-	-	-	-	-
SOLOv2 [15]	0.224	0.342	0.232	0.050	0.172	0.237	-	-	-	-	-	-
SparseInst [31]	0.243	0.346	0.247	0.000	0.166	0.251	-	-	-	-	-	-
Ours	0.335	0.451	0.374	0.021	0.204	0.353	0.350	0.453	0.411	0.067	0.325	0.351

C. Quantitative Evaluation

This section provides the instance segmentation results of marine animal detection and segmentation methods. All experiments are evaluated on our recently collected undersea aquatic animals (UAAIS2K) dataset. To quantitatively evaluate the results of object detection and semantic segmentation, average precision (AP) is used to assess the performance for object detection and semantic segmentation.

We compare the performance between our proposed method with other state-of-the-art methods on the tasks of object detection and semantic segmentation, respectively. The quantitative comparison of object detection is summarised in Table IV with regard to AP , AP_{50} , AP_{75} , AP_S , AP_M , and AP_L . The following observations can be drawn.

First, for the overall performance, our method outperforms other state-of-the-art methods for both object detection and semantic segmentation by introducing cascade boosting, contrastive learning, and Seesaw loss. Our proposed method can significantly improve instance segmentation performance in terms of AP , AP_{50} , and AP_{75} . More specifically, our method can reach 35.0% of AP for object detection (bbox) and 33.5% of AP for semantic segmentation (segm). For AP_{50} , our method provides the best performance which are 45.1% for bbox and 45.3% for segm. For AP_{75} , our method also delivers the best results for bbox and segm, which are 37.4% and 41.1%, respectively.

TABLE V
PERFORMANCE ON SEGMENTATION AND DETECTION (BBOX/SEGM)

Classes	MRCNN	C MRCNN	MS RCNN	Instaboost	Yolact	Point Rend	Queryinst	Ours
Clupeid	0.135/0.129	0.18/0.154	0.121/0.152	0.249/0.262	0.16/0.158	0.113/0.162	0.177/0.171	0.293/0.28
Haddock	0.404/0.437	0.516/0.485	0.382/0.431	0.491/0.499	0.531/0.549	0.398/0.509	0.479/0.486	0.67/0.641
Prawn	0.336/0.256	0.373/0.277	0.325/0.265	0.408/0.301	0.397/0.281	0.325/0.269	0.327/0.248	0.444/0.326
Squid	0.001/0.002	0.179/0.161	0.02/0.019	0.071/0.084	0.034/0.034	0.036/0.061	0.192/0.202	0.326/0.338
Unk gadoid	0.137/0.159	0.258/0.205	0.154/0.16	0.325/0.299	0.274/0.288	0.206/0.265	0.227/0.239	0.384/0.398
Whiting	0.111/0.158	0.208/0.207	0.107/0.147	0.303/0.323	0.151/0.184	0.128/0.184	0.187/0.188	0.467/0.48
Cod	0.0/0.0	0.483/0.418	0.0/0.0	0.275/0.285	0.094/0.09	0.098/0.122	0.019/0.02	0.783/0.764
Dab	0.601/0.628	0.694/0.677	0.598/0.626	0.683/0.695	0.652/0.71	0.604/0.699	0.678/0.69	0.79/0.755
Herring	0.05/0.063	0.016/0.011	0.0/0.0	0.017/0.034	0.0/0.0	0.001/0.005	0.029/0.026	0.053/0.041
Sole	0.349/0.456	0.533/0.563	0.311/0.388	0.583/0.632	0.412/0.473	0.463/0.579	0.303/0.31	0.786/0.794
Unk fish	0.043/0.036	0.08/0.037	0.044/0.038	0.091/0.073	0.123/0.109	0.071/0.095	0.105/0.098	0.199/0.176
Unk organism	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.003/0.003	0.003/0.003
Dogfish	0.0/0.039	0.126/0.099	0.006/0.019	0.017/0.013	0.0/0.0	0.081/0.107	0.188/0.192	0.288/0.289
Plaice	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.0/0.0	0.007/0.006	0.012/0.016
Sprat	0.021/0.052	0.074/0.085	0.0/0.0	0.152/0.103	0.083/0.052	0.032/0.066	0.028/0.025	0.046/0.036
Unk flat fish	0.019/0.05	0.079/0.066	0.036/0.048	0.055/0.082	0.219/0.197	0.08/0.106	0.114/0.13	0.17/0.145
Unk round fish	0.039/0.051	0.092/0.093	0.04/0.057	0.13/0.107	0.099/0.12	0.095/0.119	0.078/0.084	0.242/0.214

Second, for various classes, there are 17 classes out of total 17 classes achieving the best performance in our method for both object detection (bbox) and semantic segmentation (segm). Our method has a superior performance on the classes of haddock, cod, dab, sole. For instance, our method achieves 79.0% and 75.5% on segmenting and detecting underwater animals while the second best result is provided by Cascade MRCNN [11] which are only 69.4% of semantic segmentation and 67.7% of object detection. For recognising the rest classes, our method can provide satisfactory results.

Third, for performance of AP_S , AP_M and AP_L , our method shows the best performance with regard to AP_M and AP_L . For semantic segmentation, our method can achieve 20.4% for AP_M and 35.3% for AP_L which is much better than MRCNN [9] whose AP_M and AP_L are only 3.6% and 17.4%. Although our method does not provide the best performance of AP_S , small fishes will be normally released with considering the sustainability of fishery industry.

D. Qualitative Evaluation

The qualitative results of instance segmentation are illustrated in Fig. 5, where various underwater scenes are provided in different columns. In order to further illustrate the benefits of our proposed model, we provide compelling visual results comparing various models. As shown in Fig. 5, our method demonstrates superior performance yielding results that closely align with the ground truth, which performs well in diverse scenes. It also notes that the our method can segment the details of boundaries well. In addition, compared to other advanced models, the bounding boxes and segmentation maps of our model are with more complete objects and accurate bounding boxes. Our model has less false detection and segmentation when comparing with other methods who provide multi-boxes for the same object with some labels of wrong categories. Finally, our model is robust to detecting fish and recognising their corresponding species in the middle and large size. For instance, our method can accurately locate fish (i.e., bounding box and mask) and predict their species as shown in the second and third column of Fig. 5.

E. Ablation Study

In this section, we analyse the contributions of various components of our method. Extensive experiments are conducted to figure out their roles in our proposed method. The improvement of AP_{bbox} for object detection and AP_{segm} for semantic segmentation by considering one more component at each stage is presented in Table VI, where Baseline is MRCNN model; Boosting stands for instance-level boost; Cascade stands for the introduction of Cascade mechanism; SCL stands for Seesaw Contrastive Learning. We can see that a poor performance of AP will be obtained with only 14.8% for AP_{segm} and 13.2% for

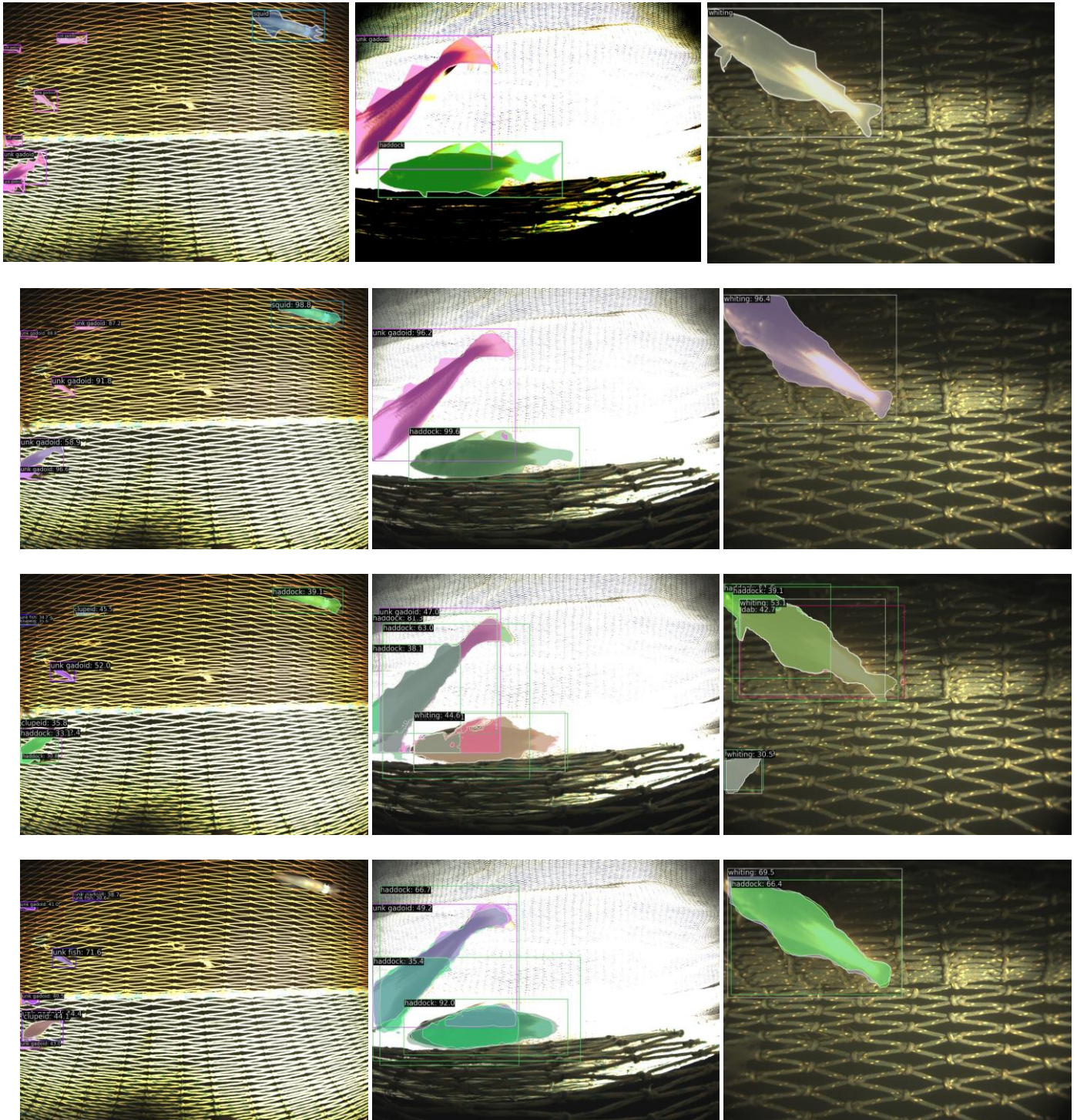


Fig. 5: Visual comparison of various models. Our method is compared with a number of SOTA models and groundtruth (GT). First row is the **ground truth**. Second row is instance segmentation performance of **our method (CBSC-Net)**. Third row is instance segmentation performance of **Point Render**. Forth row is instance segmentation performance of **QueryInst**.

AP_{bbox} if simply using baseline model. If we introduce Boosting, it will bring the performance gain as 7.5% of AP_{segm} and 9.4% of AP_{bbox} with the help of instance-level boost. If we add Cascade mechanism, there is a further improvement obtained for AP_{segm} and AP_{bbox} , which are 3.3% and 4.8%, respectively. Moreover, if SCL is introduced as well, a further gain can be obtained 7.9% of AP_{segm} and 7.6% of AP_{bbox} . When all of them are adopted in our method, we can achieve the best performance for both smantic segmentation and object detection, 33.5% of AP_{segm} , and 35.0% of AP_{bbox} .

TABLE VI
THE ABLATION STUDY ON VARIOUS COMPONENTS OF OUR METHOD

Baseline	Boosting	Cascade	SCL	AP_{segm}	AP_{bbox}
✓				0.148	0.132
✓	✓			0.223	0.226
✓	✓	✓		0.256	0.274
✓	✓	✓	✓	0.335	0.350

VI. FISH SIZING TECHNIQUE

A. Fish Size Estimation

Fish size estimation is a multi-stage process that accounts for potential differences between the number of fish visible from the left and right camera positions. The first two stages involve post-processing steps to refine the raw mask outputs from the network. Mask predictions are merged to eliminate cases where multiple masks are predicted for a single fish instance. When Intersection over Union (IoU) score is 0.3 or above, it will trigger merging. The merged instance is assigned a class label based on the maximum score of the component masks and the combined area.

The second stage filters out isolated peripheral regions that may appear in masks. Each instance must be attributed to a single, connected region of the image. Isolated areas, typically corresponding to fish fins with low image contrast, are safely removed by applying a small area filter. This ensures that the resulting mask corresponds to the main body of the fish.

Once the post-processing is complete, the sizing algorithm converts binary mask instances from both the left and right cameras into contour representations. A best-fit ellipse is generated from these contours, and the extrema on the major axes of the ellipses are identified as the "head" and "tail" points of each fish. This ellipse-matching technique provides a stable method for locating corresponding points in space across the left and right images.

Before sizing can occur, the sets of left and right ellipses need to be paired. The process considers the ordering of ellipse centers from left to right on both sides, taking into account that fish appearing on the extreme left or right may not be visible in the opposite camera. The resulting pairings are checked against an estimated lateral shift between the left and right cameras, and any necessary reordering is performed to obtain a set of ellipse pairings.

In the final stage, the "head" and "tail" points for each instance are projected to 3D space using stereoscopic projection techniques and the recorded calibrations for both the left and right cameras. The Euclidean distance between these projected points represents the estimated fish length.

B. Implementation Details of Fish sizing

The fish sizing algorithm can be summarised in six steps: Binary mask merging, contour extraction, small area filtering, ellipse fitting, instance matching and point projection. Details of each stage follow along with observations on future improvements.

- 1) **Binary mask merging:** The output of the prediction network provides a tensor of binary masks with height and width matching that of the original input image (1536x2048) and a depth corresponding to the number of detected fish. Occasionally, the network over-predicts by detecting the same fish more than once. To compensate for this, we the IoU (Intersection over Union) is calculated for all

detected instances and those which overlap sufficiently (we use a threshold of $IoU = 0.3$). The species of the merged mask is assigned to be that with the maximum confidence score.

- 2) **Contour Extraction:** We use the opencv findContours function applied layer-wise to convert each binary mask layer to a list of coordinate sequences.
- 3) **Small area filtering:** The ellipse-fitting stage requires that we have a single contour for each fish instance, but this is not the case in reality. In a small number of cases, the peripheral edges of a fin are more visible under the light conditions and hence appear disconnected from the main body. The simplest and most computationally effective way of dealing with this is just to use the contour that corresponds to the longest sequence of coordinates from the previous step.
- 4) **Ellipse fitting:** We use the EllipseModel class from scikit-image to convert the sequence of points for each detected instance to fit a corresponding ellipse. This proves to be a good method of approximating the shape of a fish, although some exceptions are noted in the following section. We guard against degenerate masks/contours by forcing an assertion that each contour should have at least ten points - the ellipse fitting algorithm cannot find a unique ellipse if there are less than five points. Where this test fails, twenty new points are uniformly resampled from the path described by the contour and the algorithm attempts to fit a new ellipse. If this fails again, it is assumed that the error occurs because the degenerate mask is co-linear and this detection is discarded as a false positive.
- 5) **Instance matching:** All the previous steps are applied to both the left and right camera images so that we have a list of ellipses for both sides. Each ellipse will correspond to an organism, but the lists are not ordered in any canonical way. There may not even be the same number of fish visible on both sides. To rectify this, we perform a transformation on the centre points of the ellipses of the left camera which mimics a shift to the left caused by moving the camera in the right direction. In effect, we want to create an overlay of the two images and identify each fish with its pair by looking at closest pairs across images.

The magnitude of the shift differs depending on the distance of the fish in the depth direction - distant fish are shifted less between cameras when compared to those close to the lens. For this reason, we decide to use an average estimated shift. This, however, needs knowledge of matched pairs - what we are attempting to do in this step. In order to achieve this an assumption is made. We assume that, when different numbers of fish are detected on both sides, either the leftmost fish in the right camera or the rightmost fish in the left camera drops off the field of view between shots. We discard the extra fish and reorder the fish ellipses by x-coordinate of the centre before pairing and estimating the average lateral shift.

This process was tested empirically using generated data. Between 2 and 7 fish were generated using random positioning and sizing and shifted to represent the opposite camera. Noise was added to the position in both axis on the opposite camera to represent inaccuracies in the system, then the fish were reordered. The algorithm could reliably recover the correct order and pair the fish correctly for a reasonable level of noise.

- 6) **Point projection:** We arbitrarily assign the leftmost extrema on the major axis of each ellipse as the “head” and the rightmost extreme as the “tail”. Paired heads and tails are projected to points in 3D space using standard stereoscopic projection techniques and the recorded calibrations for both the left and right cameras. The Euclidean distance between these head and tail projections is the estimated fish length.

C. Problem cases

We identify some problem cases where the sizing is either not possible or gives erroneous results. These are detailed below, along with some possible solutions

- **Partial view:** The most common case is when we get a detection of the tip of a fish head or tail on one side and a more complete view of the fish on the other side. While the algorithm can infer the

shape of the ellipse given a partial view, the accuracy of this decreases sharply when only a small area is visible. A method to solve this would be to compare the ratios of area between left and right areas and rejecting any sizing attempt where this lies outside of an empirically determined threshold.

- **Fish too close:** Where the fish is too close to the camera in one or both shots, the projection part of the algorithm becomes highly sensitive to error and/or cannot get a complete view of the fish to be able to accurately estimate the ellipse. In this case, a threshold can be created to ignore the cases where a single instance takes up more than a given proportion of the field of view. Some further experimentation or alignment with the goals of the SmartTrawl project can help in this case if it can be determined that this only occurs in cases where we either want to always accept or always reject the catchment.
- **Turning fish:** Where the fish is turning or contorting, the ellipse template does not fit well, but also the projection in space will be shorter than the length of the fish as it becomes curved in space. There is no real solution to this, but it seems to be a relatively rare occurrence.
- **Bad angle:** In some cases, where the fish is laying in the depth dimension of the field of view, it is possible for the matching algorithm used to pair instances on left and right sides and subsequent point location to falter. The former can be solved by a more robust matching to take this into account, although lack of accuracy in the depth dimension may still remain an issue.

The suggested solutions have a very small overhead, but will have a significant positive effect on the accuracy of the overall project. Further large-scale analysis to determine the frequency of occurrence of these problem cases when compared to the ideal case is necessary. A tentative qualitative observation is that the partial view happens quite often which can be solved with a faster image retrieval and detection process in order to be able to capture most/all organisms that pass through the aperture.



VII. CONCLUSION

In this work, we make substantial strides in underwater aquatic animal instance segmentation, providing a valuable dataset and an effective model that contribute significantly to the field of sustainable fishing practices.

Firstly, we have constructed the first undersea aquatic animals (UAAIS2K) dataset specifically tailored for trawl scenarios, providing rich aquatic animal images with comprehensive object-level and pixel-level annotations. This dataset supports various tasks such as object detection, semantic segmentation, and instance segmentation, thereby facilitating the development of effective underwater animal detection and segmentation techniques. Additionally, the dataset includes diverse underwater fishing scenes and paired stereo camera images, offering potential utility for fish size estimation. These distinctive features set UAAIS2K apart from existing datasets and greatly contribute to the advancement of sustainable fishing practices. The new progress of data collection and annotation is presented.

Furthermore, we propose a novel seesaw contrastive learning framework and integrate it seamlessly with a cascaded network architecture and instance-level data boosting. Our proposed model, CBCS-Net, proves to be an effective solution for aquatic animal instance segmentation. This integration harnesses the synergy among these components, resulting in superior detection and segmentation performance. Through extensive experiments, we demonstrate that CBCS-Net surpasses the performance of 10 state-of-the-art instance segmentation models, establishing its efficacy in undersea instance segmentation.

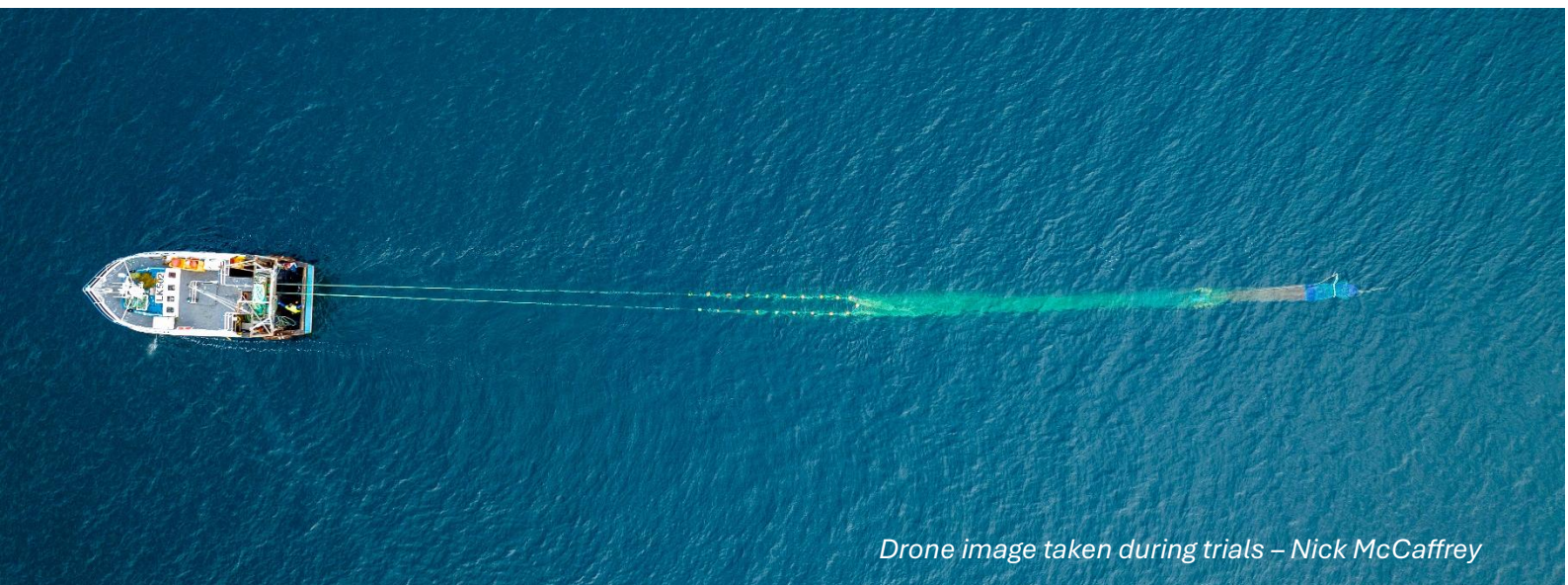
In addition, the fish sizing technique is discussed, where the overall framework of fish size estimation is discussed first. Then, the implementation of fish size estimation is discussed in detail. Moreover, some challenging cases are presented and discussed as well.

REFERENCES

- [1] P. G. Fernandes, K. Coull, C. Davis, P. Clark, R. Catarino, N. Bailey, R. Fryer, and A. Pout, "Observations of discards in the scottish mixed demersal trawl fishery," *ICES Journal of Marine Science*, vol. 68, no. 8, pp. 1734–1742, 2011.
- [2] C. Li, R. Cong, S. Kwong, J. Hou, H. Fu, G. Zhu, D. Zhang, and Q. Huang, "Asif-net: Attention steered interweave fusion network for rgb-d salient object detection," *IEEE transactions on cybernetics*, vol. 51, no. 1, pp. 88–100, 2020.
- [3] C. Li, C. Guo, W. Ren, R. Cong, J. Hou, S. Kwong, and D. Tao, "An underwater image enhancement benchmark dataset and beyond," *IEEE Transactions on Image Processing*, vol. 29, pp. 4376–4389, 2019.
- [4] L. Li, B. Dong, E. Rigall, T. Zhou, J. Dong, and G. Chen, "Marine animal segmentation," *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 32, no. 4, pp. 2303–2314, 2021.
- [5] M. Palmer, A. Álvarez-Ellacuría, V. Molto, and I. A. Catalá, "Automatic, operational, high-resolution monitoring of fish length and catch numbers from landings using deep learning," *Fisheries Research*, vol. 246, p. 106166, 2022.
- [6] M. Pedersen, J. Bruslund Haurum, R. Gade, and T. B. Moeslund, "Detection of marine animals in a new underwater dataset with varying visibility," in *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*, pp. 18–26, 2019.
- [7] P. G. Fernandes, P. Copland, R. Garcia, T. Nicosevici, and B. Scoulding, "Additional evidence for fisheries acoustics: small cameras and angling gear provide tilt angle distributions and other relevant data for mackerel surveys," *ICES Journal of Marine Science*, vol. 73, no. 8, pp. 2009–2019, 2016.
- [8] X. Yu, Y. Wang, J. Liu, J. Wang, D. An, and Y. Wei, "Non-contact weight estimation system for fish based on instance segmentation," *Expert Systems with Applications*, vol. 210, p. 118403, 2022.
- [9] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask r-cnn," in *Proceedings of the IEEE international conference on computer vision*, pp. 2961–2969, 2017.
- [10] Z. Huang, L. Huang, Y. Gong, C. Huang, and X. Wang, "Mask scoring r-cnn," in *IEEE Conference on Computer Vision and Pattern Recognition*, 2019.
- [11] Z. Cai and N. Vasconcelos, "Cascade r-cnn: High quality object detection and instance segmentation," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, p. 1–1, 2019.
- [12] H.-S. Fang, J. Sun, R. Wang, M. Gou, Y.-L. Li, and C. Lu, "Instaboost: Boosting instance segmentation via probability map guided copy-pasting," in *Proceedings of the IEEE International Conference on Computer Vision*, pp. 682–691, 2019.
- [13] A. Kirillov, Y. Wu, K. He, and R. Girshick, "Pointrend: Image segmentation as rendering," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 9799–9808, 2020.
- [14] X. Wang, T. Kong, C. Shen, Y. Jiang, and L. Li, "SOLO: Segmenting objects by locations," in *Proc. Eur. Conf. Computer Vision (ECCV)*, 2020.
- [15] X. Wang, R. Zhang, T. Kong, L. Li, and C. Shen, "Solov2: Dynamic and fast instance segmentation," *Proc. Advances in Neural Information Processing Systems (NeurIPS)*, 2020.
- [16] Z. Zheng, P. Wang, D. Ren, W. Liu, R. Ye, Q. Hu, and W. Zuo, "Enhancing geometric factors in model learning and inference for object detection and instance segmentation," *IEEE Trans. Cybern.*, vol. 52, no. 8, pp. 8574–8586, 2021.
- [17] Y. Fang, S. Yang, X. Wang, Y. Li, C. Fang, Y. Shan, B. Feng, and W. Liu, "Instances as queries," in *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pp. 6910–6919, October 2021.
- [18] J. Redmon and A. Farhadi, "Yolo9000: better, faster, stronger," in *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 7263–7271, 2017.
- [19] C. Rasmussen, J. Zhao, D. Ferraro, and A. Trembanis, "Deep census: Auv-based scallop population monitoring," in *Proceedings of the IEEE international conference on computer vision workshops*, pp. 2865–2873, 2017.
- [20] E. Hossain, S. M. S. Alam, A. A. Ali, and M. A. Amin, "Fish activity tracking and species identification in underwater video," in *2016 5th International Conference on Informatics, Electronics and Vision (ICIEV)*, pp. 62–66, 2016.
- [21] R. Prados, R. García, N. Gracias, L. Neumann, and H. Va˚gstøl, "Real-time fish detection in trawl nets," in *OCEANS 2017 - Aberdeen*, pp. 1–5, 2017.

- [22] S. Villon, M. Chaumont, G. Subsol, S. Ville'ger, T. Claverie, and D. Mouillot, "Coral reef fish detection and recognition in underwater videos by supervised machine learning: Comparison between deep learning and hog+ svm methods,"
- [23] S. A. Siddiqui, A. Salman, M. I. Malik, F. Shafait, A. Mian, M. R. Shortis, and E. S. Harvey, "Automatic fish species classification in underwater videos: exploiting pre-trained deep neural network models to compensate for limited labelled data," *ICES Journal of Marine Science*, vol. 75, no. 1, pp. 374–389, 2018.
- [24] Z. Shen and C. Nguyen, "Temporal 3d retinanet for fish detection," in *2020 Digital Image Computing: Techniques and Applications (DICTA)*, pp. 1–5, 2020.
- [25] Z. Zhao, Y. Liu, X. Sun, J. Liu, X. Yang, and C. Zhou, "Composited fishnet: Fish detection and species recognition from low-quality underwater videos," *IEEE Transactions on Image Processing*, vol. 30, pp. 4719–4734, 2021.
- [26] J. H. Christensen, L. V. Mogensen, R. Galeazzi, and J. C. Andersen, "Detection, localization and classification of fish and fish species in poor conditions using convolutional neural networks," in *2018 IEEE/OES Autonomous Underwater Vehicle Workshop (AUV)*, pp. 1–6, 2018.
- [27] G. French, M. Mackiewicz, M. Fisher, H. Holah, R. Kilburn, N. Campbell, and C. Needle, "Deep neural networks for analysis of fisheries surveillance video and automated monitoring of fish discards," *ICES Journal of Marine Science*, vol. 77, pp. 1340–1353, 08 2019.
- [28] M. Caron, I. Misra, J. Mairal, P. Goyal, P. Bojanowski, and A. Joulin, "Unsupervised learning of visual features by contrasting cluster assignments," 2020.
- [29] J. Yu, J. Yao, J. Zhang, Z. Yu, and D. Tao, "Sprnet: single-pixel reconstruction for one-stage instance segmentation," *IEEE transactions on cybernetics*, vol. 51, no. 4, pp. 1731–1742, 2020.
- [30] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, and L. Fei-Fei, "Imagenet: A large-scale hierarchical image database," in *Proc. IEEE Conf. Comp. Vis. Patt. Recogn.*, pp. 248–255, Ieee, 2009.
- [31] T. Cheng, X. Wang, S. Chen, W. Zhang, Q. Zhang, C. Huang, Z. Zhang, and W. Liu, "Sparse instance activation for real-time instance segmentation," in *Proc. IEEE Conf. Computer Vision and Pattern Recognition (CVPR)*, 2022.

Milestone 4: Six days of field trials in Shetland



Drone image taken during trials – Nick McCaffrey

Smartrawl: Shetland field trials November 2023

14th- 17th November 2023

By: Rosie Ashworth

Introduction

Smartrawl is an in-water discard and bycatch reduction device for demersal fishing trawls, allowing for a species-specific catch and the release of all unintentional catch back into the marine environment. The system is comprised of three components; a stereo camera integrated with AI algorithms to identify and size fish and a patented gate system. The stereo camera, positioned in the trawl extension obtains high-quality images of fish passing through into the cod end. Images obtained are then analysed utilising AI algorithms by an onboard computer to determine fish species and size. Upon classification, a signal is sent to the gate component which is also positioned in the trawl extension to either catch or release fish. When the camera and AI system identify non-target species or protected species, a signal is sent to open the gate, allowing for fish or other marine species to be released back into the marine environment. However, if a commercially targeted species of legal size is identified, a signal is sent to close the gate, ensuring fish transverse into the cod end to be landed.

Previous field trials have been conducted to test and obtain high-quality images from the stereo camera in 2022.

This report provides details on the field trials of the gate component carried out between the 14th and 17th of November 2023 in Scalloway, Shetland aboard research vessel Atlantia II operated by University of Highlands and Islands (UHI) staff. Pre-trial objectives were determined to test the gate functionality by obtaining video footage during tows of the gate fixed in both catch and

release mode along with gate rotation. Additionally, the operation of the new latch mechanism was tested.

Aim and pre-trial objectives

Aim

To obtain video footage of the Smartrawl gate in operation.

Specific objectives

1. To obtain video footage of the Smartrawl gate whilst fixed in **release mode**. Camera orientated towards the open door. Take care to ensure the footage is clear and video has been recorded (last time kept switching to time-lapse). Ideally in an area where might expect high bycatch (long tow time ~ 30 mins).
2. To obtain video footage of the Smartrawl gate whilst fixed in **catch mode**. Camera orientated towards open quadrant. In areas where expect high catches of cod (but only need to have short tow time ~ 15 mins).
3. To obtain video footage of the latch in operation at multiple depths. Gate in **free mode**, camera orientated towards latch, the cod end can be opened to avoid catch.
4. To obtain video footage of the gate in rotating mode to see if it does rotate. To adapt as necessary (e.g. higher speeds, open cod end, bend part of structure into shape etc.). Need only be short tows and can be close to the surface.
5. If 4 successful, then get video footage of the gate rotating and then stopping as a result of the latch engaging.
6. If time permits, get more camera footage in various areas with different fish/shellfish/elasmobranch compositions (~1 hour tows). Gate in catch mode. Enumerate catch (length frequency of catch by species) to compare species and length compositions between camera and trawl catch.

Day 1 – 14th November 2023

The Smartrawl gate was retrofitted within the extension section of the demersal trawl net and the Catchcam system (composed of an underwater camera and a torch) was set up and attached to the net to face the base of the gate in order to provide a whole view of the gate. Tow 1 lasting 30 minutes was then completed in Scalloway Deeps (~50m depth) with the gate fixed in release mode by cable ties. During this tow, video footage was recorded using the Catchcam system. The latch was not set up and GoPro was not mounted. A review of video footage obtained by Catchcam revealed that upon submersion the gate doors collapsed inward towards the gate centre (Figure 1). The aluminium gate doors appeared too weak and folded as seen in Figure 6. Video footage can be located in Smartrawl library.

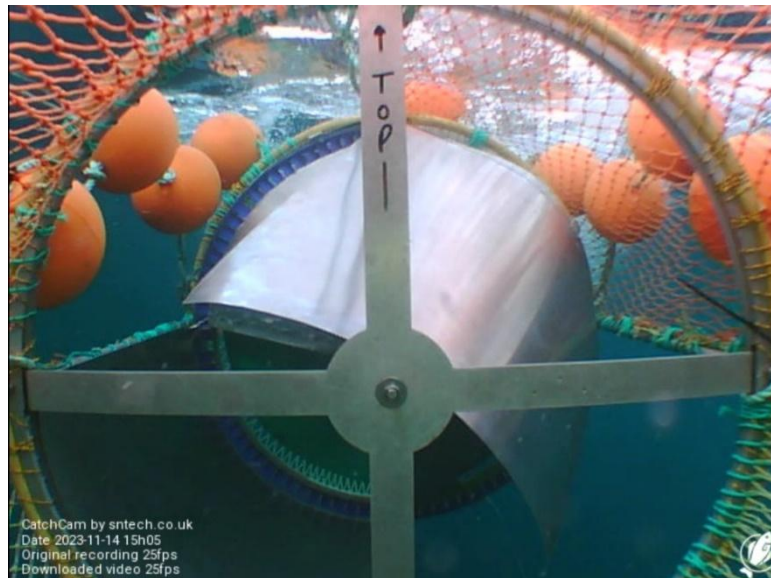


Figure 6 – Image of gate door bent inwards upon submersion.

Upon the haul of the trawl net and gate, the doors were visibly bent inwards along with two large sections of blue radial fins missing at the top section of the cone and a section of aluminium bordering had been detached. Further inspection revealed the majority of the radial fin segments on the cone had become loose as the glue had unstuck. All are displayed below (Figure 7).

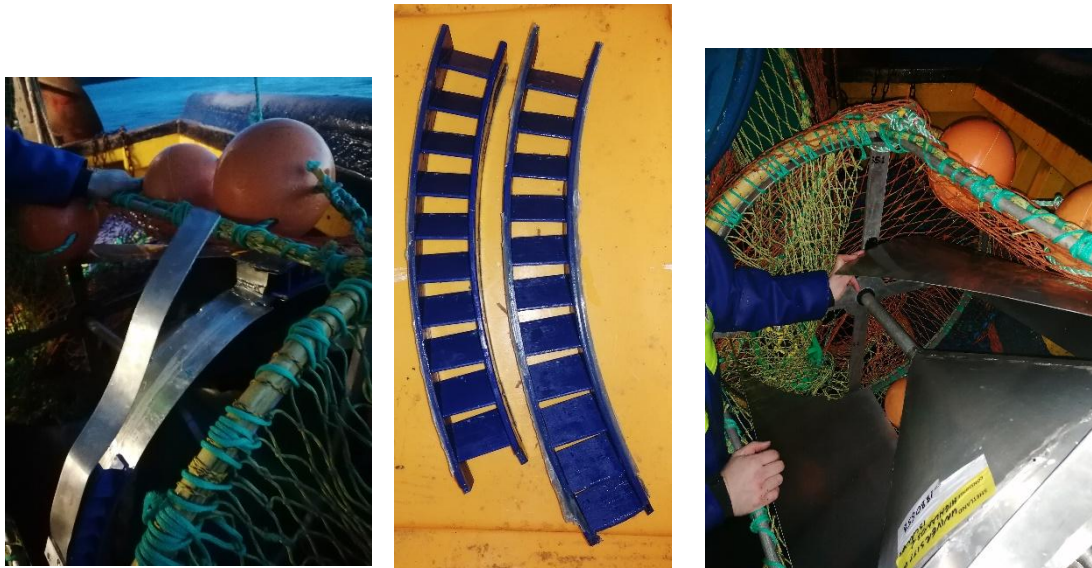


Figure 7 – Images of faults post haul of Tow 1. Left – displaced bordering, middle – detached radial fins, right – bent doors.

Day 2 – 15th November 2023

AM:

The morning of day 2 was spent making repairs to the gate in order to fix issues encountered on Day 1. To increase the stability and strengthen the gate doors an internal metal ring was fixed to the base of the doors using three bolts on each door (Figure 8). To secure the radial fins in place within the metal bordering, long bolts were inputted along the whole circumference (Figure 8). New fixtures strengthened the gate as a whole. Upon test rotation on the pier, several bolts created friction between the bolt top and supporting rods. These bolts were then shaved down to prevent friction upon rotation.



Figure 8 – Gate repairs. Left – addition of metal internal ring to strengthen doors with three bolts secured per door. Right – Long bolts placed through metal bordering and radial fins along the whole circumference of the cone.

After the gate was repaired, two torches were angled on the shaft positioned at doors to allow illumination to scatter across escape panels. A GoPro was also mounted on the base of the gate (Figure 9). Videos were taken in 4k video resolution.



Figure 9 – Two torches and a GoPro were set up at the base of the gate. Torches cable tied to shafts with sponges to angle torches.

PM:

A total of 4 tows were carried out in the PM.

Tow 2 and Tow 3 were both completed in Hamnavoe as drone footage was being taken. As a result, the net and gate remained close to the surface to allow for the drone to get close-up videos and photos of the gate in the water (Figure 5).

Tow 2 – During this tow, the gate was unhinged, and the gate was seen to visibly rotate during the haul. Videos were obtained via Catchcam.

Tow 3 – Drone footage was taken again during this tow for communications purposes (Figure 10). During this tow, the gate was fixed in release mode with cable ties and the Catchcam was on. Videos were obtained via Catchcam.

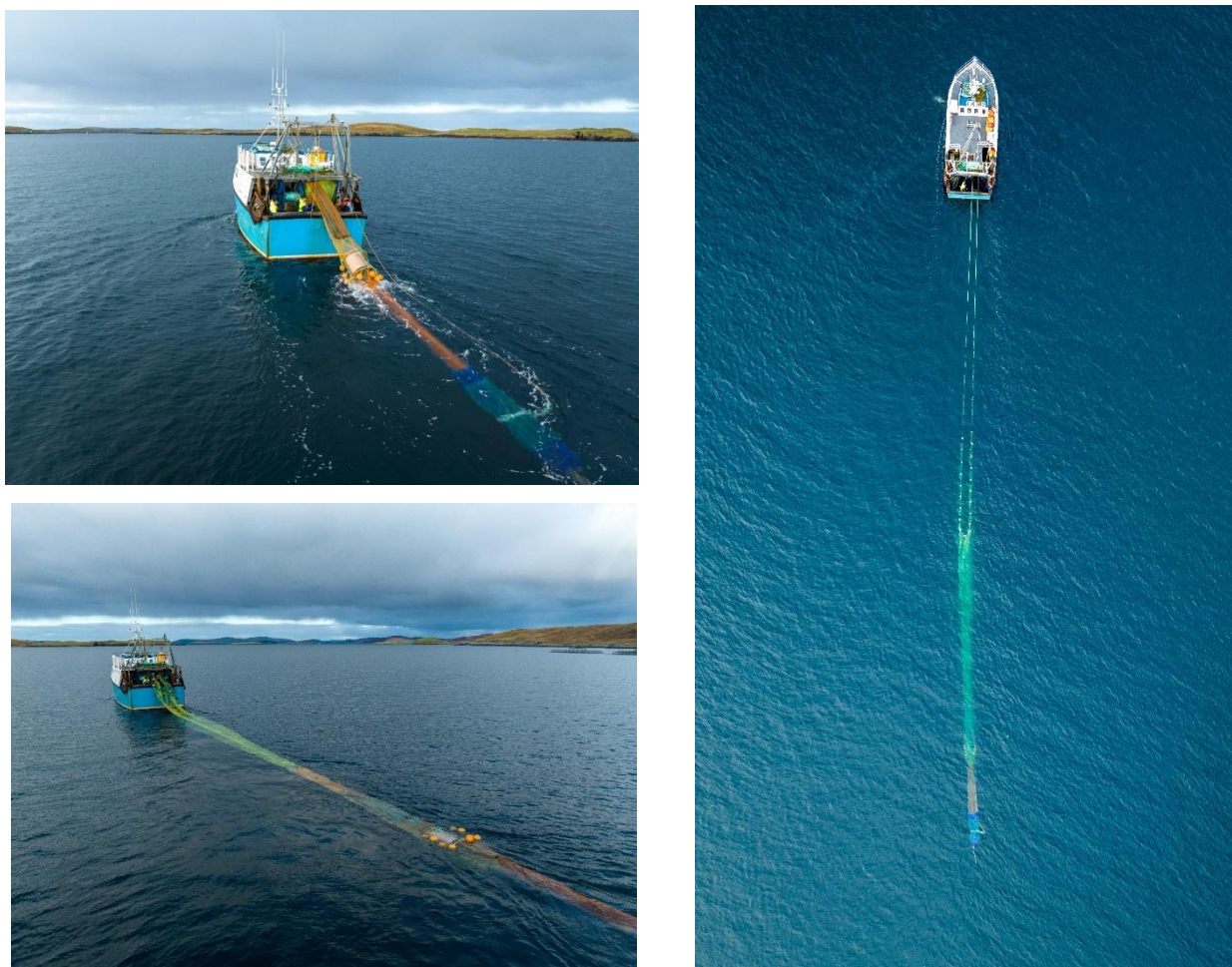


Figure 10 – Pictures taken from drone footage.

Tow 4 – Tow 4 was completed in Scalloway deeps with the gate fixed in release mode. This tow occurred at approximately 50m depth for 30 minutes. The Catchcam and GoPro were obtaining footage and two additional torches were fixed to the shaft to provide additional light.

Tow 5 – Tow 5 was also completed in Scalloway deeps. During this tow, the gate was fixed in catch mode. This tow occurred at approximately 50m depth for 15 minutes. The Catchcam and the GoPro were on, along with two additional torches mounted onto the shaft. Upon haul and opening of the cod end revealed the capture of a juvenile flapper skate (Figure 11). This skate was then measured and tagged and returned to sea. The catch of this tow was weighed and measured (Table 2). A total of 7 different species were caught including Haddock, Whiting, Monkfish, Plaice, Lemon sole, Flapper skate and Squid (Table 2).



Figure 11 – Image of the dorsal side of a female juvenile flapper skate.

Day 3 – 16th November 2023

On day 3, the lithium battery for the latch was inputted into the battery holder on the gate and the latch was tested on the pier (Figure 12). Clear instructions provided by engineers at the National Robotarium were followed to set up the latch system. Battery lights and the battery system were seen to be functioning; however, the latch was not moving. Engineers from the National Robotarium were then consulted, and a Multimeter Tester was used revealing the battery component was working but not the latch. As a result, the latch could not be tested during tows, however, the gate rotation could still be tested by fixing the latch to disengage.



Figure 12 – Image of battery system positioned next to the top end of the gate and latch.

Tow 6: On day 3, two tows were completed. Due to day 2 revealing gate-free spinning upon hauling, the gate rotation was tested during tow 6. The Catchcam and GoPro were recording during the tow. The net was shot at 50m depth for 15 minutes. At the surface the gate was seen to rotate while the boat was traveling at 5.3 knots, and upon submersion and with the speed decreased the gate was seen to rotate once fully submerged at 4.3 knots. Despite the latch not functioning, clear footage of the gate rotating was recorded.

Tow 7- Tow 7 was completed at Scalloway deeps at 53m depth for 30 minutes with the gate fixed in release. Both Catchcam and GoPro were used to record video footage. Upon video review, no footage was recorded on GoPro.

Tow 8 – Tow 8 was completed at the Side of Skeld at 21m depth for 20 minutes with the gate fixed in release. Both Catchcam and GoPro were used to record video footage.

Tow 9 – Tow 9 was completed at Scalloway Deepes at 51m depth with the gate unhinged. The Catchcam and GoPro were used to record footage of the gate rotating. High speed was used during this tow to trial rotation, however, trawl wires broke as a result of the increased speed. This tow was then abandoned.

Results

Table 1 displays details from each tow carried out during the trials, with a total of 9 tows completed over the 4 days.

Table 1 - Table of Shetland Trial 2023 tow data.

tow	date	location	shot		haul		depth(Fa)		warp (Fa)	singles (Fa)	wind		net		comments
			time	lat/long	time	lat/long	shot	haul			force	direction	headline height (Fa)	wind spread (Fa)	
T1	14/11/2023	Scalloway Deepes	1515	60 07 636, 1 24 097	1545	60 06 429, 1 24 314	50.6	53.5	125	0	1	NE	0.9	23.2	Gate fixed in release. Catchcam ON
T2	15/11/2023	Hamnavoe	1320	60 06 609, 1 19 980	1324	60 06 868, 1 20 219	17	17	10	0	1	NE			Gate unhinged, Catchcam ON, Drone overhead
T3	15/11/2023	Hamnavoe	1341	60 06 609, 1 19 980	1343	60 06 868, 1 20 219	17	17	10	0	1	NE			Gate fixed in release, Catchcam ON, Drone overhead
T4	15/11/2023	Scalloway Deepes	1431	60 07 484, 1 24 197	1501	60 05 770, 1 23 926	50.6	46.9	125	0	2	N			Gate fixed in release, catchcam ON, GoPro ON, 2 torches on shaft
T5	15/11/2023	Scalloway Deepes	1547	60 08 247, 1 24.033	1603	60 07 341, 1 24 253	50.6	51.3	125	0	2	N			Gate fixed in catch, catchcam ON, gopro ON
T6	16/11/2023	Scalloway Deepes	1040	60 08 999, 1 24 630	1055	60 09 991, 1 24.203	50	50	10	0	1	N			Gate free (unhinged), catchcam ON, GoPro ON, trying to get gate to spin
T6	16/11/2023	Scalloway Deepes	1429	60 06 570, 1 24 353	1459	60 05 512, 1 24 561	53.8	53.9	125	0	2	N			Gate fixed in release, catchcam ON, Gopro ON, 2 torches on shaft
T7	17/11/2023	Side of Skeld	929	60 08 801, 1 27 042	950	60 09 640, 1 27 042	21.3	21	100	0	1	N			Gate fixed in release, Catchcam ON, Gopro ON, 2 torches
T8	17/11/2023	Scalloway Deepes	1100	60 07 530, 1 24 092			51.3		10	0	1	N			Gate free, Catchcam ON, Gopro ON, 2 torches, towed at speed until wire broke

The footage was mainly obtained with the gate component fixed in release mode, however when fixed in catch mode in tow 5 the species caught were identified, measured and weighed (Table 2). A total of 7 species were identified with haddock and whiting making up the majority of the catch.

Table 2 – Catch data from Tow 5—gate in catch mode.

tow	T5												
date	15/11/2023												
fishing ground	Scalloway deeps												
species	weight (kg)		length of measured (cm)										
	measured	unmeasured	1	2	3	4	5	6	7	8	9	10	11
cod													
haddock	7.03		31	35	38	38	38	39	39	39	39	40	45
saithe													
whiting	3.35		29	31	31	31	31	33	34	34	36	42	
ling													
monkfish	3		44	45									
hake													
lythe													
john dory													
plaice	0.38		32										
lemon sole	0.3		28										
witch													
megrim													
turbot													
halibut													
brill													
cuckoo ray													
thornback ray													
spotted ray													
sandy ray													
lesser spotted dogfish													
common dab													
long rough dab													
common skate	measured - data on other sheet (Shaun Fraser)												
grey gurnard													
red gurnard													
squid		0.99											
norway pout													
poor cod													
argentine													
sandeel													
total weight	15.05												

Conclusion

Upon completion of gate field trials, pre-trial objectives 1, 2, 4 and 6 were achieved, despite some early gate failures in tow 1. A large amount of video footage was obtained with the gate fixed in release and catch mode. The Catchcam and GoPro footage provided two different fields of view of the gate system, along with obtaining footage of different fish and elasmobranch compositions including recording flapper skate and thornback rays. Objectives 3 and 5 were not met due to the latch malfunctioning, and as a result, the latch could not be tested at depth. However, despite the latch malfunction, objective 4 was successfully met with video footage obtained of the gate rotation particularly upon haul and at speed (4.3-5.3 knots). These trials provided clear evidence of fish and other species assemblages being either effectively caught or released by the Smartrawl gate with minimal injury or contact with the gate. The gate can freely rotate fast when pulled at speed, however, alterations to the design and robustness are required for future trials. Upon completion of these trials, engineers were able to inspect the latch and discovered loose cabling caused the latch malfunction. This has now been fixed and the latch system is awaiting further trials in June. The gate component is also being modified to increase its strength and secure radial fins to ensure gate rotation and robustness at sea/depth.

Acknowledgements

Thanks to Shaun Fraser (Senior Fisheries Scientist UHI), Kenneth Pottinger (Skipper), Davie Riley (Fisheries Technician and Crew) and Sarah Ayres (Fisheries Research Assistant).

Milestone 5: Smartrawl 5.0: AI Component Final Report

Dewei Yi, Chris Moorhead, Yiren Li, and Paul G. Fernandes

June 2024

1. Overview

This project report presents a comprehensive study on developing and enhancing methodologies for identifying and sizing commercial fish species using advanced AI algorithms. The report is structured into several key sections in the rest of this report. Each section focuses on a critical aspect of the project, from data collection to performance improvements and practical implementation.

2. Building Dataset for Commercial Fish Species

After huge amount of work for the data processing, we extract data for nine commercial fish species. Qualified ichthyologists individually labelled the individual fish species, and an external cleaning process was undertaken beforehand to eliminate mislabelled and empty images. The final dataset includes a set of nine fish species swimming in water tanks, making a total of 2918 fish, taken at various angles, positions, and obstructions. Each image tries to contain a single fish instance, meaning most images contain only one fish per image. The species and amount of fish in the training and test set include 414-115 Cod, 8-1 Dogfish, 73-19 Flatfish, 163-175 Haddock, 16-2 Herring, 43-11 Monkfish, 125-33 Prawn, 9-4 Saithe and 899-208 Whiting, taken at a consistent resolution of 2048 by 1536 pixels, with varied lighting effects by mimicking effect of sunlight underwater to create instances of deep shadow. This imbalanced the dataset, which will be addressed in the implementation stage through additional data augmentation prioritisation of complex classification cases. The range of fish provides diverse colour and shape variations, ensuring robustness and differentiation between classes. All images are relevantly labelled and separated into training (2350 fish) and test sets (568 fish). This dataset is an excellent option for identifying fish species due to its direct capture of a natural working environment; by taking pictures of fish captured from the sea, the instance segmentation model learns on a closely real-life simulated environment, making it a logical conclusion in increasing generalisation capability. Initial data augmentation for the training and test set will be taken from base file configurations as presented for each model in MMDetection, which were configured in the best interest to serve as a standardised starting point [9]. Adaptations will be introduced to the training and testing process as the project progresses to extract essential information better and personalise results. To follow ethical guidelines in line with this project's aim for sustainability in fishing practices, the fish are taken and released, meaning no harm is done to them during the process. Moreover, some examples of these nine commercial fish species are provided in Fig. 13

Table 3: Distribution of Fish Species in Training and Test Sets

Species	Training Set	Test Set
Cod	414	115
Dogfish	8	1
Flatfish	73	19
Haddock	163	175
Herring	16	2
Monkfish	43	11
Prawn	125	33
Saithe	9	4
Whiting	899	208

3. Installation and Usage Instructions

Documentation is provided here for installation of prerequisites to run the existing A.I. algorithm and any subsequent updates. There are varying considerations that must be taken into account, or may simply be useful background, for prospective users who are non-experts:

- **Windows vs Linux OS:** In general, most things work the same on either operating system. The algorithm is written on/for Linux, however, and some few lines of code are different between the two.
- **Inference vs Training:** Inference allows processing of images based on a model¹ that can be provided by the user and a source of images. Training has a much greater set of requirements and allows for the creation of new models using new data.
- **CPU vs GPU:** The algorithm itself runs on either, but the prerequisites are slightly different for GPUs. Inference will run close to real time on CPU (0.5 image pairs per second) and much faster on GPU, but training cannot be done using CPU alone. The time above includes saving annotated side-by-side comparisons of each image to the results directory.
- **Command Line vs Application:** It is necessary to use command line and virtual environments during installation and running. Explanation, justification and useful commands are given later.

¹ The model is the neural network that the main A.I. component of the algorithm. It is just one ingredient in a larger framework. Models come in various structures, but each structure has a capability to be trained on different data.

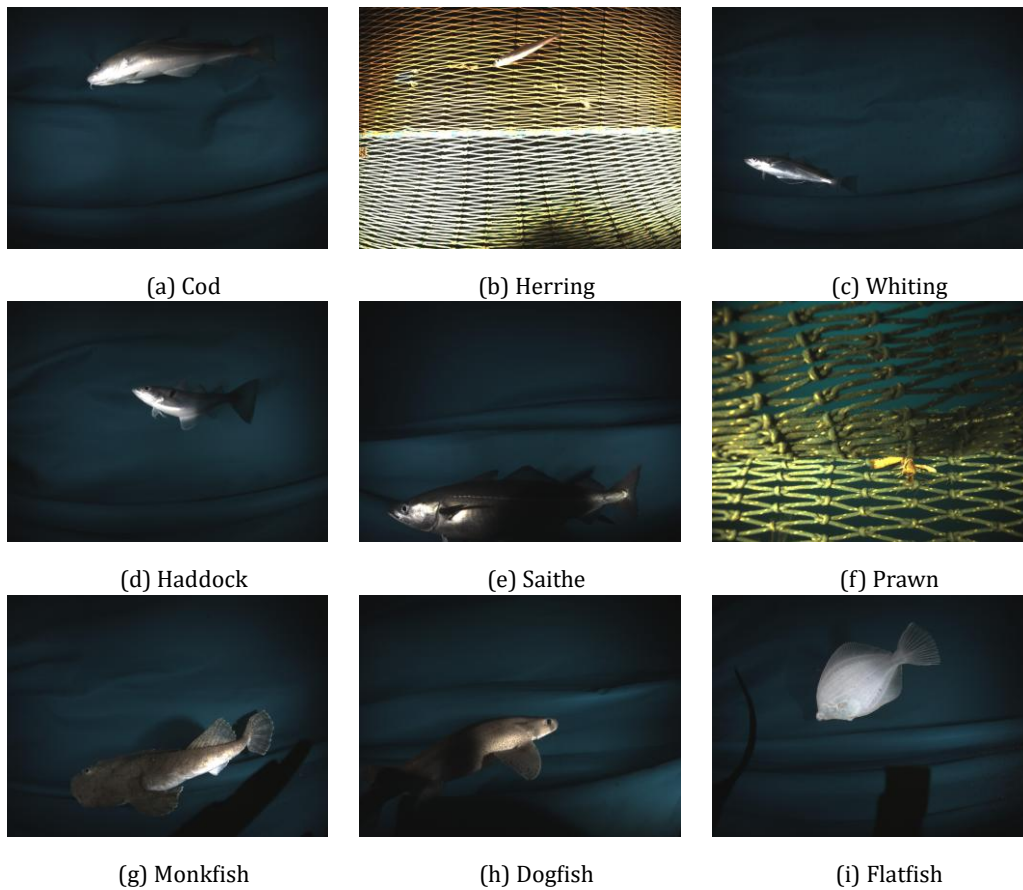


Figure 13: Nine species occurring in the dataset

- **Jetson Nano Orin vs Workstation:** Note that the Jetson Nano Orin has a completely different computer architectures compared to most computers (AARCH vs AMD) and it has its own version of Linux called Tegra, also known as Jetpack. The version we employ on workstation is designed to have as minimal differences, so the working copy requires more specific version to be installed.

3.1 Inference Only

The following instructions are for inference on CPU only. It would be preferable, but not necessary, to run with a GPU. Changes for this case are described afterwards.

3.1.1 Installation (CPU)

The following installation was made on a Windows system.

Install Conda

Conda is a package manager that allows us to create a virtual environment that allows packages to be installed in isolation without affecting the main version of Python. This is important because we will need to be able to separate environments for the inference and data preparation stages of training. Installation instructions can be found [here](#). Some extra notes:

- Miniconda is the recommended option.
- This will be installed to the user. If your PC has multiple user profiles, it won't be available to others. This applies for everything at the installation stage.

- You may see a box with the option "Add to PATH". This must be ON. • To test, go to the directory you want to keep your project. Type cmd and ENTER into the address bar of the window. This will open a command line interface at that location. Any command window will work, but we will require this location for later installation steps.
- In the command window type: conda list

And press ENTER. This should show conda options and will be evidence of installation.

- If this is not the case and you get "conda not recognised", you may need to add conda to the Environment Variables (add to PATH). The location of conda you need to add will be in C:\Users\username\AppData\anaconda3 or similar.²

Create conda environment

Now we need to create the environment which will hold all our packages. In the command window in the desired location from above we need to type and enter:

```
conda create --name smartrawl python=3.8
```

The word following smartrawl can be whatever you wish. We are constrained to use Python 3.8 on the Jetson device, so we need to specify this earlier version. We need to activate this environment with the following:

```
conda activate smartrawl
```

Or equivalent. The command window should show the name of the activated environment to the left. You may test to see if Python is available by typing python and enter. This will give you interactive version of Python where you can give line by line instructions. Exit using the exit() command and ENTER.

Note that the activation of the conda environment MUST be done every time we are using the program or making installations

Download MMDetection

We need to now download the MMDetection framework that controls the object detection portion of the algorithm. This has the code for loading, training

² If you have only use-level access, it may be the case that this location is hidden to you by the administrator. To avoid problems, install at the default location.

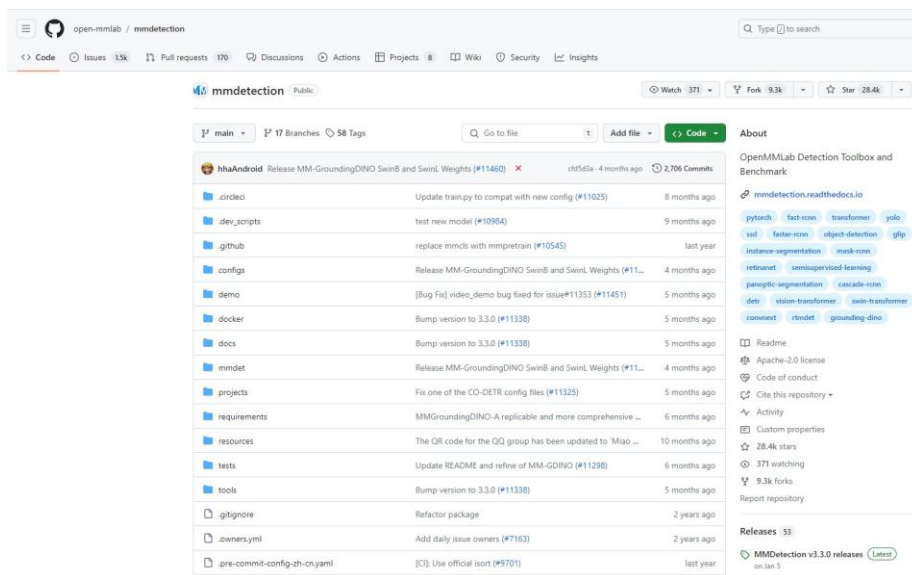


Figure 14: MMDetection github repository

and using the neural network model and is a flexible and widely-used pipeline for the computer vision available on both Linux and Windows systems. The github repository can be found here.

The whole thing can be downloaded as a zip using the code button at the top right. Unpack this zip file and organise how you wish. This will be the main project directory where we must place the remaining components.

Place additional files

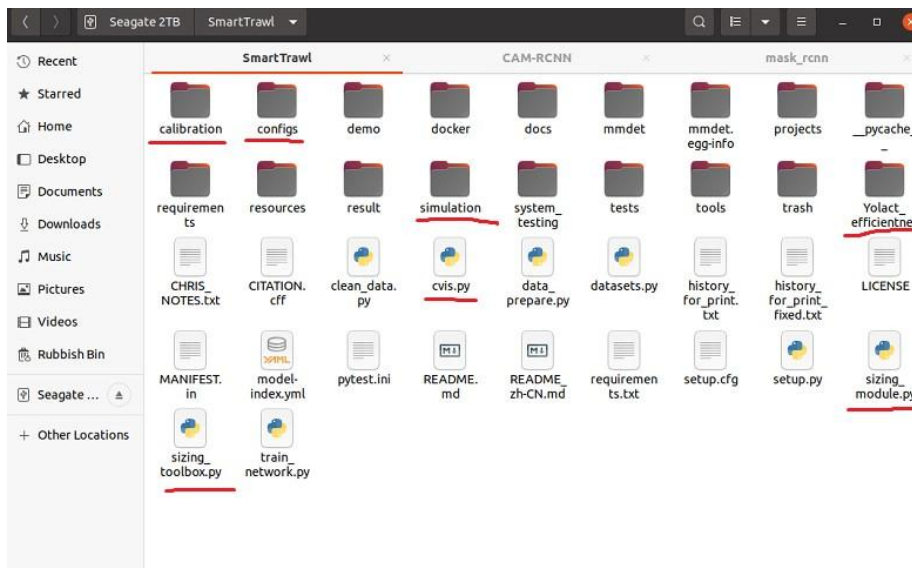


Figure 15: Necessary components to be able to run the code. Yolact efficientnet contains the example model .pth and .py config files. The current model may have a different name like rtmddet

The following files are needed to run the algorithm:

- **sizing module.py**: Contains the main code that runs the simulation and processes the images.
- **sizing toolbox.py**: Contains code snippets for different steps of the sizing and error correction.
- **cvis.py**: Contains tools for custom visualisation of the results.
- **Model and config file**: This will be a sub-directory of the main project directory that contains a model (with extension .pth) that is the structure and trained weights of the neural network and the configuration file (with extension .py). The latter is a requirement of the MMDetection framework that instructs it how to use the model.
- **Simulation directory**: Contains sub-directories “left” and “right” which have matching images for the left and right cameras. This is the default location for the simulation image capture process.
- **Camera calibration file**: Create a sub-directory of configs called “stereo cam”. Inside this directory, place the “stereo calibration.npz” file.

Install packages

We need to install the packages in the environment. If not already open from the previous step, open the command line from the main directory using the instructions in 2.1.1, note 4.

Install the packages using the following commands:

```
pip install torch==1.13.1 torchvision==0.14.1
pip install -U openmim
mim install mmengine
pip install terminaltables
pip install pycocotools
pip install shapely
pip install scikit-image
pip install mmcv==2.0.1 -f https://download.openmmlab.com/mmcv/dist/cpu/torch1.13/index.html
```

These are versions compatible with Python 3.8 and mirror those available on the Jetson Nano Orin. These will take some time to install.

3.1.2 Installation (GPU)

The installation for a machine with GPU³ differs only in a few key aspects. Follow all steps as above with the following differences.

- **CUDA**: Used to interface with the GPU hardware. This needs to be installed separately using the following instructions. We have used CUDA 11.6 for the current project. This is done either before or after installation of conda, but prior to installing pytorch in the next step.
- **pytorch**: The first line of the package installation in 2.1.5 must be amended to:

```
conda install pytorch==1.13.1 torchvision==0.14.1
pytorch-cuda=11.6 -c pytorch -c nvidia
```

³ Note that GPUs must be made by NVIDIA as others are not compatible with the installed tools.

All in one line

- **mmcv:** Similarly the address in the last step must be amended to:
<https://download.openmmlab.com/mmcv/dist/cu116/torch1.13/index.html>

Apart from these two exceptions, follow the same steps as 2.1.2 "Install packages".

The installation can be verified by entering the following in command line:

```
python import torch torch.cuda.is_available()
```

This should provide an output of "True". Close using `exit()`. If it is "False", it means that Python isn't communicating with the GPU and will require further troubleshooting.

3.2 Usage

The following are instructions on how to use the algorithm to perform the detection and sizing of paired fish images.

Activate environment

Navigate to the main directory and open a terminal. Activate the conda environment using:

```
conda activate smartrawl
```

Or whatever name was used during setup.

Place files

For the full algorithm, including sizing, place the images from the left and right cameras into the subdirectories "left" and "right" found in the simulation directory in the main directory.

The left and right sequences must contain matching images only i.e. there cannot be image IDs that exist in the left directory and not the right directory, or vice versa.

Edit Python File

The python file containing the algorithm has a number of constants listed under "SET-UP". These can be modified to change the default behaviour. The most useful is to be able to change the SOURCE directory to another that contains a dataset with matching left and right images in similarly named subdirectories.

If you don't want to modify these, skip to the next step. It may provide useful context for any potential troubleshooting. The following are a list of the modifiable parts of the algorithm that you may or may not wish to modify for simulation purposes.

- **SOURCE:** This is the path of the root directory containing the left and right directories of paired test images. By default it is the "simulation" subdirectory.

- **CAPTURE TIMEOUT:** Number of seconds before the detection process closes down because the cameras have stopped recording. In the simulation this will be when the image sequence is fully processed. By default this is 5 seconds.

```

35 # SET-UP
36 # SOURCE = Path.cwd()/"simulation"
37 SOURCE = Path.cwd()/"testing_datasets/long_seq/deployment_27"
38 CAPTURE_TIMEOUT = 5 # Number of seconds before the detection process closes down because the cameras have stopped recording
39 CAPTURE_RATE = 0.005 # Number of seconds between image capture instances. This value is only used to test timing.
40 DETECTION_THRESH = 0.4 # Confidence threshold below which we reject possible detections
41 TEST_LENGTH = 10 # How many images will be run in the initial test to time the speed of the system.
42 REFRESH_INTERVAL = 0.0003 # Delay in continuous checking of queued items
43 MODEL_CONFIG = "/rtm_det/rtmdet-Ins_tiny_8xb32-380e_d26_27_9Cls_3.py" # Model setup
44 MODEL_WEIGHTS = Path.cwd()/"rtm_det/epoch_300_rtmtdt_R1_1.pth" # Path to trained weights
45 LOG_DIR = Path.cwd()/"result"/"preds" # CSV file containing all detections. See documentation for format
46 CALIB_PATH = Path.cwd()/"configs" / "stereo_cam" / "stereo_calibration.npz" # Path to the calibration file.
47 SCALE = 0.4 # Factor for inference of smaller image. <= 1.
48 SIZER = FishFitter(str(CALIB_PATH), SCALE) # Sizer object needed for fish size estimation
49 VERBOSITY = 0 # Sets the default behaviour for printed display.
50 BOOT_IMAGE = Path.cwd()/"simulation"/"left"/"01712_D20220401-T101925.218_19462900.jpg" # Single image for first_inference.
51 #This reduces initial delay and also signals when the camera capture is ready to start.
52
53 # If "simulate" argument is chosen, these are the default directories where the left and right images are sized
54 LEFT_SIM = SOURCE/"left"
55 RIGHT_SIM = SOURCE/"right"
56
57 # OPTION TO CHECK AGAINST GROUND TRUTH.
58 # Change from above if .jsons not in same directory.
59 GTS_ROOT = SOURCE
60 LEFT_GTS = GTS_ROOT/"left"

```

Figure 16: SET-UP section to the sizing module.py that needs to be edited for custom use

- **CAPTURE RATE:** Number of seconds between image capture instances. The module should function at a rate faster than 0.5 seconds per image capture. This is initially very fast (0.005 s/image) in order to test the speed of the system and determine a capture rate that is appropriate for the current system and settings e.g. it will run a faster speed when we don't need to save annotated images. It will load up a number of images and then process them.
- **DETECTION THRESH:** The detection threshold below which any suspected objects will be rejected. This mainly prevents the occurrence of a multitude of low-confidence predictions in the region of less than 10% likelihood which would otherwise need to be removed in postprocessing the predictions prior to the sizing algorithm.
- **MODEL CONFIG:** The path to the model's config file. This will remain fixed unless you are training your own or passed a different model at a later date.
- **MODEL WEIGHTS:** The path to the trained model weights. This will remain also remain fixed.
- **LOG DIR:** Path and name of .csv file used for logging results.
- **CALIB PATH:** Path to the calibration file. The calibration .npz file should be here.
- **SIZER:** Sizer object needed for fish size estimation using FishFitter class. This is what takes in the lens properties found in the above calibration path in order to make stereo projection and size estimates.
- **VERBOSITY:** Default value for seeing output in colour of the various processes in the algorithm. In effect, this overrides the –verbose flag we will describe in the next section by setting it to always be on.
- **SCALE:** This is the size of the images that will be loaded and annotated by the algorithm. By default, this is 0.4, which proved to be a suitable factor of down-scaling without loss of accuracy in species detection or sizing. The input to the object detection network is of fixed size and a reshaped version of the initial image captured by the camera. Processing a smaller-scale version greatly improves the speed of the system.
- **LEFT SIM, RIGHT SIM:** The name of the subdirectories of SOURCE where the left and right camera image sequences will be found. Only used in the simulation and are “left” and “right” by

default. These can be overridden to specify locations in a different file structure than the default one. Otherwise they are automatically expected as sub-directories of SOURCE called “left” and “right”.

- **BOOT IMAGE:** The path of the initial image to be processed in order to “warm up” the system and only begin the main image capture and sizing algorithms until after the model has been successfully loaded. On the Jetson Nano Orin, this takes considerably longer than on a workstation.
- **GTS ROOT, LEFT GTS, RIGHT GTS:** GTS are the ground-truth species labels from the json files if the images have been previously annotated using the labelme tool. See the `-gts` flag in the next section for more details,

Running the code

Activate the environment named in the initial setup and run the script using: `python sizing_module.py --sim`

Optional flags can be added to this. The possible options are defined as follows:

- `--sim`: Used when the camera module is not present and will source data from the SOURCE path defined in the code.
- `--vis`: Used if we wish to save the left and right images with class and size detections. Outputs found in the “result” subdirectory of the main directory.
- `--logall`: While the module will always create a .csv file with detections and sizes, the default behaviour is that it will not record cases where no fish are detected on one or more sides. `--logall` will record results for all frames.
- `--verbose`: When this flag is used, each concurrent subprocess will give output to the screen in a different color.
 - **Red**: Shows the filename of the image being loaded.
 - **Blue**: Shows the images waiting in queue to be processed by the object detection algorithm.
 - **Green**: Shows the results of the detection process.
 - **Yellow**: Shows the queue for the sizing process and the results of the steps in the algorithm.
 - White: General output generated by the default algorithm without the `--verbose` flag.
- `--gts`: Will add a column to the .csv file that shows the ground truth labels for the species of the fish on left and right sided. If corresponding .json files do not exist, this will be ignored.

If you want to have visualisation, for example, use: `python sizing_module.py --sim --vis`

Alternatively, for no visualisation and a record of all images whether they contain fish or not, and showing the steps in the algorithm as output, use:

```
python sizing_module.py --sim --logall --verbose
```


View results

The results will be saved in the “results” sub-directory of the main directory. They will be inside “vis” and a sub-directory generated from the date and time of running. The log file will be saved here (see next section).

If the `-vis` flag is used, three sub-directories can be found here:

- **NO DETS:** This will store all images where no detections are made.
- **PART DETS:** This will store all images where only one side of the camera detects any fish.
- **STEREO DETS:** This stores all images where at least one fish is detected on both sides. Two subdirectories are also created **TOO LARGE** and **TOO SMALL** which will contain degenerate cases for when the size estimates are too large or too small. The default sizes for these will be greater than 100cm and smaller than 10cm.

3.2.1 Note on CSV log

The detections for paired left and right images are recorded in `detections.log` each row will be in the following format:

```
0 D20231020-T163304.987 1 haddock 0.9108 haddock 0.9628 61.556
```

The following table shows how to interpret each row of the CSV for N detected objects: Each row begins with the image ID, datetime (also the same as the image filename) and number of detected instances. This is followed five entries for each detected instance which correspond to the label and confidence scores for the left and right images and the size for that instance. When `-logall` is selected, the number of instances and left/right labels will be left empty in the relevant fields.

Entry	Meaning
0	Image ID
D20231020-T163304.987	Datetime/Filename
N	Number of instances
haddock	Left label for Fish #1
0.9108	Left score for Fish #1
haddock	Right label for Fish #1
0.9628	Right score for Fish #1
61.665	Length of Fish #1 (cm)
haddock	Left label for Fish #2
0.9340	Left score for Fish #2
haddock	Right label for Fish #2
0.9429	Right score for Fish #2
42.923	Length of Fish #2 (cm)
...	...
whiting	Right label for Fish #N
0.9821	Right score for Fish #N
37.292	Length of Fish #N (cm)

If the `--gts` flag is used and where labels exist, an extra column or extra columns will be added to show a comparison with the ground truth labels.

3.3 Training

We do not provide detailed guidance for how to do this, but a record will be given for the sake of recommendations for future work in the project. Here are important points to consider:

- **Data Preparation:** There is substantial data preparation involved before it can be used to train a network. The steps would be:
 1. **Label** species and contours of each fish using the labelme tool.
 2. **Filter** the images from left and right size so we have only instances where each pair has a corresponding opposite. Mismatches may occur because the left and right images are generally labelled separately or one side has no fish visible. It also flags any labels which have a small number of contour points that may require relabelling. An existing script called `clean data.py` will do this.
 3. **Combine** the individual .json files containing the original labels into a single, curated .json file in COCO format. This is the format required by mmdetection for training.
 4. **Curate** the training data to only include species of interest and correct any labelling errors as a result of typos or combination of merging classes eg. plaice and sole are relabelled flatfish. There is an existing script that does this and the previous step called `filter st species.py`. This is also use to split the data into training and test sets.
- **Same Species:** The same .pth model file and config file can be used to retrain or fine-tune the network with new data.
- **New Species:** Increasing the number of species will require a change in parameters in config file. The output of the model produces a “scores” vector which will be of length N where N is the number of fish species being predicted. If the number of species is different, the structure of the network needs changing. Some other training parameters may need changing in the config file. The names of each class will need editing too.
- **GPU Usage:** Training needs access to a GPU made by NVIDIA in order to be able to use CUDA required by mmdetection for acceleration. If no workstation is readily available, a cloud computing service like Google Colab can be used at economical cost.
- **Training Script:** Colab would require steps similar to our installation instructions for inference on GPU so that the relevant package versions align with those required for the algorithm. An example of using mmdetection to train can be found [here](#). See the “Open in Colab” at the top. This shows the installation steps and the full training process.
- **Data Upload:** The training data must be uploaded from local device to the cloud. This can be done by mounting a GDrive or other methods.

Details of any of these can be provided for future contributors.

3.4 Extracting Calibration File

The .npz calibration file may or may not be provided. If provided, this section can be safely ignored. Instructions to generate this and a brief description of the problem follows.

The algorithm requires information on the focal and other lense parameters for both the left and right cameras and the relative positions of the left and right cameras. This allows for an accurate projection of matching points into 3D space and thus calculation of fish lengths.

The calibration is done using the calibrate.py file and requires a dataset of 30-40 images of a checkerboard pattern taken in an underwater environment similar to that where the recordings take place. The script can be found in the calibration subdirectory of the main repository. The data for calibration should be stored in a directory containing two subdirectories labelled “left” and “right”. The default directory location for the source directory is a subdirectory of calibration called “calibration images”. This can be modified by editing the SOURCE variable in the calibrate.py script. Similarly, the default checkerboard pattern is (6, 8). This pattern refers to the number of points where four squares meet and, in this case, refers to a checkerboard 7 squares in height and 9 in width. Once run, the script will generate a calibrations file which can be stored in the config subdirectory “stereo cam”.

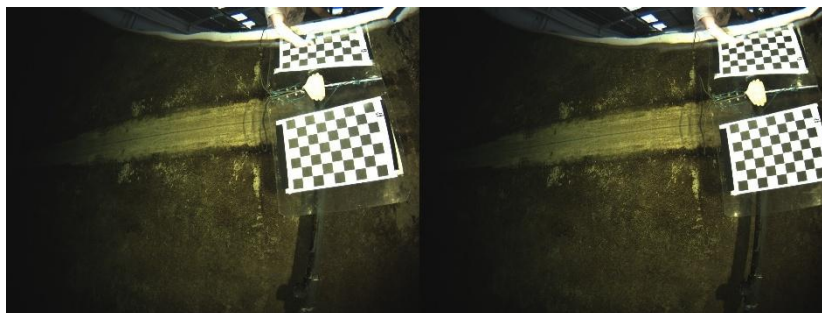


Figure 17: One example of recorded checkerboard pattern.

4. Improved Performance statistics to Identify Different Species on Jetson Orin Nano

4.1 Jetson Orin Nano

To create an instance segmentation model that can be utilised in a fishery environment, we require portable hardware for accessible deployment on aquatic vessels and powerful enough to efficiently run real-time computationally expensive formulations. As such, the MMDetection toolbox will be set up on the Jetson Orin Nano 8GB Developer Kit, a micro-computer and toolkit purpose-built for running artificial intelligence-based code, boasting seventy trillion operations per second [7]. Its compact build maximises cost-effectiveness by keeping running costs low and outperforming general-purpose computers in its price range. With 8GB of dedicated memory as well as a dedicated graphics processing unit, the Jetson Orin Nano leverages NVIDIA’s CUDA architecture for GPUaccelerated tasks by fostering communication between the memory and GPU [8]. In addition to functionality such as mixed precision training provided by tensor cores, it provides a flexible product for real-time instance segmentation.

The Jetson Orin Nano 8GB developer kit is equipped with a micro-SD card slot for system image insertion alongside an Ampere architecture GPU, holding 1024 cores and 32 tensor cores paired with a 6-core Arm Cortex-A78AE v8.2 64-bit CPU with 1.5MB L2 + 4MB L3 cache. This

makes it ideally suited for deep-learning tasks for its GPU-acceleration technology and tensor cores for optimising performance.

With support for camera integration, the Jetson comes equipped with four 3.2 Gen2 USB Type-A Connectors and a USB Type-C Connector. Additionally, the Jetson allows two power modes, the seven-watt and fifteen-watt, making it adjustable for purpose and lightweight. In an energy-dependent environment of fishing vessels [4], the utility of lower power allows for longer voyages and, thus, bigger yields. Watt usage can be adjusted based on catch size and does not cost any pipeline backlogs. The Jetson's already low power requirements, with laptops averaging between thirty to seventy watts, make it a cost-efficient addition. However, special care will be required in deploying the model in marine spaces, as the Jetson is not waterproof; however, with solutions that can enclose the device inside waterproof material, assuming logical planning will not pose significant deployment issues.

4.1.1 CUDA

Utilising MMDetection models efficiently requires using both the GPU and the CPU. Without frameworks that can integrate cohesion between both components, the CPU cannot perform the bulk of calculations. The reasoning comes from CPU architecture, which is designed for flexibility over pure performance; it is the CPU's job to interact with system components, and therefore, it is integrated with control-flow methods to handle complex decision-making processes. This is unlike the GPU, which specialises in parallel executions of singular operations. By generating thousands of lightweight threads instead of the CPU, called CUDA threads, the GPU is designed for raw performance, making it a superior option for arithmetic operations. Combining both benefits requires the CPU's efficient use of GPU resources; thus, NVIDIA developed a solution called CUDA. By utilising GPU bound functions for execution called kernels, CUDA allows for efficient execution of code on the GPU; therefore, leveraging CUDA functionality on the Jetson Orin Nano will greatly benefit model FPS and test efficiency [2].

4.1.2 Jetson Orin Nano Use Cases

The Jetson Orin Nano and its NVIDIA counterparts have been a stable part of edge processing methodology. Recent research has utilised these tools for various endeavours that would benefit from on-site development; drone video feeds for research into autonomous drone movements allowed for a transition to real-time autonomous in-flight capabilities [3]. Projects related to critical operations, such as onboard guidance systems, have been established. With research that simulates space flight powered by the NVIDIA Jetson Nano, reinforcement learning was utilised for the powered descent problem, where thrusters look to assist in the safe landing procedure of spacecraft. Results have showcased positive performance by the agent and another exciting way of utilising NVIDIA Jetson architecture [11].

Autonomous on-site research is extended to driving capabilities, with the lightweight nature of Jetson architecture being utilised for real-time model adaptation during training [5]: showcasing positive results in a semi-supervised environment with a 92.19% accuracy in testing. While the Jetson's utility comes from its lightweight characteristics, research into CAV (connected autonomous vehicles) [1] has showcased that intelligent use of these devices can be cheap alternatives to solutions requiring powerful performance. XTENTH-CAR is an open-source solution to expensive CAV research featuring a stereo camera and 2D LiDAR with ROS (Robot Operating System) utility and tools. Simulating natural environments on a tenth scale on the

NVIDIA Jetson AGX Orin allows for direct applicability while saving costs and preventing development complexity.

The Jetson Orin Nano has been used in healthcare as an inexpensive research method. As is the case with the research addressing home-related fall incidents, the project by Yuanpeng Wang and their collaborators [10] looks to address such a problem that mainly affects the elderly. An extended approach, utilising the base of YOLOv5s architecture, was used to create a lighter and superior model for detecting images of fallen versus standing individuals. Additional research has looked into remote photoplethysmography with the Jetson AGX Orin related to non-contact heart rate detection. By utilising the relationship between heart rate and blood pulsation, which produces signals that can be recovered, the Jetson device efficiently utilised this information based on a facial feature dataset [6].

4.1.3 Software and Hardware Requirements

We first need to attach system memory to an SD Card to use the Jetson Orin Nano. The Jetson is compatible with Linux, which will be utilised with the Ubuntu version 20.04 as the operating system. Before utilising the Jetson, we must flash the necessary software using the NVIDIA SDK Manager. This will be performed by using a separate native Linux machine and downloading the SDK manager on there. After installation, the Jetson requires a consistent power source and an external monitor, mouse and keyboard plugged into it. The Jetson powers automatically upon receiving power and will be usable with the Linux 20.04 Ubuntu operating system.

4.2 Improved performance statistics to identify commercial fish species

To access the performance to identify commercial fish species, both quantitative and qualitative evaluation are carried out in the following.

4.2.1 Quantitative Evaluation on Jetson Orin Nano

This section provides the instance segmentation results of fish detection and species identification. All experiments are evaluated on our previous built commercial fish species dataset. To quantitatively evaluate the results of fish detection, segmentation, and species identification, average precision (AP) is used to assess the performance.

We compare the performance between our used method with other state-of-the-art methods on the tasks of fish detection and segmentation, respectively. The quantitative comparison of fish detection is summarised in Table 2 with regard to AP^{segm} , AP_{50}^{segm} , AP_{75}^{segm} . We found that, for the overall performance, our used method outperforms other state-of-the-art methods for both fish detection and segmentation. Our proposed method can significantly improve instance segmentation performance in terms of AP^{segm} , AP_{50}^{segm} , and AP_{75}^{segm} . More specifically, our method can reach 75.5% of AP for object detection (bbox) and 76.2% of AP for semantic segmentation (segm). For AP_{50} , our method provides the best performance which are 93.8% for bbox and 93.9% for segm. For AP_{75} , our method also delivers the best results for bbox and segm, which are 78.1% and 85.8%, respectively.

Table 2: Average Precision on Jetson Orin Nano (AP^{segm} is the AP of segmentation and AP^{bb} is the AP of bounding box).

Model	AP^{segm}	AP_{50}^{segm}	AP_{75}^{segm}	AP^{bb}	AP_{50}^{bb}	AP_{75}^{bb}
BoxInst	0.500	0.874	0.464	0.805	0.923	0.868
Cascade MRCNN	0.455	0.592	0.497	0.449	0.590	0.443
InstaBoost	0.713	0.915	0.824	0.701	0.910	0.844
Mask R-CNN	0.338	0.405	0.394	0.329	0.404	0.376
PointRend	0.463	0.601	0.496	0.392	0.589	0.464
SOLO	0.233	0.341	0.242	NA	NA	NA
SOLOv2	0.421	0.513	0.442	NA	NA	NA
SparseInst	0.742	0.916	0.721	NA	NA	NA
YOLACT	0.465	0.618	0.515	0.421	0.613	0.452
Ours	0.762	0.939	0.858	0.755	0.938	0.781

4.2.2 Qualitative Evaluation

The qualitative results of instance segmentation are illustrated in Fig. 6-12, where left images are the raw images and right images are our used model identified fish species. In order to further illustrate the performance of our used model, we provide compelling visual results in these figures. From Fig. 6-12, our used method demonstrates good performance in providing good segmentation maps and more complete objects along with accurate bounding boxes.

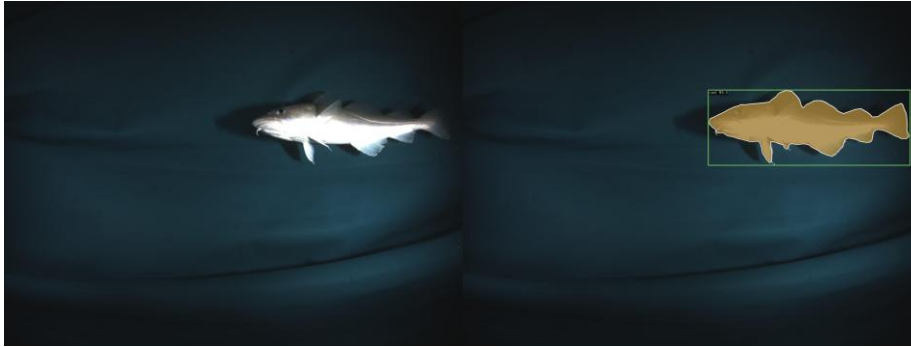


Figure 18: Cod Identification

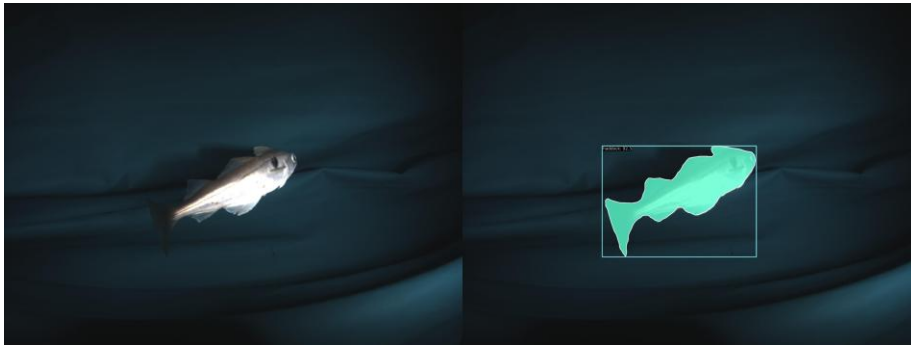


Figure 19: Haddock Identification

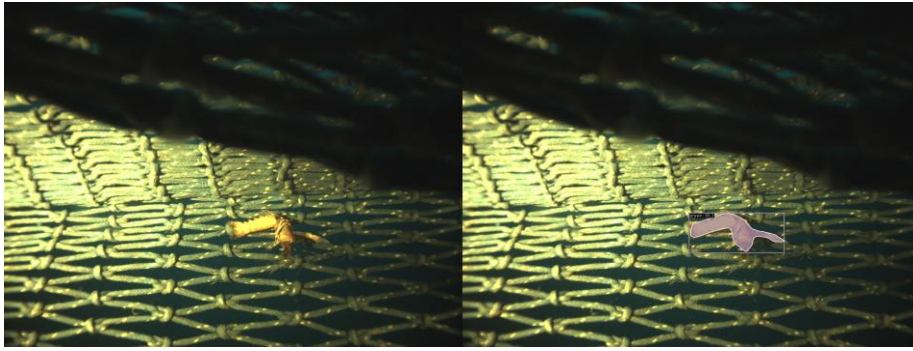


Figure 20: Prawn Identification

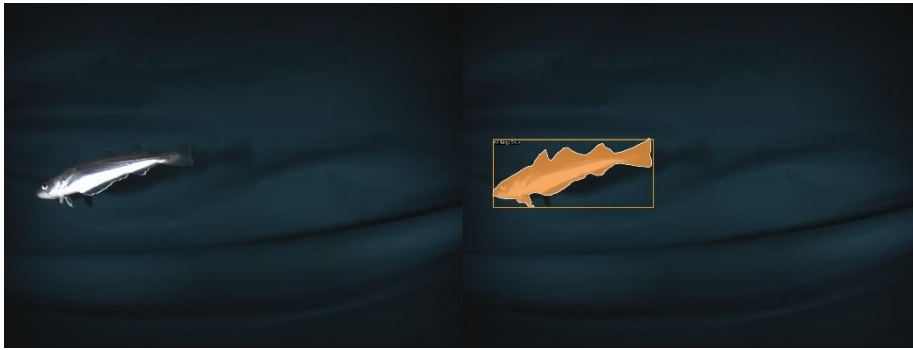


Figure 21: Whiting Identification

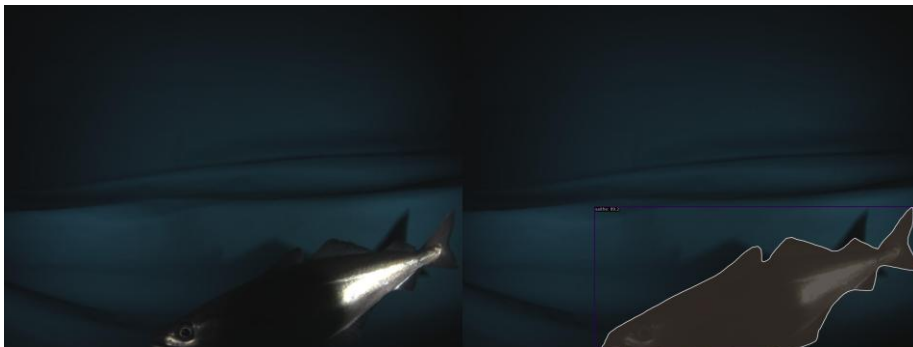


Figure 22: Saithe Identification

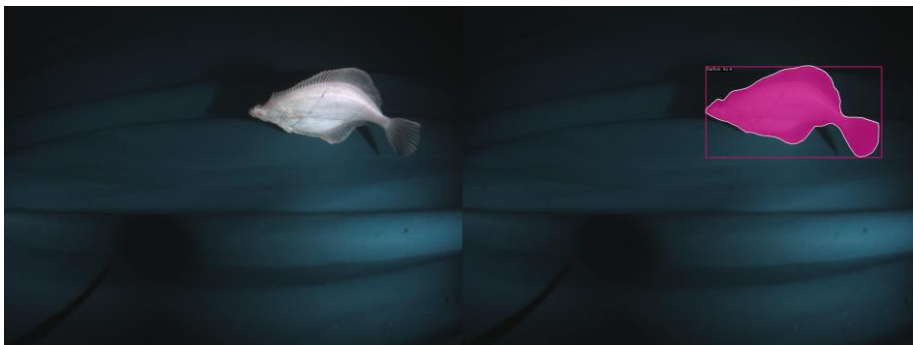


Figure 23: Flatfish Identification

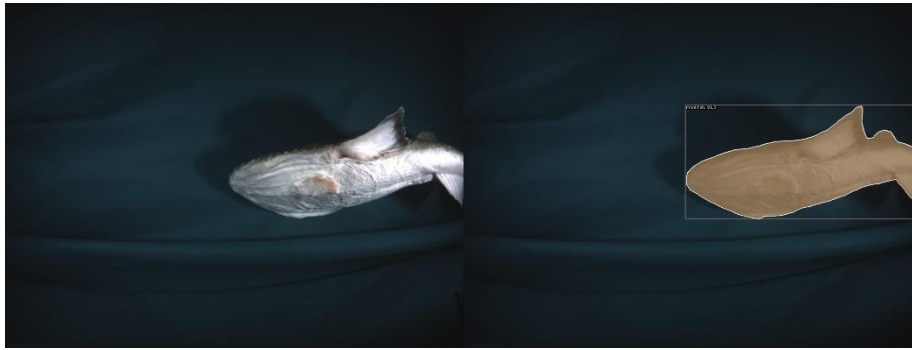


Figure 24: Monkfish Identification

5. Fish Sizing Improvements

In order to improve accuracy and confidence in the sizing methodology, we describe and pursue several avenues of investigation.

5.1 Background

The model used in sizing is the standard fisheye model used in OpenCV.⁴⁵ In this model, each lens has two properties:

- **Camera Matrix:** Which looks like this:

$$M = \begin{pmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{pmatrix}$$

Where f_x and f_y is the focal length of the lens and c_x and c_y are the optical centres.

- **Distortion co-efficients:** Which looks like this:

$$d = (k_1 k_2 p_1 p_2 k_3)$$

The distortion coefficients are used to model the radial and tangential distortion. The radial and tangential distortions are expressed by:

$$x'_{radial} = x(1 + k_1 r^2 + k_4 + k_3 r^6) \quad y'_{radial} = y(1 + k_1 r^2 + k_4 + k_3 r^6)$$

$$x''_{tangential} = x + p_1 xy + p_2(r^2 + 2x^2)$$

These make up the *intrinsic* properties of each camera i.e. those pertaining to the hardware itself.

In addition to this are *extrinsic* properties that need to be used in stereo projection. This is the relative positions of each camera to align them to a shared reference coordinate system. This comprises of a rotation matrix, R, and a translation vector, T.

5.1.1 Stereo projection

The steps in the stereo projection process are:

- Gather left and right images

⁴ Camera Calibration Tutorial

⁵ OpenCV documentation. See `calibrate()`

- Locate matching points on each image
- Undistort the images to correct for lens properties
- Project these points into 3D space using the R and T variables.

The last is not an output of the calibration process using checkerboards described elsewhere in this documentation.



Figure 25: Radial and Tangential Distortion

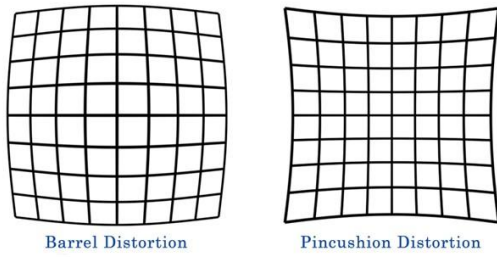


Figure 26: Radial distortion types

5.1.2 Notes

f_x and f_y should be equal as they are calculated using pixel size and density. If pixels are the same size in x and y dimensions, these should be equal. c_x and c_y are the position of the focal centre. This should be approximately the centre of the image in pixel coordinates. Denoting LM and RM as the matrices for the left and right cameras, we have:

$$LM = \begin{pmatrix} 1155.68 & 0 & 1020.14 \\ 0 & 1152.53 & 798.95 \\ 0 & 0 & 1 \end{pmatrix}$$

And:

$$RM = \begin{pmatrix} 1181.36 & 0 & 990.96 \\ 0 & 1182.49 & 818.16 \\ 0 & 0 & 1 \end{pmatrix}$$

The centre of the image in each case would be (1024, 768) pixels. The distortion properties are:

$$ldist = (-0.317, 0.159, -0.00135, 0.0000652, -0.0516)$$

$$rdist = (-0.302, 0.132, -0.00110, -0.000344, -0.0341)$$

The left and right matrices show a difference of 3.15 and 1.13 pixels in f_x and f_y , which is presumably acceptable. There is more of a difference between the focal lengths between the left and right lenses. It is not known if these should be the same, or how close these should be. The current difference is 30 pixels.

The rotation and translation properties are:

$$R = \begin{pmatrix} 0.99975711 & 0.00374768 & 0.02171822 \\ -0.00365076 & 0.99998321 & -0.00450081 \\ -0.02173473 & 0.00442043 & 0.999754 \end{pmatrix}$$

And:

$$T = \begin{pmatrix} 3.83546033 \\ -0.01094449 \\ 0.49746291 \end{pmatrix}$$

5.2 Fish Sizing Results

5.2.1 Stereoscopy

Stereoscopy is utilised to produce the final estimation of fish length in the third dimension. To allow this calculation, two cameras are placed defined as the left and right cameras. Both cameras produce images of fish that are fed into our AI model. Fish sizing can only occur when the same fish is detected on an image in both cameras, otherwise the script produces no result.

When a fish is detected on both sides, the instance segmentation model produces its segmentation mask over the fish. This mask is then extracted to produce a contour (the outline) for each mask by taking a uniform amount of points to avoid bias. We form a two-dimensional ellipse that represents the fish by using the general form, accounting for ellipse rotation:

$$Ax^2 + Bxy + Cy^2 + Dx + Ey - 1 = 0$$

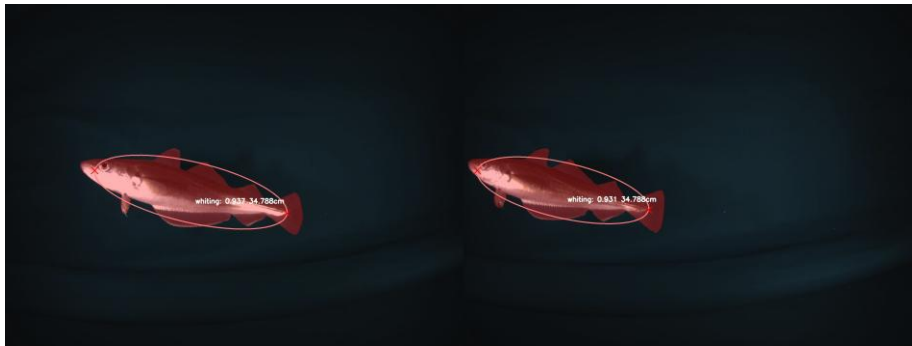


Figure 27: Fish Image result produced by Smart Trawl with ellipses defining the fish.

This ellipse representation and calculation is performed by a function in skimage, that requires 3 inputs:

1. The centre of the object, defined by (x_c, y_c)
2. The major and minor lengths from the centre of the object, relating to the head and tale, defined by (a, b)
3. Theta - the rotation of the ellipse.

Each fish instance is paired to its counterpart in both cameras, where image shifting is utilised to ensure the pairings are set appropriately.

5.2.2 Demos of fish Sizing

One limitation of the current dataset is that we have no ground truth for the size of fish. Although we can use the stereo images from left and right cameras to give the predictions of fish size, it is not straightforward to know who accurate they are. Therefore, we use the common sense to demonstrate the performance of fish sizing. For example, the predicted size of the cod in Fig. 16 is 63.988 cm which matches the common sense of a cod. Moreover, the predicted size of the whiting in Fig. 17 is 25.2 cm. The predicted size of the haddock in Fig. 18 is 29.32 cm. The predicted size of the monkfish in Fig. 19 is 43.636 cm. The predicted size of the prawn in Fig. 20 is 7.932 cm. The predicted size of the flatfish in Fig. 21 is 34.716 cm. The predicted sizes of the cod (63.988 cm), the whiting (25.2 cm), the haddock (29.32 cm), monkfish (43.636 cm), prawn (7.932 cm), and flatfish (34.716 cm) match with common sense as well.

In addition, we also test the performance of fish sizing when there are more than one type of fish as shown in Fig. 22. The predicted sizes of the whiting and haddock in Fig. 17 are 31.82 cm and 31.8 cm respectively. These predicted sizes of whiting and haddock align with common sense as well.

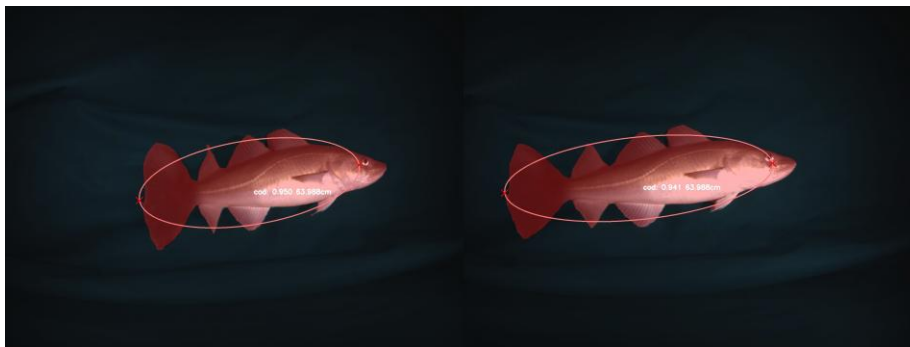


Figure 28 Cod



Figure 29: Whiting



Figure 30: Haddock

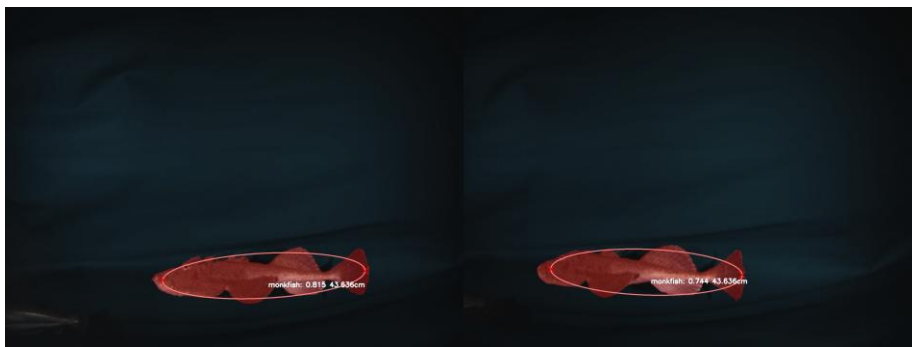


Figure 31: Monkfish

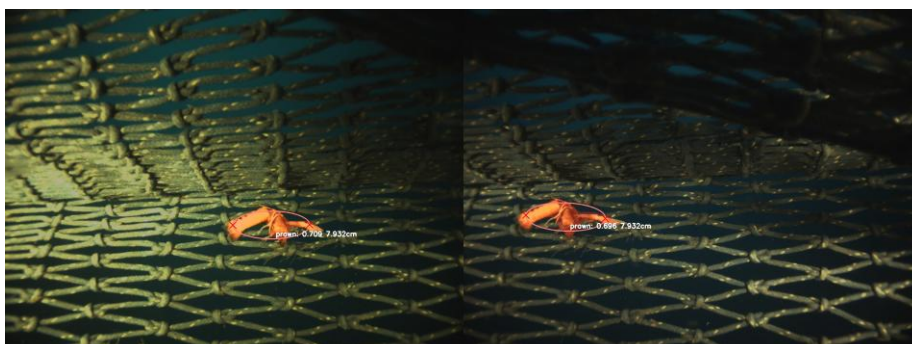


Figure 32: Prawn

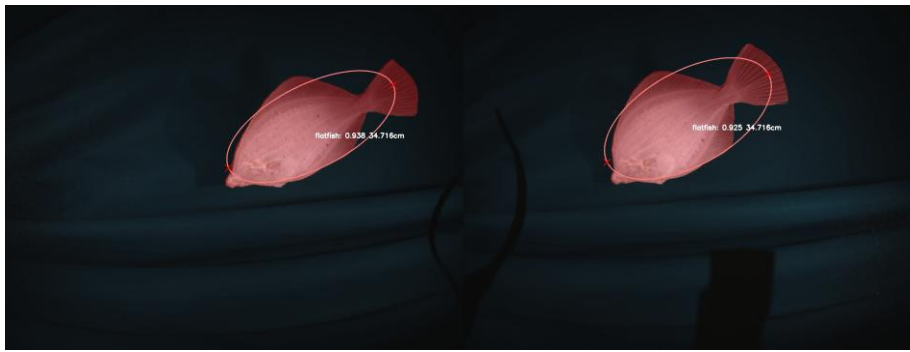


Figure 33: Flatfish



Figure 34: Haddock and Whiting

References

- [1] *XTENTH-CAR: A Proportionally Scaled Experimental Vehicle Platform for Connected Autonomy and All-Terrain Research*, volume Volume 6: Dynamics, Vibration, and Control of *ASME International Mechanical Engineering Congress and Exposition*, 10 2023.
- [2] Toru Baji. GPU: the biggest key processor for AI and parallel processing. In Kiwamu Takehisa, editor, *Photomask Japan 2017: XXIV Symposium on Photomask and Next-Generation Lithography Mask Technology*, volume 10454, page 1045406. International Society for Optics and Photonics, SPIE, 2017.
- [3] Mark Barnell, Courtney Raymond, Steven Smiley, Darrek Isereau, and Daniel Brown. Ultra low-power deep learning applications at the edge with jetson orin agx hardware. In *2022 IEEE High Performance Extreme Computing Conference (HPEC)*, pages 1–4, 2022.
- [4] Oihane C. Basurko, Gorka Gabin~a, and Zigor Uriondo. Energy performance of fishing vessels and potential savings. *Journal of Cleaner Production*, 54:30–40, 2013.
- [5] Kshitij Bhardwaj, Zishen Wan, Arijit Raychowdhury, and Ryan Goldhahn. Real-time fully unsupervised domain adaptation for lane detection in autonomous driving. In *2023 Design, Automation Test in Europe Conference Exhibition (DATE)*, pages 1–2, April 2023.
- [6] Jianwei Li, Zitong Yu, and Jingang Shi. Learning motion-robust remote photoplethysmography through arbitrary resolution videos. *Proceedings of the AAAI Conference on Artificial Intelligence*, 37(1):1334–1342, Jun. 2023.
- [7] Abu Bakar Aakif Muhammad, Adli Aulia Fattah Harahap, Angelita Cindi Viani, Christofer, Ester Vinia, Evans Hebert, Fauzan Valdera, Felix Yaman Kusuma, Gemilang Bagus Ramadhani, Glene

Felix, Hansel Matthew, Miftahul Khoir Shilahul, Muhammad Akbar Attalah, Muhammad Gavin Dirgantara, Muhammad Hurricane, Muhammad Miftah Faridh, Nathaniel Faustine, Ones Sanjerico, Prajna, Pratama P. Rachmat, Reynard Henderson, Ricad Ragapatri, Rizky Rivaldi, Valerie Olive Suryono, Vincent Brendli, and Virdian Harun Prayoga. RoboBoat 2021: Technical Design Report. Technical report, Universitas Indonesia, 2021.

- [8] Fred Oh. What is cuda? [https://blogs.nvidia.com/blog/ what-is-cuda-2/](https://blogs.nvidia.com/blog/what-is-cuda-2/), September 2012. Accessed: [insert date you accessed the site here].
- [9] OpenMMLab. Mmdetection: Openmmlab detection toolbox and benchmark. <https://github.com/open-mmlab/mmdetection>, 2023. Accessed: 2024-04-04.
- [10] Yuanpeng Wang, Zhaozhan Chi, Meng Liu, Guangxian Li, and Songlin Ding. High-performance lightweight fall detection with an improved yolov5s algorithm. *Machines*, 11(8), 2023.
- [11] Callum Wilson and Annalisa Riccardi. Enabling intelligent onboard guidance, navigation, and control using reinforcement learning on near-term flight hardware. *Acta Astronautica*, 199:374–385, 2022.

Milestone 6 : Shetland field trials of the full Smartrawl system integration

17/02/2025

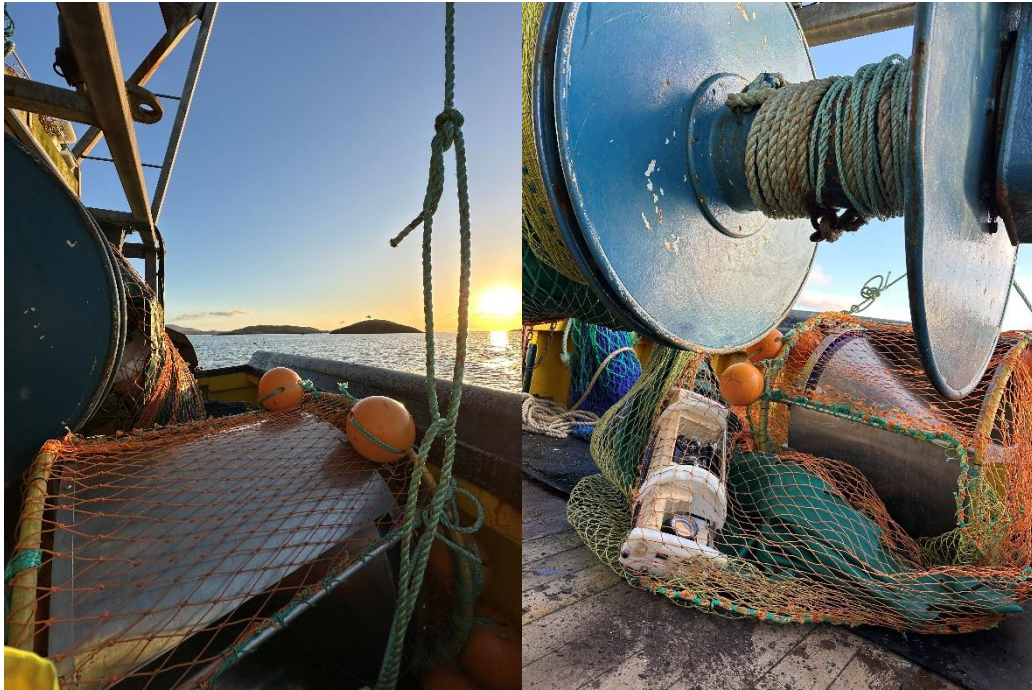


Figure 35 – Image of Smartrawl rigged into trawl net aboard the Atlantia II.

Smartrawl is an in-water discard and bycatch reduction device for demersal fishing trawls, allowing for a species-specific catch and the release of all unintentional catch back into the marine environment. The system is comprised of three components; a stereo camera integrated with AI algorithms to identify and size fish and a patented gate system. The stereo camera, positioned in the trawl extension obtains high-quality images of fish passing through into the cod end. Images obtained are then analysed utilising AI algorithms by an onboard computer to determine fish species and size. Upon classification, a signal is sent to the gate component which is also positioned in the trawl extension to either catch or release fish. When the camera and AI system identify non-target species or protected species, a signal is sent to open the gate, allowing for fish or other marine species to be released back into the marine environment. However, if a commercially targeted species of legal size is identified, a signal is sent to close the gate, ensuring fish transverse into the cod end to be landed.

After several months of equipment delays, these trials represent the first time the full integrated Smartrawl system is trialled at sea. The initial mobilisation of gear occurred on 07/11/2024.

Poor weather conditions over winter resulted in several delays in field testing due to periods of high winds and swell in Shetland. As result, the trials occurred over a period of 7 non-consecutive days.

Trial objectives:

1. Demonstrate the ease of mobilisation, deployment and recovery of Smartrawl.
2. Test the full integrated Smartrawl system at sea aboard the Atlantia II in Shetland.
3. Prove the programmable nature of the system e.g catch all AI trained species except cod.
4. Test the updated gate (version 4), demonstrate gate rotation and engagement of latch.

Day 1 - 21/11/2024:

During day one, the gate was lashed in into the extension of the trawl net and due to poor weather the vessel was only able to get out in harbour area for testing. Here the gate was lashed in, there was no latch attached with the main objective was to confirm free spinning of the gate.

The team was satisfied with the rigging and floats were configured appropriately. It was attached in a manner that was free from obstructions however during testing the gate was catching on its own internal components which prevented it from spinning even at speeds over 5 knots at the surface.



Figure 36 – Image of Smartrawl gate lashed into net extension.

At the end of day one, stereo camera tank tests were carried out to perform a stereoscopic calibration of the camera. Post tank test revealed a fare few images than expected.

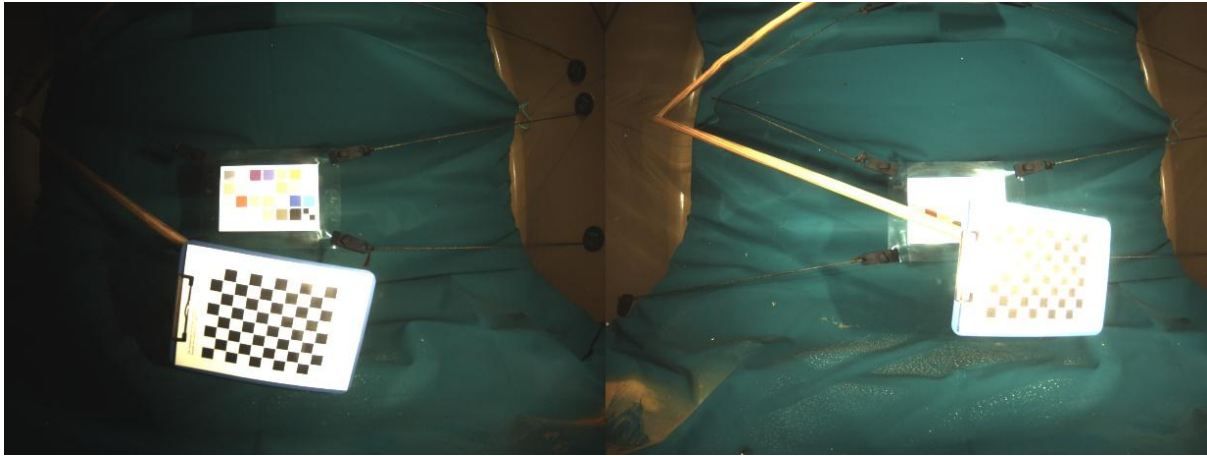


Figure 37– Paired images from tank test of stereoscopic calibration.

Day 2 - 22/11/2024:

At the start of day 2 the fasteners were re-tensioned and by doing so, the gate got it to spin at the surface at speeds over 4 knots, however it was spinning inconsistently and the central bearing appeared to loosen over time.

Nonetheless, it was deployed it at depth with Catchcam system rigged up to monitor the gate. Review of footage revealed no sign of spinning. It was evident that there was a fundamental problem with bearing and was then sent back to the National Robotarium for the bearing to be re engineered out of aluminium.

Day 3 - 28/01/2025:

Due to bad weather on day 3 the modified gate was attached in the trawl and its performance was assessed in air which appeared to spin more freely than the previous iteration. Due to weather the improved gate was unable to be tested.

Day 4 - 29/01/2025:

Day 4 was a full day doing hauls with the updated gate.

This was the first time the gate was spinning at depth. The net was shot at 11:13 at 51 fathoms at position 60 07.215 N, 1 23.978 W with the Catchcam fitted to record the forward area of gate. The vessel speeds over ground was 2.5 knots and towed for 20 minutes. The gear was then hauled at 51.9 fathoms at position 60 07.335 N, 1 24.263 W.

During this haul a small haddock became stuck in the gate which prevented the free spinning of the gate for majority of haul.

The gear was shot again at 11:13 after clearing obstructions in 50.6 fathoms at position 60 07.604 N, 1 24.138 W at 2.5 knots with the Catchcam fitted as before. The gear was then hauled at 12:05 in 54.4 fathoms at position 60 06.454 N, 1 24.286 W. The catch comprised of mostly haddock, skate and plaice.

Day 5 - 31/01/2025:

Day 5 was the first day the fully integrated system was tested. The camera, latch, latch bottle and wiring was all rigged in securely. The camera and gate was a total of 2.5 meter distance apart.

It was immediately clear that once latch was installed that the gate paddles were catching on the hinge for the latch rather than the intended retaining surface. As a result, gate paddles were cut to provide more clearance. This was a rough field fix as the eccentricity of gate was still a problem.

First deployment: Shot at 12:21 at 51.3 fathom at 60 08.072 N, 1 24.013 W towed at 2.5 knots and hauled at 13:06 at 54.6 fathom at 60.06.220 N, 1 24.306 W. The Catchcam was positioned to view gate. Upon haul both strobes were not flashing, however there was a steady green light on stereo camera system.

The system was reset but after resetting no strobes flashing at all, it was confirmed at all plugs and cables were normal and latch housing was switched on.

Initial review of Catchcam footage indicated that gate was in release for entire haul.

A hard reset was attempted by pulling out power plug, however after this it was flashing only once on deck and stopped, following another hard reset is flashed more consistently so moved Catchcam forward of stereo camera and shot again (Figure 38).



Figure 38 – Image from Catchcam, with the stereo camera and gate visible.

Second deployment: The gate was in catch mode and was shot at 14:16 at 53.1 fathom at 60 06.134 N, 1 24.309 W towed 2.5knots hauled at 15:05 at 59.6 fa at 60 07.156 N, 1 24.006 W

After hauling it was clear that only one strobe was still flashing.

Day 6 – 05-07/02/2025:

After testing the fully integrated system and reviewing the AI processed images it was evident that the images were over exposed, resulting in the images to be bright white. There were only several instances where fish were captured in the images, however due to their close proximity to the camera.

Troubleshooting and other issues encountered:

Latch battery: Post deployment when attempting to charge it, an error message 'Voltage invalid' occurred. The battery was then tested with a multimeter to check the voltage and indicated 4 V (much too low). With communications with engineers from the National Robotarium it was clear that the battery had become discharged which caused the battery to be damaged. As a result the battery could not be re charged due to alterations in the chemical balance inside the battery. A new battery was promptly ordered and replaced.

Camera and strobe exposure:

In order to fix the overexposed images, the camera parameter file was altered through adjusting the gain and exposure during bench testing in a dark room. However, after a few hours of testing the camera parameter file was crashing and caused the camera system to stop working and stop acquiring images.

The following day, this issue was resolved as an engineer was able to patch into the system to fix it and alter the gain and exposure settings.

Day 8 (10/02/2025):

Before sea trials, the stereo camera system underwent further tank tests to test the camera parameters to ensure the gain and exposure setting were correct. The gain was set to 0 and the exposure to 600. These setting worked well and the colour squares were clear.

Further bench testing of the system was undergone and images analysed to ensure system functionality before sea deployment.

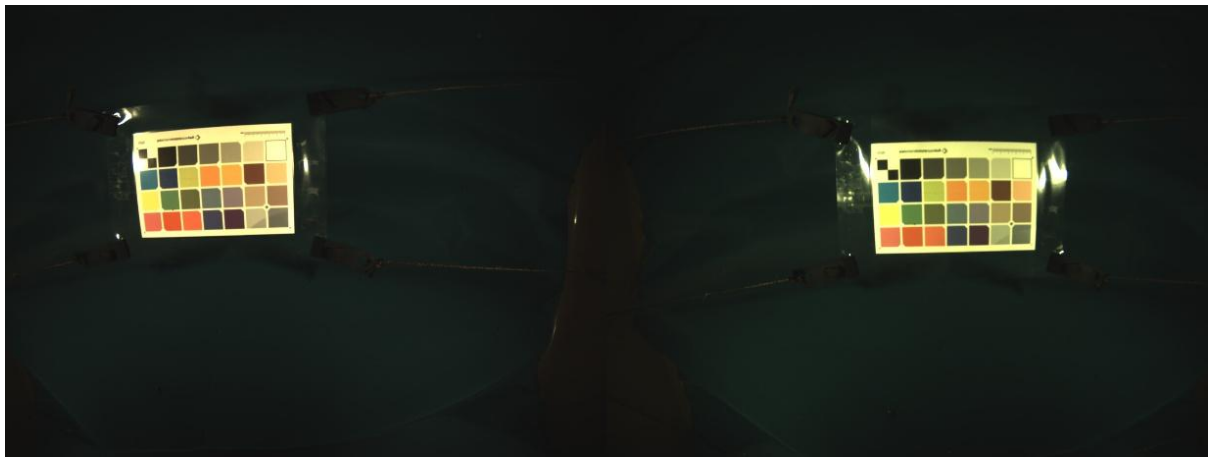


Figure 39 – Paired image from tank test showing all visible colours on colour square.

Another issues encountered when bench testing was overwriting of image files, with the new images acquired on the bench overriding the current images from tank testing.

Day 9 – 12/-2/2025:

Day 9 was the second day the fully integrated system was tested. Prior to at sea testing the 'control.csv' file that is used to program Smartrawl catch species and size was set to catch all species and their minimum conservation reference size apart from cod, as seen below.

```
type,size
haddock,30
whiting,27
saithe,35
hake,27
flatfish,27
prawn,8.5
monkfish,0
```

The camera parameter file was set to gain=0, and exposure=600.

A total of three hauls were carried out on Day 9 at Scalloway Deep as seen in table 3.

Table 3 – Haul data from two days at sea 12.02.25 and 14.02.25

tow	date	location	shot		haul		depth(Fa)		warp (Fa)	singles (Fa)	wind		net		comments
			time	lat/long	time	lat/long	shot	haul			force	direction	headline height (Fa)	wind spread (Fa)	
T1	12.02.2025	North Deep	10:57	60 08.245,1 24.140	11:33	60 06.761, 1 24.298	50.3	55	150	On	2	NE			Catchcam +gopro facing latch
T2	12.02.2025	South Deeps	12:58	60 05.120, 1 24.415	13:30	60 03.963, 1 24.748	53.8	52.8	150	On	2	NE			Catchcam and go pro on
T3	12.02.2025	South Deeps	14:33	60 07.699, 1 24.415	15:04	60 07.040, 1 24.335	53.2	53.1	150	On	2	NE			Took 2 reboots before flashing, gate in catch mode
T4	14.02.2025	Scalloway deeps	10:17	60 05.044, 1 24.260	10:48	60 05.826, 1 24.335	53.8	53.8	150	On	1	E			Deployed at green flashing, catchcam in front of stereo camera, gopro on latch - in
T5	14.02.2025	Scalloway deeps	11:53	60 06.999, 1 24.288	12:54	60 08.999, 1 23.994	55	51.3	150	On	1	E			Catchcam moved to before gate, camera not reset as latch came up moving.

Haul 1:

Haul 1 was shot at 10:57 in Scalloway deeps, with the Catch Cam facing the gate and the GoPro and torch facing the latch. Before deploying the gear, the system was fully booted and strobes flashing. The gate was set in default release position. Upon recovery the stereo camera strobes were not flashing and very little was caught (4 fish) which indicates that there was an issues triggered the latch and hence gate rotation into catch position. Review of the CatchCam footage revealed no gate rotation and go pro footage documented the latch in the same position throughout (no movement of latch).

Haul 2:

Haul 2 was shot at 12:58 in South deeps, with CatchCam facing the gate and the GoPro facing the latch. The gate was again set in release position. There was inconsistencies in the system boot up, where sometimes the strobes would flash. In order to make sure the system was fully booted up, the system was rebooted several times. Upon recovery the stereo camera strobes were not flashing and again very little catch was caught (5 fish) which again was caused by the latch not engaging and gate not rotating (Figure 40).

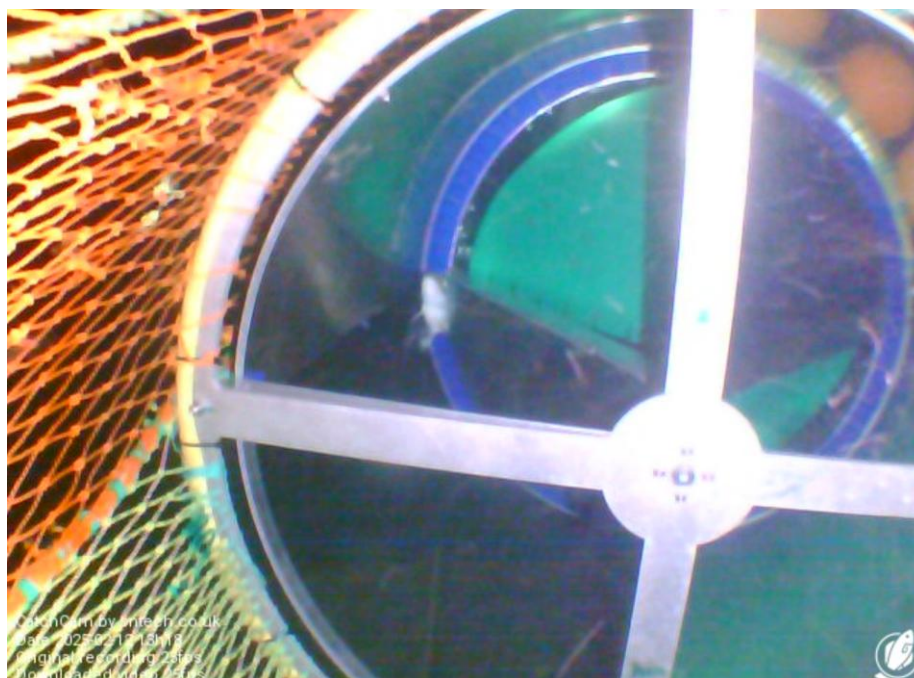


Figure 40 – Image from Catchcam of gate fixed in release position.

Haul 3:

Haul 3 was shot at 14:33 in the South Deeps, again with the Catchcam facing the gate and the GoPro facing the latch. However, for this haul the gate was set in default catch position. There was again inconsistencies with the re boot of the system. Upon recovery, there was only one strobe flashing, and an increased catch as the latch again was not engaged so the gate remained in catch position. The catch was mixed with haddock, cod, plaice, whiting, thornback ray, flapper skate and lemon sole.

Catchcam footage and GoPro footage again revealed that the latch was not triggered and hence the gate did not rotate.

Post demobilisation, issues were encountered with accessing the stereo camera data, although the camera was powered on the ethernet connection was not pinging as a result the data was not accessed and the control file could not be changed.

The following day consisted of communications with engineers from the National Robotarium in order to try and fix the camera, strobes and data acquisition. All batteries were then charged, however this did not fix the issues. All troubleshooting that could be done remotely occurred in an attempt to solve the stereo camera issues, however we were unable to fix the above issues.

Day 10 - Feb 14/02/25

Haul 4:

Haul 4 occurred on the 14/02/25 and was shot at 10:17 at Scalloway Deeps. Before deployment the Catchcam was positioned in front of the stereo camera in order to see if the camera was functioning, again the GoPro was positioned facing the latch. The gate was set into default release position.

Due to the camera issues encountered from day 9, we were unsure whether the camera system would work, so a separate torch was attached to the stereo camera frame to illuminate the camera frame in case strobes were not working.



Figure 41 – Dive torch rigged to stereo camera frame to ensure illumination in case strobes were not flashing.

Post deployment, the stereo camera system was not flashing, however the latch was continuously moving even with all gear on deck. It seemed that the system had a bug which caused the latch the continuously engage. As a result, clear footage was obtained of the latch engaging and catching the gate paddled (Figure 42 and 43). The CatchCam captured the gate clearly rotating between catch and release position.



Figure 42– Latch engaged on gate paddle.

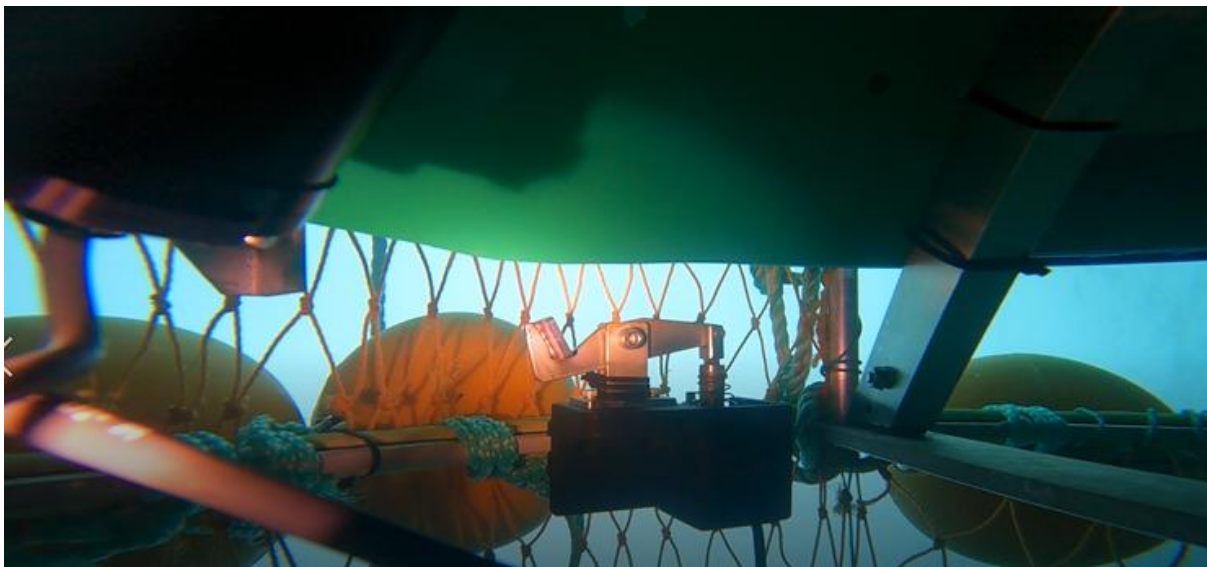


Figure 43 – Latch not engaged, allowing gate to rotate.

There was a mixed catch as a result of the latch constantly moving which included plaice, haddock, cod, whiting and thornback ray.

Haul 5:

Haul 5 occurred on the 14/02/25 and was shot at 11:53 at Scalloway Deep. The gate was set into default release position. After the 4th haul, with the latch continuous moving it was decided not to reset the system, as this will allow for clear footage to be obtained of the gate rotating between catch and release position. Before deployment the Catchcam was moved in front of

the gate in order to see if the gate switch between catch and release position. Again the GoPro was positioned facing the latch and obtained footage of the latch engaging.

The catch was also mixed including haddock, plaice, flounder, dab, grey gurnard and cuckoo ray.

During these two hauls, there was no evidence of the stereo camera and strobes operating. However, due to the camera issues we are unable to view the images acquired. Images will be reviewed upon system optimisation.

Conclusion:

The Shetland field trials for the full Smartrawl system demonstrated both promising advancements and significant technical challenges. Despite delays caused by equipment issues and poor weather conditions, Smartrawl was successfully mobilized and tested the system in various configurations. However, technical setbacks, such as issues with the gate rotation, damaged battery, the stereo camera image exposure, and the latch system, hindered the system's performance during key deployments. These challenges were compounded by difficulties in data acquisition and the camera and strobe synchronisation.

Nevertheless, the trials also showcased valuable learning experiences, including the importance of system calibration and the need for ongoing troubleshooting. The issues were addressed with remote aid from engineers from the National Robotarium, with several fixes, such as altering the gate design and adjusting camera settings, leading to more effective results. Despite the obstacles, the trials provided critical insights into the Smartrawl system's capabilities and initial limitations during testing, paving the way for further improvements and optimization in future testing phases. Despite challenges it was demonstrated how easily the Smartrawl system can be mobilised and deployed and recovered on a small fishing vessel. The latch proved to work well, catching the gate paddled and rotating the gate quickly into catch and release position.

Acknowledgments: Big thank you to Shaun Fraser (Senior Fisheries Scientist UHI), Davis (crew), Victor Duncan (Skipper).

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: Ashworth, R., Fernandes, P., Yi, D., Morrison, D., Fraser, S. (2025) SMARTRAWL 5.0 Final Report. A study commissioned by Fisheries Innovation & Sustainability (FIS) <https://fisorg.uk>

Title: SMARTRAWL 5.0 Final Report

First published: 2025

© FIS



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

