Image pre-processing in marine debris detection

Bachelor's Project Thesis

Yusuf Köse, s4239458, y.e.kose@student.rug.nl
Supervisors: Dr B.J. Wolf

Abstract: Tons of debris end up in the seas and oceans each year, a lot of which sink to the seafloor. This paper is an attempt to make it easier to rid our seafloors of debris and aiding the SeaClear Project. The goal is to reduce the murkiness and improve the colour balance of underwater footage through various image pre-processing methods, namely; CLAHE (Contrast Limited Adaptive Histogram Equalisation), UCM (Unsupervised Colour Correction Method), IBLA (Underwater Image Restoration Based on Image Blurriness and Light Absorption) and funieGAN (Fast Underwater Image Enhancement for Improved Visual Perception), in order to make object detection easier and more consistent. The YOLOv8 model was chosen to be trained with these, as it is shown to be one of the highest performing object detection models both in terms of detection rate and processing speed. The images resulting from each pre-processing method were used to create a corresponding model for that method. These models are assessed based on operational speed (FPS), mAP, accuracy, precision, recall and F1 score, and compared to a model trained with the original (non-pre-processed) images. The combined predictions of all models was also assessed in order to see what the best results are that can be achieved. The UCM and the combination of the models achieved higher overall mAP and F1 scores than the original model, although their processing speeds render them inefficient for real-time use. CLAHE and funieGAN models can be considered if specific objects are being targeted.

1 Introduction

Our oceans are responsible for about 50 percent of all oxygen production on our planet, while also absorbing about 31 percent of our CO2 emissions (*How much oxygen comes from the ocean?* (2023), *Quantifying the Ocean Carbon Sink* (2022)). On top of this, a lot of countries rely on them for their tourism and fishing, which can account for up to 38 percent of a country's GDP (Frost (2019)). We dump over 14 million tons of trash into our oceans, 70 percent of which sink towards the ocean floor. The debris that end up in our seafloors can harm the ecosystem by altering and degrading marine habitats. They do this by infusing with the algae, releasing chemicals into their surroundings and being consumed by wildlife either directly, or indirectly by altering their food sources (*Habitat: Marine Debris Impacts on Coastal and Benthic Habitats* (2016)). These alterations are sure to regress the way our oceans function. Although most of those debris are micro-plastics, there is still a lot of it that could be held by hand. That is what SeaClear attempts to take advantage of (Evers (2021), *Marine Plastic Pollution* (2021)).

SeaClear is a project which aims to clear out the trash that can be physically held from the ocean and seafloors. They do this by sending a boat to the target area, mapping the seafloor, and then sending remotely operated vehicles (ROVs) with visual systems which make use of object detection algorithms to detect trash or other objects on the parts of the ocean floor that seem most likely to contain debris. An issue that can be faced during these dives of ROVs is the visual limitations that the environment poses. Water in the deep tends to be murky and colours tend to be shifted or muted due to the properties of water. These may impair the vision of the ROVs on mission, and therefore make debris detection difficult. This paper is an attempt to tackle this problem.

There are multiple papers that discuss improvement of underwater images through pre-processing. These methods aim to enhance images in different ways; colour correction, dehazing, contrast enhancement and more. Four of such improvement methods were chosen to be tested against the original images. The goal of this paper is to find out **"To what extent can underwater object detection be improved through image preprocessing?"**.

The chosen four methods to be tested will be explained in the Background section. The Methods section goes over how these methods were implemented and assessed, the Results section goes over the numerical outcomes that came from testing, Discussion section is where the results are analysed and the Conclusion is where the outcomes are drawn. The Further Research section goes over how this research can be added onto or improved upon.

2 Background

2.1 CLAHE

Contrast Limited Adaptive Histogram Equalisation (CLAHE) is a contrast enhancement technique whose original purpose was to improve medical imaging although it was and is used also in other contexts. It was developed by Zuiderveld as an improvement to Adaptive Histogram Equalisation (AHE) which was an improved version of Histogram Equalisation (HE).

HE is a contrast enhancement technique, that relies on the frequency distribution (histogram) of image grey levels to make grey-level assignments across the whole image. When there are more pixels in a certain class of grey levels, it is optimal to assign a larger part of output grey ranges to the corresponding pixels. Grey-level transform is made through cumulative histograms in order to meet this condition. The result is ideally an approximately flat histogram. This method although can lead to a higher visibility of noise as irrelevant areas may be saturated, and does not take local contrast requirements and minor contrast differences into account.

With AHE attempts improve on this by taking local contrast requirements and minor contrast differences into account. This is attempted to be done by dividing the image into a grid of contextual regions. Separate histograms are calculated for each of these regions, therefore optimising the contrast enhancement for each region independently. Visibility of regional boundaries that may occur during this process is tackled through a bilinear interpolation scheme. Although these methods helps with improving contrast enhancement in regions of interest, it also leads to a substantial increase in background noise as contrast enhancement is attempted in each of the grid regions, some of which fall completely on background. This results in a sub-optimal image representation.

CLAHE is an attempt to fix the background noise that comes with AHE, by limiting the contrast enhancement in homogeneous areas. These areas are identified by regions where the histogram shows a high peak, indicating that most pixels are within the same grey-level range. CLAHE limits the slope associated with grey-level assignments by allowing only a maximum number of pixels to be associated with a local histogram. The rest of the pixels are considered “clipped” and are redistributed across the histogram. Clipping limit is an argument of CLAHE, with lower clipping limit resulting in less intense contrast enhancement with prevention of background noise and possible loss of information in regions of interest, while higher clipping limits lead to higher enhancement in regions of interest, while also adding background noise (Zuiderveld (1994)).

2.2 UCM

UCM is an underwater image enhancement method whose function is to improve underwater images by colour balancing (equalisation of RGB colours) and the contrast correction of both the RGB and HSI colour models, and was developed by Iqbal et al. (Iqbal et al. (2010)). The equalisation of RGB colours is done by determining the dominant colour by finding the mean R, G and B values across all pixels and determining which is the highest. We can assume that this will be the B channel, as the B (Blue) channel is often the dominant colour in underwater images.

Thereafter, two gain factors are created for the two non-dominant channels;

$$A = B_{\text{avg}}/R_{\text{avg}} \tag{2.1}$$

$$B = B_{\text{avg}}/G_{\text{avg}} \tag{2.2}$$

And then these factors are applied to every pixel iteratively with below given equations, where R and G are the old values for the current pixel's R and G channels and the R' and G' are the new values for the current pixel's R and G channels;

$$R' = A * R \tag{2.3}$$

$$G' = B * G \tag{2.4}$$

This creates a colour equalisation effect.

Contrast correction of the RGB model is done by stretching the intensity values to the desired range of values. The upper and lower limits of the image are determined which are often (0-255) in the RGB model. In the following, usually the maximum and minimum values in the histogram are determined and stretched to the desired range. In order to prevent being affected by outliers, a certain upper and lower percentage of pixel values (such as x $<$ 0.2% and 99.8% $<$ x) are ignored. The contrast correction is applied to the pixels within this range (0.2% $<$ x $<$ 99.8%) via the formula below;

$$P_0 = (P_i - c)\frac{(b-a)}{(d-c)} + a \tag{2.5}$$

where *P0* is the contrast corrected pixel value, *Pi* is the considered pixel value, a is the lower limit value (0), b is the upper limit value (255), c is the minimum pixel value currently present in the image and d is the maximum pixel value currently present in the image.

This is applied to the upper values of the intensity range considering the lowest color value component, (often) Red (R) in underwater images, and the lower values of the intensity range, considering the highest color value component, (often) Blue (B), and to the entire range with the default formula. When applying to the upper values of the intensity range, a is substituted as minimum of R and when applying to lower values of the intensity range, b is substituted as maximum of B.

Contrast Correction of the HSI model is done through the Saturation (S) and the Intensity (I) components. The above mentioned process is applied to both of these components, where the application to the S component helps obtain the true colours of the underwater image while the application to the I component creates better image illumination (Iqbal et al. (2010)).

2.3 funieGAN

The goal of FUnIE-GAN(Fast Underwater Image Enhancement for Improved Visual Perception), developed by developed by Islam et al., is to learn a mapping G: X -> Y given a source domain X and desired domain Y in the context of underwater images (Islam et al. (2020). It is based on a conditional Generative Adversarial Network (GAN) model where the generator and discriminator compete against one another in order to create results closest to the desired outcome. The generator is made up of an encoder-decoder network. The input to the generator are of dimensions 256 x 256 x 3, and the output dimensions are the same by the end of encoding and decoding.

A conditional adversarial loss function is created, where the generator tries to minimise it, therefore minimising the discriminability of the output from the desired image. The discriminator tries to maximise the loss function, by being good at discriminating between the output and the desired outcome even at higher similarities. This leads to both components pushing one another to perform better.

The GAN is trained in two different ways; paired training and unpaired training. In paired training, the generator attempts to generate an image (from a low quality image) as close to an existing ground truth (high quality image) as possible. In unpaired training, there is no ground truth and instead the generator attempts to both generate an output from the source, and also the source from the corresponding output (Islam et al. (2020)).

2.4 IBLA

IBLA (Underwater Image Restoration based on Image Blurriness and Light Absorption) is an image restoration method proposed by Peng & Cosman which restores images based on their blurriness and light absorption (Peng & Cosman (2017)). Image blurriness and light absorption are determined through estimates of Background Light (BL) and depth.

Image blurriness is determined on a filtered greyscale version of the input image where more visible regions appear relatively highlighted while the less visible regions appear darker.

BL estimation helps determine the colour and restore scene radiance of an underwater image. A high estimated (bright) BL results in a low scene radiance while a low estimated (dim) BL results in high scene radiance upon restoration. A similar effect applied to the colour of the image, where the estimated BL colour values will be reversed (an image with a blue BL will appear more red upon restoration and vice versa). The BL value is taken from a region of the image with high blurriness and low variance (as such a region is likely to be further back on the scene and reliable).

Depth estimation is derived through the relative appearance of blue, red and green light in the scene, based on the location of appearance and abundance and through image blurriness.

Finally Transmission Map (TM) estimation and scene radiance recovery is done through calculating the distance from the camera to scene points and restoring the image accordingly with colour adjustments(Peng & Cosman (2017)).

These methods could possibly be applied in real time, where each frame that the camera records would be put through a pre-processing method before the object detection algorithm is applied. This of course takes time, the duration of which depending on the method used, and therefore will drop the rate of which the area can be analysed. As the object detection model itself is also a tying factor of the rate of analysis, it would be to our best interest to keep the running time of it to a minimum.

3 Methods

3.1 Model Selection

Some studies have shown that YOLO algorithms can overall match or surpass the performance of other commonly used model types in terms of mAP, such as Faster R-CNN (Fulton et al. (2019), Uras et al. (Unpublished)). They can also achieve this while working significantly faster and putting out higher frame rates on most processors (Fulton et al. (2019)). At the time of this study, YOLOv8 is the latest version of YOLO models. Ultralytics display that the YOLOv8 Nano model has the best mAP to latency ratio in comparison to all the previous YOLO versions (Ultralytics (2023)). YOLOv8 Nano, the smallest YOLOv8 model, was therefore chosen to be utilised for model training in order to minimize frame-rate loss.

3.2 Dataset

The data used to train the pre-trained model was chosen to be the TrashCan dataset which is a data set containing 7212 images taken from underwater footage. This data set has two different versions; material, which categorises objects into the classes of the general material they are made of (metal, plastic), and instance, which categorises objects into classes of the more specific type of object that they are (can, bottle). Both were annotated in the JSON format. In this paper, the instance version, which contains 22 classes, was chosen in order to be better informed of what exactly is being detected as this may help the collection process in future expeditions if certain objects are to be targeted (Hong et al. (2020)).

3.3 Data Pre-processing

To prepare the data for model training, validation and testing, Roboflow, a website in collaboration with Ultralytics was used (*Everything you need to build and deploy computer vision models* (2023), Ultralytics (2023)). The dataset was uploaded to the website with the annotations, which are automatically read from the website. The data was split into train, validation, and test sets. The data split into each set in terms of percentage were chosen to be 85/15/5, differently from the traditional 80/10/10 to prioritise validation. The images were resized into 480 by 480 pixels (nearest square) each, which was done for consistency as there are two different sizes of images in the data set. This also adjusts the annotations to fit the new image size. Within Roboflow, the annotations were converted to the YOLOv8 format.

In addition to the dataset being used as is, an augmented version was created for each of the models. The augmentation of choice was 1.5% noise, where in an image, 1.5% of the pixels were randomly set as white. This was applied to all the images in the training set (while retaining the original images) and resulted in double the number of datapoints in order to enable more thorough training.

Letterboxing was also used when resizing the images in addition to regular resizing. Letterboxing is a way to sustain the original aspect ratio of an image by adding black padding to the part of the enlarged/shrunk image until it reaches the intended aspect ratio. For example, some of the images in the TrashCan data set are 480(height) by 270(width). In this case, we could add ((480-270)/2) 105 by 480 padding to the top and the bottom of the image to turn it into a square image while the relevant area of the image keeps its original aspect ratio. A demonstration of this can be seen in figure 3.1. This would help prevent the image distortion with regular resizing, which causes the images to be stretched and therefore decreasing in quality and the objects therefore being slightly altered. This could in turn lead to a difference in model performance. Letterboxed versions of the same pre-processed images were created in Roboflow, in the pre-processing section, by resizing with option 'fit (black edges) in'.

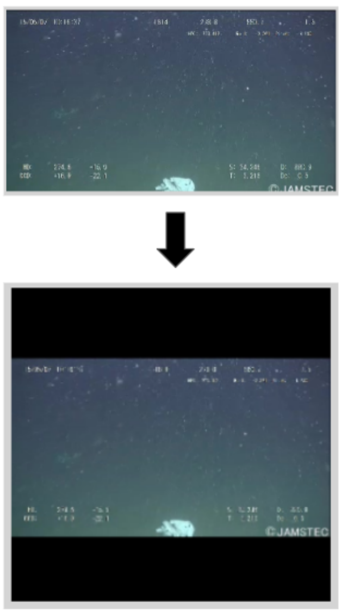


Figure 3.1: A demostration of the letterboxing method

For the application of the pre-processing methods, available code implementations were utilised *(wangyanckxx (2021), xahidbuffon (2020)). 4 new different versions of the dataset were created. Some of the aforementioned methods, namely UCM IBLA and funieGAN, were used to create thier corresponding pre-processed images (datasets). For CLAHE, the method was applied via the CLAHE function of OpenCV to the V component (in the HSV model) of a coloured image and the clipLimit (2) was set to 5.0 as lower clip limits did not result in enough visible contrast and higher clipLimits added noise. For each of those pre-processing methods, a new folder was created, with each file retaining the original filenames for easy accessibility. During the pre-processing, the time taken for each application was recorded in order to be able to obtain the mean processing time for each method. All pre-processing methods, except for funieGAN, were applied to the original images with the original size, then were resized in Roboflow as previously described. funieGAN's implementation only worked on images 252 x 252, therefore they had to be first resized with Roboflow in order to have matching annotations, and after the processing they were resized back to 480 x 480 in order to have consistent sizes with the other pre-processed data sets.

*The repositories where the code implementations of the pre-processing methods that were used can be found at `https://github.com/wangyanckxx/Single-Underwater-Image-Enhancement-and-Color-Restoration`. and `https://github.com/xahidbuffon/FUnIE-GAN`

3.4 Model Training and Testing

For each model, a remote access to the prepared data which is given by Roboflow was used in Google Colab. T4, the fastest of the available free GPUs was utilised across all models and all processes. The the number of epochs was set to 27 across all models, as improvements in metrics were not observed at higher epochs. The models were then each validated, from which the model's performance in terms of precision, recall, mAP50 and mAP95 are obtained for both boxing and segmentation, and also provides how many times each of the classes were observed. Finally each model's performance was visually observed on test data. After validation, confusion matices and other results were generated to be used for analysis.

These trained models were manually tested for their accuracy. This was done by creating a list of images with their corresponding labels. Each model was run on their corresponding pre-processed test set. When any of the model's predictions matched any of the image labels or when there was no label in the image and the model made no predictions, that was counted as a correct prediction on the image, and otherwise not. The accuracy was calculated as the n of correct predictions/n of total predictions for each model.

Each of the models' predictions were combined and assessed in the same way. The combined predictions were achieved though storing the predictions of each of the models for each image in separate lists, and then combining them into one list by image. This results in a list where each item corresponds to all of the predictions made by each model for a specific image. This was done to combine each of the models' strengths and achieve a higher accuracy score. This was also used for recall and precision, as described below.

The recall in this case is given by

$$R = \frac{TP}{TP + FN} \tag{3.1}$$

where TP is the number of predictions that are a part of the labels and FN is the number of labels that are not a part of the predictions. Similarly, the precision is given by

$$P = \frac{TP}{TP + FP} \tag{3.2}$$

where TP means the same and FP is the number of predictions that are not in the labels. Recall refers to the capability of the model to detect an object when its there, and the precision refers to the model's ability to predict only what is present and nothing else. Both of these metrics can be combined into one with the *F1* score, which is

$$F1 = 2\frac{P * R}{P + R} \tag{3.3}$$

The *F1* score is viewed as a better metric than accuracy, as its considers specific attributes of a model's detection and identification capabilities, namely, recall and precision.

4 Results

In table 4.1, the corresponding processing times for each model are displayed. The processing time of the original model indicates the inference time of the YOLOv8 model which was identical for all models, and the rest of the processing times indicate how much additional processing time is required on top of the model inference time. The FPS indicate the resulting frame rate from the addition of inference time and pre-processing time.

In figure 3.2 below, some example produced outputs of each of the pre-processing methods, in addition to the original images can be seen.

In *Table 3.1*, the Box mAP scores for the 22 classes, with the addition of mAP accross all classes combined, can be seen for each of the models with their corresponding pre-processing methods. UCM model has achieved the highest score across 9 classes, the Original model 7, the CLAHE model 2, the IBLA model 3, and the funieGAN model 2.

To test whether these mAP score differences between the models are statistically significant, one-tailed t-tests were performed with each of the models against one another. The input to these t-tests were the models' corresponding mAP scores for each class. The most notable differences were between UCM and CLAHE (t=2.381, df=42, p=0.021) and Original and CLAHE (t=1.995, df=42, p=0.053). On the basis of p(0.021) $<$ 0.05, we can consider the UCM model as significantly better in terms of mAP than the CLAHE model. While the p=0.053 obtained from comparing the Original model to the CLAHE model is not statistically significant, it is close enough to being significant for the mAP differences to be taken into consideration. The mAP differences between the two highest overall performing models, UCM and Original, were found to be highly statistically insignificant (t=0.207, tf=42, p=0.837).

Despite the non-significant overall differences between the Original and UCM model their performances were observably skewed to opposite sides. This can be seen in the confusion matrices in figure 3.3 below, where in the top left corner, the UCM model has visibly less divergence from the true class in comparison to the Original model, which is where the organism classes are located. In addition to this, when looked at the class mAP scores, the UCM model scores higher than the Original model for 5 of the 7 classes that are organisms (crab, eel, fish, shells, starfish, plant, etc) while the Original model scores higher than the UCM model for 8 of the 14 classes that are debris. The mAP scores of the UCM model and the Original model for organism and debris classes can be seen in table 4.2. We test for significance with a t-test with only these

Object Instance	original	clahe	ucm	ibla	funieGAN
all	0.86	0.792	0.866	0.795	0.817
animal_crab	0.746	0.74	0.863	0.664	0.624
animal_eel	0.838	0.688	0.824	0.724	0.811
animal_etc	0.609	0.735	0.833	0.515	0.653
animal_fish	0.893	0.826	0.865	0.844	0.88
animal_shells	0.626	0.697	0.651	0.477	0.5
animal_starfish	0.712	0.674	0.864	0.835	0.754
plant	0.873	0.788	0.919	0.835	0.856
rov	0.937	0.927	0.957	0.93	0.944
trash_bag	0.908	0.928	0.913	0.886	0.909
trash_bottle	0.937	0.868	0.975	0.819	0.839
trash_branch	0.937	0.897	0.899	0.933	0.94
trash_can	0.907	0.894	0.906	0.883	0.859
trash_clothing	0.885	0.895	0.995	0.995	0.884
trash_container	0.969	0.965	0.939	0.89	0.953
trash_cup	0.995	0.829	0.946	0.881	0.928
trash_net	0.791	0.588	0.665	0.564	0.793
trash_pipe	0.921	0.824	0.815	0.938	0.929
trash_rope	0.79	0.523	0.811	0.646	0.523
trash_snack_wrapper	0.968	0.658	0.827	0.609	0.884
trash_tarp	0.826	0.785	0.801	0.838	0.757
trash_unknown_instance	0.88	0.859	0.9	0.857	0.862
trash_wreckage	0.974	0.842	0.889	0.942	0.884

Table 3.1: (Box) mAP scores of each model with the corresponding pre-processing method, for each of the object classes. The highest score for each class is marked red (green if even), which indicates that the corresponding model for that score detects this class the best.

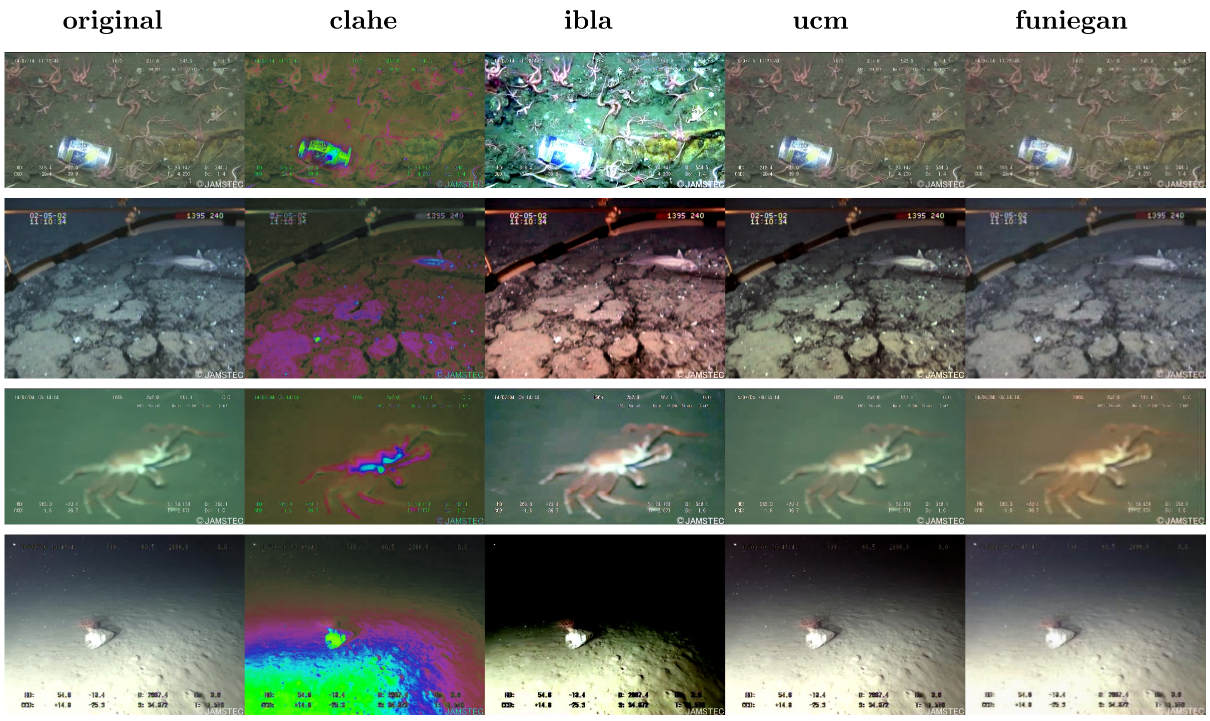


Figure 3.2: Some examples of how the given pre-processing method changes an image that it's applied to.

	original	clahe	ucm	ibla	funieGAN
Processing Times	7ms	+3ms	+2.9s	+9.8s	+30ms
FPS	142	100	0.34	0.10	27

Table 4.1: Processing speeds of each model with their corresponding pre-processing method.

	Organism	**Debris**
original	0.756	0.906
ucm	0.831	0.877

Table 4.2: The mAP scores of the UCM model and the Original model for organism and debris classes (YOLOv8 validation output)

	Accuracy
original	0.122
clahe	0.112
ucm	0.134
ibla	0.114
funieGAN	0.118
combined	0.151

Table 4.3: Accuracy scores of each model with the corresponding pre-processing method, including the accuracy of their combination (manually calculated).

specific classes as inputs. For organism classes, the difference between the UCM model's and Original model's mAP scores was found to be insignificant (t=1.377, df=12, p=0.193) and for debris classes, the difference between the UCM model's and Original model's mAP scores was found to be also insignificant (t=1.000, df=26, p=0.326).

The accuracy scores obtained by each of the models, and the accuracy score of their combination can be seen in table 4.3 below. The rankings were in accordance with the overall mAP scores. The combination of all models (explained in section 3) resulted in the highest accuracy score of 0.151.

	original	clahe	ucm	ibla	funieGAN	combined
Precision	0.074	0.066	0.081	0.066	0.71	0.075
Recall	0.079	0.077	0.091	0.074	0.078	0.125
F1	0.076	0.071	0.086	0.070	0.074	0.094

Table 4.4: Recall, Precision and F1 scores of each model with the corresponding pre-processing method, including the scores of their combination (manually calculated).

The results displayed in table 4.4 show that the UCM model is the best model on all the metrics. In contrary to the mAP scores and the accuracy scores, the IBLA model has performed worse than the CLAHE model, evidently by the lower F1 score. The combination of all models provides the second highest Precision, while resulting in the highest Recall by a substantial amount. Given this, it also results in the highest F1 score of 0.094.

The overall mAP results from the models which were trained and validated with Letterboxed images against the models which were trained and validated on normally resized images can be seen in table 4.5.

The overall mAP results show that letterboxing causes a slight improvement for UCM and CLAHE models while being a detriment to the Original and IBLA models.

Finally, the overall mAP results from the models which were trained on augmented dataset (as explained in section 3) against the models trained on the original dataset can be seen in 4.6.

The overall mAP results from the augmented-dataset models show that data augmentation is an improvement for all of the models. The CLAHE, IBLA and funieGAN models saw the highest improvement with over 5% higher mAP scores.

5 Discussion

To answer the question of **"To what extent can underwater object detection be improved through image pre- processing?"**, all aspects of each of the models have to be considered.

Given the processing speed and the overall peformance, the IBLA model can be left out of consideration when it comes to its real-time deployment in ROVs. Its slow functioning in comparison to other models is caused by the relatively high complexity of its process as described in section 2 (Peng & Cosman (2017)). This is reflected by the number of different processes the image goes through before the output is obtained. The evidence of colour shifting into red and blue hues can be observed in Figure 3.1, as some images appear relatively blue while the others red. As this inconsistency is reflected along the whole dataset, it may have resulted in an inefficient training process. In addition to this, this method was initially constructed for larger images which affects the filtering process during BL calculation (as described in section 2). These images in the TrashCan dataset being of smaller size may have caused a faulty or non-ideal filtering process. Conclusively, the IBLA model performs the best among only the trash_pipe and trash_tarp classes, and the second worst in terms of overall mAP. Even if those two classes were the only ones to be focused on, its long processing time (9.8s on Colab T4 GPU) can't justify its usage for two reasons;

1) If the ROV is to move at a constant rate, it is likely to miss any debris that do not stay in frame for at least 9.8 seconds.

2) Stopping the ROV for at least 9.8 seconds on suspected debris would be inefficient for the overall debris removal efforts.

When it comes to the CLAHE model, given its very quick processing time (3ms on Colab T4 GPU), it can achieve a frame rate that is more than necessary to analyse any frame that may contain and object of interest (100 FPS with the model inference time and pre-processing combined). This is a result of the lack of complexity of its process as described in section 2 (Zuiderveld (1994)). Its poor performance across the metrics in comparison to the other models is likely caused by the process's original intent, which is to be used on greyscale images. In this case, it was applied to the V component of the HSV model, which resulted in the brighter areas of the images appearing green and a

	original	clahe	ucm	ibla
Letterbox-resized	0.838	0.795	0.869	0.782
Regular-resized	0.860	0.792	0.866	0.795

Table 4.5: The mAP scores of each model, with letterboxing and with regular image resizing (YOLOv8 validation output).

	original	clahe	ucm	ibla	funieGAN
Augmented Dataset	0.882	0.855	0.896	0.852	0.869
Original Dataset	0.860	0.792	0.866	0.795	0.817
Difference	+0.022	+0.063	+0.030	+0.057	+0.052

Table 4.6: The mAP scores of each model, trained and validated on the augmented dataset and with the original dataset (YOLOv8 validation output).

slight hue shift and a reduction of brightness in the rest of the image. This causes a reduction in visibility towards the objects that aren't fully lit, causing a high inconsistency in object appearances depending on the light reflection from the object in the scene. This is made apparent by the low precision obtained by the CLAHE model, as seen in Table 4.2. Despite this, the CLAHE model performs the best in the animal_shells and trash_bag classes. In case of bags being the exclusive focus of an expedition, the usage of the CLAHE model can be considered. Shelled creatures often only settle on hard surfaces, therefore being able discern the best between bags and shells or shells on bags is unlikely to pose an advantage for the CLAHE model during an on-field deployment.

In all of the overall metrics, both the ones outputted by the YOLOv8 validation procedure and the ones manually obtained, the UCM model has achieved the highest scores, which makes it seem to be the most suitable model for deployment. This is likely caused by the UCM process being structured based on widely applicable generalisations, such as the blue colour often being the dominant colour in underwater images and the red colour being the first to disappear (Iqbal et al. (2010)). The steps taken throughout the process are simple and consistent, without leaving any variables that have to be fine tuned to the dataset, thus giving consistent results. The one detrimental factor of this model, like the IBLA model, is the long pre-processing duration (2.9s on Colab T4 GPU). Although it takes less than 1/3 of the time that the IBLA model takes, it is still prone to missing debris while the ROV is in motion. Rather than being used for debris detection, the UCM method seems to be more suitable for identifying underwater creatures, for example, for a documentary, as it has shown to be better at detecting animal classes than the other models, and if an animal is in motion, it would likely be followed for over 2.9 seconds, making the pre-processing duration potentially applicable.

The funieGAN model, although lower than that of CLAHE, is able to process visuals at a real-time suitable frame rate of 27 FPS. The one challenge faced with the funieGAN model was the forced resizing of the images, which had to be first shrunk and then enlarged. The shrinking before inputting into the GAN model has likely caused a decrease in image quality and detail. Another detriment to the functioning of this model is that the paired training of this model was done on a certain dataset (Islam et al. (2020)), which appeared to consist of images of higher resolution and quality and different context. Instead of the camera pointing diagonally to the ocean floor, it is pointed horizontally in open ocean in most images. This may have resulted in the model not being accustomed to images of lower or quality or images of this specific context and thus not functioning optimally on the TrashCan dataset. Despite this, given that it performs the best in two classes, namely trash_branch and tash_net, its deployment can be considered if either of those two classes are to be targeted specifically.

With all the models having been considered, the Original model still remains the most real-time deployable out of them. It can achieve a working speed of 147 FPS (on Colab T4 GPU), which makes it certain that it won't miss any frames, given that the maximum speed of the ROVs do not surpass 2.6 m/s. It is also the best performing model in terms of mAP when it comes to detecting debris (as opposed to living creatures, in which UCM displays the higher performance). As it does not suffer from any delay due to image pre-processing, the original model would be the most efficient model for deployment, and would likely result in the most debris being cleaned of the ocean and sea floors.

Something else that can be considered upon deployment is combining the predictions of the fastest running models. As shown in the Results section, the combination of all models has resulted in the overall highest measured accuracy and recall. Ideally, the Original, CLAHE and funieGAN models could be run in parallel and retain the minimum FPS of 27 FPS. If they were to be run sequentially, this would drop the fame rate to 18 FPS, which is still considerable given that the ROVs deployed by SeaClear have a maxmimum speed of 5 knots (2.57 m/s) , and assuming that a visible distance of an object is 1 meter, each possible object of interest would have a minimum of 7 chances to be detected (Subsea Tech (2023)).

The models trained on augmented data (as explained in section 3) have displayed a substantial increase in performance. While their relative performance stayed similar, CLAHE, IBLA and funieGAN models have seen further increase than the rest. This makes the deployment of CLAHE and funieGAN models more considerable, as with augmentation, the gap of performance between them and the Original model is reduced.

Finally, the letterboxing results indicate that they are unlikely to cause any effective improvements to the models, as displayed in Table 4.3, no substantial increase was observed, although a decrease can be seen for two models, namely Original and IBLA. This may be a result of the relevant area of the image being smaller in size (although less distorted), and therefore containing less information for the model to learn from.

6 Conclusion

In summary, the pre-processing methods did not prove to create an overall advantage when it comes to underwater debris detection. The usage of the CLAHE and funieGAN models can be justified when the specific classes that they perform the best at are targeted, especially when data augmentation is involved in the training process. The UCM and the IBLA models are rendered unusable due to their long pre-processing times. These models could only be effectively used if they are run on a more capable GPU which can boost their operating speed up to at least 3 FPS. This would be the minimum number of frames necessary to process any frame containing an object at least once if their visible distance is considered to be 1 meter away or less, at maximum ROV speed. All things considered, the original model without any image pre-processing, especially with data augmentation, is observably the most suitable model for real-time debris detection due to having the highest mean debris mAP and the highest processing speed.

7 Further Research

An improvement on this study could be getting rid of the irrelevant information in each of the images, or using a different data set that does not contain them. In the images that were used in this study, there is technical and recording information overlaid, which may interfere with the model's learning of object properties, in which some of the objects will have such text overlaid on top of them and the others will not. Not having such noise would lead to a more consistent training and detection process.

References

Evers, J. (2021). National Geographic. Retrieved from https://education.nationalgeographic.org/resource/great-pacific-garbage-patch/ (Accessed: 2023-12-09)

Everything you need to build and deploy computer vision models. (2023). roboflow. Retrieved from https://roboflow.com/ (Accessed: 2023-12-09)

Frost, N. (2019). *These are the countries most reliant on your tourism dollars.* Retrieved from https://qz.com/1724042/the-countries

-most-reliant-on-tourism-for-gdp (Accessed on 2023-12-09)

Fulton, M., Hong, J., Islam, M. J., & Sattar, J. (2019). Robotic detection of marine litter using deep visual detection models. In *2019 international conference on robotics and automation (icra)* (p. 5752-5758). doi: 10.1109/ICRA.2019.8793975

Habitat: Marine debris impacts on coastal and benthic habitats. (2016). NOAA. Retrieved from https://marinedebris.noaa.gov/sites/default/files/publications-files/Marine_Debris_Impacts_on_Coastal_%26_Benthic_Habitats.pdf

Hong, J., Fulton, M., & Sattar, J. (2020). *Trashcan: A semantically-segmented dataset towards visual detection of marine debris.*

How much oxygen comes from the ocean? (2023). NOAA. Retrieved from https://oceanservice.noaa.gov/facts/ocean-oxygen.html

Iqbal, K., Odetayo, M., James, A., Salam, R. A., & Talib, A. Z. H. (2010). Enhancing the low quality images using unsupervised colour correction method. In *2010 ieee international conference on systems, man and cybernetics* (p. 1703-1709). doi: 10.1109/ICSMC.2010.5642311

Islam, M. J., Xia, Y., & Sattar, J. (2020). Fast underwater image enhancement for improved visual perception. *IEEE Robotics and Automation Letters*, *5*(2), 3227-3234. doi: 10.1109/LRA.2020.2974710

Marine plastic pollution. (2021). IUCN. Retrieved from https://www.iucn.org/resources/issues-brief/marine-plastic-pollution#:~:text=Over%20400%20million%20tons%20of,waters%20to%20deep%2Dsea%20sediments. (Accessed: 2023-12-09)

Peng, Y. T., & Cosman, P. C. (2017). Underwater image restoration based on image blurriness and light absorption. *IEEE transactions on image processing : a publication of the IEEE Signal Processing Society*, *26*(4).

Quantifying the ocean carbon sink. (2022). NOAA. Retrieved from https://www.ncei.noaa.gov/news/quantifying-ocean-carbon-sink#:~:text=The%20ocean%20acts%20as%20a,2%20levels%20in%20the%20ocean.

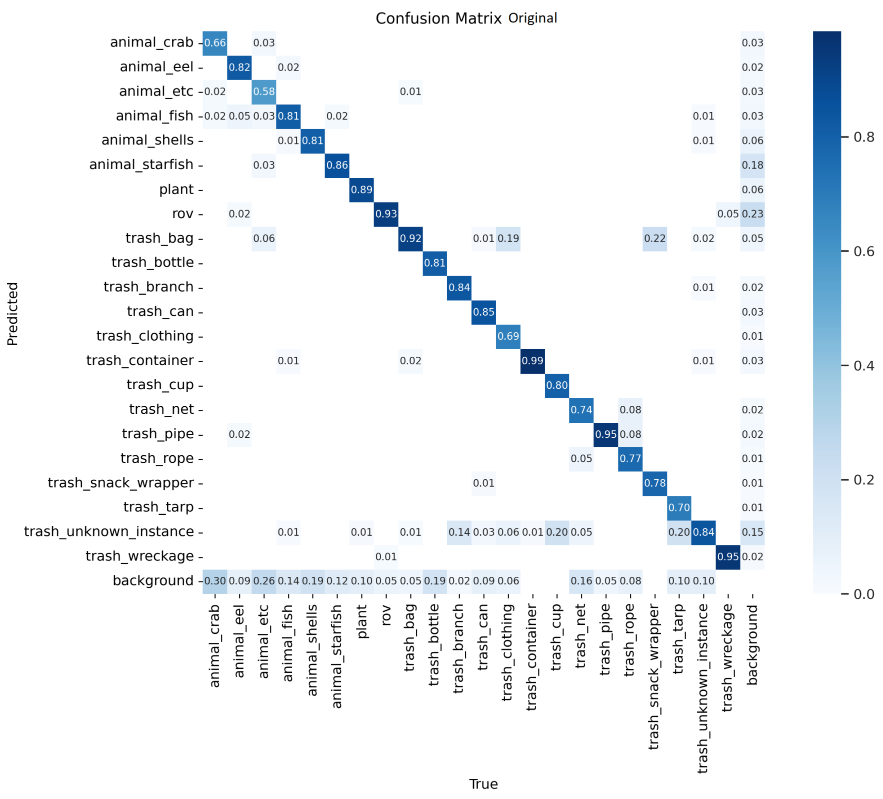
Ultralytics. (2023). *Yolov8.* https://github.com/ultralytics/ultralytics. (Accessed: 2023-12-09)

Uras, A., Wolf, B. J., Ilioudi, A., Palunko, I., & De Schutter, B. (Unpublished). Seaclear marine debris dataset.

wangyanckxx. (2021). *Single-underwater-image-enhancement-and-color-restoration.* https://github.com/wangyanckxx/Single-Underwater-Image-Enhancement-and-Color-Restoration. (Accessed: 2023-12-09)

xahidbuffon. (2020). *Funie-gan.* https://github.com/xahidbuffon/FUnIE-GAN. (Accessed: 2023-12-09)

Zuiderveld, K. (1994). Contrast limited adaptive histogram equalization. *Graphics gems.*



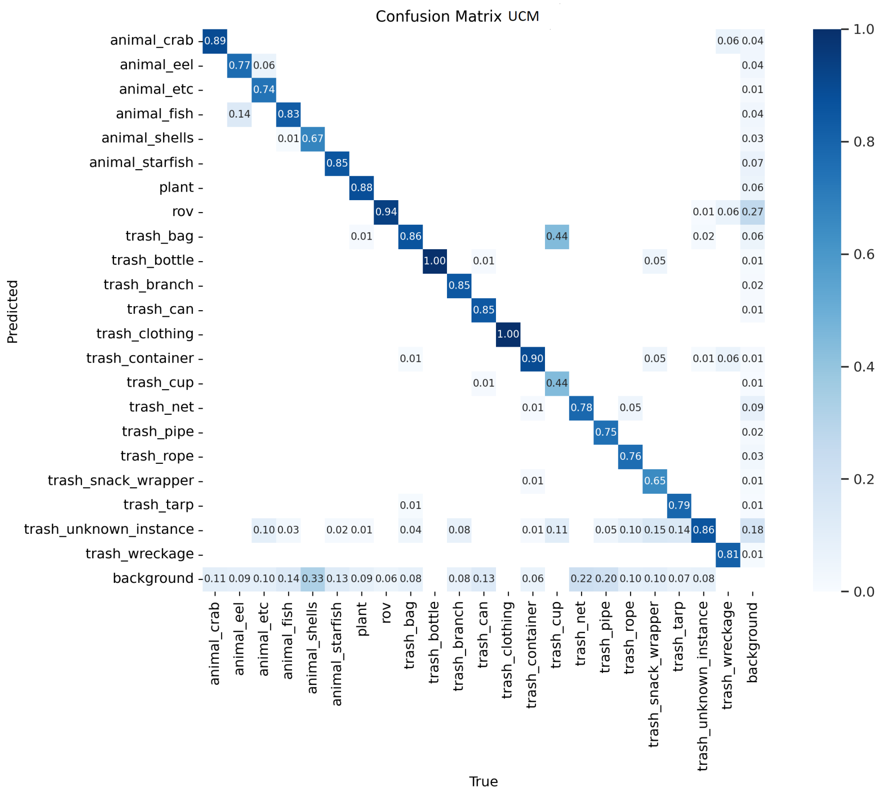


Figure 3.3: The confusion matrices for the original model and the UCM model