Scottish Government
Riaghaltas na h-Alba
gov.scot

# Automated Identification of Fish and Other Aquatic Life in Underwater Video

## Scottish Marine and Freshwater Science Vol 11 No 18

S Blowers, J Evans and K McNally

marine scotland

# Automated Identification of Fish and Other Aquatic Life in Underwater Video

Scottish Marine and Freshwater Science Vol 11 No 18

S Blowers, J Evans and K McNally

Published by Marine Scotland Science
ISSN: 2043-7722
DOI: 10.7489/12333-1

Marine Scotland is the directorate of the Scottish Government responsible for the integrated management of Scotland's seas. Marine Scotland Science (formerly Fisheries Research Services) provides expert scientific and technical advice on marine and fisheries issues. Scottish Marine and Freshwater Science is a series of reports that publishes results of research and monitoring carried out by Marine Scotland Science. It also publishes the results of marine and freshwater scientific work that has been carried out for Marine Scotland under external commission. These reports are not subject to formal external peer-review.

This report presents the results of marine and freshwater scientific work carried out for Marine Scotland under external commission.

© Crown copyright 2020

You may re-use this information (excluding logos and images) free of charge in any format or medium, under the terms of the Open Government Licence. To view this licence, visit:
http://www.nationalarchives.gov.uk/doc/open-governmentlicence/version/3/ or email: psi@nationalarchives.gsi.gov.uk.

Where we have identified any third party copyright information you will need to obtain permission from the copyright holders concerned.

## Table of Contents

**Executive Summary 3**

**List of Names and Acronyms 7**

**Introduction 11**

Objective 12

Problem Space 12

**Computer Vision 14**

Introduction 14

Feature Maps 16

Algorithms 16

Neural Networks 17

Artificial Neural Networks 18

Convolutional Neural Networks 18

Drawbacks to CNNs 20

**Computer Vision in Underwater Video 21**

Introduction 21

Algorithms Used in Literature 21

Convolutional Neural Networks 22

Competitions 24

**Review of Models for Case Study 25**

Introduction 25

Available Platforms not Incorporated into Case Study 26

FishTick 26

BIIGLE 26

VIAME 27

Online Service Portals 27

Chosen Platforms for Case Study 27

Open CV 27

Object Detection API (TensorFlow) 28

DeepSORT (Object Tracking) 29

Feature Detection Training Pipelines 29

Extracting Images and Annotations 29

Training Pipeline: Bag of Visual Words 31

Training Pipeline: Convolutional Neural Networks 33

Object Detection Pipeline 33

Frame Extraction 33

Background Subtraction 34

ROI Thresholding 35

Feature Classification 35

Object Tracking 35

**Case Study** **36**

Training Models 37

Detection Measurements 37

Fish Detection, Classification, and Counting 38

Processing Times 40

Detections 42

*Videos 1 - 5* 42

*Video 6 - Classifying Sprats vs Smolts* 44

Sea Pen Detection and Counting 45

Detections 46

Nephrops Burrow Detection 48

Detections 49

Analysis of Models from Case Study 50

**Conclusion** **52**

Actionable Recommendations 53

Considerations towards a Complete End-User Solution 54

**Acknowledgements** **57**

**References** **57**

# Automated Identification of Fish and Other Aquatic Life in Underwater Video

Stephen Blowers, Jonathan Evans and Kyle McNally

MarynSol Ltd
45 Timber Bush, Leith
Edinburgh EH6 6QH

## Executive summary

Marine Scotland tasked MarynSol to provide an overview of the current state of computer vision technologies for automated detection of aquatic life in underwater video, the objective being to provide a development route for a tool to analyse the large amount of historic video footage without the need for human supervision. This task was split into two parts: the first being a review of the literature of the current technologies and the second being a case study incorporating some of the more promising candidates for this problem space.

There is considerable immediate potential to make use of such methods for the automated detection and identification of fish and other fauna in underwater video material collected in camera boxes attached to open ended trawl nets; during monitoring at tidal turbines and at wind turbine bases; and at video based fish counters and in the video validation of fish counters using other technology. It is hoped that such methods may well allow effective progress to be made with archived video material which various parties hold which is waiting for review.

### Introduction to computer vision

The field of computer vision is wide with many researchers trying to tackle similar problems in various fields. Most approaches take the form of creating ‘features’, such as edges or corners, and mapping them together to form objects. A number of different algorithms have been proposed to perform this with varying levels of success. However, current trends involve applying a form of artificial neural network (ANN) algorithm to extract and compare features automatically.

Artificial neural networks are powerful statistical models for estimation and classification for many different data types and uses. Convolutional Neural Networks (CNNs) are a type of ANN, which can be used to identify the contents of images by applying multiple convolution stages to extract out features. These CNNs can then be trained to detect objects such as fish or other aquatic life.

## Review of relevant approaches

The literature for this specific problem space follows the same trends as the wider field of computer vision. Previous works have focused on employing feature detection algorithms to identify aquatic life but current trends adopt CNN based algorithms. There are various examples and platforms where CNNs have successfully identified subjects in a range of video environments. Certain platforms, such as BIIGLE and CoralNet, adopt CNN algorithms to assist in annotating image and video data by prompting users with potential regions or classifications. Similarly, a platform called VIAME hosts a number of algorithms in a single location with an interactive interface to enable researchers to adopt these methods. Currently, these platforms offer potential but are still under development so they might not quite yet be fit for purpose.

## Case study scope

A comparison of models was subsequently performed in the case study. This case study employed mainly open-source algorithms to minimise the time spent in algorithm development, with the OpenCV (open-source) and Google Object Detection API (open-source based in Google TensorFlow) libraries chosen to develop end-to-end pipelines for training and executing models. Background subtraction algorithms were adopted from OpenCV to determine whether each frame contained any potential regions to investigate. Subsequently, various feature detectors were applied to establish whether the region contained an object of interest. If a certain number of frames were tagged in succession, then they were flagged as a positive identification. Additionally, a third-party algorithm, DeepSORT (also open-source), was incorporated to track objects frame by frame to determine whether the video subjects could be counted automatically.

The OpenCV library offered three feature detectors to incorporate into the pipeline, SIFT (Scale-Invariant Feature Transform), SURF (Speeded Up Robust Features) and ORB (Orientated FAST and Rotated BRIEF). These had to be wrapped into a method to convert the returned feature descriptions into object detections. The BoVW (Bag of Visual Words) method was chosen whereby the feature descriptions were converted into histograms by summing together similar appearing features. This histogram could then be compared to histograms created by training data to determine whether the ROI contained the object of interest.

Three CNN models were chosen from the Google Object Detection API library for use in this case study. These were FRCNN (Faster Region-based Convolutional Neural Network), SSD (Single Shot Detector) and RFCN (Region-based Fully Convolutional Network). The FRCNN and RFCN algorithms are generally considered more accurate whilst the SSD algorithm has a much faster processing time. The Google Object Detection API already contained methods for training and applying these models with custom data so minimal effort was required to include them into the working pipeline.

**Evaluation on real data scenarios**

Video samples were extracted from available data to test the various trained models. These included samples of Atlantic salmon smolts (*Salmo salar*) and European sprats (*Sprattus sprattus*) passing through a camera box attached to a trawl net, a mobile sea-bed rig looking at phosphorescent sea pens (*Pennatula phosphorea)*, and a mobile sea-bed rig looking at Norwegian lobster (*Nephrops norvegicus,* termed Nephrops) burrows. This demonstrated the capabilities of the models to handle a variety of inputs and test the different aspects of the pipeline.

In all cases, the CNN based algorithms outperformed the BoVW based algorithms in terms of accuracy. For fish counting, the CNN algorithms returned zero false positives when tagging sets of frames, and the RFCN algorithm was able to detect 95% of smolts passing through. Accuracy was lower when trying to automatically count fish using the DeepSORT tracking algorithm, with the FRCNN algorithm successfully tracking 74% of smolts but also returning multiple instances of double counting. In the classification test for distinguishing between smolts and sprats, the RFCN algorithm was correct in 98% of detections. In the sea pen trial, the RFCN algorithm performed best with 89% correctly detected and tracked instances.

Although, in terms of speed, the SSD algorithm performed best with near real time analysis in some cases, it suffered from poor accuracy compared to the other CNN algorithms. With the adoption of a cloud computing instance to speed up processing, the FRCNN and RFCN algorithms were able to process video in 1.6x-1.8x the video length. The BoVW models tended to be moderately faster unless the frames contained many regions to analyse, but this speed increase did not make up for the lack of reliability present in these models.

Unfortunately, identifying the Nephrops burrows in video frames proved to be too complex to provide robust detections. The combination of blurred footage and very subtle indicators meant that the object detection algorithms were not able to detect entrances or even confidently detect holes in the ground for further processing. Alternative processing steps would be required for detecting this type of object in underwater video.

The speeds and accuracy values quoted here are only indicative of the models used as their performance can vary greatly with the type of data provided and training parameters given. However, it is clear that the CNN algorithms outperform the other feature detection models and should form the basis of any future work. This is also reflected in the literature analysis, which demonstrated a trend towards this type of approach in the field of automated aquatic video analysis.

## Recommendations

In conclusion to this report, a number of actionable recommendations are provided if further development in this field is to be performed:

1. For detection of aquatic fauna in video, the convoluted neural networks far surpass the older algorithms in terms of both accuracy and ease of development.

2. Develop or document a database of images with annotations on aquatic species for training purposes.

3. It would be beneficial to develop a platform, or adopt an existing platform, for aquatic biologists to gain familiarity with the process of training and adopting these types of models.

4. Some of these computer vision models may outperform humans in some tasks and underperform in others. Therefore, a certain level of standardisation of the various elements involved, which might include benchmark tests and standard data libraries, will be required to compare the performance of different available processes.

## List of names and acronyms

| | |
|---|---|
| *AdaBoost* | *Adaptive Boosting* - A machine learning classifying technique that combines many statistical models with a weighted sum to produce a stronger statistical model. |
| *AlexNet* | A CNN that won the 2012 ILSVRC by a large margin, which some consider to be a turning point in the field of computer vision to adopt more CNN methods. |
| *ANN* | *Artificial Neural Network* - A statistical analysis tool that is built up of nodes and trainable connections which resembles neural networks of the human brain. |
| *AWS* | *Amazon Web Services* - A cloud computing platform provided by Amazon. |
| *BIIGLE* | A web service for the efficient and rapid annotation of still images and videos. |
| *BoVW* | *Bag of Visual Words* - A method for classifying images based on a Bag of Words classifying model. |
| *BRIEF* | *Binary Robust Independent Elementary Features* - A feature descriptor standard that improves speed of matching features by reading straight from binaries. |
| *CATAMI* | *Collaborative and Automated Tools for Analysis of Marine Imagery* - A classification scheme for scoring marine biota and substrata in underwater imagery. Also has a tool to assist annotations. |
| *Computer Vision* | A field of study associated with the automated interpretation and extraction of information from images and videos. |
| *Convolution* | A mathematical term for the combination of two functions to form a third which represents the shape of one modified by the other. In terms of CNN, the convolution stage is represented by a pixel filter on images to derive spatial relationships between pixels. |
| *CoralNet* | A web service by the same creates as VIAME for organising and annotating images of coral. It is still under development. |
| *CPU* | *Central Processing Unit* - The main processor in computers. This favours performing complex calculations in series (as opposed to GPUs that favour many simple calculations in parallel). |
| *CNN* | *Convolutional Neural Network* - A type of ANN that processes images through numerous convolutional stages before a classification stage. |
| *Darknet* | A bespoke neural network creation platform from the creators of the YOLO CNN. |
| *DeepSORT* | *Simple Online and Realtime Tracking with a Deep Association Metric* - A tracking algorithm that combines a variety of tracking algorithms together with a neural network framework. |
| *DrivenData* | A website that hosts machine learning and data analysis competitions. Hosted the “N+1 Fish, N+2 Fish” classification competition. |
| *FAST* | *Features from Accelerated Segment Test* - A corner feature detection algorithm with notably increased speeds over other methods such as |

| | |
|---|---|
| | SIFT or SURF. |
| *Feature* | In the field of Computer Vision, a mathematical description of part of an image such as an edge or a corner that can be used to build profiles of objects and content present in the image. |
| *Feature Map* | A combination of different features that together build a profile of a specific object or classification. |
| *FFMPEG* | *Fast Forward Moving Picture Experts Group* - An open source video manipulation software toolkit. |
| *FishTick* | A piece of software developed by Salmonsoft for analysing and counting fish in dams and weirs. |
| *FRCNN* | *Faster Region-based Convolutional Neural Network* - An object detection CNN method. This algorithm has efficient region proposals allowing for highly accurate detections in large images. |
| *Gabor Filters* | Linear filters used in image processing to determine textures. |
| *Gaussian Filter* | A convolution step performed using Gaussian functions. |
| *Gaussian Function* | A function that forms a characteristic bell curve shape. |
| *Google Object Detection API* | An open source repository of object detection training and testing algorithms built with TensorFlow. |
| *GPU* | *Graphical Processing Unit* - An optional processor in computers, generally used for graphical processes. This favours performing many simple calculations in parallel (as opposed to CPUs that favour complex calculations in series). |
| *Haar features* | Filters used in image processing to determine whether pixels fall into particular spatial relationships. |
| *Haar wavelets* | A signal wavelet built up of square shaped oscillations. Can be used as filters to extract information from pixels. |
| *HOG* | *Histogram of Orientated Gradients* - A feature descriptor in image processing that measures the sum of various pixel gradients over an image or ROI. |
| *ILSVRC* | *ImageNet Large Scale Visual Recognition Challenge* - A competition for object detection algorithms to determine their effectiveness on a wide variety of objects. |
| *JPEG* | *Joint Photographic Experts Group* - A format for saving image files. |
| *Kaggle* | A website that hosts machine learning and data analysis competitions. Hosted the “The Nature Conservancy Fisheries Monitoring” classification competition. |
| *Kalman Filters* | A filter that estimates the internal state of a dynamic system from a series of measurements. Used for predicting motion of objects in object tracking. |
| *KNN* | *K-Nearest Neighbours* - A machine learning clustering method that describes data in terms of its closest distance to existing cluster centres. |
| *Marine* | A Scottish governmental department responsible for growing |

| | |
|---|---|
| *Scotland* | Scotland’s marine assets and protecting marine ecosystems. |
| *Marine Scotland Science* | A division of Marine Scotland focused on providing expert scientific, economic and technical advice and services on issues relating to marine aquaculture, renewable energy and environment. |
| *MarynSol* | The company contracted by Marine Scotland to perform this study. MarynSol’s business involves providing marine data analytics, advanced marine survey, and marine data database and warehousing. |
| *MOG* | *Mixture of Gaussians* - A machine learning clustering method that describes data as a combination of various Gaussian profiles. |
| *ORB* | *Orientated FAST and Rotated BRIEF* - A feature detection algorithm created by OpenCV as an alternative to SIFT and SURF. |
| *OpenCV* | *Open Computer Vision* - An open-source library that provides many computer vision tools and algorithms. |
| *OpenCV KNN* | A background subtraction method in OpenCV that applies KNN classification to pixels. |
| *OpenCV MOG2* | A background subtraction method in OpenCV that utilises a MOG method. |
| *Pooling* | The act of combining values into a single representative value (e.g. through averaging or taking a maximum value). |
| *Random Forest* | A machine learning classification tool that takes the weighted output of many different decision trees. |
| *Rekognition* | A platform provided by Amazon for video analysis using a selection of pre-built models such as object detection and facial recognition. |
| *RFCN* | *Region-based Fully Convoluted Network* - An object detection CNN method. Uses an alternative region selection method to FRCNN by splitting the ROI into a 3x3 box. Has comparable speeds and accuracies to FRCNN. |
| *ROI* | *Region Of Interest* - A subsection of an image that has been flagged for further analysis. |
| *Salmonsoft* | The company that produced FishTick software. |
| *SIFT* | *Scale-Invariant Feature Transform* - A feature detector that uses maxima and minima from a difference of Guassians function. |
| *Squidle* | A tool for managing, exploring and annotating images, video and large-scale mosaics. |
| *SSD* | *Single Shot Detector* - An object detection CNN method. This algorithm values speed over accuracy and is used in many real-time image processing tools. |
| *SURF* | *Speeded Up Robust Features* - A feature detector based on the sum of the Haar wavelet response around blob detections. |
| *TensorBoard* | A visualisation tool for TensorFlow that displays training and testing metrics. |
| *TensorFlow* | An open source platform for ANN development and training provided by Google. |
| *VIAME* | *Video and Image Analytics for a Marine Environment* - An open-source system for analysis of underwater video and imagery which |

| | |
|---|---|
| | pulls together a variety of different image analysis tools and algorithms into one place. |
| *Viola-Jones framework* | An object detection framework built using Haar features and AdaBoost. One of the earliest successful forms of facial recognition. |
| *YADIF* | *Yet Another DeInterlacing Filter* - A filter to remove the interlacing effect in some video formats. |
| *YOLO* | *You Only Look Once* - A fast and accurate CNN built using the Darknet platform. A very popular and widely used algorithm. It is included in the VIAME software. |

## Introduction

Scottish Government are committed to studying and maintaining the environment and ecology of coastal waters around Scotland. Monitoring underwater habitats, such as those around renewable energy and aquaculture sites, is essential in ensuring the preservation of marine life. Unfortunately, these sites tend to be remote and inaccessible, limiting the availability of data to analyse them.

Underwater video cameras offer a relatively simple and cheap method to rapidly collect substantial amounts of data across a wide range of locations for study. Unfortunately, survey campaigns can generate hundreds of hours of footage which require distilling into manageable datasets. Manual processing of this footage is slow, tedious, and is limiting in the amount of imagery that can practically be processed. Automated and semi-automated analysis algorithms for these videos have been around for many years in order to extract information, such as the number or the type of fish present. However, these methods tend to have varying degrees of success when installed outside of controlled laboratory settings.

Furthermore, capturing high quality video for analysis can be difficult in underwater environments. The devices can be affected by poor lighting due to the turbidity of the water. Lenses can become fouled over time due to biofouling or deposition of sediment, obstructing the view. Additionally, rapid motion of fish in the shot can produce blurred effects, which further compounds the difficulties with automated identification of subjects present. There also exist many algorithms that try and cope with these limitations but, as before, their effectiveness can be variable in real-world environments.

However, image analysis technologies have been revolutionised recently with advancements in machine learning and neural networks. This, in combination with established image processing analysis techniques has the potential to have a powerful impact in this problem space.

There is considerable immediate potential to make use of such methods for the automated detection and identification of fish and other fauna in underwater video material collected in camera boxes attached to open ended trawl nets; during monitoring at tidal turbines and at wind turbine bases; and at video based fish counters and in the video validation of fish counters using other technology. Such methods may well allow effective progress to be made with archived video material which various parties hold which is waiting for review.

In this project, Marine Scotland tasked MarynSol to deliver an overview of the current technologies and methods for fauna/object identification in underwater video, alongside the current developing object detection technologies developed within recent years. Additionally, a test-bed of some of the more promising technologies will be presented, using video provided, to assess their potential impact on future research and development.

## Objective

The objective of this study is to present the current state of technologies for automatic detection of aquatic life in underwater video footage, the purpose being to extract relevant data from these sources without resorting to the large amount of expert man-hours required to sift through the footage manually. This data could then be applied immediately to address knowledge gaps in aquatic ecological monitoring, such as biodiversity tracking or assessing the environmental impacts of aquaculture farms and renewable energy infrastructure.

This review will present a general overview of the current technologies available commercially or open source that directly relate to the problem space (i.e. current software to aid in aquatic fauna/flora detection) and the current direction of technologies and algorithms in the field of video object detection in general. These technologies will be investigated in a subsequent case study to determine the feasibility of adopting them into the current problem space.

### Problem space

Although the task of recognising objects in images and video seems relatively straight-forward for humans to perform, replicating this feat in computers is not so easily done. Even though it is second nature for humans and animals to see and interpret the world around them, the actual underlying process that is performed is not well understood. Therefore, translating it into programming scripts and algorithms for computers to execute is challenging.

In breaking down the task of analysing aquatic life in video, there are varying levels of information that could be extracted which involve different algorithms and approaches. For instance, if taking an example where the objects of interest are fish, then the tasks could be summed up as follows:

- Where are the fish?
  - This involves detecting whether or not the image contains fish and where they are in the image.

- What type of fish are they?
  - Once the fish has been found, further analysis is required to determine what type of fish it is, based on the visible features.
  - This step usually requires expert knowledge and experience which can sometimes be difficult to translate into an algorithm.

- How many fish?
  - To count fish in video, each fish needs to be followed between frames in order to prevent double-counting.
  - This can quickly become complicated as fish may move erratically and overlap one another, making tracking difficult for even human observers.

- How big are the fish?
  - In some cases, fish length can be estimated by comparing the snout-to-tail pixel distance to a known length in the image. For automatic measurements, this process requires calibration and also detection of fish orientation and features.
  - Alternatively, stereo-camera systems can use perspective to estimate length, but the fish detected need to be identified in two frames simultaneously.

These four tasks represent classic computer vision problems that have been studied over the last 50 years. In that time, various approaches and advances have been put forward, but these problems still remain the focus of a wide array of research institutions today.

Furthermore, underwater video also presents additional difficulties to the traditional task of object detection due to the environment in which it is obtained. These include:

- Low light level.
  - Light penetration of water decreases at lower depths. Without alternative light sources, video taken at these levels can be dark with low contrast between background and objects of interest. Even when additional artificial light is used, the scene illumination is often variable, and optical backscatter effects can be significant.

- Colour faded.
    - In the absence of an additional light source, the light that does reach low depths of high transparency water tends to be blue-shifted, giving everything a greenish-blue tinge. Additionally, materials dissolved or suspended in the water can absorb other wavelengths of light causing variations of colour in salt and freshwater sources. The lack of colour variation, other than in shallow high transparency water or with additional lighting, represents a loss of information in the video and can make object identification harder.

- Blurred elements from rapid motion.
    - High quality video capture underwater is often difficult to obtain. Lighting levels and standard video shuttering rates can cause motion blur, and other effects. Both the camera and subjects can be buffeted by tidal flows or move erratically causing motion blur in obtained footage. This can conceal features making identification of objects more difficult.

- Bio-fouling on lens in installed cameras.
    - If cameras are mounted on a permanent or semi-permanent fixture for extended periods of time, the video imagery can be subject to bio-fouling as marine growth forms on the lens. This obscures any potential objects and renders detection and identification much more difficult.

From this, it can be seen that this problem space is highly complex with many different challenges to tackle. There is not currently a stand-alone algorithm that can be used to solve all of the above. However, advances in computer vision have already addressed parts of these problems which will be demonstrated in the following sections.

## Computer vision

### Introduction

‘Computer vision’ is a highly interdisciplinary field of study that affects many areas of research. Extracting information from images automatically allows for automation in tasks that previously were thought to be only possible by a human controller. From reading the pieces of a chessboard to allowing a self-driving car to navigate populated roads, computer vision can be useful in almost any field[1].

For this task, the objective is to have a computer automatically review what may be thousands of hours of footage to count, annotate, and classify objects of interest. Therefore, the ideal algorithm must be able to first detect an object of interest in a frame, and subsequently track the same object in following frames to avoid double-counting. Alternatively, a simpler algorithm could just classify whether particular sequences of frames in a video probably contains “content of interest”, and flag for further human inspection. By discarding irrelevant frames, the workload of the human inspector could be dramatically reduced. However, even the latter of these options is not a straightforward task. A computer does not necessarily ‘see’ an image in the same way humans do. Images are stored in computers as arrays of pixel values, usually with red, green, blue (and sometimes alpha or transparency) channels. Somehow, these values and the relationship between them need to be translated into lines and shapes which then need to be further translated into descriptions of objects to recognise such as animals or plants[1].

Furthermore, computers do not automatically infer context in the same way humans do. For example, observing only the head of a fish in an image would usually be enough for a human to perform an identification. The human observer would infer the remaining part of the fish that is obscured or out of frame. Training a computer to accomplish the same task would require imparting some form of knowledge and understanding of what fundamentally constitutes a fish and then performing the same deduction through a programming script.

There are many approaches to automating detection in images and video, but the overall task can be generally split into three parts[2]:

- Object detection: Given an image, or a frame in a video, is there an area of interest in the image and if so, what part of an image does it occupy?

- Object classification: Given an image/frame, or a subsection of that image/frame, what object (if any) is present?

- Object tracking: Given an object has been found in an image/frame, can the same object be recognised in a subsequent image/frame?

There exist numerous algorithms that have been proposed over the years that tackle one or more of the above tasks with varying degrees of success and with varying ranges of applicability.

## Feature maps

The fundamental backbone behind computer vision algorithms is that objects present in images can be summarised and described in the form of ‘feature maps’. A ‘feature’, in the context of object detection, is a mathematical description of a part of the image that can range from being as simple as a prominent edge or corner, to a complex convolution process of a set of neighbouring pixels. Different algorithms adopt different methods for obtaining features, meaning a feature is not necessarily universal. Some algorithms look for local changes in raw pixel values while others apply mathematical filters to the image to generate new relationships between these pixel values. The ‘map’ represents the characteristic features and their spatial relationship that construct an object. For example, a rectangle is generated by two perpendicular sets of parallel lines. The features in this case would be the four separate lines and the map would be the amount of lines and their corresponding parallel or perpendicular nature. Thresholds can then be applied to determine how confident the algorithm is that there is a rectangle present.

It is possible to manually construct feature maps to describe objects, although this rapidly becomes a complex and awkward process as any manner of object orientations and scales need to be incorporated into the map. Instead, the usual practice is to allow a feature detector algorithm to extract prominent features from a set of training examples and derive its own map. If the detector subsequently encounters a similar set and orientation of features on a test image it can assume it is the same object.

## Algorithms

There are too many algorithms that have been created over the last few decades in the field of computer vision to investigate and comment on them all individually. Therefore, only the most prominent and successful ones will be presented here. Additionally, this report will limit the description of mathematical detail behind the algorithms to only necessary information, with references provided for readers who are interested in the mathematics that constitute these algorithms. If a more in depth look at the recent history of computer vision is desired, Zhang *et al.* provide a comprehensive review of the major techniques used in image classification[3].
A list of the more prominent object detection, or feature detection, algorithms are described below:

Scale-Invariant Feature Transform (SIFT)[4] was considered to be the best computer vision algorithm in the early 2000s. It creates features by detecting maxima and minima from a difference of Guassians function. It generally performs the best in benchmark tests against other algorithms of a similar age[5]. However, the algorithm is patented in the United States, limiting its availability for commercial use.

Viola–Jones object detection framework[6] is one of the earliest frameworks to provide consistent, real-time detections. It combines Haar features with AdaBoost training and classification to create a pyramid shaped detection set up whereby features are ranked by importance. This rapidly eliminates potential regions that are not similar to the object in question allowing for swift detections to be performed.

Speeded Up Robust Features (SURF)[7] is a faster alternative to SIFT that uses a different set of feature descriptors based on the sum of the Haar wavelet responses around a specific blob detection (a region of varying contrast or colour). It also benefits from utilising the precomputed integral image to speed up the detection of features. However, like SIFT, this method is patented in the United States.

Gabor Filters[8] look for patterns within an image by passing 2-Dimensional Gaussian filters and observe regions that have similar frequency responses to known patterns. This method works well for texture analysis and pattern recognition for object classification.

Histogram of Oriented Gradients (HOG)[9] is a method that compares colour and contrast gradient slopes and angles over an image. Features are derived from a histogram of these gradients that can be compared to expected histograms from known objects.

Oriented FAST and Rotated BRIEF (ORB)[10] is a method developed by the OpenCV (Open Computer Vision) library as a substitute for SIFT and SURF as they are patented methods. This method uses the FAST (Features from Accelerated Segment Test) algorithm to determine keypoints and then BRIEF (Binary Robust Independent Elementary Features) algorithm to create feature descriptors. Both of these algorithms have been modified to improve accuracy in rotated objects.

### Neural networks

Until the early 2010s, all advances in computer vision relied on some modification, or incremental advancement, to one of the techniques in the previous section. Nowadays, the field has pivoted and almost all computer vision algorithms rely on neural networks instead.

## Artificial neural networks

Artificial neural networks (ANNs) are powerful pattern recognition algorithms that are loosely based on the architecture of a human brain. Various layers of nodes, or ‘neurons’, are interconnected by weighting functions, shown in Figure 1. Each node takes a weighted input of all outputs from the previous layer and then returns a value between zero and one based on a sigmoid (or similar) function. This value is then passed onto every node on the next layer and so on until it reaches the output layer. Here the value can be converted into a class probability, for classification problems, or to a value via an external function, for regression problems.

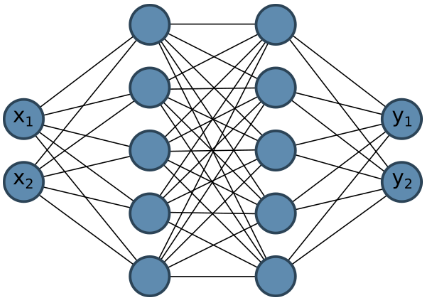


**Figure 1:** An example of a small, two-layer neural network with two input values (x1, x2) and two output values (y1, y2). Every black line (45 in total) corresponds to a connection with a trainable weighted value.

However, for these networks to work, the weighting values for each neuron connection have to be determined. This is achieved through a training process whereby known inputs are plugged into the network and the outputs are compared to the expected outputs. A correction function, called backpropagation, is then applied to all the weights in the network to adjust them slightly. This process is repeated numerous times until the outputs match the expected ones.

The concept of ANNs has been around for decades, but the technological requirements to train and apply them effectively were beyond the capabilities of most computers until the late 2000s. Since then, almost all machine learning tasks incorporate some form of neural network, yet there is no single architecture (i.e. number and size of layers) that applies to every task, and researchers are constantly developing new methods and orientations.

## Convolutional neural networks

Convolutional neural networks (CNNs) are a particular branch of ANNs focused on deriving features and patterns from an array like input, such as an image. Their basic structure consists of one or more convolution and pooling steps feeding into a layered ANN, as shown in Figure 2. These convolution stages allow the network to extract arbitrary features from an image which can then be used for training an ANN to classify as objects. However, each sample of pixels creates another connection that requires a weighted value. Therefore, covering an entire image makes training highly computationally expensive due to the size of the resultant network.

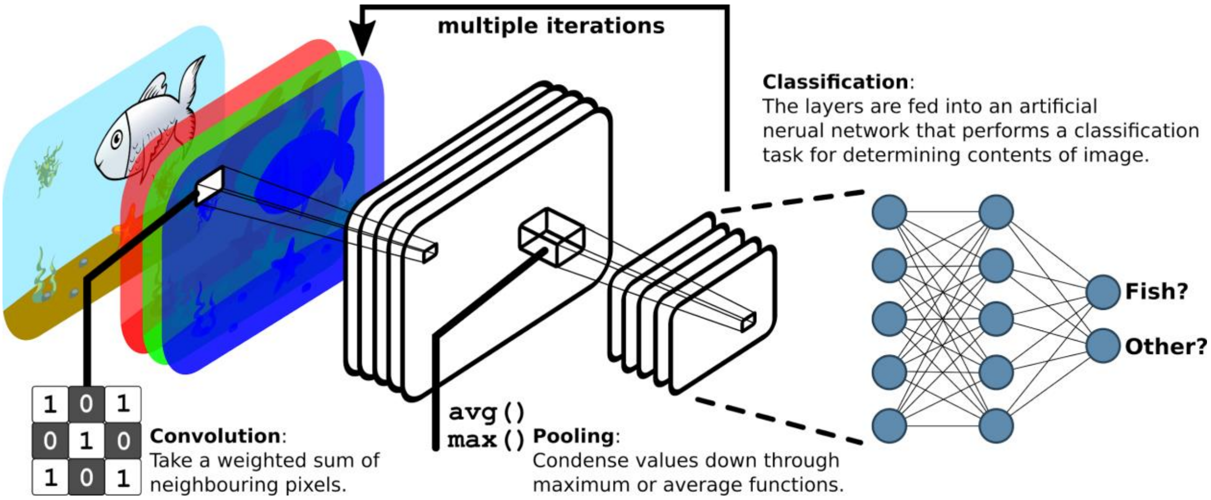

multiple iterations
Classification:
The layers are fed into an artificial nerual network that performs a classification task for determining contents of image.
Fish?
Other?
1 0 1
0 1 0
1 0 1
Convolution:
Take a weighted sum of neighbouring pixels.
avg()
max()
Pooling:
Condense values down through maximum or average functions.

**Figure 2:** An example of the architecture of a convolutional neural network. The convolutional stage takes the weighted sum of neighbouring pixels in different configurations and forms new ‘image’ layers from the results. The pooling layer then condenses these layers into fewer ‘image’ layers through averaging or taking the maximum values. This process can then be repeated multiple times before flattening the image into inputs for a classification ANN.

Prior to the adoption of CNNs in computer vision, most solutions were built on variations of SIFT, SURF and other presented in the previous section. Regular competitions were held to showcase these cutting-edge algorithms by attempting to identify a variety of objects in thousands of pictures. One of these was the ImageNet Large Scale Visual Recognition Challenge (ILSVRC)[11]. In the 2012 ILSVRC, a convolutional neural network called AlexNet[12] beat the runner-up algorithm by an unprecedented ten percentage points and was the first team ever to score above 75% accuracy. (AlexNet achieved 84.7% precision)[13]. This sparked a paradigm shift as research focus in the field shifted almost entirely towards CNNs, yielding dramatic improvements in accuracy over previous algorithms. In the 2017 ILSVRC, all teams incorporated some form of CNN into their methodology and 29 out of 38 teams scored above 95% precision[14].

As before with the ANNs, there is no single architecture that provides a solution and researchers are continually developing new ones. The complexity of these models grows exponentially with researchers combining different types of network together to form new ones. With that, the computational load to train and execute these networks increases too. Today, researchers rely on the increased computational output provided by Graphical Processing Units (GPUs) as compared to traditional Computational Processing Units (CPUs). GPUs are optimised to perform many small calculations in parallel (such as calculating pixel values for faster graphical rendering) whereas CPUs are designed to perform more complex calculations in series. As the updating of weight functions in neural networks is a relatively cheap task computationally, the training process can be sped up by running it on one or more GPUs.

### Drawbacks to CNNs

Although proven to be very powerful in the field of object detection, there are a couple of drawbacks to CNNs as compared to the simpler non-CNN based algorithms. Primarily, the resultant trained model becomes a black-box processor, meaning it is very difficult to interrogate the model as to why it has provided a certain output. This makes debugging the system next to impossible. Usual practice is to generate many models with a variety of parameter inputs and then select the one that returns the best results.

The correct architecture for a CNN is a topic of considerable academic research and there is no consistent correct answer, only certain architectures that have been shown to perform well in certain tasks. Luckily, these can usually be taken and adapted for other tasks. Sometimes, solutions require a combination of various different CNN architectures to work in collaboration. Similarly, model performance is reliant on data provided, but exactly how much and what quality that data should be is unknown. A general rule of thumb is “more data is better”, but the benefits from increased data plateau after a certain amount.

Development of a CNN solution tends to involve iterative experimentation to determine what available technologies and algorithms are suited to the task at hand. Once a model has reached an adequate level of accuracy, it is deemed acceptable, even if the underlying reasons for why it is performing well remains unknown. This overall process can sometimes be considered more of an art rather than a science.

## Computer vision in underwater video

### Introduction

With a wealth of video being produced from various marine environmental and aquaculture practices, a number of attempts to incorporate automated detection of aquatic life has been performed over the years.

In particular, fish monitoring and counting are popular focuses for automation tasks. Examples include fish aquaculture industries looking to improve data acquisition by incorporating machine learning algorithms into their monitoring techniques[15, 16]. Jovanović *et al.* demonstrate how to automatically monitor for splashing in fish pens using surface cameras[17]. Williams *et al.* were able to develop an automatic detection and measurement algorithm for fish in a trawl from stereo cameras[18]. However, tracking of individual fish still had to be performed manually.

A report of the 2010 National Marine Fisheries Service Automated Image Processing Workshop[19] indicated that the direction of computer vision in underwater images followed the same trends of the time. Most algorithms focused on the current feature extraction methodologies, such as SIFT or shape contours, which struggled with complex, non-uniform backgrounds. The workshop mainly focused on fish identification and classification in images, with only two of the eight presentations from invited experts discussing identifying other marine life.

### Algorithms used in literature

An overview of trends in fish detection as of 2013 is given by Shortis *et al.*[20]. Even though it focuses on stereo-video footage, many of the applied techniques would work on single-camera video as well. It summarises three techniques by Spampinato *et al.*[21], Khanfar *et al.*[22], and Charalampidis *et al.*[23] and then proposes a general model architecture based on the more promising techniques. A similar methodology is incorporated by Westling et al.[24] to measure fish with a stereo camera and Lantsova *et al.*[25] for identifying fish in low quality video.

In the papers mentioned above, the typical workflow to identify fish in underwater video comprises of the following steps:

1. Establish a background by averaging or processing frames without any fish present.

2. Detect regions where a fish might be, generally by using a comparison with the background frame among other techniques.

3. Check if region contains a fish through a combination of feature comparisons colour histograms and/or contour fittings.

4. Check if two or more fish occupy the same region in frame (this step is highly complex and difficult to perform rigorously, therefore it is usually omitted).

5. Classify the type of fish using further feature comparisons against a database of fish of interest.

6. Measure fish if stereo camera is in use by comparing lengths between head and tails. Sometimes multiple measurements are averaged due to the contortion of fish as it swims.

7. Track fish in subsequent frames to prevent multiple counting by predictive positioning, usually by incorporating Kalman filters.

It can be seen that these workflows follow the general Detection/Classification/Tracking framework incorporated by most computer vision algorithms. The feature comparisons are generally performed with a Viola-Jones object detection framework[26].

Image enhancements, such as colour balancing[27, 28], tend not to be incorporated as a step in these models. These enhancements generally only benefit the human interpretation of the image. Background subtraction and feature detection algorithms generally work by comparing the relationship between pixels. These relationships are generally preserved in the context of colour histogram stretching[29] and brightness/contrast enhancement[30]. Similarly, two frames processed using the same enhancements should display the same foreground/background appearances, meaning detecting changes for background removal should be the same regardless of whether it has been enhanced or not. Unless the enhancement applies some complex, conditional, non-linear filter, they are considered unnecessary in the context of computer vision.

### Convolutional neural networks

In recent years CNNs have also been applied to the task of detection[31, 32] and classification[33] of marine life with promising outcomes. Modern object detection algorithms allow for both regional detection and simultaneous classification within the same architecture[34, 35], which would allow for both identification and counting of aquatic life in underwater video. Initial forms of this have been attempted by tracking detections in subsequent video frames using SURF features[36] and CNN features[37]. However, the results tracking metrics are not presented clearly so it is unclear how well these perform.

Many software applications have started to incorporate CNNs and machine learning into their workflow to facilitate the process of detecting and annotating aquatic life in images and video. Annotation software such as BIIGLE, CATAMI, and Squidle incorporate some form of automated object identification or classification[38]. However, these applications are primarily designed for images and not video.

The automated processes in CATAMI and Squidle focus on classification of images or other media via suggestions, but actual annotation has to be performed manually. They also serve as a repository for storing data and allowing further collaboration with other researchers. BIIGLE contains a CNN to detect potential objects of interest automatically in the image set provided. Rather than classifying the image, this method actually detects and bounds the objects as well. Refinement and confirmation of the detection also has to be performed manually. CoralNet is another web-based software for classifying images of coral reefs and other biological growths[39], which also has an inbuilt CNN classifier to assist annotation of images. This was shown to be effective in assisting with assessing biodiversity of offshore pipelines in the North Sea[40].

FishTick is a software system built by Salmonsoft which is used for fish counting in a variety of locations. The current version of the software only allows for automated tagging frames to be subsequently reviewed by an operator to count the fish manually. In 2016, they developed a machine learning feature to automatically detect and measure fish passing through video. Unfortunately, this has not currently been brought forward into development and is not available in their current version (v3). The developers have stated than in their upcoming version 4.0 they will offer an embedded hardware solution to process object detection and tracking, with the end goal being for the system to be able to classify species, detect features and perform fully automated counting.

The developers of CoralNet have also developed an all-inclusive environment for allowing researchers to develop computer vision pipelines for their data, called VIAME (Video and Image Analytics for a Marine Environment). This platform hosts a variety of algorithms that can be used to detect and annotate images and video and also measure subjects. There is a modular framework that allows for customisation of workflows and once models have been trained, the process can be automated. Whilst still under development, VIAME holds a lot of potential for addressing this problem space.

## Competitions

In 2017, two online competitions were held by the US Nature Conservatory concerning the detection and classification of fish in video. The first, called "The Nature Conservancy Fisheries Monitoring", hosted by Kaggle and with a prize pool of US$150,000, aimed to identify what type of fish (out of a selection of six) were present in each frame from the video footage from a variety of boat cameras[41]. The second, called "N+1 fish, N+2 fish", hosted by DrivenData with a prize pool of US$50,000, aimed to identify, count and measure the length of fish discarded by New England trawling vessels[42]. In both competitions, the footage used comprised of a variety of different boats and viewing angles which had to be taken into account by the algorithm.

The latter of these competitions, "N+1 fish, N+2 fish", went on to publish the five winning entries in an open source format[43], allowing an insight into the current top-of-the-range algorithms used for these types of tasks. All five of these entries incorporated CNNs into their algorithms, although their approaches to the task varied greatly. For instance, the winning entry first implemented a CNN to identify the ruler used to measure fish in the frame, and then subsequently used another CNN to identify and measure fish on images reoriented about this ruler. The second-place entry, however, searched for the fish directly in each frame, and then used subsequent CNNs to extract information on a cropped augmented image of the fish. Additionally, the architecture of the CNNs themselves varied between entries, demonstrating that there is not necessarily a singular approach to this problem domain. The second-place entry included a flow chart of their methodology, shown in Figure 3. This entry incorporated twenty neural networks of four different types into their final algorithm for detecting, counting and classifying fish.

Accuracies for the winning entry were given as follows:

*"The winning algorithm alone achieved above 90% identification accuracy across 5 of 7 species, and 99% accuracy for three species. Meanwhile, predicted counts were within just 1 fish – on an average of 44 fish per video – 83% of the time, and the average error in length was under 2%."*

This demonstrates that these methods, although intricate and powerful, are not currently 100% accurate, especially for tasks that involve a large range of parameters. In controlled environments, such as identifying and counting fish on a conveyor belt, simpler algorithms can yield >95% precision[44]. However, these tend to be very task specific and lack the versatility offered by the competition algorithms that had to deal with varying camera locations and lighting effects.

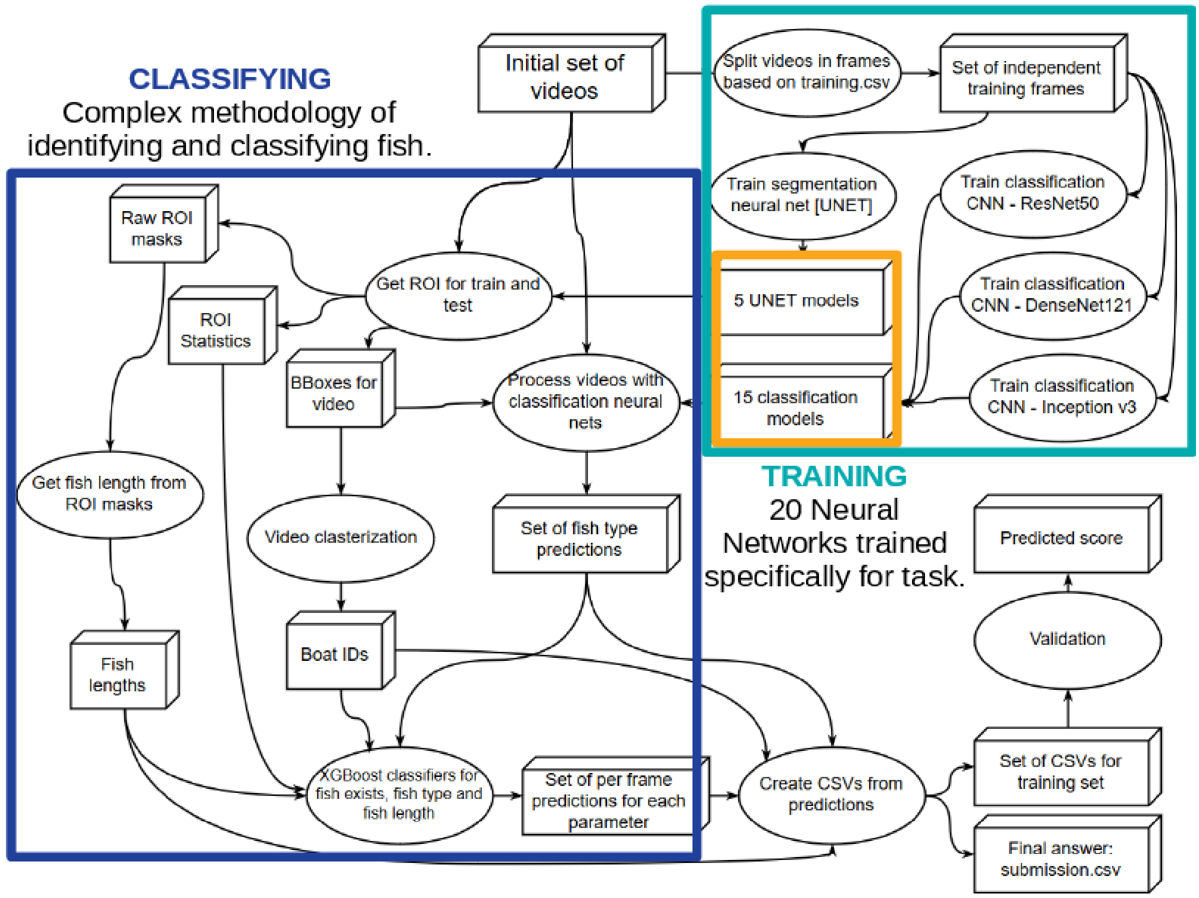

CLASSIFYING
Complex methodology of identifying and classifying fish.
Initial set of videos
Split videos in frames based on training.csv
Set of independent training frames
Train segmentation neural net [UNET]
Train classification CNN - ResNet50
5 UNET models
Train classification CNN - DenseNet121
15 classification models
Train classification CNN - Inception v3
TRAINING
20 Neural Networks trained specifically for task.
Raw ROI masks
ROI Statistics
Get ROI for train and test
BBoxes for video
Process videos with classification neural nets
Get fish length from ROI masks
Video clasterization
Set of fish type predictions
Predicted score
Validation
Fish lengths
Boat IDs
XGBoost classifiers for fish exists, fish type and fish length
Set of per frame predictions for each parameter
Create CSVs from predictions
Set of CSVs for training set
Final answer: submission.csv

**Figure 3:** A flow chart describing the methodology for the second place entry in the "N+1 Fish, N+2 Fish". This methodology incorporated 20 CNNs for identifying and classifying fish in the videos. Classifying and Training coloured boxes added by the current author. Original image taken from https://github.com/drivendataorg/n-plus-one-fish/tree/master/2nd-place.

## Review of models for case study

### Introduction

In addition to presenting the current technologies available for this problem space, the objective of the case study is to demonstrate the available algorithms in a working environment. Although research publications present methods that are successful in detecting fish in underwater video, it is rare to find them available in an open-source or in a ready-to-use format. Additionally, the results they present tend to be specific for the environment in question and do not necessarily represent a general-purpose solution. Therefore, this case study will attempt to produce a methodology that is accessible and can be recycled for further use by aquatic researchers.

To make it accessible, and due to the short timeframe of the project, it was decided that focus would be put on currently available open-source solutions to allow the rapid generation of simple working end-to-end pipelines upon which tests can be run. These pipelines are relatively simple compared to competition winning methods, shown in Figure 3, that are highly specialised to a single task, therefore, they serve to illustrate what can be performed and indicate what technologies are suitable for further development.

### Available platforms not incorporated into case study

There are a number of other open-source platforms available that could have been used in this case study, but these were not trialled, due to limitations in their method or complexities in their acquisition.

#### FishTick

Although widely used, FishTick by Salmonsoft was not included here due to the current version not possessing the ability to automatically count targets. Various sources and video demonstrate its frame tagging capabilities in a narrow, controlled environment but the methodology is not flexible and is only suited for a specific task. Although the developers indicate that new machine learning tools will be incorporated in the future, they were not available for this case study.

#### BIIGLE

BIIGLE is a web-hosted platform that only requires a registration to access. Unfortunately, it does not support hosting data without specific permissions, so that task has to be performed by the user which is not always simple, especially with sensitive data. Once data is hosted and accessed through the BIIGLE interface, annotation is simple and straightforward. Accessing the neural network annotation assistant is also straightforward but the actual processing that is performed is hosted on a separate server. Simple tests of a few images then took multiple hours to return with annotation suggestions, with multiple iterative steps required in between. Although the suggestions were good, the time that would be required to process hundreds of frames of video makes it inadequate for this task.

### VIAME

VIAME appears to be a platform to host all the different pipeline tasks in a single location, with interface tools designed to assist in all forms of annotation and identification. However, the installation of the software was not straightforward and a suitable version of the software could not be obtained. From the tutorial videos, the interface looks highly dense in that there are many required options and selections required to perform tasks such as annotating data and training a CNN. This software is still under development and so these steps may be simplified in the future but it in its current state it does not appear accessible to an untrained user.

### Online service portals

There are various portals online provided by companies like Amazon, Google and Microsoft that allow training and managing object detection and classification networks by supplying images and data alone, extracting out the complexities involved with developing and training the CNNs. However, these were avoided due to the monetary cost associated with training and subsequent analysis of a large number of images when free alternatives exist. The Rekognition tool from Amazon allows for video analysis without prior decomposition into individual frames, but it only allows processing using provided algorithms, such as facial recognition, and not custom trained models. However, the cloud computing platform, Amazon Web Services (AWS), was utilised in the case study to expedite processing times.

### Chosen platforms for case study

This section outlines the various platforms and algorithms that were incorporated into the case study. As mentioned above, the feature detection algorithms chosen for this case study represent those that could be easily incorporated into a working pipeline. For the sake of balance, it was also decided that versions of older algorithms, such as SIFT and SURF, should be included alongside the more modern CNN algorithms. However, it would be an inefficient use of time and resources to attempt to create custom versions of these algorithms so it was necessary to seek out and adopt available, open source implementations.

### Open CV

The OpenCV (Open Computer Vision) library contains a wide variety of processing tools for image manipulation and computer vision, including implementations of SIFT and SURF (however, since these are patented, they are only supported until

OpenCV version 3.4.2.16, which was used in this case study). Due to the open source nature of the library and the Python code interface it offers, OpenCV provided an efficient platform on which to build quick and simple workflows for image processing and object detection. In addition, due to the SIFT and SURF algorithms being patented, OpenCV offers a similar, but open source, variation of these algorithms called ORB (Oriented FAST and Rotated BRIEF). These three feature detection algorithms were chosen to compare against the CNN algorithms.

### Object detection API (TensorFlow)

For the CNN implementation, TensorFlow by Google offered a wide range of capabilities for developing machine learning models and pipelines. Additionally, researchers have created an Object Detection API[45] to facilitate rapid production and prototyping of certain models with many examples to choose from, making it easy to implement into our working pipeline. This is similar to the approach performed by Piechaud *et al.*[35]. Using TensorFlow as a platform, three types of CNN were chosen to take forward to the case study. The first, Faster Region-based Convolutional Neural Networks (FRCNN)[46], is considered to be one of the more accurate networks available but tends to have a slower performance speed. The second, Single Shot Detector (SSD)[47], is a very fast network, used mainly for detecting objects in live video, but suffers from lower accuracy. The third, RFCN (Region-based Fully Convolutional Networks)[48], claims improved accuracy from FRCNN with comparable performance speed.

The Object Detection API hosts a number of these models (known as a model ‘zoo’) from which we can download pre-constructed versions of these models. There are various versions of each model type to choose from, so versions that represent a balance between speed and accuracy were chosen. The specific models chosen are listed in Table 1.

## Table 1

The names of the CNN models chosen from the Google Object Detection API model zoo found at https://github.com/tensorflow/models/blob/master/research/object_detection/g3doc/detection_model_zoo.md.

| Model | Object Detection API Name |
|---|---|
| FRCNN | faster_rcnn_inception_v2_coco |
| SSD | ssd_inception_v2_coco |
| RFCN | rfcn_resnet101_coco |

A notable omission from the chosen networks is a CNN called YOLO v3 (You Only Look Once)[49] which has demonstrated both high speed and accuracy in a variety of tasks. It is also the algorithm incorporated into the VIAME processing toolbox. Unfortunately, the developers implemented this CNN in their own proprietary software format, called Darknet, which was incompatible with the TensorFlow pipeline platform and would have been too complex to train and implement independently within the scope of the project. Future research should certainly incorporate this algorithm into any tests performed.

**DeepSORT (object tracking)**

While various object tracking algorithms exist in OpenCV using a variety of methods, preliminary testing showed that none of them could match the capabilities provided by a third-party algorithm called DeepSORT[50] (which is not included in OpenCV). The DeepSORT algorithm combines a multitude of tracking algorithms, such as Kalman Filters, with a neural network to keep track of detected objects, and it refines its predictions based on frame by frame detections. This method works well retaining the detection if objects became obscured and recognising objects across multiple frames if the detection algorithm failed.

**Feature detection training -pipelines**

The chosen feature detection algorithms all required training on the data that was to be used in the case study. This involved two steps: converting the provided videos into trainable image sets and annotations, and creating the pipelines for the training algorithms.

**Extracting images and annotations**

A couple of hundred frames that contained the objects in question were extracted from the videos as images using the open source video conversion tool, FFMPEG. Due to interlacing in the videos causing a combing effect, as seen in Figure 4, a deinterlacing filter, called YADIF (Yet Another DeInterlacing Filter), was applied as these images were extracted.

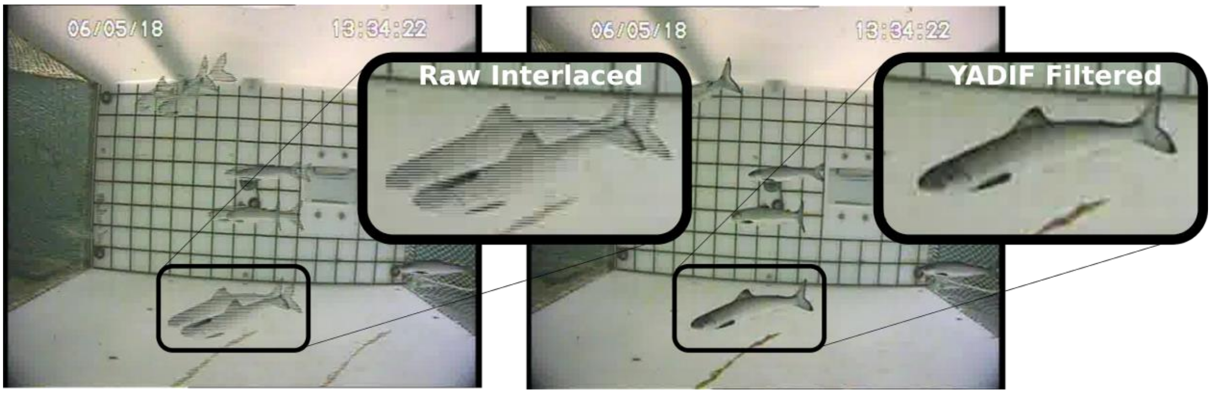


**Figure 4:** Left: an example extracted frame depicting the combing effect that appears from unfiltered interlaced video. Right: the same frame after filtered using YADIF filter in FFMPEG.

These images were then inspected and annotated manually using custom software to produce locations of objects of interest. This involves describing a box that identifies the pixels in the image that contain the object of interest, usually in some form of [X, Y, width, height] as shown in Figure 5. The best practice involves depicting the object of interest in a variety of positions, scales, and backgrounds, and in some cases having certain features obscured. This can help prevent overfitting of training data where the models do not generalise the features and therefore cannot predict well in practice. Around 200 images of aquatic flora/fauna appear to be sufficient for training[35]. There exists a multitude of programs for performing this annotation step open-source (such as LabelIMG and LabelMe) which could have been used in place of the custom software developed for this task. However, creating this software allowed rapid interfacing with the following sections of the training pipeline.

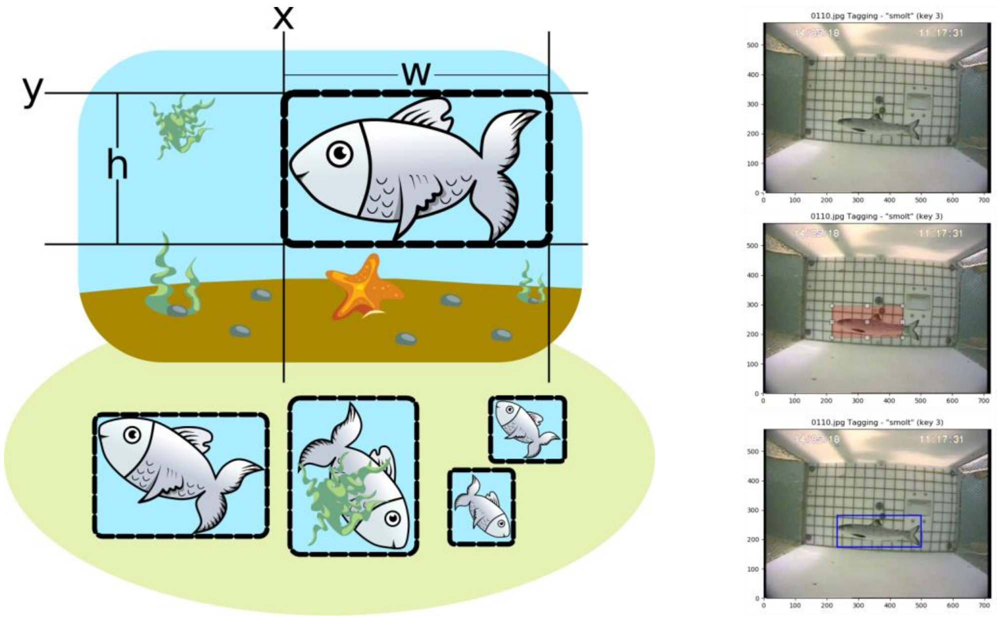

x
y
w
h
0110.jpg Tagging - "smolt" (key 3)
0110.jpg Tagging - "smolt" (key 3)
0110.jpg Tagging - "smolt" (key 3)

**Figure 5:** Left: An example of a bounding box annotation of a fish. The data for the box is stored as an [X, Y] coordinate in pixels representing the top-left corner of the box, along with the width and height of the box in pixels. Multiple boxes can appear in a single image. For improved training, multiple examples of the subject should be given in a variety of poses. It is good practice to get the subject at different orientations, with different levels of obstruction, and at different scales, as shown in the bottom row. Right: An example of the custom annotation software used in action, tagging the pixels in a frame that represent an instance of a salmon smolt.

This step involves the most manual intervention from the user in the pipeline, but it only has to be performed once for each dataset. All existing feature detectors, even those outside the focus of this study, use a similar input format making the resultant annotated datasets valuable for reuse. A recommended action should be to document and store created image/annotation pairs in a database. Platforms such as BIIGLE and CoralNet are offering this as a service so that data can be transferred easily between researchers around the globe and more powerful models can be trained.

### Training pipeline: Bag of visual words

The SIFT, SURF, and ORB feature detectors cannot classify by themselves and require a framework to convert their outputs into recognisable and trainable features. To accomplish this, a custom pipeline using a Bag of Visual Words (BoVW) methodology was created.

The BoVW method is based on the ‘Bag of Words’ method for classifying text documents. If a document is broken down into its individual words, then they can be classified based on the occurrence and frequency of certain words. In the case of “visual words”, descriptive features of an image, such as edge shapes and colours, take the place of the words.

Figure 6 gives an overview of the BoVW method used in this case study. To train this model, the feature detectors first sample all the examples (along with some null examples of only background) and extract feature vectors from each. These are then clustered in a K-Nearest Neighbours (KNN) classifier to create standard ‘words’. This set of standard ‘words’ then forms the feature ‘dictionary’ from which all features will be defined. For each example (and null example), the detected features are then classified in the KNN classifier and the number of words are counted in a histogram. This histogram forms a feature vector of the example.

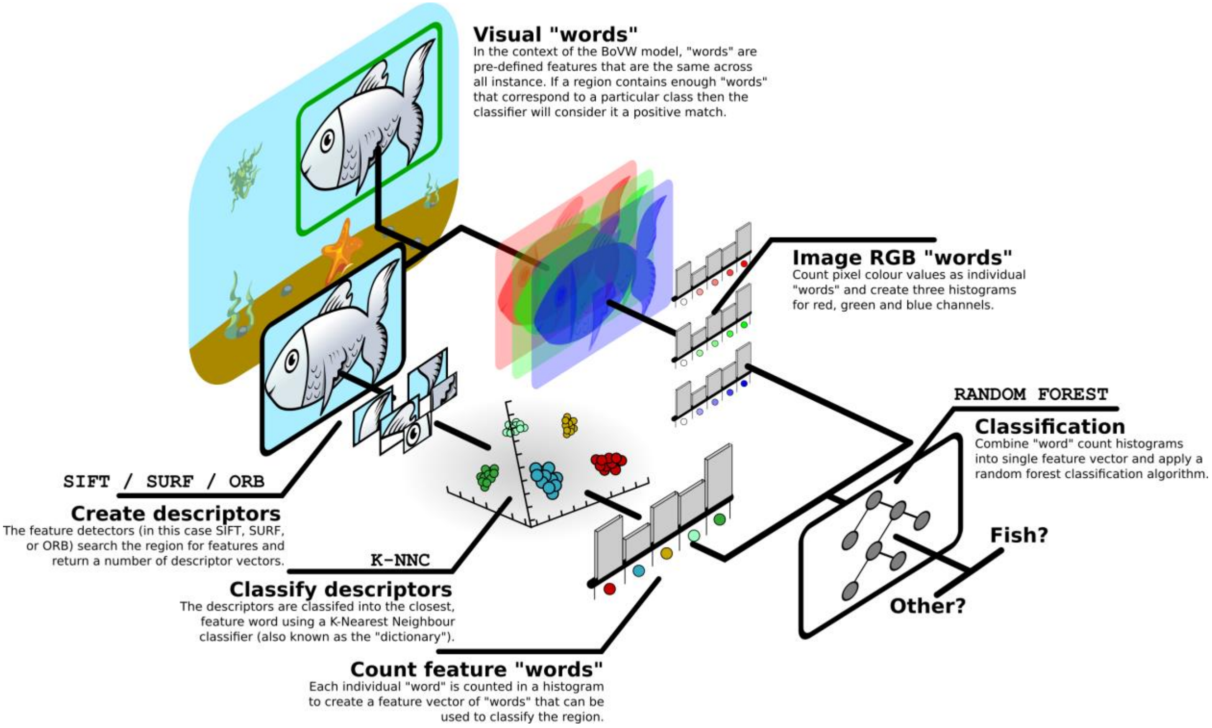

Visual "words"
In the context of the BoVW model, "words" are pre-defined features that are the same across all instance. If a region contains enough "words" that correspond to a particular class then the classifier will consider it a positive match.
Image RGB "words"
Count pixel colour values as individual "words" and create three histograms for red, green and blue channels.
RANDOM FOREST
Classification
Combine "word" count histograms into single feature vector and apply a random forest classification algorithm.
Fish?
Other?
SIFT / SURF / ORB
Create descriptors
The feature detectors (in this case SIFT, SURF, or ORB) search the region for features and return a number of descriptor vectors.
K-NNC
Classify descriptors
The descriptors are classifed into the closest, feature word using a K-Nearest Neighbour classifier (also known as the "dictionary").
Count feature "words"
Each individual "word" is counted in a histogram to create a feature vector of "words" that can be used to classify the region.

**Figure 6:** A summary of the Bag of Visual Words (BoVW) model used in this case study.

At the same time, three similar feature vectors are formed using histograms of pixel colour values in the red, green, and blue channels from the example image. These form feature vectors describing the colour of the object in question.

The four vectors are combined and fed into a classifier (a Random Forest network in this example) which is trained to distinguish between vectors belonging to the examples and those that do not. Once trained, any example image can be processed by creating the feature and colour histograms and passing that through the classifier.

### Training pipeline: Convolutional neural networks

Training a CNN from scratch requires substantial computational time and data in order to create weights in the initial convolution stages of the models. However, it has been shown that the feature weights trained by a model on certain generic objects (such as cars and people) can be reused to classify other objects. Therefore, only the classification stage at the end of these models requires re-training if they are to be repurposed. This is known as ‘transfer learning’.

The Google Object Detection API already contains a basic pipeline for re-training CNNs via transfer learning with custom annotated data. Minor additions were added in order to streamline the process. This resulted in a single step for training these models which was to curate the example images and annotations into a single folder and then execute the provided training code.

Although transfer learning is faster than training all levels of the CNN, to achieve sufficient accuracy, models had to be left to train overnight on a laptop. This process could have been quickened substantially by incorporating a GPU enabled laptop or cloud instance. However, these were not available during this stage of the case study.

Unlike the custom BoVW training, the CNN training is iterative and various levels of progress can be visualised in the TensorBoard suite, also provided by Google. This allows for early termination of training if it can be seen that there has been no improvement over the last few iterations.

### Object detection pipeline

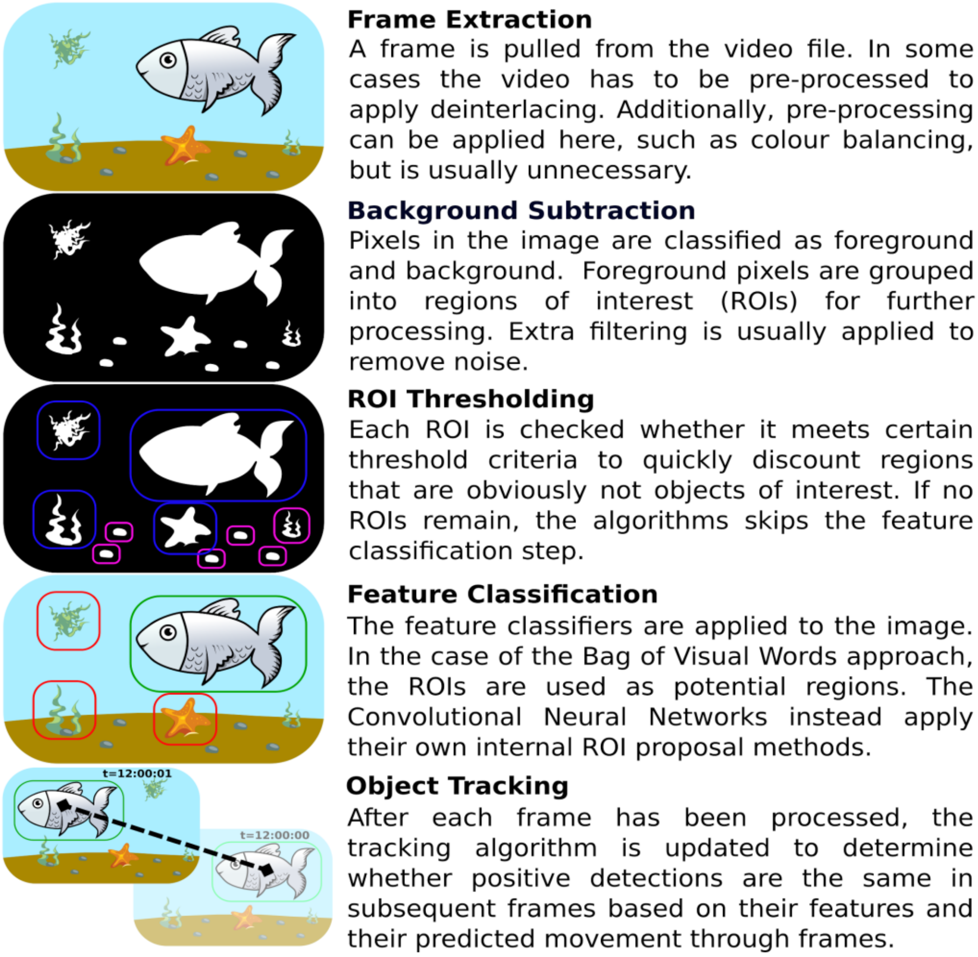
This section outlines the framework for the object detection algorithms once the feature classifiers have been trained. An overview of the algorithm is presented in Figure 7 and a more detailed flowchart is shown in Figure 8. Each entry in Figure 7 is described in more detail in the following sections.

### Frame extraction

OpenCV contains a tool for reading videos frame by frame which is used to convert the video to processable images as needed. However, some video is saved in an interlaced format which creates a combing effect (as shown in Figure 4) when read directly. Therefore, the video has to be deinterlaced initially to allow for processing which is outside the capabilities of OpenCV. Instead, the whole video needs to be converted using FFMPEG and the YADIF filter.

### Background subtraction

Two methods from OpenCV were used to perform the background subtraction: OpenCV MOG2 and OpenCV KNN. Both these models operate by taking a sample previous frames and developing a predictive model for background pixels, a Mixture of Gaussians (MOG) model in the former and a K-Nearest Neighbours (KNN) model in the latter. Foreground pixels are those which fall outside these predictive models which are constantly updated as frames progress. OpenCV MOG2 worked best with a static background, such as the fish trawl video, when adopted with a large history. However, it did not function well on a moving background, such as the seabed rig. Instead, OpenCV KNN performed much better on the moving background with a short history but not well on the static background.



**Frame Extraction**
A frame is pulled from the video file. In some cases the video has to be pre-processed to apply deinterlacing. Additionally, pre-processing can be applied here, such as colour balancing, but is usually unnecessary.

**Background Subtraction**
Pixels in the image are classified as foreground and background. Foreground pixels are grouped into regions of interest (ROIs) for further processing. Extra filtering is usually applied to remove noise.

**ROI Thresholding**
Each ROI is checked whether it meets certain threshold criteria to quickly discount regions that are obviously not objects of interest. If no ROIs remain, the algorithms skips the feature classification step.

**Feature Classification**
The feature classifiers are applied to the image. In the case of the Bag of Visual Words approach, the ROIs are used as potential regions. The Convolutional Neural Networks instead apply their own internal ROI proposal methods.

**Object Tracking**
After each frame has been processed, the tracking algorithm is updated to determine whether positive detections are the same in subsequent frames based on their features and their predicted movement through frames.

**Figure 7:** An overview of the object detection pipeline used in this case study.

The output of the background subtraction algorithms was generally noisy so further filtering was applied to reduce this. The number and frequency of the filters were determined through a manual trial and error method with various combinations of filters. Once the output appeared sensible, this process was fixed for the remainder of the case study. An initial median filter (9x9) was applied to remove noise specks, followed by five loops of dilation (5x5), closing (7x7), erosion (5x5), then opening (7x7) to close gaps in the remaining regions (the numbers in brackets represent the pixel kernel size for the operation).

**ROI thresholding**

The remaining regions underwent an additional thresholding to determine whether they were of sufficient size to pass onto the feature detection stage. This again was an arbitrary choice through trial and error to obtain a good balance of region de-selection. The minimum height and width of a region both had to be at least ten pixels in length in addition to the total rectangular area being greater than 200 pixels.

If the region passed thresholding, an extra 20 pixel boundary was added to the region prior to passing it to the feature detector to ensure all elements of the object were included in the region.

**Feature classification**

One of the five pre-trained feature classifiers are applied to the image, the 3 BoVW models (SIFT, SURF or ORB based), or the three CNNs (FRCNN, SSD, RFCN). In the case of the BoVW models, all of the remaining regions are passed individually to the classifier to determine whether the region contains an object of interest. However, in the case of the CNNs, each one contains its own method of ROI selection, therefore the regions created from the background subtraction and subsequent thresholding are discarded. In these cases, they only serve to determine whether a frame is worth investigating or not. In the absence of any regions of interest, the CNN would not be called to save time as it is computationally expensive.

The resultant regions containing objects of interest (if any) are then passed onto the object tracker.

**Object tracking**

The DeepSORT algorithm was attached to the end of the pipeline to assess detections and track objects of interest between frames. In addition, a frame tagging system was implemented to log times when objects of interest are detected for future manual inspection by a person. Although not a fully automated counting system, this method would dramatically reduce the work time of manual inspection by eliminating the majority of unnecessary frames. To filter out many false positives from debris, a second of footage is only tagged if it contains a detection in a minimum of three out of five sequential frames.

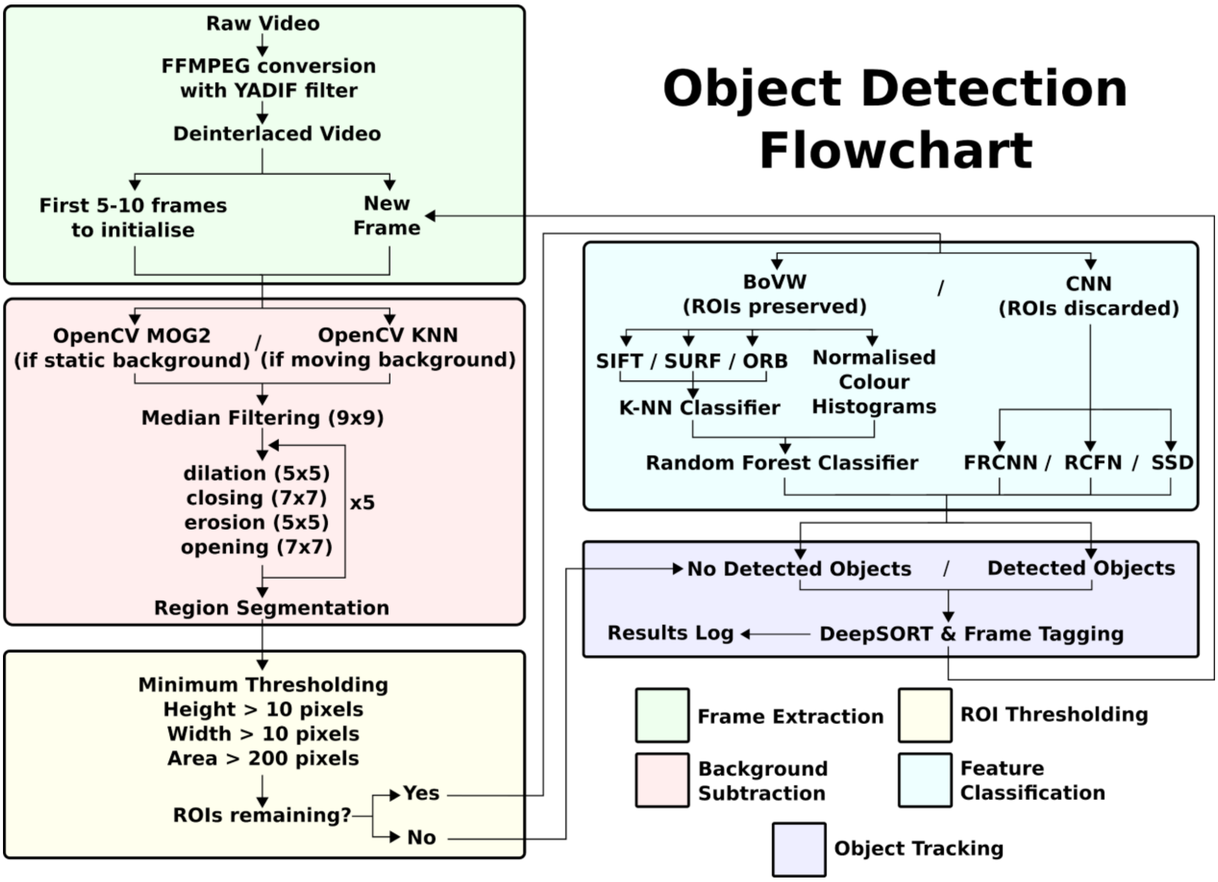


**Figure 8:** A detailed flowchart of the algorithms and programs used in the object detection pipeline.

## Case study

This section describes the execution and results of the case study. Various video clips were extracted for analysis by the different models described in the previous section. Both quantitative and qualitative analysis is provided to describe the effectiveness of the tested models. Only a short amount of video data could be analysed in the timeframe permitted, therefore, this case study serves as a feasibility study into the effectiveness of current computer vision technologies in this problem space, rather than providing a final solution to the problem.

As mentioned in the previous section, only algorithms that could be integrated rapidly into the pipeline were taken forward to the case study, meaning there may exist more advanced versions that would perform better in these scenarios. Additionally, the quantitative results depicted here are only a representation of the performance of the models and should be used as an indication of their potential rather than exact benchmarks. Variations of training data quantity and quality, as well as variations in model architecture, might serve different scenarios better or worse.

**Training models**

All models were trained on frames not included in the case study videos. A combination of ~300 frames from various videos were combined for each model. Frames were extracted from the videos using the open source software FFMPEG using the YADIF deinterlacing filter and saved as JPEG images. For the CNN feature detections, the frames and images were automatically reduced to have a maximum dimension of 1024 pixels, except for the SSD model which has a maximum dimension size of 300 pixels.

For training the CNNs, the frames were randomly split 70%/30% into training and testing sets respectively in order to prevent overfitting of the data. Training progress is monitored by applying the current model iteration to the training set and assessing the accuracy. The model is considered trained once this accuracy plateaus. To achieve this, the models are individually left to train overnight.

For the BoVW model, the same 70% of data is applied for training but does not undergo multiple iterations and therefore the 30% testing data is not used. Training of the BoVW models were completed within 10-15 minutes.

However, the BoVW models generally returned high numbers of false positives on areas that were not objects of interest. This included areas such as a bright reflection off the netting and the clock time printed on the frame. To rectify this, sections of video with no objects were processed with the BoVW model and all instances of false positives were saved. These were added to the training data and the BoVW model was retrained to refine the classification step to ignore these regions.

**Detection measurements**

The various models were run on each of the videos independently, with each generating listed outputs from the frame tagging and DeepSORT algorithms.

Additionally, an annotated video visualising the detections was produced for each model to manually determine the accuracy of these outputs.

In each case the number of true positives (TP) indicate correct counting and identification of objects in video. False negatives (FN) indicate objects that should have been detected but were missed. False positives (FP) indicate instances where wrong frames were tagged or wrong objects were tracked. In the case of the DeepSORT algorithm, there were instances where object tracking was lost and then reassigned a new value, creating multiple counts (MC) for each object.

In addition, overall metrics are used to compare the performance of each model. These are the measures of ‘precision’ and ‘recall’ which are given in Equations 1 & 2 respectively. Precision represents how sensitive a model is to similar looking objects, and recall represents how well the model is able to detect objects in the environment. These were measured using the total counts across all of the five videos.

$$Precision = TP / (TP + FP)$$

*Equation 1: Precision accuracy, given by percentage of positive instances detected out of the total of all positive detections (both true and false).*

$$Recall = TP / (TP + FN)$$

*Equation 2: Recall accuracy, given by percentage of positive instances detected from all possible true instances.*

### Fish detection, classification and counting

A large proportion of research for computer vision in underwater environments goes to fish counting and fish species identification. For the case study, models were trained primarily to identify and count Atlantic salmon (*Salmo salar*) smolts passing through a camera box attached to a specially designed, near-surface trawl net. Various other animals and debris are also present, so it is important that the model can detect and track smolts specifically. Furthermore, the models were simultaneously trained on two types of fish, smolts and European sprats (*Sprattus sprattus*), shown in Figure 9, to determine whether the models could distinguish between them.

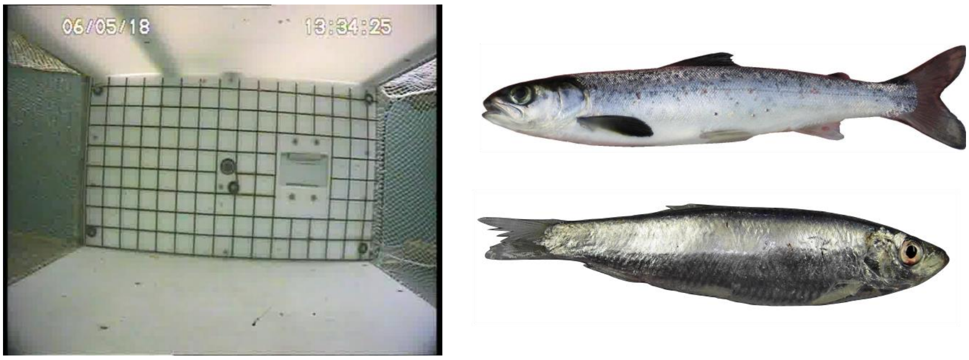


**Figure 9:** Left: an example frame showing the background of the trawl video used for training and testing. Right, top: An Atlantic salmon (*Salmo salar*) smolt, the main fish of interest in this case study. Right, bottom: A European sprat (*Sprattus sprattus*) used for testing the classification capabilities of the models.

The trawl videos were recorded by Marine Scotland Science on board both the RV Scotia and the fishing vessel Sunbeam FR487 during several research voyages to capture salmon smolts over a period between 2017-2019. Trawls took place on the east coast of Scotland, from south of the Firth of Forth to near Orkney. This work was undertaken to attempt to identify salmon migration routes in the context of the development of marine renewable energy.

For training the fish detection and classification algorithms, frames were extracted from the available footage contained either smolts, sprats or neither. Smolts and sprats were given separate classes for the purposes of classification upon detection.

A set of six short videos were extracted from the available footage (that did not contain any training frames) to assess the performance of the various models, shown in Table 2. Although Video 6 contained too many fish to accurately count manually, this video only served for testing processing speeds for large numbers of objects and the classification accuracy of models so the exact number of fish present was not needed.

## Table 2

Video segments used for fish detection and counting model comparison. Processing Times**.**

| Video | Length [mm:ss] | Flora/Fauna | # Objects (manually counted) | Notes |
|---|---|---|---|---|
| 1 | 01:00 | Smolt | 1 | Single smolt but appears only on 5 frames. Rest of video is clear. |
| 2 | 01:00 | Smolt | 1 | Single smolt over ~10 frames. Rest of video is clear. |
| 3 | 01:00 | Smolt | 2 | 2 individual smolts sightings. Moderate amounts of debris in rest of video. |
| 4 | 01:00 | Smolt | 6 | 2 individual smolts sightings. Moderate amounts of debris in rest of video. |
| 5 | 00:15 | Smolt | 12 | Many smolts passing through simultaneously. Testing tracking accuracy for counting. |
| 6 | 00:10 | Sprat | Many | Many sprats passing through (too many to count manually). Testing fish classification accuracy. |

For Videos 1-6, all processing times for each BoVW and CNN algorithm were benchmarked on a GPU enabled laptop (Laptop specifications: Intel Core i7-7700HQ CPU @ 2.80GHz, 16GB DDR4 RAM @ 2400MHz, GeForce GTX 1050 Mobile GPU). Additionally, the CNNs were also benchmarked on an AWS cloud instance (AWS Cloud Instance specifications: 6 vCPU custom Intel Xeon Scalable Skylake @ 2.5GHz, 61GB RAM, NVIDIA Tesla V100 GPU). The results for these benchmarks are shown in Figure 10.

For Videos 1 & 2, processing times across all models are similar. This is due to the fact many frames were empty, meaning the feature detectors were not run and the overall processing was similar in all cases. However, in Videos 3 & 4 the processing times start to diverge as there was more debris and objects in the frames triggering more instances of the feature detection process. This is most notable in Video 6 where processing times were generally multiple times the length of the video in all cases due to every frame contain many objects to track.

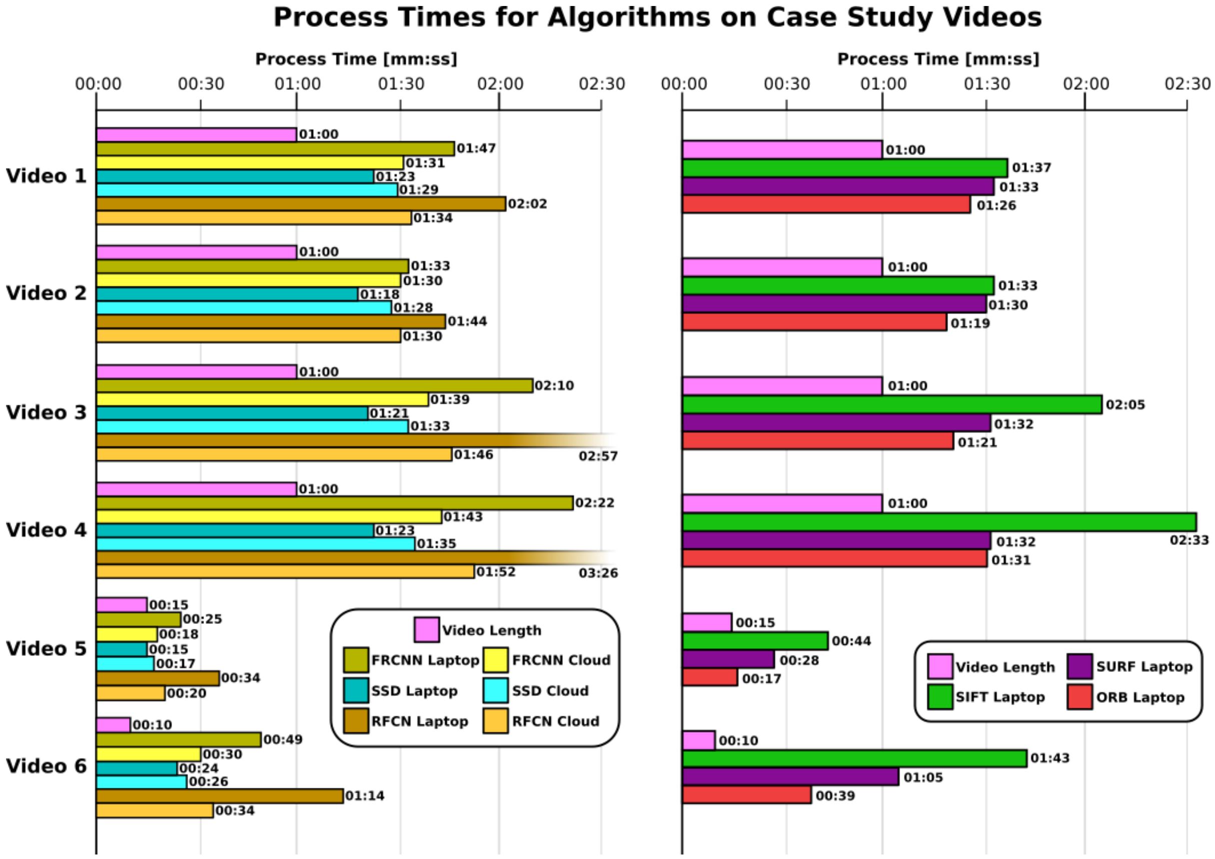


**Figure 10:** Process times for the various algorithms running through the case study videos. For the CNN algorithms on the left, the times cloud times denote the processing time running on an AWS cloud instance. All other times were measured when running on a GPU-enabled laptop. The FRCNN and RCFN speeds were improved when using a more powerful cloud instance; however, the SSD ran slower. This is most likely due to the model not being optimised for that environment.

The processing times for the BoVW models is generally longer than the CNN models. This is most likely due to the fact the BoVW models are not optimised for GPU operations and do not take advantage of the hardware. For the CNNs, the RCNN and RFCN models performed better on the cloud instance whereas the SSD model performed slightly worse. Again, this might be due to the SSD model not being optimised for performance on the cloud hardware. However, on the laptop, the SSD model performed the fastest.

No model was able to complete the analysis faster than the length of the video. This means that live analysis of this footage type would not be possible using these pipelines. However, the computation time could be reduced by skipping frames or running multiple instances of the pipeline in parallel. The current CNN methods are limited to running on a single GPU instance; increasing the number of processors does not improve performance. However, if designing these networks from the ground up, it would be possible to include the option for more parallel processing that would potentially enable real-time analysis.

## Detections

Videos 1-5

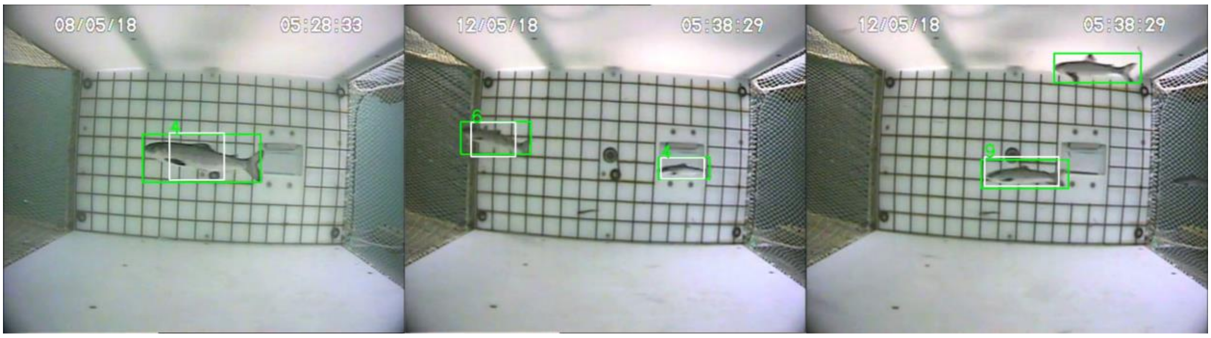


**Figure 11:** Fish detections in Video 1-5 using the FRCNN model. Detections from this CNN are shown as green boxes. The white boxes are from the DeepSORT tracking algorithm. The green number refers to the DeepSORT identification given to the tracked instance and does not necessarily reflect the count of instances.

As the background of the trawl videos remained predominantly static, the OpenCV MOG2 background subtraction method was used. The detection pipeline shown in Figure 8 was implemented with both frame tagging and DeepSORT tracking operating simultaneously.

Examples of processed frames are shown in Figure 11. The detection results for Videos 1-5 are given in the tables above (Table 3 - Table 7). Additionally, the overall metrics for each model over these five videos are given in Table 8.

In terms of frame tagging, all the CNNs had no false positives and so scored 100% precision. The FRCNN and RFCN models also scored 91% and 95% respectively for frame tagging recall displaying an almost perfect solution to this problem. The SSD showed much lower recall most likely due to the low pixel resolution it uses. The FRCNN model also scored highly for the DeepSORT tracking algorithm whereas the RFCN did not. This was due to the fact the RFCN model kept dropping detections every few frames, meaning that, although the frames were tagged, the DeepSORT algorithm could not track the instances effectively.

| VIDEO 1 | | Detections | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model | | Frame Tagging | | | DeepSORT Tracking | | | |
| | | TP | FN | FP | TP | FN | FP | MC |
| CNN | FRCNN | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | SSD | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| | RFCN | 1 | 0 | 0 | 0 | 1 | 0 | 0 |
| BoVW | SIFT | 1 | 0 | 8 | 0 | 1 | 6 | 0 |
| | SURF | 1 | 0 | 3 | 0 | 1 | 2 | 0 |
| | ORB | 1 | 0 | 0 | 0 | 1 | 0 | 0 |

**Table 3:** Video 1 detection results.

| VIDEO 2 | | Detections | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model | | Frame Tagging | | | DeepSORT Tracking | | | |
| | | TP | FN | FP | TP | FN | FP | MC |
| CNN | FRCNN | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | SSD | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | RFCN | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| BoVW | SIFT | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | SURF | 1 | 0 | 0 | 1 | 0 | 0 | 0 |
| | ORB | 1 | 0 | 0 | 1 | 0 | 0 | 0 |

**Table 4:** Video 2 detection results.

| VIDEO 3 | | Detections | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model | | Frame Tagging | | | DeepSORT Tracking | | | |
| | | TP | FN | FP | TP | FN | FP | MC |
| CNN | FRCNN | 2 | 0 | 0 | 2 | 0 | 0 | 0 |
| | SSD | 2 | 0 | 0 | 2 | 0 | 0 | 0 |
| | RFCN | 2 | 0 | 0 | 2 | 0 | 0 | 1 |
| BoVW | SIFT | 2 | 0 | 14 | 2 | 0 | 6 | 0 |
| | SURF | 2 | 0 | 9 | 2 | 0 | 3 | 0 |
| | ORB | 2 | 0 | 0 | 1 | 1 | 0 | 0 |

**Table 5:** Video 3 detection results.

| VIDEO 4 | | Detections | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model | | Frame Tagging | | | DeepSORT Tracking | | | |
| | | TP | FN | FP | TP | FN | FP | MC |
| CNN | FRCNN | 5 | 1 | 0 | 4 | 2 | 0 | 1 |
| | SSD | 4 | 2 | 0 | 2 | 4 | 0 | 3 |
| | RFCN | 5 | 1 | 0 | 4 | 2 | 0 | 2 |
| BoVW | SIFT | 5 | 1 | 16 | 4 | 2 | 8 | 1 |
| | SURF | 5 | 1 | 9 | 3 | 3 | 5 | 1 |
| | ORB | 3 | 3 | 1 | 2 | 4 | 1 | 0 |

**Table 6:** Video 4 detection results.

| VIDEO 5 | | Detections | | | | | | |
|---|---|---|---|---|---|---|---|---|
| Model | | Frame Tagging | | | DeepSORT Tracking | | | |
| | | TP | FN | FP | TP | FN | FP | MC |
| CNN | FRCNN | 11 | 1 | 0 | 9 | 3 | 0 | 2 |
| | SSD | 6 | 6 | 0 | 4 | 8 | 0 | 2 |
| | RFCN | 12 | 0 | 0 | 6 | 6 | 0 | 4 |
| BoVW | SIFT | 8 | 4 | 3 | 7 | 5 | 1 | 2 |
| | SURF | 8 | 4 | 0 | 8 | 4 | 0 | 3 |
| | ORB | 2 | 10 | 0 | 1 | 11 | 0 | 0 |

**Table 7:** Video 5 detection results.

The BoVW models did not perform well, with multiple false positives, in the case of SIFT and SURF, and false negatives, in the case of ORB, in both frame tagging and object tracking. The false positives from the SIFT and SURF were mainly focused on some netting in the background reflecting light as it moved. Further refinement might have been able to address this and improve the precision of these models.

## Table 8

Overall precision and recall results for each model in Videos 1-5. For the purposes of overall precision, instances of multiple counts (MC) are considered false positives (FP).

| Tracking Algorithm | | CNN | | | BoVW | | |
|---|---|---|---|---|---|---|---|
| | | FRCNN | SSD | RFCN | SIFT | SURF | ORB |
| Frame Tagging | Overall Precision | 100% | 100% | 100% | 29% | 45% | 90% |
| | Overall Recall | 91% | 66% | 95% | 77% | 77% | 41% |
| DeepSORT Tracking | Overall Precision | 84% | 64% | 65% | 37% | 50% | 83% |
| | Overall Recall | 73% | 41% | 59% | 64% | 64% | 23% |

### Video 6 - Classifying sprats vs smolts

All models were trained with two possible fish types to classify, smolts and sprats. In Videos 1-5 all instances of fish were smolts. The models all correctly identified the smolts in these videos save for the occasional frame. In Video 6, all instances of fish were sprats, which can be seen in Figure 12.

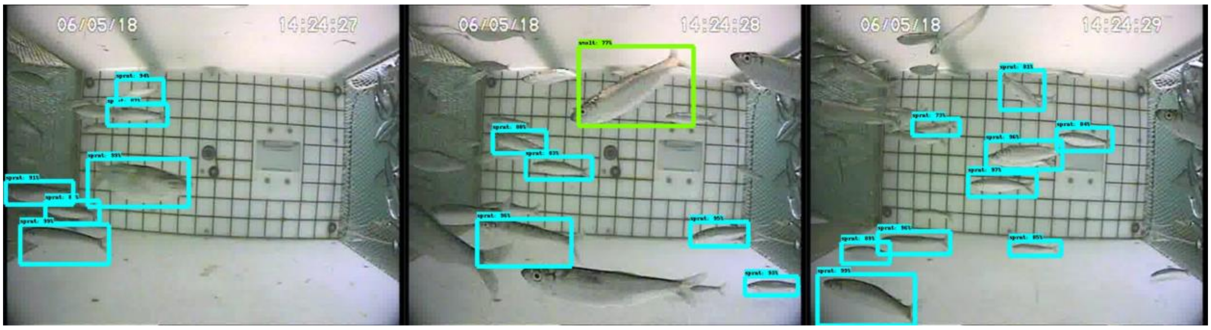


**Figure 12:** Fish detections in Video 6 with only sprats present using the RFCN model. Detections classed as sprats are shown in blue boxes, whereas detections classed as smolts are highlighted in green. It can be seen that the number of fish present in the video makes accurate counting and tracking very difficult.

Table 9 shows the number and percent of sprats correctly identified in Video 6. There were too many sprats in the short segment of video to correctly count the number present. However, due to the density of fish, it is assumed that there are no false positive identifications. The BoVW models all struggled with this task with low numbers of detections and at least two thirds of these mislabelled. The SSD model had a low detection rate but high precision in classification. The RFCN model performed the best scoring the second highest number of detections whilst also achieving 98% classification precision, demonstrating that classification of types of fish under these conditions is feasible.

**Table 9**

The number of detections and sprat classification percent for each model in Video 6.

| | CNN | | | BoVW | | |
|---|---|---|---|---|---|---|
| | **FRCNN** | **SSD** | **RFCN** | **SIFT** | **SURF** | **ORB** |
| **Number of Detections** | **557** | **217** | **463** | **366** | **290** | **55** |
| **Percent Identified as Sprats** | **79%** | **95%** | **98%** | **32%** | **31%** | **15%** |

In the preliminary studies, where the models were only trained on instances of smolts, the video segments with sprats generated many false positives as the model considered them too similar to smolts. By training on two types of fish, these false positives have now been converted into correct classifications. This result not only shows that the CNN models are capable of distinguishing between different types of fish based on example data alone, but demonstrates that for accurate detection models to work, prior knowledge of the types of fish expected to be present is required. That way, the model can be trained to distinguish between them and generate fewer false positives.

### Sea pen detection and counting

Another common application of video data collection and inspection in aquatic environments involves identifying and counting species found in stretches of footage along reefs or seabeds to determine the levels of eco-diversity present and how that changes over time[51]. For this case, footage of phosphorescent sea pens (*Pennatula phosphorea*), shown in Figure 13, were used to determine how well the model pipelines performed in this different scenario.

The sea pen video was collected by Marine Scotland Science in May 2015 on board the MRV *Scotia*. The video was recorded approximately 90 km due east of Kinnaird Head in the northern North Sea. A drop-frame TV camera system was towed behind the vessel at ~1 knot. A digital stills camera (Canon) was mounted on the drop-frame together with a high definition and standard definition video (Kongsberg Simrad). The drop-frame was suspended 1 m above the seabed, guided by a steel weight attached by a line to the drop-frame. Maintaining the steel weight (63.5 mm diameter) on or just above the seabed ensured the correct height for accurate focusing of the video and digital camera. Video was recorded continuously together with digital photographs taken at one minute intervals for the duration of the transect. Two laser pointers set 68 mm apart provided a scale for identifying features.

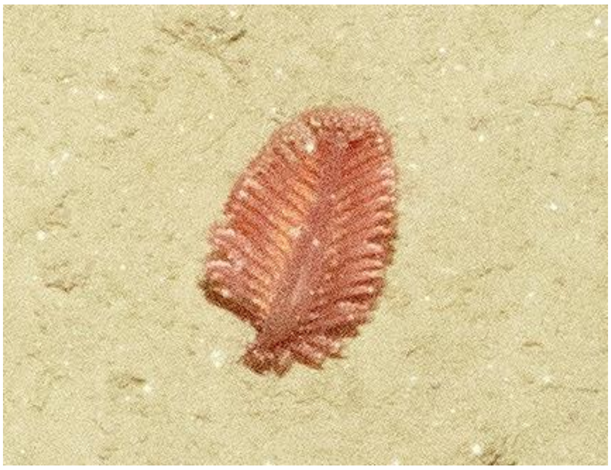


**Figure 13:** An example of the phosphorescent sea pen (*Pennatula phosphorea*) which was the subject of interest in this video.

For training the sea pen detection algorithm, the supplied high-resolution images were used in conjunction with frames extracted from the first minute of test footage. In the case of the sea pen data, images and video were of much higher quality than the other cases. The images had 3648 x 2736 pixel resolution and the video was 1920 x 1080 pixel resolution whereas the resolution of the videos used in fish detection were only 720 x 576 pixels. This meant that parts of the pipeline struggled to handle the large quantities of data.

A single one minute video segment (that did not contain any training frames) was analysed to assess the performance of the various models in detecting sea pens, shown in Table 10.

## Table 10

Video segment used for sea pen detection and counting model comparison.

| Video | Length [mm:ss] | Flora/Fauna | # Objects (manually counted) | Notes |
|---|---|---|---|---|
| S | 01:00 | Sea pen | 45 | Many sea pens on the seabed of varying sizes. Occasional dust clouds obscuring vision temporarily, otherwise generally clear. |

### Detections

As in the previous videos, the detection pipeline outlined in Figure 8 was implemented to identify sea pens in this case example. However, unlike the static background of the fish trawls in Videos 1-6, the background in Video S was a moving sea floor. Therefore, the OpenCV KNN background subtraction was used instead. Unfortunately, as the video resolution was high and there were many moving objects, every frame contained multiple potential targets. This meant the BoVW models returned hundreds of false positives which did not improve with refined training and the CNNs were run on every frame. Additionally, due to the high-resolution of each frame, the processing time was multiple times longer than the video length, even when running on a cloud server.

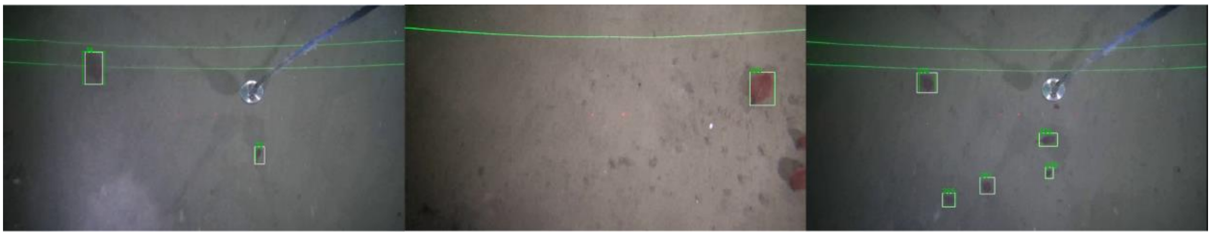


**Figure 14:** Sea pen detections in Video S using the RFCN model. Detections from this CNN are shown as green boxes. The white boxes are from the DeepSORT tracking algorithm. The green number refers to the DeepSORT identification given to the tracked instance and does not necessarily reflect the count of instances.

**Table 11**

Video S detection results and metrics. The high-resolution images were unable to be processed by the SSD model in the current pipeline so results are not available for this model. Results and metrics for the BoVW models are not shown due to their overall poor performance in this task.

| VIDEO S | | Detections | | | | | | | | | | | Processing Time [mm:ss] | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|---|---|
| Model | | Frame Tagging | | | | | DeepSORT Tracking | | | | | | Laptop | Cloud |
| | | TP | FN | FP | Precision | Recall | TP | FN | FP | MC | Precision | Recall | | |
| CNN | FRCNN | 38 | 7 | 12 | 76% | 84% | 35 | 10 | 3 | 7 | 78% | 78% | 10:03 | 08:43 |
| | SSD | - | - | - | - | - | - | - | - | - | - | - | - | |
| | RFCN | 42 | 3 | 10 | 81% | 93% | 40 | 5 | 2 | 4 | 95% | 89% | 13:01 | 09:09 |

Example frames from the detection process are shown in Figure 14 and the detection results and metrics are given in Table 11. Due to a memory limitation, the SSD model was unable to be trained on the large-scale images. Upon training completion, the remaining CNN models claimed 94% and 98% precision (FRCNN, and RFCN respectively) for recognising sea pens in the test images. This is most likely due to the distinguishing shape and colour of the animal against other debris on the seabed. The RFCN model maintained this level when analysing Video S with DeepSORT tracking, scoring 95% precision and 89% recall. However, the FRCNN algorithm only achieved precision of 76% with a recall of 84% in frame tagging, and 78% precision with 78% recall with DeepSORT. Unlike in the fish detection videos, in this case, the RFCN network outperformed the FRCNN network which demonstrates that no single model performs best in every task.

### Nephrops burrow detection

Identifying Nephrops (*Nephrops norvegicus*) burrows is very challenging for the untrained observer. There are a number of criteria that distinguish them from other similar looking burrows in sandy seabeds[52]. Examples of these are shown in Figure 15. As the identification of these burrows is complex, only a preliminary identification of entrances was attempted. This involved identifying all hole and hole-like debris in the video frames which would then be hypothetically post-processed to determine whether they pertain to a Nephrops burrow or not. Unfortunately, the full process of burrow detection is outside the scope of this case study.

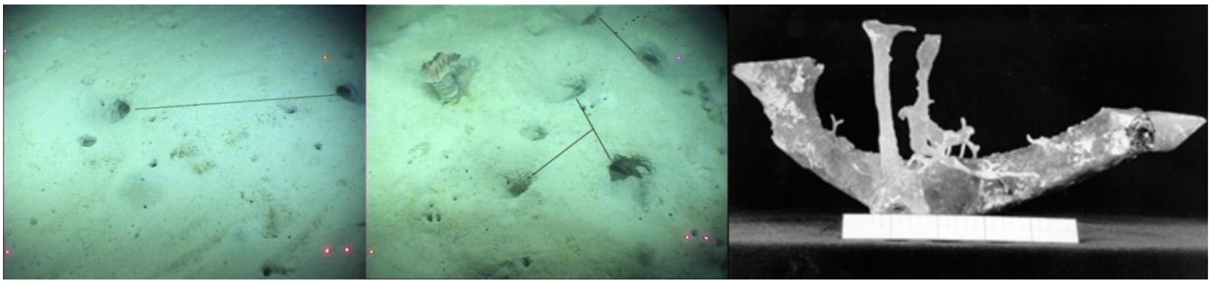


**Figure 15:** Left and middle: examples of Nephrops burrows highlighted by straight black lines. Right: a resin cast of a Nephrops burrow. Distinguishing features include a crescent shaped entrance holes, u-shaped complex, and entrances extending from the vertical apex in a linear fashion.

Underwater video surveys are used as a fishery-independent method to assess the size of Nephrops stocks. The footage provided was obtained during these research vessel surveys by means of an underwater camera and lights, mounted on a sledge and towed along the seabed at approximately one knot. Counts of the burrows and information on the area covered on each tow are used to estimate the density of Nephrops on the ground.

## Table 12

Video segments used for Nephrops burrow detection model comparison.

| Video | Length [mm:ss] | Flora/Fauna | # Objects (manually counted) | Notes |
|---|---|---|---|---|
| NB | 02:53 | Nephrops Burrows | Hundreds | Attempting preliminary detection of holes in ground for hypothetical subsequent analysis for Nephrops burrows. Video frames are very blurry when camera is moving. Also, video contains too many holes for accurate counting. |

For training the Nephrops burrow detection algorithm, one frame each second was extracted for the first 150s of a sample video. The remaining footage in the video was analysed to assess the performance of the various models, shown in Table 12. In this case only one frame per second was analysed.

### Detections

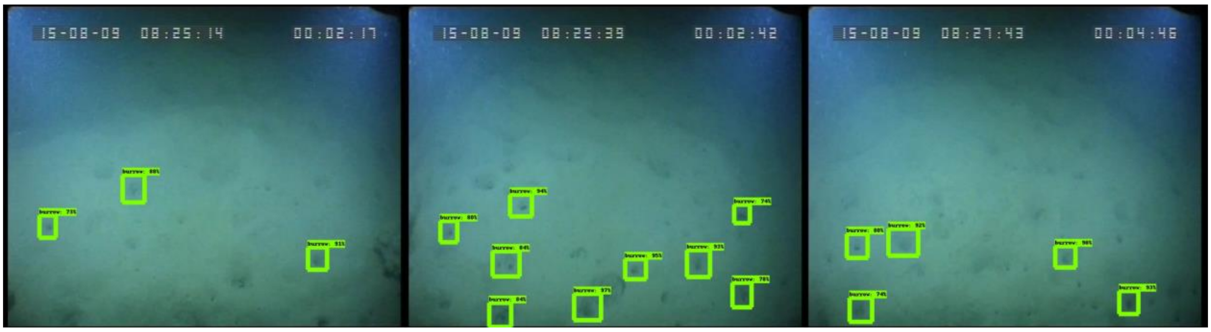


**Figure 16:** Potential Nephrops burrow detections in Video NB using the FRCNN model. Ideally the model should identify all hole-like objects in each frame for further post-processing. However, it is clear from these images that many of these have not been identified.

Nephrops burrows require multiple steps for identification and to distinguish them from other burrows on the seabed. These include subtle shape differences of the entrance surrounded by characteristic claw marks in the sand and the connectivity between various burrow entrances to produce the structure shown in Figure 15. These factors are not easily translatable into features for performing object detection. Instead, further custom algorithmical analysis would have to be performed on potential entrances to determine whether or not the burrow. These steps involve, but are not limited to, determining the angle of the entrance based on the perceived angle from the camera, analysing which entrances may combine to create the particular burrow structure, and assessing whether there may be obscured entrances or entrances facing away from the camera. However, these steps are not simple to program and fall outside the scope of the current case study.

As Nephrops burrows are difficult to identify, the training frames were manually annotated with instances of holes or hole-like debris, representing potential entrances to the burrow as the first step of hypothetical detection. These were then put through the FRCNN pipeline. Unfortunately, recall never rose above 65% meaning there were many instances that were missed. The images in Figure 16 show examples of the underperformance of the model.

The main factors for the low precision and recall scores are the lack of prominent features of the hole structures combined with the blurred video from camera motion. In some cases, the holes are only visible from subtle shading in the sand which is difficult for the computer vision algorithm to recognise. The difficulty is further compounded by the camera motion blur inflicting shading on various other features of the seabed.

Although unsuccessful, this case serves to highlight that, while the current computer vision technology is powerful, there are limitations as to what it can achieve on its own. There are many steps to identifying a Nephrops burrow, beginning with identifying possible entrance holes, and even that has been shown not to be a straightforward task to automate. From the results here, it is considered that this task, although not impossible, would require a much more specialised approach which is outside the scope of the current case study.

### Analysis of models from case study

It is clear from the results that the CNN models outperform the BoVW models in terms of both speed and accuracy. With the CNN models, the high levels of precision and recall observed indicate that these are potential solutions for automating the analysis of videos from aquatic environments. However, selecting a single model for an application is not straightforward. Whilst the SSD model was the fastest tested here, which would be useful for analysing a large backlog of videos, it was not as robust as the slower FRCNN or RFCN models. The FRCNN model performed the best overall in the fish detection videos, but the RFCN model performed better with fish classification and detecting sea pens. For future development, training of various models would be beneficial to determine which performs best for the particular task in question.

It was intended to create pipelines that avoided as much manual input as possible in order to streamline the case study process. Therefore, in most cases, once enough data has been annotated the models can all be trained and applied with the execution of a single command. This also makes them approachable for further investigation by researchers without a strong background in the field of computer vision.

However, there are a couple of places where manual intervention is required. For instance, in the BoVW models, training needed to be repeated multiple times to reduce the number of false positives. This requires extracting these instances from a set of results and adding them to the original data. Additionally, fine tuning the Background Subtraction to remove noise requires various image processing steps that were derived manually. These may need to be adjusted to suit different backgrounds. However, in the CNN models, the background subtraction step only serves to improve computation time as the model itself performs its own regional selection algorithms and may not be necessary if time is of no issue.

In terms of the algorithms that constitute the pipelines, both the BoVW and CNN methods are reasonably modular lending themselves to be improved with other available methods if required. Currently, the CNN models are built around TensorFlow and only accept compatible models; however, similar training and execution pipelines in other frameworks exist that could be adopted in a similar fashion. Similarly, the feature detection algorithms for the BoVW models could be replaced by any algorithm that returns similar description vectors as the OpenCV implementations of SIFT, SURF, and ORB. For every other section, alternative algorithms are acceptable as long as the inputs and outputs are compatible with the pipeline stages that come before and after respectively.

The footage tested in this case study was indicative of the various types of video encountered in aquatic research. However, certain practices in obtaining footage could aid in the process of automatically identifying aquatic life. Steady or still camera shots prevent motion blur of subjects and backgrounds which makes distinguishing features much simpler. Although high-definition footage also allows more features to be presented in each image, there is a trade-off for processing times as seen when comparing the computation time for a one minute 720 x 576 fish trawl video (1.5-2 minutes) and a one minute 1920 x 1080 sea pen video (8-9 minutes). Furthermore, the scene should be well lit from artificial light to prevent colour loss, but with care to avoid harsh reflections off subjects of interest that can obscure features.

This case study has shown that these algorithms, specifically the CNN based algorithms, are fit for purpose in identifying aquatic fauna in underwater video. They are readily customisable for a variety of tasks and require a relatively low level of human intervention to initiate. Once the data pipeline has been produced, the level of input for customisation is relatively simple, only annotated data is required which can be produced by a user with little knowledge of the inner workings of the model. However, for consistently reliable, high-accuracy models, further specialist development may be required to ensure the statistical precision and recall of the models remain at appropriate levels across a wide variety of cases. Additionally, for individual tasks, multiple CNN models should be trialled initially to determine which is best suited for the job.

## Conclusion

In this report, an overview of the current computer vision technologies were presented with their contributions to the field of object detection in aquatic video. A sample of the more promising technologies were taken forward for a case study to analyse video clips of data provided by Marine Scotland. This identified some which are viable for further development.

Although much research has been performed in the field of computer vision, recent developments with convolutional neural networks have overshadowed many of the previously developed algorithms. This was also reflected in the case study, where the CNN algorithms outperformed the BoVW models by a large margin.

Additionally, the open-source variants of the technology have proven to be effective even in the short developmental time frame available. This is promising for future development as fewer resources are required to initialise the process and more work can be performed in training and testing models. The working pipeline presented here, based on Google Object Detection API, could act as a platform upon which to build a working solution. There are other platforms, such as BIIGLE and VIAME, which are also trying to bring these computer vision technologies to aquatic scientists by abstracting out many of their complexities. However, they are currently still in the early stages of development, but could possibly provide a solution in the future.

The case study methodologies and results described in this report demonstrate that the current available technology performs well in detecting aquatic fauna in underwater video which, in turn, will help improve the environment analysis capabilities of Marine Scotland. These methods could be applied to various video material collected in, for instance, camera boxes attached to open ended trawl nets; during monitoring at tidal turbines and at wind turbine bases; and at video based fish counters and in the video validation of fish counters using other technology. Although some instances, like the Nephrops burrows, would require additional development, salmon smolt and phosphorescent sea pen detection can be achieved using the methods presented here.

### Actionable recommendations

From the outputs of this case study, a variety of actionable recommendations are proposed here for future research and development in this problem space:

1. For detection of aquatic life in video, the convoluted neural networks far surpass the older algorithms in terms of both accuracy and ease of development.
    - Transfer learning is sufficient in most cases. It is generally unnecessary to retrain the convolution/pooling stages, meaning many available open source options can be capitalized on.
    - However, more complex objects that require more nuanced interpretation, such as the Nephrops burrows, will require further post-processing or alternative approaches for automatic identification.

2. Develop or document a database of images with annotations of aquatic life for training purposes.
    - All models exhibited here rely on training data to properly initialise the feature detection steps.
    - The video data supplied contained annotations of number of species identified with approximate timestamps which are insufficient for training these algorithms.
    - Regardless of the final method chosen, having a database of training data already available will greatly facilitate future development.
    - The recommended requirements are a database of images, each with corresponding [X, Y, width, height] annotations similar to Figure 5. These annotations should also denote the class of the object represented which should be unique to prevent confusion (e.g. genus and species). Although the actual annotation formats used by various training models may vary, they can usually be derived from this information.

3. It would be beneficial to develop a platform, or adopt an existing platform, for aquatic biologists to gain familiarity with the process of training and adopting these types of models.
   - The complexities of model and parameter selection should be kept to a minimum in the platform to avoid unnecessary complexity to the process.
   - This would allow researchers to attempt development of models by themselves with their own data without having to refer to an expert in the field of computer vision.
   - There exist potential solutions based on other platforms such as BIIGLE, CoralNet, and VIAME that may address some areas of the problem space, although they would require additional development or expertise to achieve a complete end-to-end solution.

4. Some of these computer vision models may outperform humans in some tasks and underperform in others. Therefore, a certain level of standardisation of the various elements involved, which might include benchmark tests and standard data libraries, will be required to compare the performance of different available processes.
   - Additionally, a standardised representative level of accuracy should be presented with models to correctly interpret generated detections if the video is never going to be reassessed by humans.

**Considerations towards a complete end-user solution**

In addition to the recommendations above, the following is a list of possible platforms upon which to potentially build a complete solution for the automatic detection of aquatic life in underwater video.

The Google Object Detection API platform is the method displayed during this case study. The others represent alternative platforms currently in varying stages of development. However, these other platforms will also require additional effort or development to be able to implement a full end-to-end solution similar to the approach presented in this case study. All these platforms are available in open source formats.

| **Google Object Detection API (combined with DeepSORT tracking)** | |
|---|---|
| **Pros** | ● Demonstrated here in the case study to be a powerful and versatile option for detecting and counting aquatic life.<br>● Can be trained on almost any subject with enough training data.<br>● Feature detection model can be chosen to suit the task at hand. |
| **Cons** | ● Requires knowledge of Python scripting and familiarity with the TensorFlow library to implement.<br>● On its own it is a bare-bones package aimed at computer vision researchers. It is not suitable for people unfamiliar with the training and implementing neural networks. |
| **Work Required** | ● As with the working pipeline presented in this case study, the process for training and executing the networks can be extracted out into simple steps.<br>● Additional work could be put in to create a user-friendly front end to simplify the process. |

| **VIAME (Video and Image Analytics for a Marine Environment)** | |
|---|---|
| **Pros** | ● Has lots of functionality in regards to computer vision technology, including the ability to track and count species in video.<br>● It is aimed towards marine scientists and the problems they would encounter with visual data.<br>● Allows for a modular approach to customise your own solution. |
| **Cons** | ● Installation is not straight forward. Obtaining a version was not possible in the time frame of this project.<br>● From the tutorial videos it contains, the interface looks very dense and not necessarily user-friendly for people unfamiliar with computer vision technologies.<br>● It appears to only contain a single type of CNN to work with out of the box. It may be possible to include more through the modular framework but it is unclear. |
| **Work Required** | ● Assistance from developers would be required to setup and configure the software.<br>● This would involve training and instruction on how to use the various features that it contains. |

| **CoralNet** | |
|---|---|
| **Pros** | ● Contains a powerful CNN backend to assist in automatically detecting coral in images.<br>● Can upload your images for processing to enable public sharing of data. |
| **Cons** | ● Specialised for reefs and surface biological growths, not compatible with detection and tracking algorithms. |

| | |
|---|---|
| | ● Only works with images, so videos would have to be processed into frames manually before passing to this software.<br>● The CNN model is hosted by the website meaning customisation of the approach is not possible. |
| **Work Required** | ● This solution performs well in identifying types of surface biological growth in images.<br>● For video processing, extraction of frames and subsequent analysis on the frame detections would be required to incorporate tracking and counting.<br>● Alternatively, a custom interface to the website could be produced; however, there could be unforeseen integration problems with this approach. |

| **BIIGLE** | |
|---|---|
| **Pros** | ● An accessible web-based interface that has a simple workflow pattern (once data has been hosted in an online repository).<br>● The machine learning suggestions that are returned are useful and generally accurate.<br>● Has an established user base. |
| **Cons** | ● The process is image focused (not video) meaning the ability to track and count objects is not present.<br>● Only works with images, so videos would have to be processed into frames manually before passing to this software.<br>● Requires users to host images themselves (no option to directly upload to the software without additional permissions), which presents a barrier to entry if researchers are not familiar with the process. |
| **Work Required** | ● This solution is functional and approachable for the task it performs, which is identifying subjects in images.<br>● Manual pre-processing of video and subsequent analysis on the frame detections would be required to enable video processing.<br>● Alternatively, a custom interface to the website could be produced; however, there could be unforeseen integration problems with this approach. |

| **FishTick** | |
|---|---|
| **Pros** | ● Used widely for fish counting in North America.<br>● Interface is simple and straight-forward to set up. |
| **Cons** | ● Current version only contains frame-tagging capabilities. Manual classification and counting still need to be performed.<br>● Currently requires simple stationary backgrounds (i.e. fish ladders). |

| | |
|---|---|
| **Work Required** | ● A new version that aims/claims to include automated classification and counting is in the beta stages of development. However, the release date for this is unknown. |

## Acknowledgements

The authors would like to thank Scottish Government for providing the funding and Marine Scotland Science for providing the video data used in this project. We would also like to thank Marine Scotland and Scottish Natural Heritage for participation in the project steering group.

## References

[1] Dawson-Howe, K. (2014). *A practical introduction to computer vision with OpenCV*. John Wiley & Sons.

[2] Panchal, P., Prajapati, G., Patel, S., Shah, H., & Nasriwala, J. (2015). A review on object detection and tracking methods. *International Journal for Research in Emerging Science and Technology*, ***2***(1), 7-12.

[3] Zhang, D., Islam, M. M., & Lu, G. (2012). A review on automatic image annotation techniques. *Pattern Recognition*, ***45***(1), 346-362.

[4] Lowe, D. G. (1999). Object recognition from local scale-invariant features. *Proceedings of the International Conference on Computer Vision*, **99**(2), 1150-1157.

[5] Mikolajczyk. K. and Schmid, C. (2005). A performance evaluation of local descriptors. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, **27**(10), 1615–1630.

[6] Viola, P. and Jones, M. (2001). Robust real-time object detection. *International Journal of Computer Vision*, **4**(4), 34-47.

[7] Bay, H., Tuytelaars, T., & Van Gool, L. (2006). Surf: Speeded up robust features. In *European conference on computer vision* (pp. 404-417). Springer, Berlin, Heidelberg.

[8] Movellan, J. R. (2002). Tutorial on Gabor filters. *Open Source Document.* https://web.archive.org/web/20090419123314/http://mplab.ucsd.edu/tutorials/gabor.pdf

[*9*] Dalal, N. and Triggs, B.  (2005). Histograms of oriented gradients for human detection.

[*10*] Rublee, E., Rabaud, V., Konolige, K., & Bradski, G. R. (2011). ORB: An efficient alternative to SIFT or SURF. *International Conference on Computer Vision,* **11**(1), 2.

[*11*] Russakovsky, O., Deng, J., Su, H., Krause, J., Satheesh, S., Ma, S., Huang, Z., Karpathy, A., Khosla, A., Bernstein, M. & Berg, A.C. (2015). ImageNet large scale visual recognition challenge. *International journal of computer vision,* **115**(3), 211-252.

[*12*] Krizhevsky, A., Sutskever, I., & Hinton, G. E. (2012). Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems* (pp. 1097-1105).

[*13*] "ImageNet Large Scale Visual Recognition Challenge 2012 (ILSVRC2012)", *ImageNet*. http://image-net.org/challenges/LSVRC/2012/results.html

[*14*] "ImageNet Large Scale Visual Recognition Challenge 2017 (ILSVRC2017)", *ImageNet*. http://image-net.org/challenges/LSVRC/2017/results

[*15*] Føre, M., Frank, K., Norton, T., Svendsen, E., Alfredsen, J.A., Dempster, T., Eguiraun, H., Watson, W., Stahl, A., Sunde, L.M. & Schellewald, C. (2018). Precision fish farming: A new framework to improve production in aquaculture. *Biosystems Engineering*, **173**, 176-193.

[*16*] Beauxis-Aussalet, E., Palazzo, S., Nadarajan, G., Arslanova, E., Spampinato, C., & Hardman, L. (2013). A video processing and data retrieval framework for fish population monitoring. In *Proceedings of the 2nd ACM international workshop on Multimedia analysis for ecological data* (pp. 15-20). ACM.

[*17*] Jovanović, V., Risojević, V., Babić, Z., Svendsen, E., & Stahl, A. (2016). Splash detection in surveillance videos of offshore fish production plants. In *2016 International Conference on Systems, Signals and Image Processing (IWSSIP)* (pp. 1-4). IEEE.

[*18*] Williams, K., Lauffenburger, N., Chuang, M. C., Hwang, J. N., & Towler, R. (2016). Automated measurements of fish within a trawl using stereo images from a Camera-Trawl device (CamTrawl). *Methods in Oceanography*, **17**, 138-152.

[*19*] Williams, K., Rooper, C. N., & Harms, J. J. H. (2012). Report of the National Marine Fisheries Service Automated Image Processing Workshop, September 4-7, 2012, Seattle, Washington..

[*20*] Shortis, M.R., Ravanbakskh, M., Shaifat, F., Harvey, E.S., Mian, A., Seager, J.W., Culverhouse, P.F., Cline, D.E. and Edgington, D.R. (2013). A review of techniques for the identification and measurement of fish in underwater stereo-video image sequences. In *Videometrics, Range Imaging, and Applications XII; and Automated Visual Inspection (Vol. 8791, p. 87910G).* International Society for Optics and Photonics.

[*21*] Spampinato, C., Chen-Burger, Y. H., Nadarajan, G., & Fisher, R. B. (2008). Detecting, Tracking and Counting Fish in Low Quality Unconstrained Underwater Videos. *VISAPP,* **2**(2008), 514-519.

[*22*] Khanfar, H. et al. "Automated recognition and tracking of fish in underwater video." *Final Report, LA Board of Regents Contract NASA (2008)-STENNIS-08*. (2010).

[*23*] Khanfar, H., Charalampidis, D., Ioup, G., Ioup, J., & Thompson, C. H. (2010). Automated recognition and tracking of fish in underwater video. Final Report, LA Board of Regents Contract NASA (2008)-STENNIS-08.

[*24*] Westling, F., Sun, C., & Wang, D. (2014). A modular learning approach for fish counting and measurement using stereo baited remote underwater video. In *2014 International Conference on Digital Image Computing: Techniques and Applications (DICTA)* (pp. 1-7). IEEE.

[*25*] Lantsova, E., Voitiuk, T., Zudilova, T., & Kaarna, A. (2016). Using low-quality video sequences for fish detection and tracking. In *2016 SAI Computing Conference (SAI)* (pp. 426-433). IEEE.

[*26*] Viola, P. and Jones, M. (2001). Rapid Object Detection using a Boosted Cascade of Simple Features. *IEEE Int. Conf. on Computer Vision and Pattern Recognition (CVPR)*, **1**(1), 511-518.

[*27*] Iqbal, K., Salam, R. A., Osman, A., & Talib, A. Z. (2007). Underwater Image Enhancement Using an Integrated Colour Model. *IAENG International Journal of Computer Science*, **34**(2).

[*28*] Peng, Y. T., Zhao, X., & Cosman, P. C. (2015). Single underwater image enhancement using depth estimation based on blurriness. In *2015 IEEE International Conference on Image Processing (ICIP)* (pp. 4952-4956). IEEE.

[*29*] Ghani, A. S. A. and  Isa, N. A. M. (2014). Underwater image quality enhancement through Rayleigh-stretching and averaging image planes. *International Journal of Naval Architecture and Ocean Engineering,* **6**(4), 840-866.

[*30*] Singh, R. and Mantosh, B. (2017). Hazy Underwater Image Enhancement based on Contrast and Color improvement using fusion technique. *Image Processing & Communications,* **22**(3), 31-38.

[*31*] Salman, A., Siddiqui, S.A., Shafait, F., Mian, A., Shortis, M.R., Khurshid, K., Ulges, A. and Schwanecke, U. (2019). Automatic fish detection in underwater videos by a deep neural network-based hybrid motion learning system. *ICES Journal of Marine Science.*

[*32*] Labao, A.B. and  Naval Jr, P. C. (2019). Cascaded deep network systems with linked ensemble components for underwater fish detection in the wild. *Ecological Informatics.* **52**, 103-121.

[*33*] Rathi, D., Jain, S., & Indu, S. (2017). Underwater fish species classification using convolutional neural network and deep learning. In *2017 Ninth International Conference on Advances in Pattern Recognition (ICAPR)* (pp. 1-6). IEEE.

[*34*] Li, X., Shang, M., Hao, J., & Yang, Z. (2016). Accelerating fish detection and recognition by sharing CNNs with objectness learning. In *OCEANS 2016-Shanghai* (pp. 1-5). IEEE.

[*35*] Piechaud, N., Hunt, C., Culverhouse, P. F., Foster, N. L., & Howell, K. L. (2019). Automated identification of benthic epifauna with computer vision. *Marine Ecology Progress Series*. **615**, 15-30.

[*36*] Huang, R. J., Lai, Y. C., Tsao, C. Y., Kuo, Y. P., Wang, J. H., & Chang, C. C. (2018). Applying convolutional networks to underwater tracking without training. In *2018 IEEE International Conference on Applied System Invention (ICASI)* (pp. 342-345). IEEE.

[*37*] Jäger, J., Wolff, V., Fricke-Neuderth, K., Mothes, O., & Denzler, J. (2017). Visual fish tracking: Combining a two-stage graph approach with CNN-features. In *OCEANS 2017-Aberdeen* (pp. 1-6). IEEE.

[*38*] Gomes-Pereira, J.N., Auger, V., Beisiegel, K., Benjamin, R., Bergmann, M., Bowden, D., Buhl-Mortensen, P., De Leo, F.C., Dionísio, G., Durden, J.M. and Edwards, L. (2016). Current and future trends in marine image annotation software. *Progress in Oceanography*, **149**, 106-120.

[*39*] Williams, I. D., Couch, C., Beijbom, O., Oliver, T., Vargas-Angel, B., Schumacher, B., & Brainard, R. (2019). Leveraging automated image analysis tools to transform our capacity to assess status and trends on coral reefs. *Frontiers in Marine Science*, **6**, 222.

[*40*] Gormley, K., McLellan, F., McCabe, C., Hinton, C., Ferris, J., Kline, D., & Scott, B. (2018). Automated image analysis of offshore infrastructure marine biofouling. *Journal of Marine Science and Engineering*, **6**(1), 2.

[*41*] “The Nature Conservancy Fisheries Monitoring”. *Kaggle Inc*. https://www.kaggle.com/c/the-nature-conservancy-fisheries-monitoring

[*42*] “N+1 Fish, N+2 Fish”. *DrivenData*. https://www.drivendata.org/competitions/48/identify-fish-challenge/

[*43*] “Winning models for the N+1 Fish, N+2 Fish competition“. *DrivenData, GitHub, Inc.* https://github.com/drivendataorg/n-plus-one-fish

[*44*] Chuang, M. C., Hwang, J. N., & Rose, C. S. (2013). Aggregated segmentation of fish from conveyor belt videos. In *2013 IEEE International Conference on Acoustics, Speech and Signal Processing* (pp. 1807-1811). IEEE.

[*45*] Huang, J., Rathod, V., Sun, C., Zhu, M., Korattikara, A., Fathi, A., Fischer, I., Wojna, Z., Song, Y., Guadarrama, S. and Murphy, K., (2017). Speed/accuracy trade-offs for modern convolutional object detectors. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 7310-7311).

[*46*] Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster R-CNN: Towards real-time object detection with region proposal networks. In *Advances in neural information processing systems* (pp. 91-99).

[*47*] Liu, W., Anguelov, D., Erhan, D., Szegedy, C., Reed, S., Fu, C. Y., & Berg, A. C. (2016). SSD: Single shot multibox detector. In *European conference on computer vision* (pp. 21-37). Springer, Cham.

[*48*] Dai, J., Li, Y., He, K., & Sun, J. (2016). R-RCN: Object detection via region-based fully convolutional networks. In *Advances in neural information processing systems* (pp. 379-387).

[*49*] Redmon, J., & Farhadi, A. (2018). YOLOv3: An incremental improvement. *arXiv preprint arXiv:1804.02767*.

[*50*] Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE International Conference on Image Processing (ICIP)* (pp. 3645-3649). IEEE.

[*51*] Bell, E., Clements, A., Dobby, H., Doyle, J., Feekings, J.P., Leocádio, A., Lordan, C., Weetman, A. and Wieland, K. (2018). Using underwater television surveys to assess and advise on Nephrops stocks.

[*52*] “Report of the Workshop on Nephrops burrow counting.”, *ICES SSGIEOM Committee,* (2016).

© Crown Copyright 2020

Marine Scotland Science
Marine Laboratory
375 Victoria Road
Aberdeen
AB11 9DB

Copies of this report are available from the Marine Scotland website at www.gov.scot/marinescotland