Scottish Government
Riaghaltas na h-Alba

A New Application for Automated Video Identification of Marine Species (AVIMS)

Scottish Government Marine Directorate

Scottish Marine and Freshwater Science Vol 14 No 7

M Mackiewicz, G French, M Fisher

Main Report

A New Application for Automated Video Identification of Marine Species (AVIMS)

Main Report

Scottish Marine and Freshwater Science Vol 14 No 7

M. Mackiewicz, G. French and M. Fisher

Published by the Marine Directorate of the Scottish Government

ISBN: 978-1-83521-505-0
ISSN: 2043-7722
DOI: 10.7489/12473-1

The Marine Directorate of the Scottish Government is responsible for the integrated management of Scotland's seas. Scottish Marine and Freshwater Science is a series of reports that publishes the results of scientific research and monitoring carried out by the Marine Directorate. It also publishes the results of marine and freshwater scientific work that has been carried out for the Marine Directorate under external commission. These reports are not subject to formal external peer-review.

This report presents the results of scientific work commissioned by the Marine Directorate and carried out at the School of Computing Sciences, University of East Anglia.

This report should be quoted as:
Mackiewicz, M., French, G. and Fisher, M. (2023). A new application for automated video identification of marine species (AVIMS). Scottish Marine and Freshwater Science Vol 14 No 7, 18pp. DOI: 10.7489/12473-1.

© Crown copyright 2023

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: Open Government Licence (nationalarchives.gov.uk) 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.

A New Application For Automated Video Identification Of Marine Species (AVIMS)

Main Report

M. Mackiewicz, G. French and M. Fisher

School of Computing Sciences, University of East Anglia,
Norwich Research Park, Norwich. NR4 7TJ

Abstract

In the course of its environmental monitoring activities, the Scottish Government's Marine Directorate collects a large amount of underwater video to, for example, obtain information on the numbers of fish in rivers or on species living on the seabed. Manual analysis of this footage is laborious and costly, but Machine Learning algorithms can now be used to automate such image analysis. The Marine Directorate commissioned the University of East Anglia to develop a web-based application to allow staff to create, train and execute machine learning-based (semi-)automated analysis of video footage without a need to interact with the underlying computer code. The application was tested using three diverse sets of video footage and found to be usable by staff without computer science or coding experience. The tool was able to detect and count sea pens in footage from towed underwater vehicles, salmon smolts at sea in underwater footage from towed fishing gear and adult salmon and sea trout in footage from underwater or overhead cameras at fixed locations on rivers. Improving the accuracy of the models at detecting and counting organisms of interest will require the use of larger annotated datasets in further training of the algorithms, but the current application provides a basis for further developing these.

Executive summary

In this project we developed a web application entitled Automated Video Identification of Marine Species (AVIMS). The project was funded by Scottish Government Contract, CASE/216380. The objective was to develop an automated video analysis capability with a user-friendly graphical interface which could be used by the Scottish Government's Marine Directorate biologists and stakeholders' non-specialist staff who do not have computer science and coding expertise.

The Marine Directorate collects a large amount of underwater video for a number of different purposes. Analyses of these video data is time-consuming, often requires a skilled taxonomist and hence constitutes a significant draw on resources. The high cost of the analyses of this large amount of data often results in situations where only a subset of the available video is fully analysed. Consequently, automated video analysis software performing the above tasks would be highly desirable as it would reduce the costs of carrying out the analyses and allow for the analysis of all available data. It is expected that due to a steady and rapid improvements in new sensor/camera technology and their decreasing costs, the amount of video data available will only increase making the current processing bottlenecks even more acute.

To achieve that goal, the Scottish Government funded an earlier piece of work in this area - Automated Identification of Fish and Other Aquatic Life in Underwater Video (Blowers et al. 2020) in which the authors reviewed current image and video analysis methods and how these can be applied to different types of video footage and data extraction requirements used by the Marine Directorate. The authors also made recommendations for how video analysis could be automated using state of the art and open-source machine learning (deep learning) algorithms. We have followed the recommendations of Blowers et al. (2020) closely, making a number of important refinements. Our machine learning solution has been integrated and deployed as a user-friendly web application - AVIMS.

AVIMS allows users without computer science / coding experience to create, train and execute machine learning models without any need for interaction with the underlying code. The application supports computer vision models which fall into the common framework of detection of individuals from a predefined set of marine species, tracking detected individuals across the consecutive frames in the video, and finally, counting all distinct entities in the video for each species of interest (detect/track/count).

The workflow of our web-based application implementing the above detect/track/count computer vision framework allows users to: create new survey types; define a set of species of interest for each survey type; upload video and image data for training machine learning models; annotate video and image data with objects of interest; create datasets comprising annotated

data for training machine learning models; train machine learning models; upload new videos for the analysis by the machine learning models created in the previous steps of the workflow and view or download the analysis results.

The web application uses distributed computing to perform the required tasks. The computationally intensive tasks which include training machine learning models and analysing new videos are sent to a separate machine specifically equipped to handle this type of computations where they wait for their turn in a queue.

The web application has been tested by the development team and Marine Directorate scientists on several survey types including overhead in-river fish counters, salmon smolts entrained by a trawl and videos of the seabed. The initial machine learning models created in the web application have been shown to perform the required tasks. This said, in order for the system to achieve the level of accuracy that is expected from a practical application, the current small amount of annotated video data will need to be expanded to allow the machine learning models to better learn the appearance of various marine species of interest. This can be achieved from within the delivered AVIMS application.

Contents

Abbreviations

AVIMS	Automated Video Identification of Marine Species
CNN	Convolutional Neural Network
GPU	Graphics Processing Unit
GUI	Graphical User Interface
mAP	Mean Average Precision
OS	Operating System
UEA	University of East Anglia

Introduction

This report describes the system called the Automated Video Identification of Marine Species (AVIMS) that was developed by the authors as part of the Scottish Government contract, Ref: CASE/216380.

The objective of this project was to develop an automated video analysis application with a user-friendly interface which could be used by Marine Directorate biologists and stakeholders' non-specialist staff, without the need for coding/computer science expertise. The requirement was that only open-source algorithms are to be used.

The project was meant to build upon a previous project funded by the Scottish Government - Automated Identification of Fish and Other Aquatic Life in Underwater Video (Blowers et al., 2020), which reviewed and recommended various approaches to automated image analysis in underwater video. The immediate aim of this project was to produce a software tool to allow the Marine Directorate to make cost-effective use of the video data which is used to underpin scientific advice for Ministers.

Marine Directorate collects a large variety of underwater video for a number of different purposes. The data comes from cameras which can be towed (e.g. images of the seabed) or fixed (e.g. attached to drop-frames or trawl nets, or deployed at underwater turbines or in-river fish counters).

Analyses of these video data is time-consuming, often requires a skilled taxonomist and hence constitutes a significant draw on resources. The high cost of the analyses of this large amount of data often results in situations where only a subset of the available video data can be fully analysed. Consequently, an automated video analysis software performing the above tasks would be highly desirable as it would reduce the costs of the existing activities and allow for the analysis of all available data. It is expected that due to a steady and fast improvements in the new sensor/camera technology and their decreasing costs, the amount of video data available will only increase making the current processing bottlenecks even more acute.

To achieve that goal, the Scottish Government funded an earlier piece of work in this area (Blowers et al., 2020) where the authors reviewed current image and video analysis methods and how these can be applied to different types of video footage and data extraction requirements used by the Marine

Directorate. The authors also made recommendations for how video analysis could be automated using open-source machine learning algorithms. Our work builds on some of the recommendations from that work as well as on our own expertise in the field of computer vision and software development.

The following three sections describe the Methodology and Design of the AVIMS web application, the computer vision approach we have chosen, the hardware and software specification, the datasets and experiments, and the other resources.

Methodology and design

The overall objective of this project was to develop an application that provides a user-friendly interface that allows Marine Directorate biologists – who do not have machine learning or programming expertise – to train computer vision and machine learning models for the purpose of analysing video footage, detecting and recognizing a range of species of fish and benthos.

Our developed system is built on the work of (Blowers et al., 2020) and our prior experience from the EU funded H2020 Smartfish (SMARTFISH H2020 – Innovation for Sustainable Fisheries (smartfishh2020.eu), French et al., 2020) and JellyMonitor (French et al., 2018; Gorpincenko et al., 2020). Blowers et al. provided a feasibility study that considered a handful of entities of interest extracted from a few hundred video frames. Their work demonstrated the power of convolutional neural networks (CNNs) over other more traditional approaches and suggested a possible system architecture. We followed Blowers et al.'s recommendations closely, making a number of important refinements based on our experience developing larger scale systems capable of handling many more entities of interest.

In the initial phase of the project, a number of meetings with Marine Directorate scientists took place which helped us to design and iteratively refine our approach so that the end product met Marine Directorate requirements. During this phase of the project we also obtained the video footage from Marine Directorate which included overhead or underwater footage from fish counters, fish surveys and seabed surveys. In this phase we considered whether our system will be a standalone desktop or a web based application. We decided to proceed with the latter as this was considered to meet the project requirements. The web based solution also offered the

possibility of concurrent and remote working with the evolving prototypes of the tool by a number of Marine Directorate scientists. This work included the upload of the datasets, the annotation of parts of the uploaded datasets and provision of feedback to the development team.

Our application allows users without computer science / coding experience to create, train and execute computer vision models without any need of interaction with the underlying code. The application supports computer vision models which fall into the common framework of detection of objects from the predefined set of classes (schema), tracking detected objects across the consecutive frames in the video, and finally, counting all distinct objects in video for each class in the schema.

The workflow of our web-based application which implements the above computer vision framework is as follows:

Survey type. The application supports users in accomplishing a wide variety of tasks, e.g. counting fish and benthos in footage of the seabed or taken in fish survey work, and counting salmon in overhead footage from in-river fish counter sites. Given the differing appearance of the targets and the background in these tasks, the best performance is obtained by training separate models, even if both tasks share a general goal - counting entities of interest in a video. The application permits users to create different survey types for the various tasks that they wish to train models for.

Schema design. Given that each survey type is aimed at accomplishing a different task, the entities of interest that the user wishes to quantify are likely to differ. The application therefore provides an interface that allows the user to specify the list of species or classes of fish or other entities that the model should count within the video. Each survey type has its own schema that corresponds to the goals of the survey.

Video import. Videos selected by the user are added to the survey type with a view to extracting training images.

Extraction of images from video. Still images are extracted from the video files to be annotated by the user in the subsequent stages. Here, the user is required to choose which frames in the video they wish to include in the dataset that will be used to train and test the machine learning model/s. While extracting and annotating every frame may yield a robust model, the effort

required to manually annotate potentially thousands of images will likely make this prohibitive. The user is presented with the graphical tool utilizing a traditional video slider where they scroll the video and choose the video frames they wish to be included in the dataset. These should be representative of the conditions in which the model is required to work.

Annotation. The selected frames from the imported video/s need to be annotated by expert taxonomists in order to provide the ground-truth data for training and testing the trained machine learning models. The annotation tool, which was built on our earlier work (GitHub - Britefury/django-labeller: An image labelling tool for creating segmentation data sets, for Django and Flask (github.com)), is presented to the user, allowing them to identify entities of interest and annotate them, either using polygonal annotations or ellipses. We anticipate that after testing the model the user may wish to annotate more data in order to improve the model performance.

Dataset construction. Machine learning requires an annotated training and testing set for training the model and evaluating its performance respectively. Segments from a single video are likely to share common visual appearance characteristics, e.g. lighting or turbidity due to time of day and weather conditions. Training a model on footage drawn from one part of a video would likely result in higher testing scores on footage from a different part of the same video than is achievable in the desired practical scenario of using the model to analyze footage from an as-of-yet unseen site. Here, our system allows for a number of approaches regarding how the data should be split so that the testing results are indicative of the future performance.

Training and Testing. A subset of annotated images is used to train the model. This involves iteratively presenting images to the neural network and optimizing the network parameters. This is computationally expensive and can take hours to complete on a GPU server. A proportion of the dataset constructed above is marked for testing. These images are held out during training i.e. are not part of the training set, and given to the model for inference at this stage. The predictions generated by the model are compared to the manually generated ground-truths. The system determines which objects of interest the model failed to detect (false negatives) and which detections produced by the model were spurious (false positives). The user is presented with these results.

The mAP (mean Average Precision) is used as a measure of system accuracy during training. This figure combines model accuracies (average precisions) for each class separately, which are then averaged, forming one figure - mAP. For example, a mAP of 80% should be highly reliable over all classes. In this case, a user can be relatively confident about the model predictions and the subsequent tracking results, as long as the video passed for the subsequent analysis/inference (see Inference below) is "similar" to what the model has been trained on.

Object tracker. Without an object tracker it is not possible to count objects in a video. The object detection model trained in prior stages detects entities in single frames. Using this alone would result in a significant over-count as an entity will be counted multiple times, once for each frame in which it is detected. The task of the object tracking system is to associate detections that correspond to a single entity across the frames in which that entity is visible. The user is presented with an interface that allows them to tune the tracker in order to achieve the most accurate results.

Inference (analysis of new videos). Once a model has been trained and achieved an acceptable level of accuracy, the system uses the trained model to analyze videos provided by the user, outputting the desired results. This stage can also be computationally expensive (especially for long videos) and usually requires high performance computer hardware.

Here, the user chooses a machine learning model for the task at hand and selects a set of videos for analysis. Once selected, the system processes the videos in turn, saving the event logs to a .csv and .json files which are available to the user for viewing either from the application GUI or for the download. The event logs contain an entry for each detected entity. The entry consists of a time stamp, the event duration, and a predicted class label of the object detected.

Moreover, we considered adding a confidence score for each detection. However, eventually we decided not to add this feature as we did not have a reliable way of generating a confidence score that is correct and meaningful. While the system could provide such scores in theory, our experiments/observations show that these would not be helpful i.e. they are not really what a human would consider confidence/probability.

At the end of this stage, the user is also given an opportunity to add the video/s that have just undergone inference to the training dataset to improve the potential subsequent iteration of the machine learning model. This may be particularly worthwhile in cases where videos are substantially different to those present in the dataset used to train (and test) the model and have consequently returned unsatisfactory results in the inference stage. This maximizes the improvement in performance obtained by expanding the machine learning dataset.

Our application supports a rapid annotate and test cycle, allowing the user to quickly assess the performance of the model and grow the training set in order to achieve the desired level of performance as quickly as possible.

Computer Vision Approach

The AVIMS system adopts a detect and track approach. An object detection model is applied to each frame in a video. The object detections resulting from this are passed to an object tracker that joins them into tracks that correspond to objects that are visible throughout multiple frames of a video.

Object Detection

Further to the recommendation of (Blowers, Evans and McNally, 2020), our application supports instance segmentation in addition to object detection. Where an object detection algorithm detects the presence of an object in an image and estimates its size, an instance segmentation algorithm proposes an outline surrounding the region of the image that the object covers. This can be advantageous in situations where the object density is very high, as the rectangular region proposed by an object detector can cover several objects, making the intended target ambiguous.

The work of (Blowers, Evans and McNally, 2020) outlines technical details of some of the components described below. They discussed the Google Object Detection API – that is part of Tensorflow – as a potential approach for object detection. Tensorflow is one of a number of open source deep learning systems that are available.

Many of the computer vision libraries provided by deep learning toolkits implement the Faster R-CNN object detection algorithm that proved to be a strong contender in (Blowers, Evans and McNally, 2020). It is expected that

the various implementations will offer similar accuracy. One such implementation we have used in the past is provided by the torchvision library (Torchvision 0.16 documentation (pytorch.org)) that is part of the PyTorch ecosystem (PyTorch - an open source machine learning framework (pytorch.org)). PyTorch is an open source deep learning toolkit developed by Facebook AI Research (FAIR) whose goals are very similar to those of Tensorflow. We also note that torchvision provides an implementation of the Mask R-CNN instance segmentation algorithm that we wished to utilise. Our choice of Pytorch over Tensorflow has been motivated by the ease of development using PyTorch in comparison to other systems and the ease with which it can be incorporated into the final deliverable application. Other systems including TensorFlow often require the installation of several dependent packages, complicating the installation process and also subsequent maintenance.

Consequently, in this application we utilize the Faster-RCNN algorithm implementation provided by torchvision. They provide a model that combines an ImageNet (ImageNet (image-net.org)) pre-trained ResNet-50 (Deep Residual Learning for Image Recognition (arxiv.org)) backbone with an FPN head (Feature Pyramid Networks for Object Detection (arxiv.org)) and is fine-tuned for object detection using the CoCo dataset (COCO - Common Objects in Context (cocodataset.org)). We adapted this network by removing the final class prediction and bounding box regression layers and replacing them with new layers with the appropriate outputs for the object classes defined by the labelling schema designed by the AVIMS site users. The network was fine-tuned using the training dataset.

AVIMS site offers the option – in fact defaults to it – of detecting objects as oriented ellipses rather than axis-aligned bounding boxes. Rather than using a specific oriented object detection network, we train a Mask-RCNN (Mask R-CNN (arxiv.org)) instance segmentation model. We convert the oriented ellipse labels to masks with oriented elliptical shapes that are used as Mask-RCNN training targets. Our algorithm takes the predicted instance mask and computes the closest fitting oriented ellipse using the *regionprops* function provided by the SciKit-Image library (scikit-image: Image processing in Python (scikit-image.com)).

Object tracking

We noted the challenges posed by the requirement of accurate object tracking. Blowers, Evans and McNally (2020) found that inaccuracies in this component resulted in overcounting. We also noted the suggestion that the DeepSORT algorithm might provide a good solution to our object tracking problem.

Accurate object tracking is dependent on accurate input from the object detector (Bewley et al. 2016); a prior stage in the inference pipeline. Inaccurate predictions from the detector are likely to negatively impact tracking performance. Furthermore, we note that the DeepSORT algorithm uses an association metric model for the purpose of object re-identification. It is used to estimate the likelihood that detections from different frames represent the same object. This model is trained using an annotated object tracking dataset. An object tracking dataset consists of annotated videos rather than single images and that each object must be annotated in every frame in which it can be seen. We note that annotating all frames in a video substantially increases the amount of manual effort required. While it is possible that the annotation tool forming the part of our web application could be extended for this purpose, the human effort required to annotate enough videos to train a sufficiently accurate model can make this approach practically infeasible. This is the reason why we have chosen the SORT algorithm (Bewley et al., 2016) for our object tracker; effectively DeepSORT without the association metric model. SORT takes the predictions of our object detection model as input and joins the per-frame predictions into object tracks. We use our own implementation of SORT. We use the Kalman filter implementation provided by the *filterpy* library.

Hardware and Software Details

In the following two subsections, we give hardware and software details of the application website (AVIMS site) and of the GPU worker (AVIMS worker) where the time consuming machine learning training and inference are performed.

AVIMS site

Server Hardware and Operating System (OS)
Our web-server is a virtual machine with 4GB of RAM running with a larger server. We use the Ubuntu Linux OS and NGINX web server. Our data is stored in a PostgresSQL database. Large files such as image and video files are stored on the file system.

Web application
Our web application uses the Django web application framework (The web framework for perfectionists with deadlines | Django (djangoproject.com)). Django provides an object relational mapper that provides object oriented access to the rows stored in the PostgresSQL database. Its template system simplifies the design of HTML web pages that incorporate data extracted from the database.

We utilized the Twitter Bootstrap 4.3.1 framework (Introduction · Bootstrap (getbootstrap.com) to provide layout and user interface controls. AVIMS uses the Django-labeller labelling tool (GitHub - Britefury/django-labeller: An image labelling tool for creating segmentation data sets, for Django and Flask (github.com)) to allow users to annotate images quickly and effectively.

Video handling
The video files provided by the Marine Directorate came in a variety of formats, depending on their source. Early in the project we found that it was necessary to convert the video files to a common format. For this, we adopted FFMPEG (FFMPEG - A complete, cross-platform solution to record, convert and stream audio and video (ffmpeg.org)). FFMPEG was also able to convert interlaced video to a non-interlaced format in the instances where this was necessary.

AVIMS worker

Hardware
Training object detection models and analysing video files requires the use of a GPU. We therefore performed these tasks on a separate machine with the nVidia 1080-Ti GPU, 32GB RAM and an Intel Core i7 CPU.

Software
We used the Python Celery distributed task queue (Introduction to Celery — Celery 5.3.4 documentation (celeryq.dev)) to provide a task queue to run the machine learning tasks required by AVIMS.

When an AVIMS user initiates a model training or video analysis job, the task is appended to the celery queue that runs on the web server. This job is retrieved by the GPU worker machine that downloads the relevant data from the web server. It runs the task while periodically reporting progress and finally uploads the results to the web server on completion.

Datasets and Experiments

During the development of the web application, we were provided with several video datasets which were meant to aid the development and testing of various aforementioned features of the AVIMS application. The datasets have been provided throughout the project by means of the evolving prototypes of the web application. They have been partially annotated by biologists at Marine Directorate using AVIMS tools. This allowed us to produce initial machine learning models and test further functionalities of the AVIMS application including model training, results reporting and inference.

The datasets provided have been split according to a survey type as described in the Methodology section and included : *Sea Pens, Rocky Reef, River – infrared, River – daylight, Vaki light boxes, Smolt Trawl, fan mussels and horse mussels*.

It is important to emphasise that the machine learning experiments performed here were not meant to measure the algorithm performance that might realistically be expected by Marine Directorate for each of the survey types in future, but rather help us to develop and debug the machine learning applications and teach the Marine Directorate future users the most appropriate ways of creating datasets, training models and analysing results.

The numbers of labelled images in each of the above datasets were very low for the standards of modern object detectors utilising deep learning such as those we use here. The datasets included no more than 200 image frames in most cases, which often came from one or just a few videos. Further, some classes in the chosen schemas were very poorly represented in the training sets (heavily imbalanced datasets). Hence, it is not surprising that the

resulting machine learning models could not generalise adequately, and the reported performance of the system as measured using mean average precision of the object detector was below the level that we would like to see in a practical application.

The most satisfactory initial results were obtained for counting fish in overhead in river counters for the *River Daylight* and *River Infrared* datasets. The machine learning models were trained from datasets comprising 81 labelled fish and 61 otters (River Daylight) and 208 fish (River Infrared). Here, the mean average precision ranged from 9% to 55% for a very challenging condition where the test camera was not part of the training set, and for an easier (and unrealistic) condition where some images from the training video (not the same as in the training set) were part of the test set. The analysis of the inference results confirms that the system was able to track fish and otters. Some overcounting took place usually in places where unusual (unseen during training) water turbulences were present, but these could be eliminated if more images were available for training.

For *Smolt Trawl* and *Vaki light boxes* datasets, we have observed that the chosen class schema (which included sex differentiation for salmon) was most likely too ambitious for the limited size of the training set and consequently resulted in low mean average precision of the object detector and ultimately significant overcounting of fish in the videos. This teaches us that despite the fact that fish were clearly visible in the videos provided (better visibility than in overhead in-river counters discussed above), having too many (similar) classes of fish may significantly degrade the performance of the detector and consequently the tracker, in particular where the training set contains few images.

The *Sea Pens* dataset has proven to be particularly challenging. The image features are arguably more difficult to spot for the human observer than the previous applications. The chosen class schema contained 10 classes of which 7 had no more than 20 samples in the dataset. Consequently, the reported mean average precision of the object detector was low and the object tracking resulted in noticeable overcounting. The overcounting was particularly significant for those parts of the videos where image features related to the disturbed sediment (not seen in the training set) were present.

The machine learning experiments performed to this stage have proven that the developed features of the web application work correctly for all datasets.

The performance of machine learning algorithms do vary between different survey types and will require careful design of the class schema for each survey type and the annotation of a greater number of images than were available during the development of this application.

Other Resources

Further guidance on the features and the use of the AVIMS web application are available to authorised users of AVIMS. The guidance includes an AVIMS User Guide and Tutorial slides (see Annex Materials).

References

Bewley, A., Ge, Z., Ott, L., Ramos, F. and Upcroft, B. 2016. Simple online and realtime tracking. In 2016 IEEE International Conference on Image Processing (ICIP), 3464-3468.

Blowers, S., Evans, J. and McNally, K. 2020. Automated Identification of Fish and Other Aquatic Life in Underwater Video. In: Scottish Marine and Freshwater Science, Vol 11 No 18. 62pp. DOI: 10.7489/12333-1.

French, G., Mackiewicz, M., Fisher, M., Challiss, M., Knight, P., Robinson, B. and Bloomfield, A. 2018. JellyMonitor: automated detection of jellyfish in sonar images using neural networks. In: IEEE International Conference on Signal Processing (ICSP), IEEE, 406-412.

French, G., Mackiewicz, M., Fisher, M., Holah, H., Kilburn, R., Campbell, N. and Needle, C. 2020. Deep neural networks for analysis of fisheries surveillance video and automated monitoring of fish discards. ICES Journal of Marine Science, 77(4), 1340-1353.

Gorpincenko, A., French, G., Knight, P., Challiss, M. and Mackiewicz, M. 2020. Improving Automated Sonar Video Analysis to Notify About Jellyfish Blooms. IEEE Sensors Journal.

Annex Materials

Please see PDF supporting documents containing

1. User Guide
2. Tutorial Slides

Contact

Email: craig.robinson@gov.scot

© Crown Copyright 2023

Marine Directorate of the Scottish Government
Marine Laboratory
375 Victoria Road
Aberdeen
AB11 9DB

Copies of this report are available from the Scottish Government Publications website at https://www.gov.scot/publications/

Scottish Government
Riaghaltas na h-Alba

© Crown copyright 2023

OGL

This publication is licensed under the terms of the Open Government Licence v3.0 except where otherwise stated. To view this licence, visit **nationalarchives.gov.uk/doc/open-government-licence/version/3** or write to the Information Policy Team, The National Archives, Kew, London TW9 4DU, 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.

This publication is available at **www.gov.scot**

Any enquiries regarding this publication should be sent to us at
The Scottish Government
St Andrew's House
Edinburgh
EH1 3DG

ISBN: 978-1-83521-505-0 (web only)

Published by The Scottish Government, November 2023

Produced for The Scottish Government by APS Group Scotland, 21 Tennant Street, Edinburgh EH6 5NA
PPDAS1347782 (11/23)