The Alan Turing Institute

Data Study Group Final Report:

Centre for Environment, Fisheries and Aquaculture Science (CEFAS)

Automated identification of sea pens using OpenCV and machine learning

12-16 December 2022

https://doi.org/10.5281/zenodo.8329169

Contents

1 Executive summary

1.1 Challenge overview

Sea pens are colonial animals closely related to the corals that live attached to the sea floor. They generally form a feather-shaped colony with a horny skeleton. They respond to multiple human interventions (contamination and human waste, fishing, boats, etc.) and therefore they are an indicator of the health of their ecosystem. Sea pen communities are Vulnerable Marine Ecosystems (VMEs), which have been proposed as indicators of good conservation status for mud habitats. The goal of this challenge is to study their distribution and abundance, so their presence can be used to evaluate whether the seabed is exposed to good water quality and is undamaged by fishing gear.

The Centre for Environment, Fisheries and Aquaculture Science (Cefas) studies the distribution of sea pens to obtain marine environmental insights about the health of mud ecosystems. From previosly acquiered video footage, they were interested in classifying two types of sea pens which are distributed at very different densities in two different survey locations: the slender sea pen (*Virgularia mirabilis*) and the phosphorescent sea pen (*Pennatula phosphorea*) (Figure 1). Both sea pens live in fine sediments ranging from sheltered inshore waters to deeper water offshore ($\sim$ 10-400 m depth). The slender sea pen has a central stem only a few millimetres thick lined by small tentacled polyps arranged in two opposing lateral rows on the central stem. The colony varies in colour from white to creamy yellow and can grow up to 60 cm long. Colonies of the phosphorescent sea pen on the other hand are stout and fleshy and up to 40 cm long. The triangular leaf-like branches formed of fused polyps range from translucent to a deep reddish-pink colour.

Prior to this DSG, Cefas had created an analysis pipeline using preliminary machine learning algorithms which would classify sea pens from seafloor video footage with varying accuracy. A key challenge identified was that of standardising all video data, which showed different levels of brightness, contrast and other characteristics. This was rather challenging, as different years use different lighting systems, and it can be hard to spot sea pens under LED lighting conditions. In summary, the

Figure 1: Pictures of the two sea pens of this challenge: *Virgularia* (left) and *Pennatula* (right).

preliminary models did not generalise well over different years, since the characteristics of the cameras and the lightning conditions changed over time. The report from this preliminary analysis can be found online.[1] The aim of this DSG project and report is to attempt to find a machine learning pipeline based on computer vision techniques that can accurately classify and detect these two types of seapens across footage from several years, as well as gain some insights into which video modification techniques can be applied to homogenise the different video conditions over the years.

1.2 Data overview

Cefas undertakes an annual Nephrops (a genus of lobster) burrow survey and has several decades of video data totalling terabytes, ranging from 2007 to 2022. The data was collected from a video camera mounted on a towed camera sledge system deployed from the Research Vessel Cefas Endeavour. These videos were used to count sea pens for this challenge, in addition to their original purpose. Cefas had performed manual annotations of sea pens using a third-party software by selecting

bounding boxes in certain video frames. These annotations were made in videos from 2014 to 2021 and were provided as ground truth labels to train computer vision models for sea pen detection and classification.

1.3 Main objectives

The overarching goal of this challenge is to improve the detection, classification and tracking of two types of sea pens in seafloor videos from different years and different recording conditions, ensuring that one sea pen is counted only once. The detailed objectives are:

1. To homogenise the brightness and hue conditions of the videos to generalise as much as possible the detection and classification of sea pens in future surveys.
2. To detect two laser beams that appear in the videos to define a region of interest so that the density of sea pens can be calculated accurately.
3. To detect sea pens in sea-floor footage with high accuracy using footage from different years.
4. To classify the two types of sea pens.
5. To track the sea pens, once classified, over several frames to make sure they are counted only once.

1.4 Approach

Aside from the writing of general-use functions, the Data Study Group decided to split the challenge into 5 work packages which could be developed in parallel: 1) Image standardisation and pre-processing; 2) Laser Identification; 3) Detection; 4) Classification; and 5) Tracking. In the following subsections, we briefly summarise the tasks performed by each of the sub-teams.

1.4.1 Image standardisation and pre-processing

A key issue with the challenge presented was that the previous machine learning models only worked well on the year of data that it was trained on (2016). Since footage in other years was filmed on a different camera-light system, the algorithm performed poorly on them. This quite likely results from a different hue, saturation and lightness in the different years. The team investigated several image pre-processing techniques that could homogenise these differences and highlight sea pens in all the videos available to us (2014-2021). Some of the techniques implemented where time-averaged hue-saturation-lightness (HSL), the grey-world (GW) algorithm, RGB histogram equalisation, and contrast limited adaptive histogram equalisation (CLAHE), as well as a multi-stage pre-processing. In particular, the CLAHE algorithm gave positive results and highlighted the presence of sea pens in the available video footage. We assess the quality of the pre-processing techniques qualitatively by visual exploration and we provide an initial quantitative analysis of colour normalisation in the appendix.

1.4.2 Laser identification

The seafloor videos have one set of parallel lasers separated a specified width of about 70 cm. Combined with knowledge of the length of the survey transect, this width can be used to standardise the number of sea pens between the laser lines to a known swept area. The identification of the lasers is, therefore, a useful outcome for Cefas, as it will allow them to only count the sea pens that fall within the lasers and compute an accurate density of sea pens in the area. The team used line-detection on the footage as the most reliable method for this task. The approach involved acquiring a grayscale image, applying Gaussian blur and Canny edge detection, and using the Hough lines transform to convert edges to lines. Finally, a binary mask was obtained, which would allow to filter out detected sea pens that fell outside the central laser region. In good visibility conditions, the algorithm performed returning an adequate mask in approximately 95% of the cases (upon qualitative assessment of whether two clear lasers were identified) in videos from 2014-2021 without applying standardisation techniques described in the previous subsection. In the remaining cases, visibility conditions make the lasers

easy to distinguish even by bare eye, but we suggest investigating the sources of errors to optimise this algorithm.

1.4.3 Detection

Before object identification, the video must first be processed to pick out objects, in this case sea pens, from the background. The team investigated detection and classification algorithms based on You Only Look Once (YOLO) v5. This model provided significantly positive results of $\sim$ 90% mean average precision (mAP) at 0.5 intersection over union (IOU) in our test dataset, which is an outstanding value for object detection. For this, we used videos from the years 2014-2020, excluding those from 2021 and some from 2016 due to some issues with the alignment of bounding boxes from the annotations, described in the next section. The videos were not pre-processed with the standardisation techniques studied in parallel due to time constraints, and we suggest as future work the implementation of the same detection algorithms on the homogenised videos to assess whether these pre-processing improves the detection of sea pens.

1.4.4 Classification

The detection algorithms based on YOLOv5 can also classify the objects as the two types of sea pens. However, due to the aforementioned difficult lightning conditions, the team suggested an approach in which the detection would be over-sensitive, that is, would detect too many false positives, and a subsequent classification algorithm would help discriminate between sea pens and background objects. The team proposed starting with baseline algorithms, such as a simple custom-made convolutional neural network (CNN) model. We also tested more complex algorithms, such as ResNet50 and VGG16. However, we should be cautious with the more complex algorithms, as they have been trained for object detection in completely different scenarios (e.g., cities, houses, etc.) and not for the detection of maritime animals like sea pens in maritime environments, with very different lightning conditions and noise levels. In any case, both the simple and complex models provided excellent results, with classification accuracy $>$94% in all cases and sometimes very close to 100% across all years. As in the detection case,

we did not use videos from 2021 and some from 2016 due to problems with bounding box alignment, and none of the videos were pre-processed beforehand. However, it needs to be noted that these results are validation scores at the end of training, and they need to be tested more cautiously with a test dataset, but the model loss function does not suggest the presence of overfitting.

1.4.5 Tracking

In order to count sea pens only once, it is necessary to have a tracking algorithm that can follow the detected sea pens as the camera moves along the seabed. Therefore, the tracking algorithm should track their movement, as well as their potential disappearance, as the footage was frequently obscured by plumes of sediment. A tracking algorithm that could detect sea pens was successfully implemented, although more testing is necessary. This algorithm was capable of very robustly following sea pens across the video, and even recover their positions when they are momentarily obscured. However, in certain occasions sea pens were occluded for a longer time and the tracker would lose them, potentially causing counting the same sea pen twice. Due to time constraints, this implementation could not be assessed systematically, neither qualitatively nor quantitatively, and therefore more research is needed to investigate the sources of errors and its generalisability. We tested this approach on one video from 2018, but previous experience of the team with this approach tells us that the methodology is robust to different video conditions and should be easily generalised, and that further optimisation of the algorithm could solve all of the issues encountered.

2 Data overview

2.1 Dataset description

The data available for this challenge was camera-sled video footage covering 2014-21 and associated box annotations of both types of sea pens (see Section 2.2). Each video was captured using a sledge-mounted camera for 10 minutes at an average speed of 0.7 knots. Two parallel lasers at a known distance are used to delimit the field of

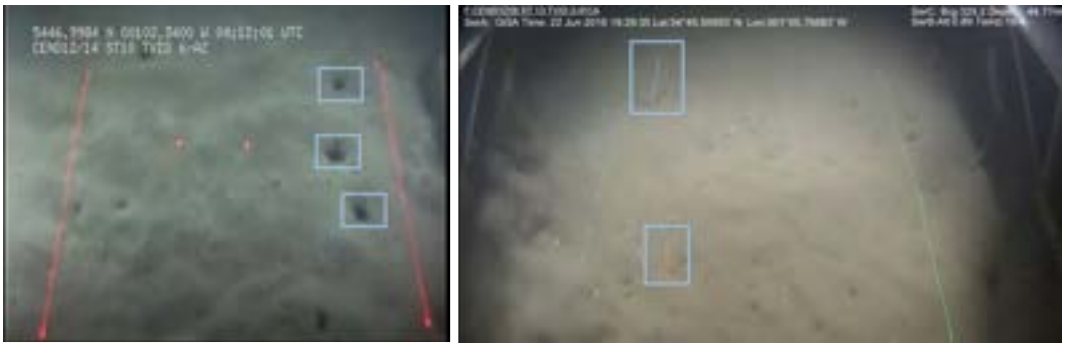

Figure 2: Two snapshots from seafloor footage from different years (2014 and 2017, respectively) where the different lightning conditions, aspect ratios and image quality can be observed. The two different types of sea pens are displayed in boxes.

view. Figure 2 shows two snapshots of videos from different years (with bounding boxes from the original human annotations) where many of the challenges of this project can be observed. Most notably, both videos have different lightning and hue conditions. The two lasers used to delimit the field of view are also visible in red and green.

2.2 Data summary

For the challenge, there were about 120 videos per year, from 2014 to 2021. Six of these videos from 2016 and 2 videos each from 2014-2021 (16 videos) are accompanied by handmade annotations using the software VIAME [2], which allows the user to delimit bounding boxes of objects (in this case, sea pens) and extract their coordinates. These annotations were done by researchers trained to identify sea pens in the videos and can be used to train computer vision models to detect these sea pens over several years. In more detail, there were three types of manual annotations:

- **Individual annotations:** each sea pen, individually identified, was delimited only once in a single frame. There is no subsequent tracking in time.
- **Track annotations:** the annotations for each sea pen are tracked in time, individually recognised, over several frames. Notice that to

simplify this process, the original videos (generally at 25 FPS) were downsampled to 10 FPS, so that human annotators would have to annotate less frames while still providing a useful time series.

- **Mixed annotations:** in this case, some datasets have a single annotation of identified sea pens, but others have track annotations. All of them come with unique identifiers, so they are easy to separate.

2.3 Data naming system and standardisation

The naming system of the videos and associated annotation files was not consistent, due to different persons at different times not completely following the standard advised. For instance, uppercase and lowercase letters were used in different files; a suffix to indicate the number of the recording station was sometimes preceded by 'ST' or 'STN', in upper- or lowercase; three-digit codes were sometimes written as, e.g., 003 or simply 3; and division between codes was indicated with a space or an underscore.

The challenge owners provided us with a glossary explaining the naming system and pointing out these discrepancies. As our goal was to obtain a fully automated system for sea pen detection, it was necessary to homogenise the naming system. We created a bash script that looped through all the files and modified their names according to the following convention: CEND*XXYY*_STN_*DDD*_TVID_*N*-*SC*, where CEND is the survey cruise code (always CEND, meaning Cefas Endeavour); *XX* is two digits identifying the number of the survey in year *YY* (last two digits of the year); *DDD* is the sampling station (STN) number of the survey; and finally the permanent station identifier, where TVID means TV survey, *N* is the area code digit(s) and *SC* are 1-4 letters identifying the station code. For instance: CEND0821_STN_002_TVID_6.

2.4 Annotations

Table 1 shows an overview of the data contained in the annotation files, focusing on the column names, the data type and their contents. Although these annotation files were already in a clean format, the team

Column name	Data type	Description
1: Detection or Track-id	INT	Unique sea pen ID
2: Video or Image Identifier	TIMESTAMP	This data was incorrect and therefore discarded
3: Unique Frame Identifier	INT	Frame number (starting at 0)
4-7: Imgbbox(TL_x	INT	Pixel coordinate of the top-left x-axis value of a bounding box
TL_y	INT	Pixel coordinate of the top-left y-axis value of a bounding box
BR_x	INT	Pixel coordinate of the bottom-right x-axis value of a bounding box
BR_y	INT	Pixel coordinate of the bottom-right y-axis value of a bounding box
8: Detection or Length Confidence	FLOAT	As annotations were manual, this column was always 1, and therefore discarded
9: Target Length (0 or -1 if invalid)	INT	This value was always -1 and therefore discarded
10-11+: Repeated Species	STRING	Name of the species
Confidence Pairs or Attributes	FLOAT	This value was always 1, since they were human annotations and was therefore discarded

Table 1: Column names in the dataset containing manual human annotations provided by Cefas (exported from the *VIAME* annotation software).

performed some further cleaning in these files so that only the relevant columns were kept and the column names were simplified to better reflect their contents. Table 2 shows an example of one of the annotation files after cleaning.

2.5 Data quality issues

The footage was collected on a towed platform (sledge) and the quality of the footage is dependent on the sea conditions at the time. For instance, waves in rough sea conditions will cause the horizontal vessel speed to

detection_id	frame_number	TL_x	TL_y	BR_x	BR_y	species
0	0	1126	508	1170	540	Pennatula
0	1	1125	508	1166	543	Pennatula
0	2	1127	519	1163	552	Pennatula
0	3	1131	518	1172	558	Pennatula
0	4	1131	523	1172	563	Pennatula
⋮	⋮	⋮	⋮	⋮	⋮	⋮
1	0	906	328	953	381	Pennatula
1	1	899	329	952	386	Pennatula
1	2	901	335	953	385	Pennatula
1	3	906	341	946	385	Pennatula
1	4	908	347	955	393	Pennatula
⋮	⋮	⋮	⋮	⋮	⋮	⋮

Table 2: An example of a cleaned annotation dataset.

be less consistent, as well as lifting the vessel vertically. Typically, some small plumes of sediment may pass in front of the camera at the peripheries. In worse conditions, the sled accelerates and decelerates abruptly. At times, the sled may even lift off the seafloor entirely. When this occurs, the transect is not useful for reliable species counting and is usually abandoned.

Before 2016, a Kongsberg/Simrad 14-408 camera system was used. The camera used for 2014 and 2015 had a width-to-height aspect ratio of 4:3, which enabled a close-up view of the seabed, with a small amount of coverage outside of the fan laser lines (left of Figure 2). For the 2016 to 2019 surveys a Subsea Technology and Rentals (STR) high-definition Internet Protocol (IP) camera was used and for the 2020 and 2021 surveys an STR SeaSpyder High Definition (HD) camera was used, further increasing the achievable pixel resolution. The IP and HD camera both have an aspect ratio of 14:9, extending the area captured outside of the laser lines and bringing the sledge frame into view (right of Figure 2).

There are also extrinsic parameters which have altered between surveys.

There have been changes in both the camera mounting height in the frame and the angle pointing towards the seabed. This is particularly noticeable in the 2019 data, where the camera angle is more vertical than the other surveys, and the sledge skids are only just visible in the frame. Due to the increase in mounting height, there has only been a slight increase in pixel resolution on the seabed.

2.5.1 Misalignment of frame rates and bounding boxes

A significant issue arose when matching the annotations' frames with the corresponding video frames, due to the software that was used to create the human annotations. Each row of the dataset specified a frame number ('frame_number' column in the clean dataset from Table 2), along with coordinates for the bounding boxes enclosing any sea pens present within that frame. However, the frame numbers specified in the dataset did not match those in the videos. When the bounding boxes were drawn on the videos, these did not seem to correspond to actual sea pens. This was a significant bottleneck for the whole challenge, as the detection, classification and tracking subteams counted on being able to manually extract bounding boxes with sea pens from the manual annotations.

These human annotations were done on a video that was downsampled to 10 frames per second (FPS) within the annotation software VIAME, from their original 25 FPS[1]. This was done so that human annotators did not need to annotate 25 frames each second, which would be significantly cumbersome. Downsampling to 10 FPS decreased the number of annotations needed while still being able to track the sea pens in time with reasonable FPS. However, when these downsampled videos were exported again from the software, it appeared that the software automatically compressed the video and converted them back to 25 FPS through an unknown process. It was unclear to what the 'frame_number' referred to: i) to the frames of the original video; ii) to the video downsampled to 10 FPS; or iii) to the new 25 FPS video, resampled from the 10 FPS video. Notice that this was not an obvious problem, as the

[1]Note that not all the original videos had exactly 25 FPS, as some of them had, for instance, 24.92 FPS. Fractional FPS are not uncommon in older recording devices, as this is done to avoid synchronisation issues between audio and video for historical reasons. For simplicity's sake, we will generally refer to the original videos having 25 FPS.

ratio between FPS, namely $25/10 = 2.5$, which is not an integer. This means that when VIAME resampled the original video into 10 FPS and then again to 25 FPS[2], some non-trivial decisions were made to adjust the frames.

We explored several approaches to try and gain an understanding about what was going on with the frame IDs and video frame rates. Comparing the number of frames between the original and 10 FPS videos revealed that, contrarily to what could be expected, the videos generated by VIAME at 25 FPS generally had more frames than the original 25 FPS videos, despite supposedly having the same contents. The difference in frames was generally small enough to come from rounding issues when working with fractional FPS and converting videos from one FPS to another.

However, one striking example of this difference happened with the 2021 videos. In one of them, the original 25 FPS video had a total of 15,207 frames. The equivalent downsampled video which also had 25 FPS had, however, 18,276 frames. The videos were of the same duration and were supposed to have the same information, so this difference in frame probably meant that extra frames had been added to the video generated by VIAME. We hypothesised that VIAME first converts the videos to 30 FPS, which allows to easily dividing them over 3 to reach 10 FPS. Then, it converts them back to 25 FPS with these extra frames, that most likely arise from frame duplication at certain points along the video. To initially corroborate our hypothesis, we performed the following calculation. The duration of the original 25 FPS video needed to be $15,207$ frames$/25$ FPS $= 608.28$ s. If it was converted to 30 FPS by duplicating certain frames, the total number of frames would be 608.28 s $\times$ 30 FPS $= 18,248.4$ frames. This number falls close to the observed total number of frames, 18,276, and the difference could be due to rounding errors.

We played the 10 FPS videos in VIAME and in Python simultaneously for comparison, observing certain differences in the frames. This confirmed that exporting the videos from VIAME caused the misalignment in the annotations. An obvious linear relationship explaining the misalignment could not be observed from simple comparisons. We read through VIAME manuals and found out that the downsampling of videos is conducted using the computer vision tool KWIVER (Kitware Vision and

[2]In this case, the FPS was always exactly 25.

Image Reasoning) from the company Kitware.[3] A code example showing the downsampling approach is available at their FAQ page[4] and further information was found at a GitHub issue where they reported the same problem as us.[5]

Based on our experiments, we concluded that the 'frame_number' column from the dataset related to the original videos, but with a certain multiplier that depended on the original FPS and the downsampled FPS. We were able to identify the multiplier that allowed matching the video frame rates with the bounding boxes. The multiplier was calculated as follows:

$$\text{New ID}_f = \lfloor \frac{\text{ID}_f \cdot f_{ds}^{FPS}}{f_{ref}^{FPS}} \rfloor \tag{1}$$

where ID_f is the frame ID in the annotation file, New ID_f is the new frame ID that exactly matches the original video, f_{ds}^{FPS} is the frame rate after downsampling (in our case, always 10 FPS) in FPS, and f_{ref}^{FPS} is the original frame rate (generally around 25 FPS), in FPS. $\lfloor \cdot \rfloor$ represents the flooring operation as a rounding method to obtain and integer frame ID. We chose the flooring operation, as this was also used by Kitware and shown in their FAQ.[4]

We tested using this multiplier calculation method for videos from different years to make sure it was generalisable to all videos. The method worked for all videos expect those from 2021, which, as commented above, showed striking frame differences. When this multiplier approach was tested on them, the bounding boxes did not match exactly the sea pens in the videos. Our intuition told us that this multiplier seemed to be non-constant, as initially the bounding boxes matched sea pens, but they moved at different speeds. Due to time constrains, we could not fix this issue and the videos from 2021 were not used the detection, classification and training. We suspected the issue could stem from a formatting issue in VIAME, as the 2021 videos were in *.avi* format, whereas the rest of the videos were in *.mp4*. The VIAME software might use a different downsampling method for different file formats.

2.5.2 Excluded data

Out of the 21 videos provided, we excluded 4 of them:

- 'CEND0821_STN_008_TVID_6-AZ' from 2021, due to the bounding box matching issue.
- 'CEND0821_STN_093_TVID_6-B' from 2021, due to the bounding box matching issue.
- 'CEND1216_ST047_TVID_6-DM' from 2016, as we believed the detection IDs for individual sea pens were incorrect.
- 'CEND1216_STN_128_TVID_6-B' from 2016 was annotated twice, and only the annotation with sea pen tracking was kept.

3 Image pre-processing

The data used within this study originated from studies on Nephrops (a genus of lobster) burrows between the years 2014 and 2021. Due to the extended period of these surveys and changes in survey equipment, there is significant variation in the appearance of the video footage available for this study. Due to differences in lighting hue, for example due to lighting type (halogen versus LED) and lighting geometry, the videos appear significantly different from each other between years (Figure 3).

Analysing images or video footage with such variance between years can lead to inaccurate conclusions about the distribution of sea pens, especially when using models trained on specific subsets of the video dataset with variable lighting conditions. A key step to avoiding the challenge of variable light conditions is to use colour normalisation algorithms to try and account for the differences between years [6]. Here we explore several image pre-processing algorithms to identify potential approaches for the normalisation of the Cefas video dataset to facilitate the complete analysis of the years 2014-2021. In Section A of the Appendix, we also provide a quantitative analysis of colour normalisation.

Figure 3: Comparison of frames from video footage between 2014 (left) and 2016 (right). Difference in lighting geometry, hue and bounding laser colour.

3.1 Time-averaged Hue-Saturation-Lightness (HSL)

One approach investigated time-averaged HSL corrections. Every *n*th frame, a copy of the current video frame was taken to contribute to a weighted accumulation image of the hue, saturation and lightness. This built an 'average' of the image to allow for correction of the spotlight effect that can be seen in some years with LED lighting. Typically videos with halogen bulb lighting were more uniform and did not exhibit this artefact, however the time averaged normalisation helped to account for variations in light due to suspended sediment.

An example of this approach is shown in Figure 4 where it is evident that the spotlight effect of the lighting has been successfully removed. Qualitatively this does not make the sea pens necessarily easier to detect. However, this pre-processing may make images from different years more uniform and therefore allow their use within models that have been exclusively trained on single year datasets.

3.2 Grey-world (GW) algorithm

Another colour normalisation (or constancy) algorithm is the GW algorithm. The GW algorithm assumes that the average pixel within an image can be assumed to be grey. [7] This assumption holds only for images where the majority of the RGB colour spectrum is present within the images and thus can be mapped to a global average of grey.

Figure 4: Original frame (left) and time-averaged HSL correction (right). Spotlight effect evident in the original image in the upper left of the image. Spotlight not longer evident in processed image, only the seabed remains in focus for analysis.

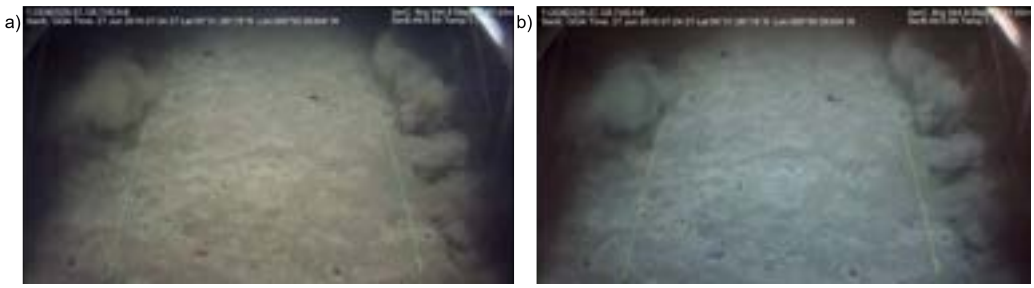

Figure 5: Example of the GW algorithm performance. Original image on the left, processed image on the right.

The video footage analysed varied significantly, as multiple years exhibited blue tinted frames and other showed warmer hues. As an attempt to reduce this variation we implemented the GW algorithm on various movies, example shown in Figure 5.

The GW algorithm improved the overall variation in image colour warmth, discussed in section A. The GW algorithm appeared to successfully remove the warmer tones that can be present in halogen-illuminated video footage, which should upon further testing allow for more applicable comparisons between LED and halogen-lit video footage.

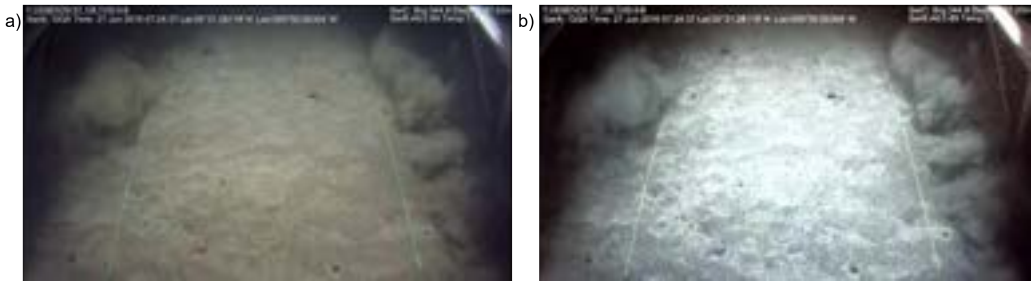

Figure 6: Example of the equalised RGB processing. Original image on the left, processed image on the right.

3.3 RGB histogram equaliser

In raw images or video frames, pixel values can vary between 0 and 255 (8-bit range). However, it is possible for an image to contain pixels within a very small range of these intensities. RGB equalisation works by re-scaling the RGB histograms such that the entire 0-255 intensity range is occupied within the RGB histogram. [8]

Using this can, however, lead to colour artefacts within images, due to potentially large adjustments to the RGB histograms. An example of an RGB equalisation on a video frame is shown in Figure 6.

It is clear that, although the overall warmth of the original video is removed, suggesting a normalisation of global warmth/colour, the central region of the seabed appears to be over-saturated. This could hinder both manual inspection and model performance and so we moved to investigate a more sophisticated histogram equalisation approach in the following subsection.

3.4 Contrast limited adaptive histogram equalisation (CLAHE) algorithm

CLAHE is a technique used to enhance the contrast in images. It works by dividing the image into small blocks, called tiles, and then applying histogram equalisation to each tile individually.[9] This can help improve the local contrast within each tile, making the image appear more vibrant. However, because applying histogram equalisation can sometimes

Figure 7: Contrast Limited Adaptive Histogram Equalisation (CLAHE). Original image (left), CLAHE enhanced image (right). Note how CLAHE reveals detail in the seafloor.

produce over-saturated colours, CLAHE includes a contrast limiting step that ensures that the overall contrast in the image is not increased too much. This helps prevent the over-enhancement of images, resulting in a more visually pleasing and natural-looking image.

We converted images to Lab colour space (also sometimes referred to as L*a*b colour space). Unlike RGB, which refer to red, green, and blue, the Lab colour space expresses colour as *L* for perceptual lightness, *a* for the green-red colour axis, and *b* for the blue-yellow colour axis. Instead of applying the CLAHE equalisation algorithm to the three channels, we applied it only to the brightness, or *L*, component in the Lab colour space, leaving the colour channels intact. In our data, this algorithm was successful at revealing detail in the seafloor (Figure 7).

The algorithm appeared to make manual detection of both *Pennatula* and *Virgularia* easier due to the increase in contrast and thus the improvement in manual identification of local features. This particular algorithm appeared to make the most impressive improvement on multiple data sets and is therefore a potential candidate for routine video pre-processing.

3.5 Multi-stage image pre-processing

The aforementioned algorithms work to adjust the colour variability and local contrast in different ways and therefore can be combined in series to

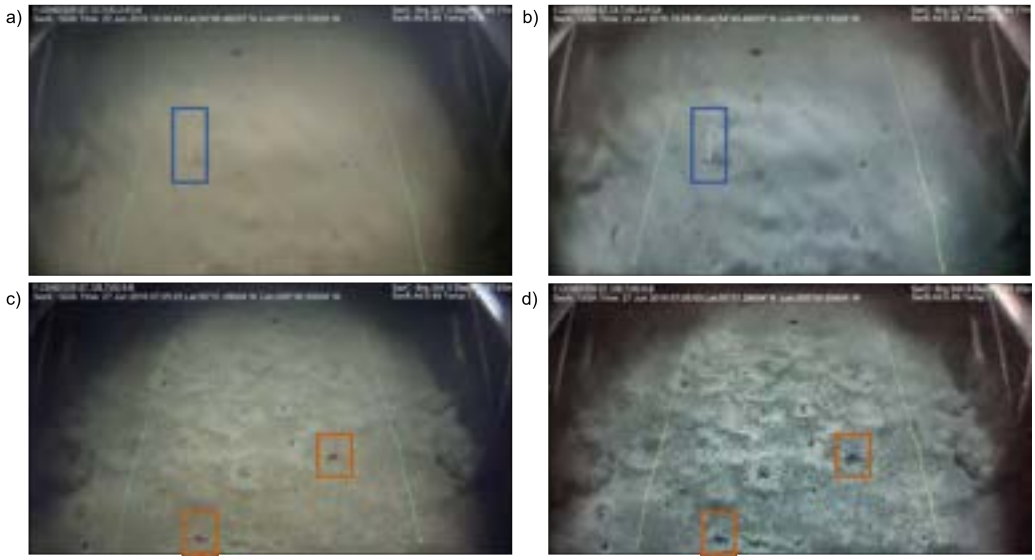

Figure 8: Example of a multi-stage video processing output. The GW algorithm and the CLAHE algorithm were applied in series. (a-b) Examples of video frames displaying *Pennatula* and (c-d) *Virgularia*, with boxes highlighting the specimens. Original images on the left and multi-stage processed images on the right.

generate more complex pre-processing of videos. Within our preliminary study, we decided to trial the GW algorithm with a subsequent CLAHE processing step. We chose to trial this combination of algorithms as they appeared to remove colour variances in overall image hue, but also aided in manual detection of sea pens, see example in Figure 8.

Interestingly, in the GW-CLAHE processed images (Figure 8) both the *Pennatula* and *Virgularia* are more visible for manual detection. Given their distinct morphology, this combination of pre-processing algorithms could prove to be an appropriate approach to ensure accurate identification of both sea pen species, which is crucial for the assessment of the sea pen ecosystem. These multi-step processing pipelines must be applied carefully as to avoid complete distortion of the original video frames and data. However, in theory any combination of these (or other algorithms) pose as potential avenues for further research.

Figure 9: Laser detection. Screenshot of the laser detection algorithm in one of the frames. Long and mostly vertical edges are detected as blue lines. The two longest lines correspond to the position of the lasers and further extrapolation with a line fitting allows to define a polygon in the central region. Notice that sometimes the sledge was visible at the edges of the video, which resulted in the identification of false positives.

4 Laser detection

The Cefas team uses parallel lasers separated by 70 cm to standardise the measurement of sea pens on the seafloor video footage. The lasers help identify the number of sea pens within a specified area, allowing for an accurate density calculation. The team must identify the lasers and differentiate the region between the lasers from outside of the lasers to omit inaccurate sea pen identifications. It was therefore an important goal to obtain an accurate automated segmentation of the regions between the two lasers, so that detected sea pens outside of them could be discarded.

Attempts were made to specifically extract the laser lines by their colour based on optical properties (e.g RBG banding or HSV ranges). A very narrow range of parameters was required to identify these pixels, and this was found to be very different based on the video, particularly since the characteristic hues were changeable from year to year. For this reason, it was decided that the most reliable method was to use automated line-detection on the footage. The chosen approach was as follows:

1. **Acquire a grayscale representation of the image**: certain algorithms used later, such as edge detection, require images in gray scale.
2. **Apply Gaussian blur with a 5x5 pixel kernel**: Gaussian blur is used to reduce the amount of noise in the image and make the edges in the image more defined, making it easier for the computer to detect the lines. The kernel of 5x5 pixels means that the blur algorithm uses a 5x5 grid of pixels to calculate the new colour value of each pixel.
3. **Apply Canny edge detection to image**: Canny edge detection is used to identify and highlight the edges in the image. This helps the computer to identify the lines more accurately and quickly.
4. **Apply Hough lines transform**: The Hough lines transform takes the edges identified by the Canny edge detection and converts them into lines, making it easier for the computer to identify the lasers. This is the step represented in Figure 9.
5. **Select lines where y1-y2 distance is over a given length threshold in pixels**: By only selecting lines that have a large difference between y1 and y2, the team ensures that the lines being selected are likely to be the lasers and not other horizontal lines in the image. This filtered out lines where the orientation was predominantly horizontal.
6. **Continuously update a pixel mask based on detected lines in each pixel for a duration of video footage**: A pixel mask is a binary image where each pixel is either white or black. By continuously updating the pixel mask based on whether lines were detected in each pixel, the team creates a mask that accurately reflects the lines in the video footage.
7. **Resulting mask is contoured**: In this steps, a function is used to find the contours of the pixel mask. It is assumed that after the previous step, the largest contours would be those corresponding to the lasers, as they are present in every frame of the video.
8. **A line fit is applied to the image contours**: This step fits a line to the image contours, which allows the computer to accurately identify the lasers in the image.

9. **Omit short lines**: Finally, the team omits any lines that are too short, as these are unlikely to be the lasers. By doing this, the team ensures that the laser lines are accurately identified and segmented from the rest of the image.

This approach provided binary mask files that separated three regions: a central polygonal-shaped region with 1's, representing the area between lasers, and two surrounding regions at the left- and right-hand side of this central region, filled with 0's. After detection of sea pens, this information can be used to filter out those sea pens falling outside the laser areas, resulting a more accurate density calculation. The laser identification method worked only under reasonably clear conditions. In those cases, the algorithm provided adequate masks in approximately more than 95% of the videos across all years under good quality footage with good visibility. For a large amount of the footage, strong currents mean the footage is obscured by bright backscatter from suspended sediment. Under such conditions, the lasers are hard to see even by eye. For this algorithm to work, a set of parameters (such as thresholds for edge detection algorithms or to discard short lines) was established by trial and error, and it is possible that fine tuning of these parameters would improve the accuracy to nearly 100%. Closer investigation of the sources of error and a systematic study of the influence of these parameters should be the necessary next step to ensure line detection works with sufficient accuracy and can be integrated in the pipeline to count sea pens.

5 Detection and classification of sea pens

5.1 Detection with YOLOv5 (You Only Look Once v5)

Object detection typically involves first extracting a set of strong features from input images, and then using classifiers or localisers to identify objects within those features. These classifiers or localisers can be applied over the entire image or on specific regions of the image. Traditional techniques for object detection involve adapting pre-existing classifiers to the task of detection. For instance, R-CNN (Regions with CNN features) is an object detection system that was proposed in 2014 [10]. It uses a two-stage approach for object detection. In the first stage, it

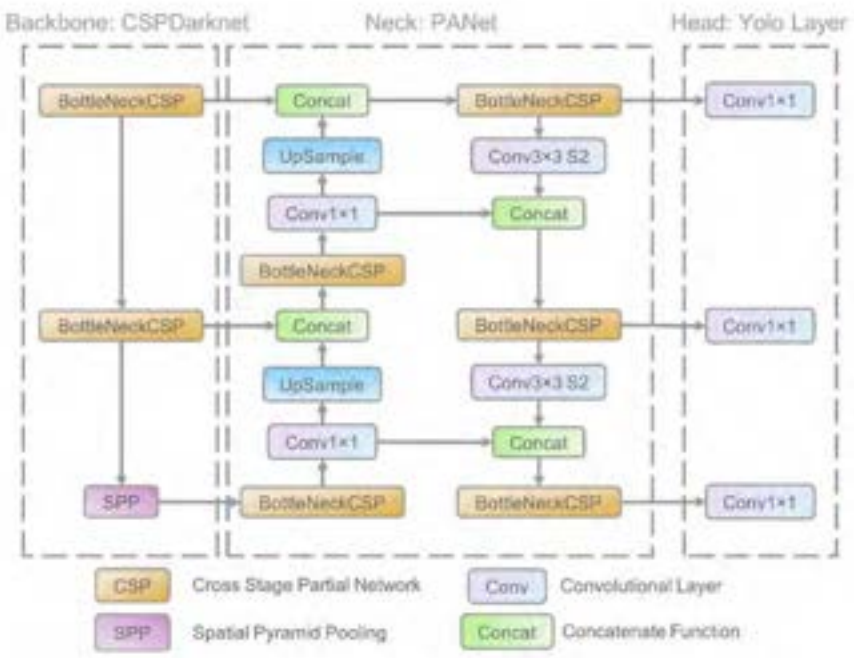

Figure 10: The network architecture of YOLOv5 consists of the following parts: i) a backbone based on CSPDarknet; ii) a neck based on PANet; and iii) a head based on the YOLO layer. The input backbone layer is in charge of feature extraction, which are then fed to the neck layer for feature fusion, and finally the YOLO layer outputs the detection results, namely class, score, location and size. Adapted from [11].

uses a technique called Selective Search to generate a set of region proposals, which are regions in the image that are likely to contain objects. In the second stage, it uses a convolutional neural network (CNN) to classify and localize each region proposal.

The YOLO (You Only Look Once) architecture, however, is trained to predict at the same time both the bounding boxes of objects in an image and the class of each object (in our case, each type of sea pen) within those boxes. In other words, it not only detects the presence of objects, but also identifies what those objects are. In particular, YOLOv5 uses a single neural network that processes the entire image at once, and this

allows it to predict bounding boxes and class probabilities directly from the full images in one evaluation. This is the main feature that makes YOLOv5 different from other object detection systems that only detect the presence of objects without classifying them, allowing for a better optimisation of the entire detection process and a significantly faster system. YOLO is similar to R-CNN in some ways, but it generates fewer bounding boxes per image, only 98 compared to around 2000 from Selective Search.[12] While Fast R-CNN[13] and Faster R-CNN[14] aim to improve the speed of R-CNN, they still do not achieve real-time performance.[12] YOLO, on the other hand, is a general-purpose detector that can learn to identify multiple types of objects at once, making it suitable for real-time detection.

The base YOLO model can process images in real-time at a rate of 45 FPS, while a faster version, Fast YOLO (which uses less convolutional layers in its neural network), can handle 155 FPS while maintaining high accuracy.[12] Despite its speed, YOLO is less prone to localisation errors and less likely to identify false positives in the background compared to other state-of-the-art detection methods. Additionally, YOLO has been shown to excel in learning general representations of objects and outperforming other techniques, such as DPM and R-CNN, when applied to diverse domains like natural images and artwork. In this project, we used YOLOv5, which is one of the latest versions of YOLO.[15] This object detection model has shown top performance in some of the most commonly used object detection datasets,[11] like Pascal VOC[16] and Microsoft COCO.[17]

It is worth noting that YOLOv5 has different versions, also known as variants, that have various numbers of parameters and different levels of accuracy. These variants are named YOLOv5n, YOLOv5s, YOLOv5m, YOLOv5l and YOLOv5x. Generally, the more parameters a variant has, the more accurate it is but also the longer it takes to train. In this specific project, the team decided to use YOLOv5n because it has the least number of parameters (1.9M) and the shortest training time. However, its accuracy is not as high as the other versions. The next sections will demonstrate that the performance of YOLOv5n on the sea pen dataset is still remarkable. A preliminary study comparing the performance of YOLOv5n with the second smallest version, YOLOv5s (with 7.2M parameters) was also performed and is reported in Section C of the

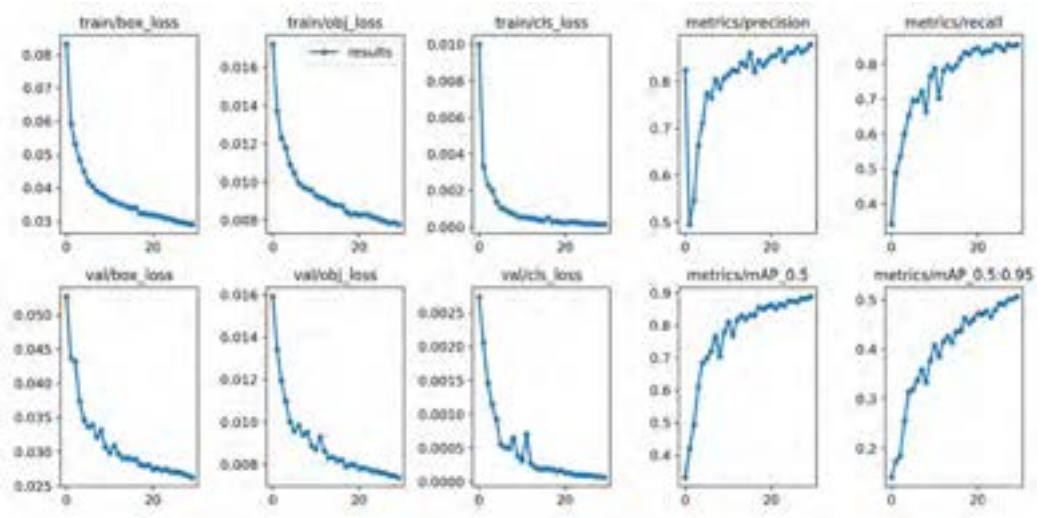

Figure 11: Training results of the YOLOv5 model. The three left columns correspond to the loss function per training epoch and the right two columns to different performance metrics per epoch.

Appendix. In future studies, larger versions of YOLOv5, as well as newer architecture versions, could be considered in order to increase accuracy.

We used a pre-trained PyTorch YOLOv5n model as our baseline model, with frames and bounding box annotations from all available years as input for both sea pen classes. No pre-processing was applied to the videos before training the model or during testing. The models were trained on different machines but, as an example, it took a maximum time of 1 h and 40 min to train the YOLOv5n model on a Nvidia GeForce RTX 2060 GPU and a maximum of 2 h and 15 min to train the YOLOv5s on the same GPU. The models were trained for 30 epochs and batch sizes of 16 images of 640x640 pixels, and training/validation splits of 0.80/0.20.

5.1.1 Results

Figure 11 shows the results of the training of the YOLOv5 model. The left three columns show the evolution of the box loss, object loss and class

Figure 12: Representation of the IOU. Source: interstellarengine.com.

loss during the training and validation. The goal of the training process is to minimise the loss function, and it can be observed how this value continuously decreased over the 30 epochs, slightly saturating to zero towards the end. The model did not suffer from overfitting, as the loss did not increase its value during the training. The right two columns show the evaluation metrics over time. Both precision and recall, as well as mean average precision (bottom) increased with epochs, reaching values close to 1, and therefore the model was successfully training to detect sea pen classes.

The intersection over union (IOU) is a metric used in object detection algorithms that compares the overlap between two bounding boxes, namely the actual bounding box that contained an object and the model prediction. This is also known as the Jaccard similarity coefficient or metric. This metric is computed by dividing the overlapping area between prediction and actual boxes over the total union area. Figure 12 shows representation of excellent, good and poor predictions of an object. Notice that the IOU is 1 when there is 100% overlap and 0 when the prediction has completely missed the object and there is no overlap between boxes. IOU defines what is considered a true positive (TP) or a false positive (FP), which in turn are used to compute the precision and recall of the model.

Figure 13 shows the distribution of IOU of our model for each year and sea pen class. It can be seen how the average value of this distribution is reasonably constant throughout the year, averaging at about 0.75. This means that the model managed to generalise over different lightning conditions and video qualities, one of the main goals of the challenge. It needs to be noted that no pre-processing or standardisation (as

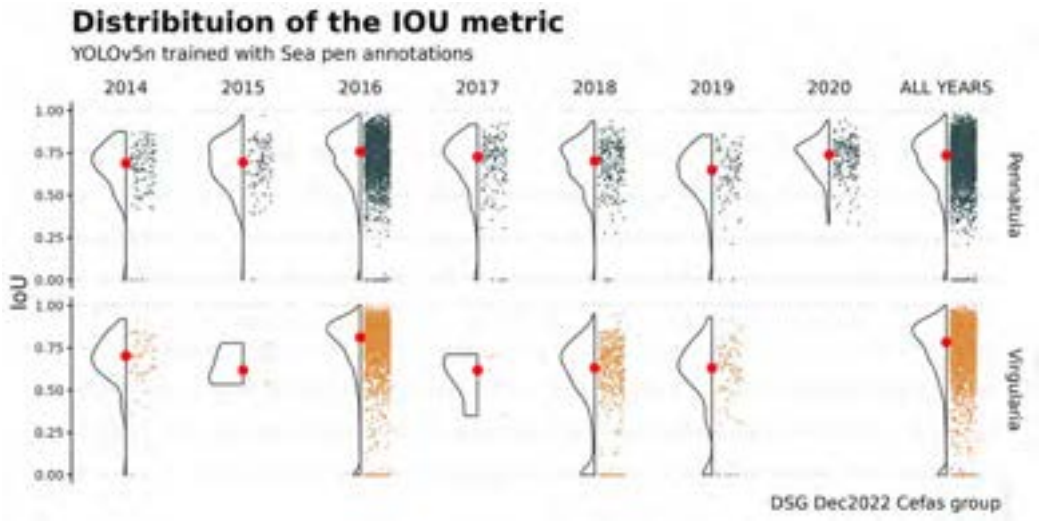

Figure 13: Distribution of the IOU of the YOLOv5 model for different years and sea pen classes. Each data point reflects a sea pen in the videos from each year. Notice the larger number of sea pens in videos from 2016, which contain the largest sample, as described in Section 2. Despite this reason, the model can generalise over sea pens in other videos with different lightning conditions.

described in Section 3) was applied to the videos before training and testing, as those were being developed in parallel. An average IOU of about 0.75 is considered a good value, as shown in Figure 12. Not many values fall under the IOU of 0, so in very few cases the predictions were completely wrong at identifying sea pens. Figure 14 displays the median values of IOU over the years, showing again how in the median case the predictions had adequate IOU's, and detection of *Virgularia* sea pens, for many of intermediate years, showed lower values of IOU. In the year 2016, which contained a higher amount of annotations than the other years (as there were more videos available for 2016), both *Virgularia* and *Pennatula* peaked in performance, as the model was probably overly relying on the conditions shown in those videos during training. This is a common problem when dealing with unbalanced datasets, but still the performance for the remaining years was also high.

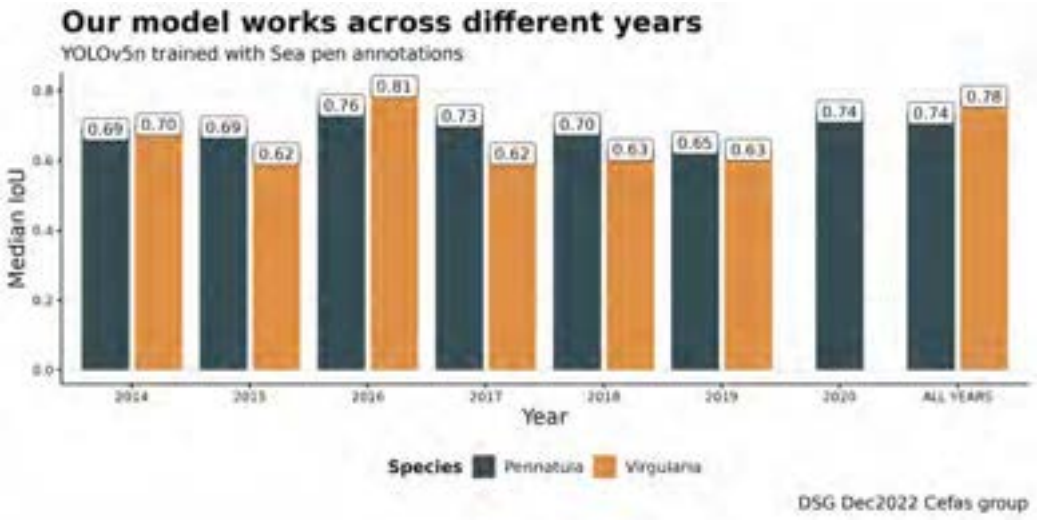

Figure 14: Median IOU of the YOLOv5 model for several years and sea pen classes.

Another popular evaluation metric is the mean average precision at IOU 0.5 (mAP@0.5), which provides a comprehensive evaluation of the model's accuracy by averaging the precision of all objects in an image. In this case, the IOU threshold is set to 0.5 to determine whether a predicted bounding box is a TP or a FP. Then, the mean precision over all the object types (in our case, the two types of sea pens) is calculated for that IOU threshold, yielding the mAP@0.5. The mAP can be calculated for any other IOU threshold, but 0.5 is a popular convention. Figure 15 shows the results for our model. In line with Figure 14, the mean average precision is high for all years and sea pen classes. Some of the years, such as 2018 and 2019 suffer a slight decrease in performance, but the average mean precision for all years is maintained at 0.91 and 0.88 for *Pennatula* and *Virgularia*, respectively.

5.1.2 Limitations

The YOLO model uses strict spatial limitations on its predictions of bounding boxes, as each grid cell can only predict two boxes and be assigned one class. This spatial constraint limits the ability of the model

to detect a large number of nearby objects. As a result, the model may have difficulties detecting small objects that appear in clusters, such as groups of sea pens.[12] As in this dataset sea pens appear rather separate from each other, we did not expect this to be an issue. The YOLO model also struggles to generalise to objects in unseen aspect ratios. In our case, since sea pens are generally symmetric and do not change much in their aspect ratios, we also did not expect this to be an issue.

This model employs relatively basic features to predict bounding boxes, as the architecture includes several layers that reduce the size of the input image. Additionally, when training the model, the loss function used to approximate detection performance does not distinguish between small and large bounding boxes when calculating errors. An error in a large bounding box is typically not as significant as an error in a small bounding box, which has a greater impact on the intersection over union (IOU) metric. As a result, the primary source of error in YOLO model is poor localisation of objects. [12]

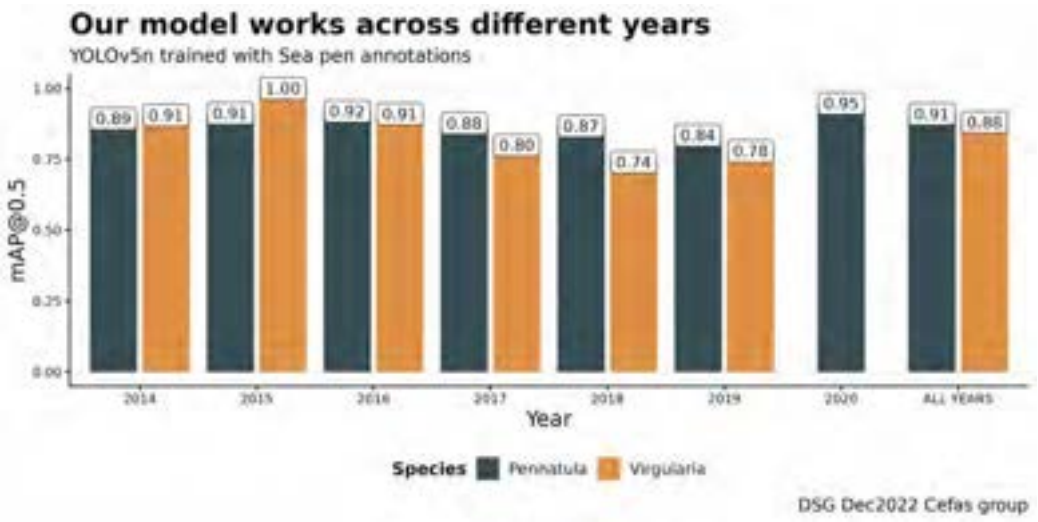

Figure 15: mAP@0.5 of the YOLOv5 model for several years and sea pen classes.

5.2 Classification

As mentioned above, other approaches for object detection and classification rely on a first step that detects any objects of potential interest in the image or video frame, and a second classification step that classifies that object into a specific class. However, a limitation of these algorithms is that they are often trained and optimised for detecting objects in human environments, such as cities or houses. This means that they may not perform well in natural settings with different lighting conditions, or with objects that look vastly different from the objects the model was optimised to detect, even if fine-tuning has been done with relevant training images.

For this reason, we suggest the following approach where detection and classification are performed separately might benefit our case study. For this parallel pipeline, we assume that an only-detection algorithm has been applied to the video frames to detect objects of interest, but the model is overfitting detections, that is, it is predicting a large number of false positives. As mentioned in the previous subsection, if the confidence threshold is kept low, the model will predict a larger number of objects with a low confidence score. Likewise, a lower IOU will decrease the overall precision of the model, but it will increase its recall and number of FP detections. A subsequent classification algorithm would then discriminate these potential objects between the two types of sea pens (*Pennatula* and *Virgularia*) and the background. Therefore, we analysed the performance of three classification models based on convolutional neural networks (CNNs): a simple CNN model, the ResNet50 model and the VGG16 model (reported in Section D of the Appendix).

5.2.1 Creating a training dataset

Before creating our classification models, we needed to create a dataset containing training images of the sea pens and background. Whereas for the dual detection and classification YOLO models we needed to provide full frames and a list of bounding boxes with the labels of the sea pens, in this case we were assuming that the detection had already been performed and we already had a series of cropped images that needed to be classified as *Pennatula*, *Virgularia* or background. For that reason, we

created a script that, using the human annotations provided by Cefas, cropped images of sea pens from the videos and saved them using the following naming convention: processing status + station + year + actual frame ID + sea pen ID + type.

1. Processing status: identifying whether the video had been pre-processed with one of the techniques from Section 3. Since we only tested classification on the original videos, this status was always set to 'orig'.
2. Station: the station were the videos came from, as per the original naming convention by Cefas.
3. Year: the year in which the videos were recorded.
4. Actual frame ID: the frame ID in the original video (not downsampled), as calculated in Section 2.5.1.
5. Sea pen ID: this ID identified an individual sea pen. As some of the annotation files contained track annotations, as explained in Section 2.2, it was possible to identify the same sea pen in subsequent frames of the same video.
6. Type: referred to the class labels: 0 for *Pennatula*, 1 for *Virgularia*, and 2 for Other or Background.

Apart from creating cropped images of sea pens according to the human-made annotations, this script also created a series of random images (of approximately the same size of the typical sea pen bounding box) that corresponded to the Background class. The script checked for duplicates and also made sure that cropped images did not contain sea pens. The class datasets, therefore, contained the following number of images:

- *Pennatula*$_n = 19,166$
- *Virgularia*$_n = 25,337$
- Other$_n = 26,447$
- Total$_n = 70,950$

5.2.2 Convolutional Neural Networks (CNNs)

As the three models we considered were based on CNNs, it is useful to define this kind of machine learning model. A CNN is a type of deep learning neural network that is commonly used for image and video analysis. CNNs are designed to recognise patterns in images and videos through the use of convolutional layers, pooling layers, and fully connected layers. The convolutional layers are used to identify patterns within specific regions of the image by calculating the inner product between a convolutional filter and various regions of the image to produce a feature map. The feature map is then processed through a non-linear function, such as tanh, logistic, softmax, or ReLU, to create activations, which are further processed in the pooling layer. These alternating layers of convolution and pooling are used to extract features at each step. Finally, the last layer is a fully connected layer that outputs the class in a recognition task.

The function of a convolutional layer is to transform the input data using a group of connected neurons from the previous layer. It uses a mathematical operation called convolution, which applies a convolution kernel to the input and returns a feature map as output. Different types of filters in a convolutional layer learn to produce the strongest activation to spatially local input patterns and the output depth controls the number of neurons in the layer. Convolutional layers have important hyperparameters such as filter size, output depth, stride, and zero-padding that affect the spatial arrangement and size of the output volume. The filter size defines the spatial dimensions of the filter and the output depth controls the number of neurons that are connected to the same region of the input volume. The stride defines the pace at which the filter is applied to the input, and zero-padding is used to maintain the spatial dimensions of the input in the output volume.

The pooling layer is used to progressively reduce the spatial size of the data representation and prevent overfitting. It uses the maximum operation to resize the input data spatially and does not have any learnable parameters. The fully connected layer acts as the output layer for the network and has the output volume dimension as [1 x 1 x N], where N is the number of output classes to be evaluated. It has general neural network layer parameters and hyperparameters.

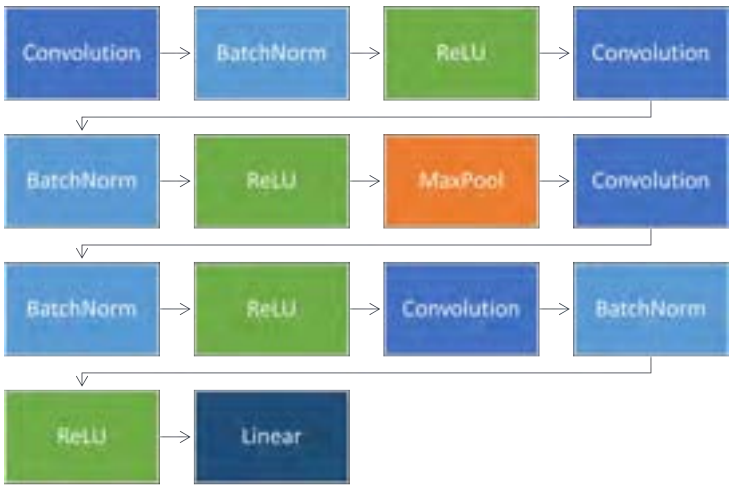

Figure 16: The 14 layers of the simple CNN.

5.2.3 Simple CNN

The first model was a simple CNN with 14 layers using code from a Microsoft tutorial[3]. Its architecture is shown in Figure 16.

Four of the layers were convolution layers. The parametrisation of each of the four convolution layers is shown in Table 16. 'In_channels' specifies the number of input channels of the layer: for example, the first layer takes a 3-channel input (RGB). 'Out_channel' specifies the number of feature detectors that the layer applies: this determines the number of inputs for the next convolution layer. The 'kernel size' was 5, meaning that the size of the kernel was 5x5. The stride is the number of pixels the convolutional filter moves in the input image in each direction (horizontally and vertically) at each step, and it was kept to 1. Padding is the number of pixels added to the edges of the input image before the convolutional filter is applied. Padding is used to ensure that the output feature map has the same size as the input image, and was set to 1.

Figure 16 also shows that the network has several Batch Normalisation layers, ReLU activation layers and a MaxPool layer. Batch normalisation is a technique that is used to normalise the activations of a neural

[3] https://learn.microsoft.com/en-us/windows/ai/windows-ml/tutorials/pytorch-train-model

Layers	In_channels	Out_channels	Kernel size	Stride	Padding
conv1	3	12	5	1	1
conv2	12	12	5	1	1
conv3	12	24	5	1	1
conv4	24	24	5	1	1

Table 3: Hyperparameters of the CNNs used in the simple CNN model.

network layer. It helps to improve the stability and performance of the network by normalising the activations so that they have a mean of 0 and a standard deviation of 1. This can prevent the activations from becoming too large or too small, which can cause problems during training, such as slow convergence or overfitting. A max-pooling layer is used to reduce the spatial dimensions of the data representation. It works by applying a max operation over a region of the input data, in our case in a 2x2 region, and taking the maximum value from that region as the output. This helps to reduce the number of parameters in the network, as well as improve the robustness of the features learned by the network, as it helps ensure that the location of an object in a frame does not affect the capacity of the network to detect its specific features. Finally, ReLU (Rectified Linear Unit) is an activation function that is commonly used in neural networks. It applies an activation function to the output of a neuron, usually element-wise. It computes the function $f(x) = max(0, x)$ to the input x. It is commonly used because it helps to introduce non-linearity into the network, which can help to improve the performance and stability of the network. Additionally, ReLU is computationally efficient and helps to reduce the vanishing gradient problem, where the gradient of the loss function becomes very small during training.

The images were resized to 32x32 pixels for this model. This can be adjusted, but too large a number of pixels will stretch the sea pen images and will make the training slower. The image dataset ($n = 70,950$) was split into training (60%), validation (20%) and test (20%) sets. The loss function was defined with classification cross-entropy loss and an Adam Optimiser. The model was run for five epochs with the learning rate set at 0.001. The model was run on a CPU device, 11th Gen Intel(R) Core(TM) i9-11950H @ 2.60GHz 2.61 GHz with 32 GB of RAM. It only took around four minutes to run an epoch, so that running the whole model only took

Epoch	Batch number	Running loss	Epoch accuracy
1	1000	0.447	82%
	2000	0.270	
2	1000	0.363	92%
	2000	0.263	
3	1000	0.225	92%
	2000	0.218	
4	1000	0.193	93%
	2000	0.195	
5	1000	0.180	94%
	2000	0.176	

Table 4: Results after training the simple CNN of Figure 16 with the sea pen dataset for a 3-label classification task, where the classes were *Pennatula*, *Virguralia* and Background. The running loss represents the training loss at the end of the batch, whereas the epoch accuracy is the accuracy at the end of the epoch in the validation dataset.

approximately 20 minutes. The exploratory model run performed remarkably well, as can be seen in Table 4, reaching a test accuracy of 94% in the three-label classification task. Future work could explore using different layer parameterisations and applying different numbers of layers and epochs.

5.2.4 ResNet50

The ResNet50 model employs a residual learning approach to refine and improve the training process for deep networks. These networks have been shown to be easily optimised and achieve greater accuracy than simple stacks of CNNs. For instance, application of ResNets resulted in a 3.57% error rate on the ImageNet test set, which won first place in the ILSVRC 2015 classification task. [18] ResNets leverage a residual mapping that can be trained on a few stacked layers, allowing for the approximation of complex functions. This solution tackles the degradation or vanishing gradient issue, which suggests that traditional training methods may struggle to approximate identity mappings through multiple nonlinear layers. With residual learning, if the identity mapping is optimal,

the training process can simply drive the weights of the nonlinear layers towards zero, resulting in an identity mapping.

In the ResNet network, the convolutional layers are predominantly comprised of 3x3 filters and adhere to two basic design principles: (i) for a consistent output feature map size, the number of filters remains constant; (ii) if the feature map size is halved, the number of filters doubles to maintain layer time complexity. Additionally, the network employs direct downsampling on convolutional layers with a stride of 2 and concludes with a global average pooling layer and a 1000-way fully connected layer with a softmax activation. The total number of weight layers is 34. The 50-layer ResNet is achieved by replacing the 2-layer block in the 34-layer model with a 3-layer bottleneck block.[18]

To make the training process more manageable, we used transfer learning. Transfer learning involves taking a model that has been trained for one specific task and modifying it for use in a different task. This allows for knowledge learned in one scenario to be leveraged in a new scenario, thereby improving performance and saving time. In this case, we used a ResNet50 model pre-trained for image recognition on the ImageNet dataset.[19]

5.2.5 ResNet50 results with 2 classes

In this section, we present the results of a ResNet50 experiment that was conducted for the two sea pen classes, *Pennatula* and *Virgularia*. The ResNet50 model was implemented in PyTorch as a pre-trained version using the IMAGENET1K_V2 weights. The training was carried out on a Standard NC8as T4 v3 (8 VCPUs, 56 GiB memory) on Azure with a Tesla GPU. The dataset was the one described in Section 5.2.1 and consisted of 44503 training images, which were split into 40000 for training and 4503 for evaluation (note that we did not use the ”Other” class in this experiment).

The training was completed in 85 minutes and 29 seconds, and the best validation accuracy was obtained as 0.997113 after 10 epochs of training, as can be observed in Table 5. This near-perfect accuracy is likely a result of the homogeneity of the physical features of the sea pens (as shown in Figure 1, they are rather distinct). It needs to be noted that this

Epoch	Train		Validation	
	Loss	Acc	Loss	Acc
0	0.0802	0.9709	0.0264	0.9913
1	0.0284	0.9907	0.0280	0.9891
2	0.0200	0.9932	0.0274	0.9951
3	0.0143	0.9957	0.0173	0.9942
4	0.0108	0.9963	0.0113	0.9960
5	0.0079	0.9974	0.0373	0.9962
6	0.0072	0.9980	0.0289	0.9924
7	0.0043	0.9988	0.0182	0.9951
8	0.0039	0.9989	0.0132	0.9962
9	0.0031	0.9993	0.0288	0.9971

Table 5: Results of the two-class ResNet50 model trained for 10 epochs

classification method assumes that a sea pen has been correctly identified after a detection process and the input image is a cropped bounding box with a sea pen. Therefore, the model only needs to learn to classify between both types of sea pens. However, it should also be noted that these results are preliminary and require further rigorous accuracy assessment on test images. The use of ResNet50, a deep residual network, and the pretraining on IMAGENET1K_V2 allowed us to obtain state-of-the-art results in a relatively short amount of time, making it a promising approach for this specific task.

5.2.6 ResNet50 results with 3 classes

In our second experiment with ResNet50, we used a ResNet50 model that was implemented in PyTorch and pre-trained with IMAGENET1K_V2 weights. We trained the model in the same GPU devices on the Azure platform. This time, we used three classes: *Pennatula*, *Virgularia*, and "Other". We had a total of 70950 training images (see Section 5.2.1), which were split into 60,000 for training and 10,950 for evaluation. The training process was completed in 65 minutes and 8 seconds, with a best validation accuracy of 0.982100, as can be seen in Table 6. As before, these results are still preliminary and will require further assessment to determine their accuracy on test images.

Epoch	Train		Validation	
	Loss	Acc	Loss	Acc
0	0.1617	0.9442	0.1218	0.9735
1	0.0593	0.9810	0.3355	0.9559
2	0.0335	0.9891	0.6497	0.9474
3	0.0217	0.9931	0.1232	0.9821
4	0.0132	0.9961	0.6877	0.9769

Table 6: Results of the three-class ResNet50 model trained for 5 epochs.

6 Tracking

Videos can be regarded as a collection of video frames (i.e. still images) which are then displayed in a rapid sequence. The detection algorithms described previously take an individual video frame as input, and produce a set of bounding boxes as an output (that correspond to the sea pens that are present in that frame), for example, as can be seen in Table 7. These detection labels correspond to the sea pens shown in Figure 17.

Since our objective is to count the number of sea pens that appear in a video, each detected sea pen needs to be tracked across the frames that it appears in, to ensure that it is only counted once. The detection algorithms from the previous section treat each frame independently and it is not possible to known whether a detection in one frame corresponds to the same sea pen in a later frame. For this reason, a tracking algorithm is necessary so that each sea pen can be assigned a unique ID so that they can only be counted once, even if they are detected in every single frame.

In this section, we review the work towards implementation of tracking algorithms to detected and identified sea pens, so they could be tracked through subsequent video frames and counted only once. We implemented an algorithm that could successfully track sea pens (assuming an initial bounding box with a detection) over several frames, which could also recover the position of the sea pen when temporary occlusion took place. Still, due to lack of time, certain limitations and errors could not be addressed, and the algorithm could only be tested in one video from 2018, albeit with overall positive outcomes.

frame_number	TL_x	TL_y	BR_x	BR_y
0	1126	508	1170	540
0	906	328	953	381
1	1125	508	1166	543
1	899	329	952	386
2	1127	519	1163	552
2	901	335	953	385
3	1131	518	1172	558
3	906	341	946	385
4	1131	523	1172	563
4	908	347	955	393
5	1131	532	1172	572
5	900	356	951	400
6	1133	540	1174	580
6	902	361	952	409
7	908	364	956	417
8	909	377	955	416
9	916	381	972	427
⋮	⋮	⋮	⋮	⋮

Table 7: A dataset of associated frame numbers ($frame\_number$) and bounding box coordinates (TL_x, TL_y, BR_x, BR_y), see Figure 17 for a visual example.

6.1 Tracking algorithms

There are many different types of tracking algorithms, each with its own set of advantages and disadvantages. Correlation-based tracking algorithms are a common technique that uses image correlation to track the movement of an object between frames. They search the area within which the object is expected to appear for a region which is visually similar to the object. Similarly, distinctive features of an object, such as edges or corners, can also be used to track its movement. Other algorithms track objects by generating a model of their trajectory. For example, Particle Filters use a set of randomly-generated samples (particles) that represent the probability distribution of the object's location. This location is iteratively improved at each frame (making use of measurements as soon as they become available).

In our case, we used the OpenCV library from Python to apply an off-the-shelf CSRT (Discriminative Correlation Filter with Channel and Spatial

Figure 17: An example of a frame and drawn bounding boxes.

Reliability) algorithm,[20] which is a widely used and easy to implement tracker. This algorithm combines the previous two methods by using both the objects past motion to predict its future behaviour, and by taking into account changes in the objects appearance. CSRT adds the concepts of channel and spatial reliability to discriminative correlation filter (DCF) tracking, allowing for the enlargement of the search region and improving tracking of non-rectangular objects. CSRT has achieved state-of-the-art performance on VOT 2016, VOT 2015, and OTB100 benchmarks, and is able to run in real-time on a CPU.[20]

Several other tracking algorithms are part of the OpenCV library[4] can be implemented as part of our tracking pipeline (alternatively to CSRT). Some examples are KCF (Kernelized Correlation Filters) and MOSSE (Minimum Output Sum of Squared Error), both of which use correlation filters. CSRT was chosen due to previous positive experiences from the group's researchers, as it is both fast and accurate, and no other trackers were assessed due to time constraints. Nevertheless, it is easy to alternate between trackers in our code, as they are all available under the same package, without needing to change any of the code.

[4]For instance, see https://broutonlab.com/blog/opencv-object-tracking and https://pyimagesearch.com/2018/07/30/opencv-object-tracking/.

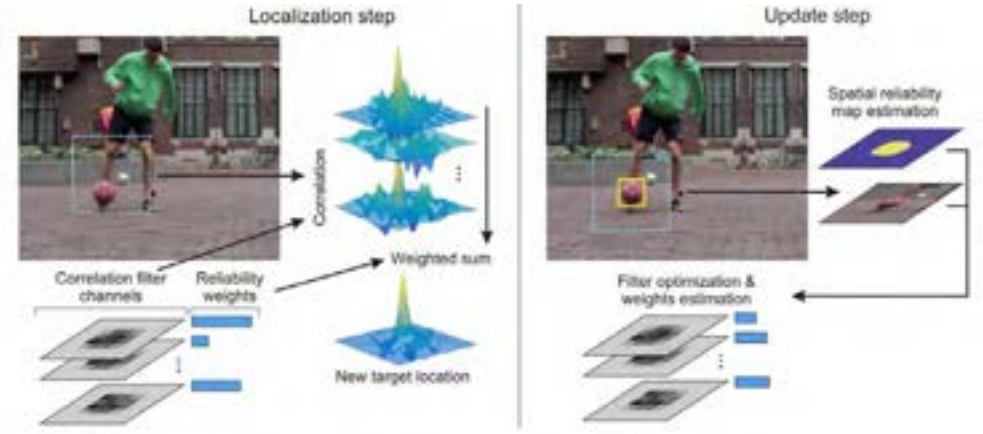

Figure 18: The CSRT tracking [20] iteration: localisation step is shown on the left and update step on the right side of the image.

6.2 Implementation

The tracking pipeline was created as a stand-alone module that, when given a dataset containing video frame indices and corresponding bounding boxes (e.g., Table 7), appends a column containing unique number ($detection\_id$) for each individual sea pen. The algorithm is shown as pseudocode in Algorithm 1 and it proceeds as follows.

The algorithm loops through each frame of the video. It starts by reading the current frame of the video and then obtaining the current 'tracked' positions of sea pens that were seen in previous frames using the CSRT algorithm (and are still present in the current frame). Next, the positions of the sea pens that were detected in that specific frame are read from the detection dataset. These can be divided into two distinct groups: (i) sea pens that have been detected in the current frame for the first time; and (ii) sea pens which had already been detected in previous frames (and therefore should already be tracked by the CSRT algorithm).

The sea-pens in group (ii) are identified by checking whether the bounding boxes obtained from the detection dataset overlap with any of the bounding boxes obtained from the CSRT tracking algorithm. Any bounding box which does not overlap with a track is considered to a be a previously unseen sea pen from group (i), and therefore is assigned a

Algorithm 1 Assigning unique IDs to sea pens

while video not finished: **do**
  read the current frame
  obtain the current 'tracked' positions of sea pens that were seen in previous frames (using the CSRT algorithm)
  obtain the positions of the sea pens that were detected in that frame (provided by the detection algorithm)
  for each detected sea pen **do**
    if detected sea pen position overlaps with a sea-pen that is being tracked from the previous frames **then**
      assign the corresponding $track\_id$ to the sea pen
    else
      assign the sea pen a new unique $track\_id$ and add it to the sea pens that are being tracked by the CSRT algorithm
    end if
    add the $track_id$s to the detection dataset
  end for
end while

unique ID and tracker.

We added a couple of fixes in Algorithm 1 that worked around certain failures of the tracker to continue tracking a sea pen. For instance, all the sea pens were 'moving' in the frame from top to bottom, as the sledge was moving across the seafloor. When the sea pens reached the bottom of the screen and disappeared, the tracker continued looking for them in the region they were seen last, before completely deciding that the sea pen was lost from the field of view. To solve this small issue, we simply added a 'height threshold'. When the sea pen was close to the bottom of the screen (as specified by the height threshold), we set the tracker status to False, completely ignoring the last unreliable tracking instances. We could only do this due to the particularities of our videos, since sea pens would move towards the bottom 100% of the time, and never towards the top. In some but very few occasions, the tracker would momentarily lose the sea pen and then wrongly find it at a different location, confusing it for a neighbouring sea pen. This was also a particularity of our challenge, as all the sea pens look very similar to each other. To avoid the tracker moving

onto a different sea pen, we added a 'jump threshold'. At each step, we calculated the distance between the current position of the sea pen and the position in the previous frame. If the distance was higher than the jump threshold (in our implementation, 20 pixels, but this should be further optimised considering more videos), the tracker was set to False and the sea pen was not tracked anymore. This meant, however, that if the tracker activated the jump threshold, that sea pen would be completely lost and therefore prone to being counted twice. Nevertheless, this occurred in very few occasions and its effect on the overall performance is expected to be low.

6.3 Limitations

Our analysis mostly focused on the CSRT tracker and only included a video from 2018. However, some quick tests were performed with videos from other years and it was evident that lightning conditions do not affect the tracking of the sea pens, as the CSRT algorithm is robust. However, it is recommended that the pipeline is tested on a larger variety of videos, and different trackers should be investigated so that their performance can be compared.

It is difficult to assess the performance of the tracking algorithm against the initial annotations provided by Cefas. As discussed in Section 2.2, many of the annotations were single annotations without any tracking information. Those that contained tracking annotations, however, were only tracked in one out of three frames, as the videos were downsampled to make them easier to work with. One potential way to assess the performance of the tracking algorithms would be to compare the total number of unique sea pens identified in each video with the numbers manually obtained by Cefas. This is, however, an indirect performance metric, as it is not assessing the performance of the tracking algorithm itself, but a subsequent computation.

In Section 6.2, we discussed that the tracker would occasionally lose track of a sea pen, causing them to be potentially counted more than once. An advantage of state-of-the art tracking algorithms is that they can predict the trajectory of objects when they are hidden from view, such that they can continue to track them when they re-appear in future frames.

However, in some occasions, such occlusions, for instance caused by plumes of sand, were too long to take advantage of this capacity of the algorithm. There is not an easy solution to this problem, as the CSRT algorithm (or any other off-the-shelf algorithms available through OpenCV) does not have any parameters to optimise. One potential solution would be to not consider the whole field of view for the tracking algorithm. As the only goal of the tracking is to avoid counting the sea pens more than once, we could consider only a portion of the vertical dimensions of the field of view (e.g., half of it, or two thirds of it) for the detection and tracking of the sea pens. It is likely that, even if we consider only half of the height, a sea pen will be detected at least once. Tracking, then, would be perform during a shorter amount of time, decreasing the possibilities of losing the sea pen and thus counting it more than once.

7 Conclusions

In this DSG challenge, we aimed at implementing a pipeline that could achieve detection and classification of two types of sea pens (*Pennatula* and *Virguralia*) from sea floor footage over a range of years with different lightning and recording conditions. Moreover, secondary objectives included (i) the standardisation of video conditions, such as brightness or hue, so that sea pens could be detected more easily; (ii) the detection of laser lights in the videos that would allow counting sea pens in a certain region of interest; and (iii) the implementation of a tracking algorithm that could follow detected sea pens over several frames in a video, so that they could be counted only once.

To achieve these objectives, the DSG team was divided into five subteams, namely image processing, laser identification, detection, classification and tracking. Although in an ideal scenario, all of these streams of work should be integrated in a single pipeline (see Figure 19 in next section, tackling future work), they were performed in parallel to ensure all subteams could achieve tangible results. Image processing and standardisation discovered an equalisation algorithm, CLAHE, that provided positive results by highlighting the presence of sea pens in the available video footage. An algorithm based on edge detection and line

fitting over several frames of the video footage was able to identify the lasers present in the videos, and provide a series of masks that could be used to segment the frames into two regions with high accuracy.

Detection and classification algorithms based on the model YOLOv5 provided outstanding results over all years with $\sim$90% mAP at 0.5 IOU. In parallel, we tested several classification algorithms that, assuming a positive detection, could classify between the two types of sea pens and background, in order to eliminate false positives. We tested models like ResNet50, VGG16 and a simple custom-made CNN, with classification accuracy of $>$94% over all years. Finally, we provided a working tracking algorith that, given an initial sea pen detection, could track such sea pen over several frames, removing double detections and allowing the user to count each sea pen only once. Due to time constraints, certain limitations of the algorithm and its generalisability over several years could not be tested. Overall, this work provided positive results for all five streams of work that could be integrated into a single pipeline to allow for an accurate detection and classification, as well as counting, of sea pens over several years in videos with different lightning conditions.

8 Future work and research avenues

There were several aspects of this work that could not be implemented due to time constraints, but which follow naturally from the results presented here:

1. The image pre-processing results from Section 3 showed an interesting augmentation of the original videos, correcting differences in lightning and hue and highlighting the presence of sea pens. This work, however, was performed on parallel to detection and classification, and there was no time to check their combined effect. A very immediate avenue of research would be to test the detection and classification algorithms on pre-processed videos. This would require creating new image datasets from these videos, which should be immediate as we provide the codes to do that, and retraining the models with these new datasets to test whether these image pre-processing can improve the results of the detection and classification models.

2. The detection of the lasers allowed to create binary masks of the field of view inside the lasers. In order to count the sea pens inside this field of view as the larger project requires, these masks should be applied to the coordinates of the bounding boxes of the sea pen detections to remove those that fall outside this region. This is not a difficult task, but a necessary one. However, for some of the videos, this algorithm failed due to laser occlusion. Closer investigation of the sources of error could improve the algorithm, for instance, adding more strict and robust rules as to what can be considered good segmentation of laser lines, considering that they are always roughly at the same position. Another way could be applying further filtering. For example, sometimes two lines were fit to one side of the image. This could be resolved by denying two lines with very close midpoints, as the laser track midpoints should be on different sides of the image. Given that turbid conditions often mean the lasers are not obscured, it may be difficult to implement a reliable computer-vision approach. Under such conditions it may be useful for the tool to contain a user-defined override option to define the 4 vertices of the laser line polygon by hand.

3. Throughout the challenge, we used rather light, fast and well-established detection and classification models to reduce risks and save time. Still, the results were unexpectedly positive with these models. Heavier models, such as the siblings from YOLOv5 that can have up to 86.7M parameters (YOLOv5x), as well as more recent architecture versions from YOLO like YOLOv8,[21] could also be tested. Likewise, we recommend testing next other models like Single Shot MultiBox Detector (SSD),[22] Localizing Objects with Self-Supervised Transformers and no Labels (LOST),[23] or Self-Supervised Transformers for Unsupervised Object Discovery using Normalized Cut.[24]

4. In Section 5.2, we proposed to overdetect objects in the videos and then use a second classification step to further discriminate over types of sea pens or background. Initial tests showed that this approach could be promising, with accuracies close to 100%. However, these results should be further assessed on test datasets over several runs to ensure that this performance is stable. Moreover, this classification step was never implemented after a

detection step, that is, it was never tested on real detections arising from the object detection model. These two pipelines should be implemented together in order to assess the overall performance of this detection/classification two-step process.

5. The tracking algorithm worked well, but it lost sea pens in some of the cases. Most of these errors were easily solvable, as explained previously, but certain occasions in which the sea pens occluded the view are difficult to solve. However, this only happened a couple of times and we do not expect it to be an issue that decreases the overall tracking performance. Steps to decrease its effect, like considering a fraction of the image height, could be implemented in the future. Most importantly, however, this tracking algorithm should be implemented in the pipeline right after the detection and classification steps to obtain a closed system in which the sea pens are counted only once.

Figure 19 shows the flowchart of all the streams of work put into a single pipeline. Solid lines roughly represent connections that already exist. Dashed lines, on the other hand, show connections that need to be implemented, such as image pre-processing, tracking of the detected objects, or disregarding objects outside the lasers. Further research should be focused on these connections to create a stand-alone system.

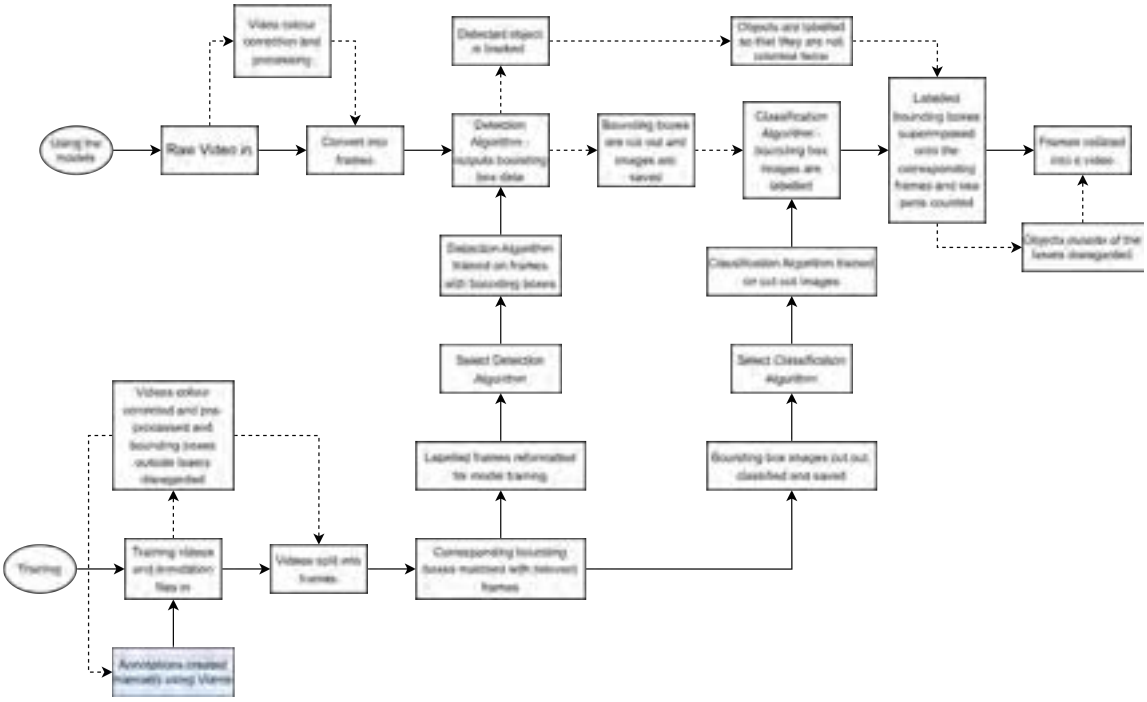

Figure 19: Flowchart of the data flow, including the use and training of the detection and classification models. Note that the dashed arrows imply non-implemented channels of the flow. Blue text boxes indicate a task that is completed externally.

9 Team members

In alphabetical order:

Meghna Asthana is a PhD Student in Computer Vision and Deep Learning. She supported the Detection and Classification sub-team. She was one of the facilitators for the project.

Robert Blackwell is a data scientist at Cefas. His principle contributions to the project were helping to fix git merge conflicts and making the tea. But also data wrangling, image pre-processing and sea pen classification models (VGG16 and ResNet50).

Sam Davis is a PhD student in Environmental Data Science at the

University of Sydney. He supported the Detection and Classification sub-team and was also one of the facilitators for the project.

Jessica Forsyth is a research associate working on Bayesian inference methods in applied mathematics and health technology assessments with a background in quantitative developmental biology. They contributed to the image processing areas of the project.

Kasia Kedzierska is a PhD student in biomedical applications of ML at the University of Oxford. She contributed to the data wrangling and cleaning, laser detection and sea pen detection from videos using YOLOv5n and YOLOv5s models.

Rafael Mestre is a New Frontiers Fellow at the University of Southampton. He was the PI of the project and contributed during project design with the challenge owners, project management, drinking coffee and putting out fires.

Zarreen Reza is an early career AI Research Scientist working in multiple industries for more than four years. She contributed to creating training data and implementing the sea pen detection and classification model using YOLOv5.

Joseph Ribeiro is a fisheries scientist at Cefas. He contributed to this project as a joint challenge owner and participant, particularly on the laser detection and image pre-processing sub-team.

Pirta Palola is a DPhil student in the Seascape Ecology Lab, University of Oxford. In her research, she uses machine learning and remote sensing technologies to study marine ecosystems. She contributed to the data preparation and the development of sea pen classification models.

Yanica Said is a PhD student in Mathematical Biology at the University of Malta and Oxford Brookes University. Her research focuses on the computational analysis of metabolic networks. She contributed to the data preparation, to the construction of the tracking algorithm, and by giving part of the final presentation.

Anna Downie was the challenge owner on behalf of Cefas.

All team members contributed to designing the project, data analysis and interpretation, presentation preparations and report writing.

References

[1] Anna Downie et al. *Automated detection of sea pens from video data: applications for time-series monitoring of Vulnerable Marine Ecosystems (VME)*. Tech. rep. 2022. URL: https://randd.defra.gov.uk/ProjectDetails?ProjectID=20916.

[2] VIAME Contributors. *VIAME: Video and Image Analytics for Marine Environments*. https://github.com/VIAME/VIAME. cff-version: 1.2.0. May 2017.

[3] Kitware. *Kitware - Open-source Software for Scientific Research and Development*. URL: https://www.kitware.com/.

[4] Kitware. *Frequently Asked Questions*. Accessed: 2023-01-20. URL: https://kitware.github.io/dive/FAQ/.

[5] Kitware. *Issue 421: FAQ*. 2020. URL: https://github.com/Kitware/dive/issues/421.

[6] Graham D Finlayson, Bernt Schiele, and James L Crowley. "Comprehensive colour image normalization". In: *European conference on computer vision*. Springer. 1998, pp. 475–490.

[7] Joost Van De Weijer and Theo Gevers. “Color constancy based on the grey-edge hypothesis”. In: *IEEE International Conference on Image Processing 2005*. Vol. 2. IEEE. 2005, pp. II–722.

[8] Farid García-Lamont et al. “Contrast enhancement of RGB color images by histogram equalization of color vectors’ intensities”. In: *international conference on intelligent computing*. Springer. 2018, pp. 443–455.

[9] Ali M Reza. “Realization of the contrast limited adaptive histogram equalization (CLAHE) for real-time image enhancement”. In: *Journal of VLSI signal processing systems for signal, image and video technology* 38.1 (2004), pp. 35–44.

[10] Ross Girshick et al. “Rich feature hierarchies for accurate object detection and semantic segmentation”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2014, pp. 580–587.

[11] Renjie Xu et al. “A forest fire detection system based on ensemble learning”. In: *Forests* 12.2 (2021), p. 217.

[12] Joseph Redmon et al. “You only look once: Unified, real-time object detection”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 779–788.

[13] Ross Girshick. “Fast r-cnn”. In: *Proceedings of the IEEE international conference on computer vision*. 2015, pp. 1440–1448.

[14] Shaoqing Ren et al. “Faster r-cnn: Towards real-time object detection with region proposal networks”. In: *Advances in neural information processing systems* 28 (2015).

[15] Glenn Jocher et al. *ultralytics/yolov5: v7.0 - YOLOv5 SOTA Realtime Instance Segmentation*. Version v7.0. Nov. 2022. DOI: 10.5281/zenodo.7347926. URL: https://doi.org/10.5281/zenodo.7347926.

[16] Mark Everingham et al. “The pascal visual object classes challenge: A retrospective”. In: *International journal of computer vision* 111 (2015), pp. 98–136.

[17] Tsung-Yi Lin et al. “Microsoft coco: Common objects in context”. In: *European conference on computer vision*. Springer. 2014, pp. 740–755.

[18] Kaiming He et al. “Deep residual learning for image recognition”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2016, pp. 770–778.

[19] Olga Russakovsky et al. “ImageNet Large Scale Visual Recognition Challenge”. In: *nternational Journal of Computer Vision* 115 (2015). ISSN: 0142-7873. DOI: 10.1007/s11263-015-0816-y. URL: https://doi.org/10.1007/s11263-015-0816-y.

[20] Alan Lukezic et al. “Discriminative correlation filter with channel and spatial reliability”. In: *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2017, pp. 6309–6318.

[21] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. *YOLO by Ultralytics*. Version 8.0.0. Jan. 2023. URL: https://github.com/ultralytics/ultralytics.

[22] Wei Liu et al. “Ssd: Single shot multibox detector”. In: *European conference on computer vision*. Springer. 2016, pp. 21–37.

[23] Oriane Siméoni et al. “Localizing Objects with Self-Supervised Transformers and no Labels”. In: 2021.

[24] Yangtao Wang et al. “Self-supervised transformers for unsupervised object discovery using normalized cut”. In: *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2022, pp. 14543–14553.

[25] Karen Simonyan and Andrew Zisserman. *Very Deep Convolutional Networks for Large-Scale Image Recognition*. 2015. arXiv: 1409.1556 [cs.CV].

A Quantitative analysis of colour normalisation

By pre-processing the video footage, the accuracy of sea pen counting could be improved at various points within the workflow: i) by improving accuracy of manual tracking, which in turn would provide better training data sets for the classifier models; ii) by potentially improving the accuracy of the detection and classification models; and iii) by improving the cross-comparison of counted sea pens and subsequent analyses.

In order to fully assess the impact of these potential pre-processing algorithms, we would have conducted tests to assess performance at each of these three stages. However, within the allotted time this was not possible. Instead, we decided to describe the colour variability through comparison of the average RGB histograms of each year's footage with a reference year's average RGB histogram. We applied this analysis to two 10-FPS videos from each of the years 2014, 2015, 2017, 2018, 2019, 2020 and 2021 and compared these to the average histogram from 7 10-FPS videos from 2016. We chose the 2016 videos to be our reference data set as this data was used as training data in the preliminary study by Cefas. We chose to compare the RGB histograms for the following pre-processing algorithms: GW, time-average HSL correction, CLAHE and GW+CLAHE. Example average RGB histograms for each year, for each processing method are shown in Appendix B.

To attempt to summarise the differences between the average RGB histograms for each year and the 2016 data we used two metrics, the Wasserstein distance and the Pearson correlation coefficient. The Wasserstein distance, also know as the 'Earth mover's distance' describes how much the bins of the histogram would have to be changed to match that of the reference histogram and therefore a value of 0.0 indicates a complete match, and larger distances describe distributions which differ more significantly. The Pearson coefficient can vary between -1, indicating a negative correlation between the two histograms and +1 indicating a strong correlation between the two histograms. A coefficient of 0 indicates no correlation.

Within this comparison we would therefore hope that the processed

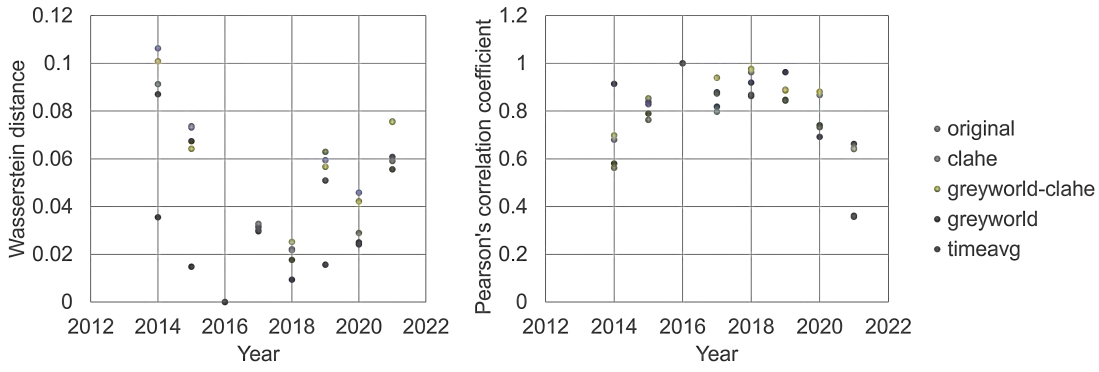

Figure 20: Comparison of average RGB histogram comparison metrics between 2015-2021 to the reference year 2016.

images had smaller Wasserstein distances and Pearson coefficients close to 1. We summarise our findings in Figure 20. We noticed that there was variable performance for each of the algorithms across the years and that no single algorithm improved either the Wasserstein distance or the Pearson coefficient, suggesting a more year specific approach based on a single reference histogram.

We did notice that in the Pearson coefficient analysis, the greatest reductions in RGB histogram similarity, even in processed images, were found in the years furthest from the reference year 2016. This could potentially suggest that the difference between these years (2015 and 2021) and 2016 was too significant for these simple video processing algorithms and that more sophisticated approaches such as comprehensive colour normalisation should be used. [6]

B Average RGB histograms for original and processed videos

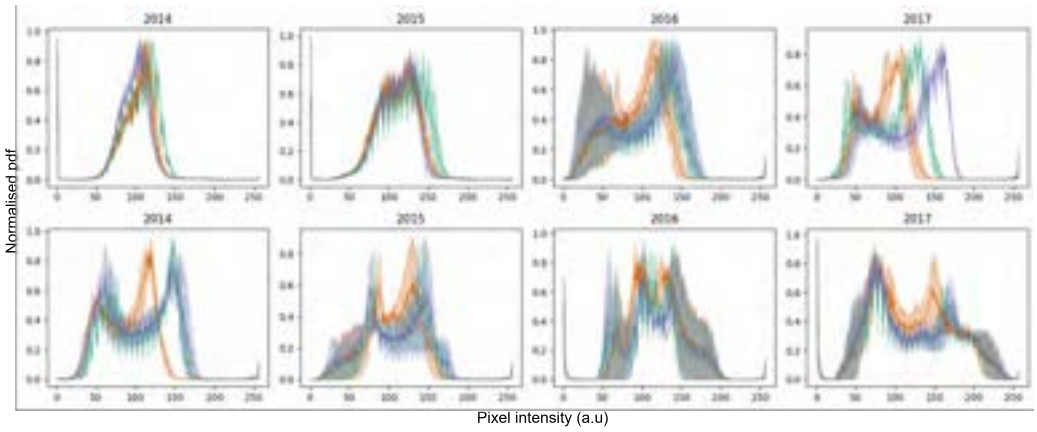

Figure 21: Average RGB histograms with associated error bars of minimum and maximum for each pixel intensity for a subset of original 10fps videos across each year.

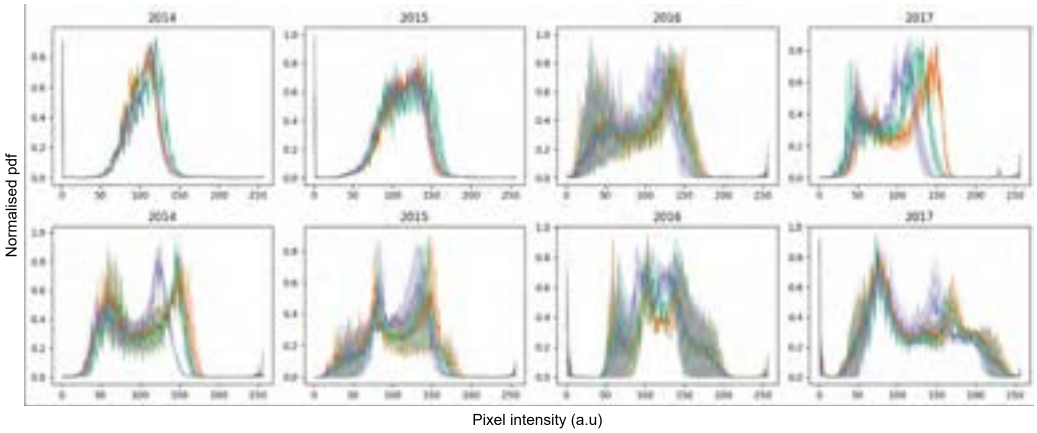

Figure 22: Average RGB histograms with associated error bars of minimum and maximum for each pixel intensity for a subset of 10fps videos processed using the GW algorithm across each year.

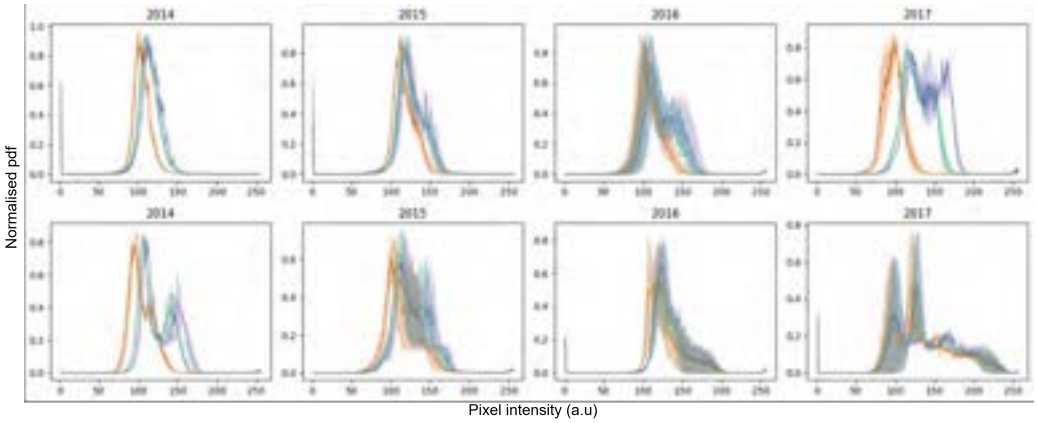

Figure 23: Average RGB histograms with associated error bars of minimum and maximum for each pixel intensity for a subset of 10fps videos processed using the time averages HSL algorithm across each year.

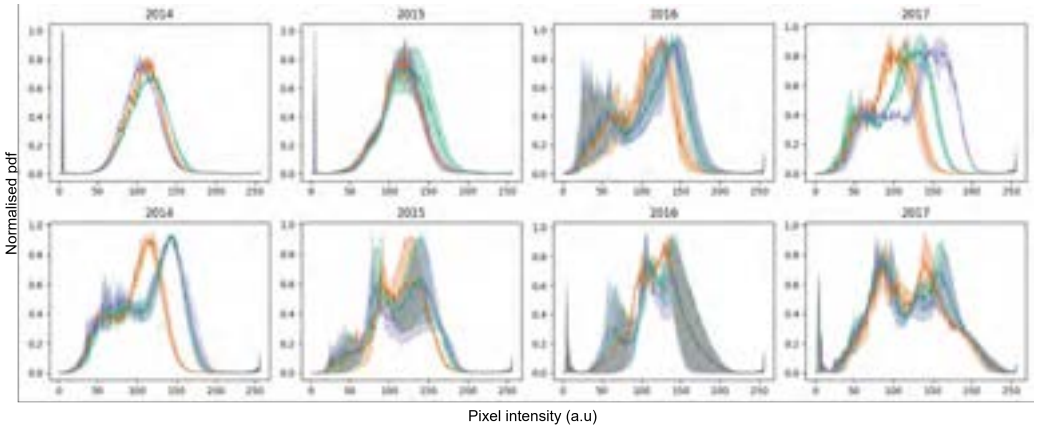

Figure 24: Average RGB histograms with associated error bars of minimum and maximum for each pixel intensity for a subset of 10fps videos processed using the CLAHE algorithm across each year.

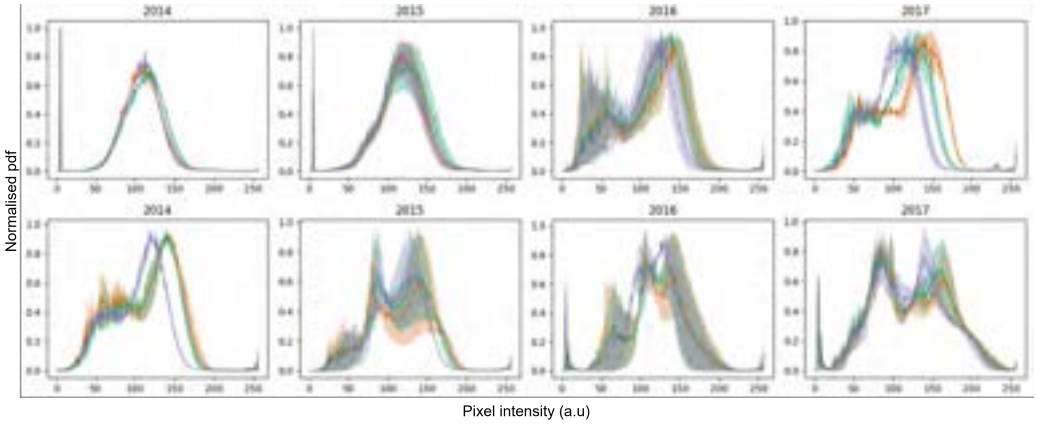

Figure 25: Average RGB histograms with associated error bars of minimum and maximum for each pixel intensity for a subset of 10fps videos processed using the multi-stage GW+CLAHE processing across each year.

C Partial results for the YOLOv5s model

As the YOLOv5n model took a short time to train, we also performed training on the next model, YOLOv5s (for 'small'). This model has a total of 7.2M parameters, whereas the YOLOv5n model has 1.9M parameters. On a NVidia GeForce RTX 2060 GPU, the YOLOv5s took only 2 h and 15 min to train, whereas the YOLOv5n model had taken 1 h and 40 min. Due to lack of time, we could not completely compare these results, and we present here the outputs of a training with the same images, suggesting a more in-depth comparison.

Figure 26 shows the results of the training. We can observe how the loss functions are saturating towards zero, without indications of overfitting to the data. The peromance metrics are also increasing with the epoch and, after 30 epochs, the mAP@0.5 reaches the value of 0.92408. In the YOLOv5n model, that value was exactly 0.90065. The improvement is not significantly higher and training took longer. This poses a trade-off between training time and accuracy, and it remains to be seend whether this small improvement is worth it. In any case, this comparison should be performed more systematically, with replicates, possibly more epochs,

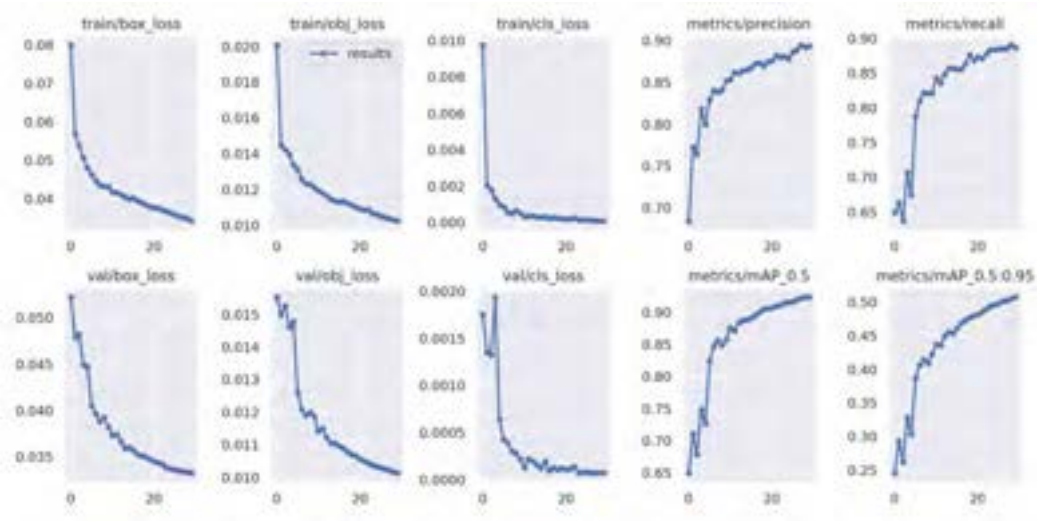

Figure 26: Training results of the YOLOv5 small model. The three left columns correspond to the loss function per training epoch and the right two columns to different performance metrics per epoch.

and it would also be interesting to test with the remaining larger YOLOv5 models to see whether the performance can significantly increase the reported here, albeit with longer training times.

D VGG16 for sea pen classification

The VGG16 model for sea pen classification was also investigated was part of Section 5.2, although with partial results. VGG16 is a CNN model that has achieved a top-5 test accuracy of 92.7% on the ImageNet dataset, which consists of over 14 million images belonging to 1000 different classes.[19]

The network architecture of the VGG16 model processes fixed-size RGB images of 224x224 pixels. Before training, the mean RGB value is subtracted from each pixel. The image then passes through a series of convolutional layers that use 3x3 filters. The stride for convolution is set at 1 pixel, with the input to the convolutional layer padded with 1 pixel to

Epoch	Train		Validation	
	Loss	Acc	Loss	Acc
0	0.4112	0.07965	0.1029	0.9645
1	0.1110	0.9617	0.0792	0.9725
2	0.0659	0.9779	0.1053	0.9734
3	0.0555	0.9816	0.0483	0.9822
4	0.0451	0.9842	0.0845	0.9687

Table 8: Partial results of the VGG16 two-class model after training for 5 epochs.

maintain its resolution after convolution. Five max-pooling layers follow some of the convolutional layers to perform spatial pooling. Max-pooling is applied over a 2 × 2 pixel window with a stride of 2. Finally, the image is processed through three fully connected layers, with the first two having 4096 channels each and the third having as many channels as output classes needed for classification. The final layer is the soft-max layer and all hidden layers are equipped with the rectification non-linearity (ReLU).[25]

In this case, we also applied the concept of transfer learning and used a pre-trained models for image recognition on the ImageNet dataset,[19] whose models weights are imported in PyTorch under the name IMAGENET1K_V1 (notice that for ResNet50, the model weights were 'version 2'). Table 8 shows the results of the two-class sea pen problem (no background images) partially trained for 5 epochs. As these are only partial results, they should be considered with care, but this first attempt shows results as promising as the previous ones.

The
Alan Turing
Institute

turing.ac.uk

@turinginst