The effects of colour enhancement and IoU optimisation on object detection and segmentation of coral reef structures

Marina Arendt[1,2 [0000−0002−4573−2462]], Johannes Rückert[1 [0000−0002−5038−5899]], Raphael Brüngel[1 [0000−0002−6046−4048]], Christopher Brumann[1 [0000−0002−4117−2541]], and Christoph M. Friedrich[1,3 [0000−0001−7906−0038]]

[1] Department of Computer Science, University of Applied Sciences and Arts Dortmund, Emil-Figge-Str. 42, 44227 Dortmund, Germany
marina.arendt@fh-dortmund.de
[2] Kairos GmbH, Bochum, Germany
[3] Institute for Medical Informatics, Biometry and Epidemiology (IMIBE), University Hospital Essen, Essen, Germany

Abstract. This paper considers approaches used to localise and annotate coral reef structures in underwater images. Beside the actual localisation and annotation the focus laid on image pre-processing and their evaluation. Underwater images differ from terrestrial images in illumination, acuity and colour which make them more blurred with a green and blue cast. To enhance those physical properties, Image Blurriness and Light Absorption (IBLA) with additional Rayleigh optimisation or additional colour reduction were used. Afterwards, for both competition tasks Mask R-CNN was used, involving on-the-fly data augmentation and oversampling to combat the coral class imbalances. Several types of post-processing were applied to the generated boxes and polygons, mostly to account for the evaluation methodologies. IBLA and Rayleigh pre-processing improved accuracy for the localisation and annotation task, while colour reduction led to overall worse results than the original images and also oversampling led to even worse mean Average Precision (mAP) and only a slightly better average accuracy. For pixel-wise parsing IBLA achieved better mAP score but worse accuracy and Rayleigh achieved worse results for mAP and accuracy. Colour reduction worked well and oversampling reduced mAP but strongly improved average accuracy. Concluding, image pre-processing – in particular IBLA and Rayleigh – has improved accuracy for both tasks and only achieved better mAP on the pixel-wise parsing task. In future work, the results could be improved by using larger images, trying other types of oversampling and train separate models for different classes and object size.

Keywords: underwater colour correction · box optimisation · Mask R-CNN · deep learning · Jaccard index

1 Introduction

This paper considers the results of the 2020 ImageCLEFcoral challenge which took place for the second time [3]. Main focus lies on automatic image detection and classification of coral reef structures. ImageCLEFcoral is a subtask of ImageCLEF [7]. Underwater images differ from terrestrial images in terms of colour, illumination and acuity which can cause problems in automatic detection. Nevertheless, automatic detection and classification is needed as manual detection is cost and time intensive [15]. To regard the issues with underwater images the approaches focused on image pre-processing to enhance image structure, illumination and colour to test the effect of these steps on detection and segmentation.

After actual training of the models, several types of post-processing were applied to the generated boxes and polygons, mostly to account for the evaluation methodologies. For this, generated polygons were validated and boxes were shrunk to achieve a better Jaccard index, also known as Intersection over Union (IoU).

The paper is separated in the sections of data set description and preprocessing showing the main issues with underwater images followed by annotation and localisation (object detection task 1) and pixel-wise parsing (segmentation task 2). The results and discussion build the main part and is completed with the conclusion. All scripts used during this project are available in a GitHub repository[4].

2 Image and annotation data

All details about the task and data can be found in the task overview paper [3]. The provided version 4 of the data set consists of $n = 440$ images in the development part (training set) and $m = 400$ images in the test part. All images are provided in the Joint Photographic Experts Group (JPEG) format with a resolution of 4032×3024 px. An annotation file for the development part holds $k = 12,082$ annotations, structured in eight variables:

`image_id`: Data set-wide unique image identifier, equal to the image filename.
`substrate`: Image identifier-wide unique substrate index, starting by 0.
`c_class`: Name of one of the 13 present coral classes.
`confidence`: Confidence of annotation, always 1 for 100% confidence.
`x_min`: Minimum x-axis bounding box coordinate[5].
`y_max`: Maximum y-axis bounding box coordinate[5].
`x_max`: Maximum x-axis bounding box coordinate[5].
`y_min`: Minimum y-axis bounding box coordinate[5].

In the following, the results of explorative analyses on annotation data and applied image pre-processing methods are presented.

[4] https://github.com/saviola777/fhdo-imageclef2020-coral/, accessed 2020-07-07
[5] On the image level the coordinate system origin ($x\_min = 0, y\_min = 0$) is located in the upper left corner.

2.1 Explorative data analyses on annotations

Explorative analyses on annotation data of the training set were conducted using the statistical language R[6] [12] in version `4.0.1` and the integrated development environment RStudio[7] [14] in version `1.2.5033`. Spatial analysis of substrate bounding boxes was performed using the Simple Features for R (`sf`) package[8] [9] in version `0.9-2`.

During a first screening minor inconsistencies were identified. These comprise (i) a single negative coordinate value of `x_min` $= -1$ present in one row (substrate 10 in `2018_0714_112502_024`), and (ii) five dot-sized bounding boxes with `x_min` $=$ `x_max` and `y_min` $=$ `y_max` (substrate 17 in `2018_0714_112534_047`; substrate 14 in `2018_0714_112535_042`; substrate 1 in `2018_0714_112535_050`; substrate 21 in `2018_0729_112613_064`; substrate 5 in `2018_0729_112458_039`). For the negative coordinate the respective substrate was checked manually on the respective image. A sign flip was performed and its bounding box was kept. Dot-sized bounding boxes were removed. Cleansing of annotation data resulted in a total of $k = 12,077$ entries, still related to $n = 440$ images. Presented results are based on the cleansed annotations.

Table 1: Class frequencies, images presence and figures of per-image occurrences in the training set[a]. Top-5 occurrences are highlighted.

Class	Frequency	Percentages	Images	Maximum
Algae Macro or Leaves	91	0.75 %	53	11
Fire Coral Millepora	19	0.16 %	9	6
Hard Coral Boulder	**1640**	**13.58** %	398	18
Hard Coral Branching	**1181**	**9.78** %	349	16
Hard Coral Encrusting	**945**	**7.82** %	310	21
Hard Coral Foliose	177	1.47 %	104	11
Hard Coral Mushroom	223	1.85 %	140	6
Hard Coral Submassive	198	1.64 %	93	12
Hard Coral Table	21	0.17 %	21	1
Soft Coral	**5662**	**46.88** %	425	66
Soft Coral Gorgonian	90	0.74 %	66	5
Sponge	**1691**	**14.00** %	342	34
Sponge Barrel	139	1.15 %	118	3

[a)] The Frequency column describes the number in the overall data set and the Images column in how many images this coral type is pictured. Percentages stands for the overall frequency distribution and Maximum is the maximum number of a coral class representatives in one image.

[6] https://www.r-project.org/, accessed 2020-07-09
[7] https://rstudio.com/, accessed 2020-07-09
[8] https://github.com/r-spatial/sf, accessed 2020-07-09

Annotations comprise 13 substrate classes, listed in Table 1 together with their frequencies, overall percentages, presence in images and maximum count they occurred in an image. The top two classes in regards to their frequency account for 60.1% of all annotations while the top five classes account for 92.1%. Soft Coral is the most common class and represents 47.2% of all annotations.

Statistics on substrate bounding box per-image frequency, aspect ratio (x:y) and area (px) are listed in Table 2. The substrate density in images features a high variety. While `2018_0712_073801_116` is the only image with a single substrate, `2018_0712_073920_154` shows the maximum of 96 substrates in a single image. The median number of substrates in an image is 24, rather few images contain a vast amount. The aspect ratio of substrate bounding boxes also shows a wide span with a minimum of 0.12 and a maximum of 8.51. Highly elongated bounding boxes are present, however, a median of 1.08 and an interquartile range of 0.30 suggest a moderate elongation in most cases. Also the areas of substrate bounding boxes show a wide span and with partially extreme low and high values. For a better understanding square area values are discussed, assuming substrate bounding boxes with an aspect ratio of 1:1. Here, the minimum area is 12.73 px^2 while also a maximum of $3,249.23$ px^2 is present. The median square area is 241.94 px^2.

Table 2: Statistics on substrate bounding box features of the training set.

	Min.	1st Qu.	Median	Mean	3rd Qu.	Max.	Std. deviation
Per-image frequency	1	16	24	27.45	36	96	16.13
Aspect ratio (x:y)	0.12	0.87	1.08	1.17	1.35	8.51	0.50
Square area (px^2)	12.73	158.39	241.94	412.26	388.81	3,249.23	261.33

The spatial analysis of substrate bounding boxes revealed a notable amount of overlaps where up to five boxes shared an intersecting area. The most common overlap scenario involve two and three substrate bounding boxes. For two substrate bounding boxes the mean (median) is 14.90 (12) overlaps per image, for three substrate bounding boxes it is 2.46 (3). A maximum of up to 81 overlaps between two and up to 32 between three substrate bounding boxes indicates numerous redundancies of annotated areas for several images. Overlaps between four and five substrate bounding boxes are rare. Also full overlaps between substrate bounding boxes have been found where a bounding box of a substrate fully covers that of another. For intra-class overlaps 146 full overlaps have been identified, while for inter-class overlaps 509 were found. Especially inter-class overlaps and full overlaps may display a challenging condition for object detection tasks.

2.2 Pre-processing

Underwater images differ from other images in their physical properties. The deeper the images where taken the darker the images get as well as red light is more absorbed than green and blue light. This often results in blurred images with a green and blue cast [10].

The idea was to enhance image quality prior to segmentation and parsing which should lead to enhanced segmentation and parsing results. The best pre-processing steps were chosen by visual inspection. Pre-processing functions [16] that have been used are described in the following sections. The images were processed first by Image Blurriness and Light Absorption (IBLA) followed by either a transformation of Rayleigh distribution or an octree colour reduction. Figure 1 (a) and (b) are examples showing the problems with underwater images described above.

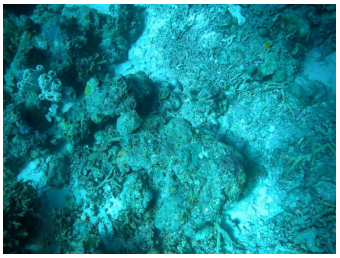
To visualise the colour distribution, normalised histograms were made using the NumPy[9] package. Figure 2 shows the histograms of images `2018_0714_112438_016` and `2018_0729_112414_024`. Clearly seen is the either high intensity of the green channel or the high intensity of the blue channel in combination with very low intensity of the red channel typical for underwater images.

Image Blurriness and Light Absorption (IBLA) Underwater image restoration based on IBLA was conducted on both the training and test set [10,16]. IBLA transformation is based on four main steps. First, the image blurriness is analysed and afterwards a smoothed and refined blurriness map is generated to optimise the image. Second, the background light pixels are estimated by image blurriness and variance via a quad-tree algorithm. Third, the actual enhancement using depth estimation based on light absorption and blurriness which results in an optimised depth map. Last, the transmission map is estimated leading to restoration rather than estimation [10]. The results are shown in Figure 1c and Figure 1d.

Enhancement based on Rayleigh distribution The method for image enhancement with Rayleigh distribution is separated into two main steps [5]. First, the contrast is corrected and second, the colour is corrected. For contrast correction a global histogram stretching is implemented followed by division into a lower and an upper side by the average point. Both parts are then Rayleigh-stretched to the full gray-scale from 0 to 255 and recombined. For colour enhancement the image is transformed into the Hue, Saturation, and Value (HSV) colour model. The saturation and value levels are stretched and reconverted to an Red, Green, Blue colour space (RGB) image. This led to an enhancement in contrast and details and reduced image artefacts [5,16]. Results are in Figure 1e and Figure 1f.

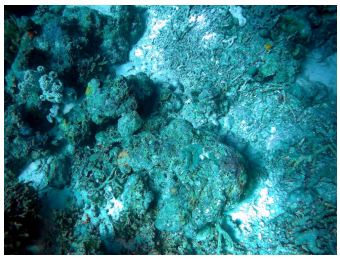
[9] https://numpy.org/, accessed 2020-07-10

(a) Original 2018_0714_112438_016



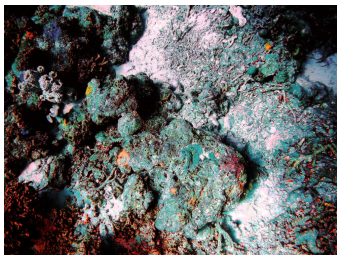
(b) Original 2018_0729_112414_024

(c) Image (a) IBLA transformed

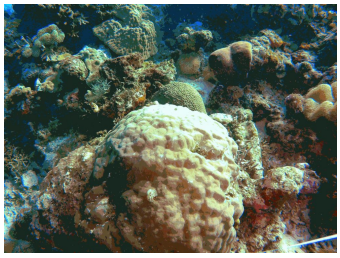


(d) Image (b) IBLA transformed

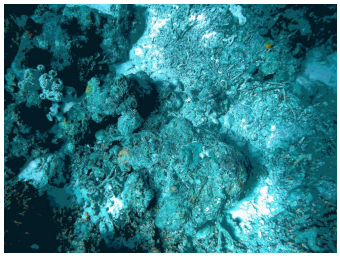
(e) Image (c) Rayleigh transformed



(f) Image (d) Rayleigh transformed



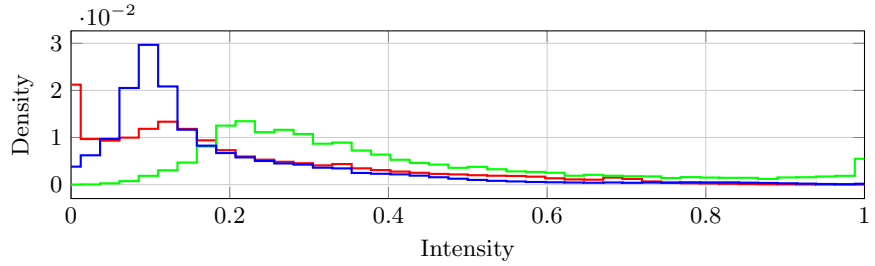
(g) Image (c) colour reduced



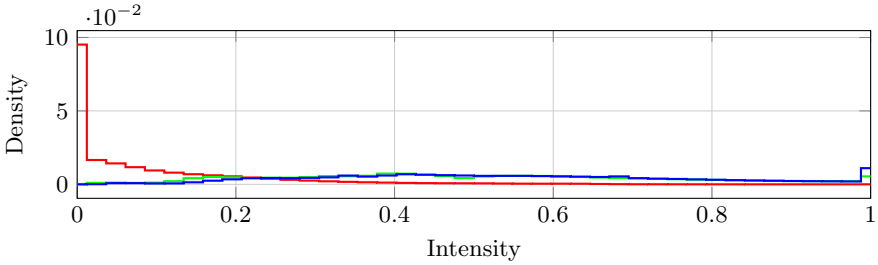
(h) Image (d) colour reduced

Fig. 1: Comparison of the original images `2018_0714_112438_016` (a) and `2018_0729_112414_024` (b) from the training set with their three different transformation (c) - (h). (a) and (b) contain the main problems with underwater images. They are blurry and show a large translation of histograms towards green or blue. The original images (a) and (b) are under copyright of the organisers [3,7]

.



(a) Histogram of `2018_0714_112438_016`



(b) `Histogram of 2018_0729_112414_024`

Fig. 2: Histograms of two training set images showing the typical unequally distributed colours of underwater images.

Colour reduction The colour reduction was conducted using the octree process reducing all IBLA transformed images to maximum 256 colours as implemented in the following code [10]. The octree colour reduction (for instance described in [2, p. 333 sqq.]) results in an image with 256 colours with a harmonised colour distribution [2]. The results are shown in Figure 1g and Figure 1h.

3 Methods

Mask R-CNN [6] is an instance segmentation framework which extends Faster R-CNN [13] with a parallel branch for instance segmentation on region of interests.

The models described in this paper were trained on a Mask R-CNN implementation using TensorFlow and Keras in Python 3[11] which was patched to support TensorFlow 2.1[12]. All models used weights pre-trained on the MS COCO data set [8].

To speed up on-the-fly pre-processing and avoid padding, all images were resized to 1536×1536 beforehand.

[10] https://github.com/delimitry/octree_color_quantizer, accessed 2020-07-07

[11] Abdulla, W.: Mask R-CNN for object detection and instance segmentation on Keras and TensorFlow. https://github.com/matterport/Mask_RCNN, accessed 2020-07-07

[12] https://github.com/DiffPro-ML/Mask_RCNN, accessed 2020-07-07

The training was split into two phases: first, only the newly added layers were trained for one epoch. Second, the complete network was trained for the remaining epochs. Then, Polyak averaging [11] was performed on the top five models based on their .632 error [4]. For the submission models, early stopping was used based on the average epoch number for the top five models during cross-validation training.

For on-the-fly data augmentation, the images were randomly rotated (up to $\pm 180°$ in each direction), flipped (up/down or left/right) with 33% chance each, and 0 to 5 of blur, sharpen, random crop (up to 20% on each side), Gaussian noise, brightness, hue/saturation, and contrast were changed.

To combat the class imbalance in the data set, oversampling was performed which entails iterative optimisation of the Shannon entropy of the data set by adding images until the number of images is tripled, with constraints on the number of times a single image can appear in the final data set.

Considering only the five most frequent classes was a consideration due to the imbalance of the data set and achieved good results in our cross-validation runs.

Table 3 lists the most important parameters used for the different training runs. The training run parameters are listed in Table 4. It includes the submission ID, the run name, which data set was used (original images, IBLA pre-processing, IBLA plus Rayleigh pre-processing, colour reduced, see Section 2.2), whether on-the-fly data augmentation as described above was applied, whether images of size 1024×1024 or 1536×1536 were used, whether oversampling was used as well as the number of epochs.

Table 3: Mask R-CNN parameters used for training variations. Image size, batch size and anchor scales depended on the image size. Learning rate η was reduced for the second phase and was different for oversampling runs.

Parameter	Value
Backbone	Resnet101
Image size (larger images)	1024×1024 (1536×1536)
Batch size (larger images)	2 (1)
Optimiser	Stochastic gradient descent with Nesterov momentum enabled
η first phase (oversampling)	0.005 (0.002)
η second phase (oversampling)	0.0005 (0.0005)
Learning momentum	0.9
Weight decay	0.0001
Epochs (augmentation, oversampling)	15 (30, 20)
Minimum detection confidence	0.6
Anchor scales	$32, 64, 128, 256, 512$
Anchor scales larger images	$48, 96, 192, 384, 768$

Table 4: Run configurations with submission ID, Run name, pre-processing, Augmentation, Larger images, Oversampling, and number of epochs.

ID	Run name	Data set	Augm.	Larger images	Overs.	Epochs
67914	Baseline	Original	No	No	No	15
67919	5 classes	Original	No	No	No	20
68188	Augmentation	Original	Yes	No	No	30
68187	IBLA	IBLA	Yes	No	No	30
68186	IBLA + Rayleigh	IBLA + Rayleigh	Yes	No	No	30
68184	Colour reduction (CR)	IBLA + CR	Yes	No	No	30
68185	Oversampling	IBLA	Yes	No	Yes	20
68183	Larger images	IBLA	Yes	Yes	Yes	20
68182	Segmentation	IBLA (polygons)	Yes	Yes	Yes	20
68181	Segmentation + IoU optimisation	IBLA (polygons)	Yes	Yes	Yes	20

3.1 Annotation and localisation

The focus of the approach described in this paper was on the annotation and localisation task, and models were trained using the bounding box annotations and optimised against the PASCAL Visual Object Classes (VOC)-style mean Average Precision (mAP) implementation in Mask R-CNN [6].

The different configurations that were analysed (as seen in Table 4) included different data sets based on the pre-processing described in section 2.2, different levels of on-the-fly data augmentation, using two different image sizes, using oversampling, as well as trying to train models for varying numbers of epochs.

3.2 Pixel-wise parsing

For this task, similar approaches as for the annotation and localisation task were applied only using the polygon annotations for training. The colour reduction run was skipped due to its poor results for the first task, instead a number of different runs with larger images were included to see the results of more and less training as well as a lower confidence threshold.

3.3 Post-processing

When the evaluation code for the challenge was published and it turned out that the submitted bounding boxes would be evaluated against the polygons instead of the bounding box annotations, it became clear that training with the polygon annotations would be much more effective. For example, this led to an evaluation F1 score of 0.8 for the bounding box training ground truth, whereas the score was 0.99 for the polygon training ground truth. The loss of 0.01 was due to invalid polygons in the ground truth annotations.

This value can be increased from 0.8 to 0.9 simply by reducing the size of the bounding boxes by 7.5% on each side as seen in Figure 3. This post-processing step was therefore used for all submissions of the first task.

Additionally, an iterative algorithm was created to approximate the best possible rectangular box for a given polygon according to their IoU. This algorithm is described in the next section. To make use of this algorithm, a model was trained on the polygon annotations and the resulting polygons were used to generate boxes.

For the second task, the polygons generated from binary masks by OpenCV[13] [1] were validated against the shapely library[14] which was used in the evaluation script since about 1% of generated polygons were not valid according to the shapely library's definition of a valid polygon and would be ignored by the evaluation script. A valid polygon may not cross itself and may only touch in a single point.

To clean up invalid polygons, first duplicate points were removed, then the polygons were split in several separated polygons using the touching / self-crossing points and the biggest polygon was used. Separately, the `buffer` function provided by the shapely library was used to generate a valid polygon. Then the polygon with the overall least absolute area difference compared to the original, invalid polygon was used.

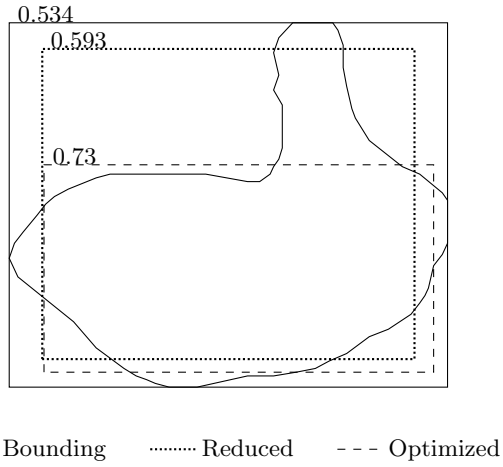


Fig. 3: IoU values for a detected polygon. The minimum bounding box shows an IoU of 0.534. After reducing the size by 7.5% in all dimensions the result was increased to 0.593 as the dotted box shows. Applying the iterative optimisation algorithm led to the highest IoU of 0.73.

[13] https://opencv.org/, accessed 2020-07-10
[14] https://github.com/Toblerity/Shapely, accessed 2020-07-09

Bounding box IoU optimisation Given a solid polygon P, which is defined by a contour as a set of points. The corresponding minimal enclosing rectangle can be defined by four parameters: $R^0 := [x, y, w, h]$. Here x and y define the top rectangles left corner and w and h the width and height, respectively. Calculating the IoU of the polygon with the rectangle showed that this value is not necessarily the best to be achieved as can be seen in Figure 3. This became particularly clear with thin long polygon arms that run parallel to the rectangle edges. In these cases it often led to a higher IoU if the rectangle was slightly reduced for the corresponding side. When considered the minimum bounding box, however, an increase in edge length can only led to an IoU deterioration. Therefore the polygons minimum bounding box was chosen as the algorithm's starting point for maximising the IoU. The parameter space was thus defined by the rectangle's parameters.

The optimised objective function was $\mathrm{IoU}(P, R) = \frac{|P \cap R|}{|P \cup R|}$ for the given polygon P and a current rectangle R^k at optimisation step k. To maximise the target function, R was iterative changed in all parameters via translation and scaling, so that the most optimised rectangle was used for the next iteration. This process was continued until no further objective function improvement was achieved. Consequently, the rectangle was accepted as optimised. During the optimisation process, step sizes for translation, shrinkage and growth are given by t, s and g. In this application all values were set to four. Each optimisation iteration was then performed by calculating:

$$\begin{aligned} R^{k+1}_{T\leftarrow} &= R^k + [-t, 0, 0, 0] & R^{k+1}_{T\rightarrow} &= R^k + [t, 0, 0, 0] \\ R^{k+1}_{T\uparrow} &= R^k + [0, -t, 0, 0] & R^{k+1}_{T\downarrow} &= R^k + [0, t, 0, 0] \end{aligned} \tag{1}$$

$$\begin{aligned} R^{k+1}_{S\leftarrow} &= R^k + [s, 0, -s, 0] & R^{k+1}_{S\rightarrow} &= R^k + [0, 0, -s, 0] \\ R^{k+1}_{S\uparrow} &= R^k + [0, s, 0, -s] & R^{k+1}_{S\downarrow} &= R^k + [0, 0, 0, -s] \end{aligned} \tag{2}$$

$$\begin{aligned} R^{k+1}_{G\leftarrow} &= R^k + [-g, 0, g, 0] & R^{k+1}_{G\rightarrow} &= R^k + [0, 0, g, 0] \\ R^{k+1}_{G\uparrow} &= R^k + [0, -g, 0, g] & R^{k+1}_{G\downarrow} &= R^k + [0, 0, 0, g] \end{aligned} \tag{3}$$

As can be seen in Equation 1, the rectangle movement was performed in every possible direction. It should be noted that the rectangle never excesses the bounding box dimensions. Equation 2 shows the corresponding contraction of the rectangle, while Equation 3 calculates its enlargement. Once all possible rectangles R^{k+1} were calculated, the corresponding objective function with P was evaluated, so that the corresponding improvement factor was known. Subsequently the operation with the most promising improvement factor was performed by setting the corresponding rectangle as the new R^k and thus setting the starting point for the next iteration. As soon as no rectangle shows an improvement, R^k was no longer updated and was assumed to be optimised with regard to the objective function. An example for an optimised rectangle is shown in Figure 3.

4 Results and Discussion

For the image annotation and localisation task (see Table 5), on-the-fly data augmentation expectedly reduced over-fitting and led to better results for both mAP and average accuracy. Pre-processing using IBLA and Rayleigh led to lower mAP values but increased average accuracy, while colour reduction produced overall worse results compared to the original images. The Rayleigh run had the highest average accuracy of all runs submitted for the first task, followed by the IBLA and augmentation runs, showing that while the models do not detect objects as accurately as some of the competitors, they classify the detected objects very well.

Table 5: 5-fold cross-validation results on the training data set along with the corresponding submission results for the image annotation and localisation task.

						Submission	
ID	Run name	F1 score train (sd)		F1 score valid (sd)		MAP_0.5	Avg. accuracy
67914	Baseline	**.688**	(.012)	.425	(.010)	.391	.113
67919	5 classes	.573	(.011)	**.569**	(.020)	.383	.083
68188	Augmentation	.492	(.005)	.433	(.011)	.440	.149
68187	IBLA	.482	(.005)	.426	(.012)	.424	.155
68186	IBLA + Rayleigh	.474	(.012)	.420	(.008)	.410	**.159**
68184	Colour reduction	.447	(.008)	.382	(.005)	.388	.137
68185	Oversampling	.511	(.010)	.432	(.011)	.369	.117
68183	Larger images	.505	(.012)	.446	(.011)	.405	.107
68182	Segmentation	.564	(.007)	.470	(.010)	.422	.108
68181	Segmentation + IoU optimisation	.579	(.007)	.483	(.011)	**.457**	.107

Considering only the 5 most frequent classes achieved worse results than the baseline in both mAP and accuracy, unlike in the cross-validation runs, where it achieved the best F1 score on the validation data.

Surprisingly, oversampling led to even worse mAP results than the baseline model, while having only slightly better average accuracy. In our cross-validation runs on the other hand, oversampling achieved results on par with the data augmentation run. This may be due to a slightly different class distribution in the test data set, or too many epochs of training for the submission models.

Using larger images led to better mAP scores but worse average accuracy. Using polygon annotations for training clearly improved the mAP, but did not improve the average accuracy.

Looking at the per-substrate accuracies, the IoU optimised run produced the worst results with seven classes having 0% accuracy. The run without IoU optimisation and the run with larger images still have six classes without any correct detections, meaning that all runs with larger images have trouble with the less

frequent classes. The baseline, for comparison, produced no correct detections for five classes. The oversampling run only has three classes without any correct detections, but three more classes with accuracies below 0.05.

The runs without oversampling performed better, each of them produced correct detections for all but one class which was the class that had 0% accuracy across all submitted runs of all participants and was only represented by less than 30 instances in the training data set.

The evaluation code was released only several weeks before the submission deadline for the models, and it produced very different results than the PASCAL VOC-style mAP evaluation recommended in the task's description and implemented in Mask R-CNN.

The evaluation strategy was mainly the same for both tasks. The focus of this work would have been much more on the second task and on training instance segmentation models if the evaluation code had been published earlier. It is easier and more effective to generate suitable boxes from polygons compared to guessing suitable boxes based on the bounding box or just using the bounding box itself.

In effect, a model which predicts perfect bounding boxes would never be able to exceed an mAP score of about 0.8 based on objects that are shaped in a way that produce bounding boxes with very low IoU. This effect was demonstrated by the bounding box refinement and IoU optimisation, which led to a significant improvement in mAP score.

For the image pixel-wise parsing task (see Table 6), IBLA achieved a better mAP score than the original data set, but a worse average accuracy than the baseline. Rayleigh performed even worse with a similar mAP score than the baseline but worse average accuracy.

Table 6: Submission results for the image pixel-wise parsing task.

ID	Run name	MAP_0.5	Avg. accuracy
67963	Baseline	.433	.134
67964	Augmentation	.449	.140
67965	IBLA	.453	.128
67967	IBLA + Rayleigh	.435	.120
68192	Oversampling	.424	.191
67968	Larger images	.469	.174
68190	Larger images, colour reduction	**.474**	.158
68191	Larger images, 10 epochs	.416	.128
67969	Larger images, 0.4 confidence	.376	**.196**
68189	Larger images, 60 epochs, more augmentation	.371	.157

Oversampling once again reduced the mAP score, but strongly improved the average accuracy. Similar to the first task, using larger images increased the mAP score but reduced the average accuracy.

Using larger images and the data set with reduced colours further improved the mAP score, reaching the overall best value for the second task among the submitted runs but produced a much lower average accuracy.

Training for more or less epochs led to overall worse results. Reducing the confidence threshold led to a very low mAP score but slightly improved the average accuracy, led once again to the highest average accuracy out of all runs submitted for the second task.

Looking at the overall and per-substrate accuracies the models with larger images interestingly had better accuracies than the one with smaller images, unlike in the first task, where it was the other way around. Excluding the run with 0.4 minimum detection threshold which achieved the highest overall accuracy with a much lower mAP value the oversampling run had the best accuracy.

The runs without oversampling achieved comparatively worse results with four to five classes having no correct detections whereas all other runs had only the one class without correct detections.

Post-processing as described in Section 2.1 did not achieved an impact on classification quality as expected. Hence, those results are not shown as they were not finally implemented.

5 Conclusion

Concluding, image pre-processing using IBLA and Rayleigh has improved accuracy for the localisation and annotation task while achieving better mAP on the pixel-wise parsing task. Colour reduction worked well with larger images for the second task in terms of mAP, but falls behind in accuracy.

Oversampling was overall not successful even though it led to better accuracy in the second task. This result was not reflected in cross-validation analysis on the training data. Therefore, oversampling was used in the majority of submitted runs decreasing the performance especially of the runs with larger images.

Larger images performed worse in early runs that were not properly fine-tuned and hence enlargement was not considered in most of the further analysis. All runs with larger images used oversampling which most likely hurt the model performance. A stronger focus on larger images would have been useful, since the results are promising at least for the second task.

Nevertheless, this work has produced competitive models especially in terms of the classification accuracy that could be improved in the future. Examples are using larger images, trying other types of oversampling, or training separate models for different classes or object sizes.

References

1. Bradski, G.: The OpenCV Library. Dr. Dobb's Journal of Software Tools **25**, 120–125 (2000)
2. Burger, W., Burge, M.J.: Digital Image Processing - An Algorithmic Introduction Using Java. Springer, Berlin, Heidelberg, 2nd edn. (2016). https://doi.org/10.1007/978-1-4471-6684-9

3. Chamberlain, J., Campello, A., Wright, J.P., Clift, L.G., Clark, A., García Seco de Herrera, A.: Overview of the ImageCLEFcoral 2020 Task: Automated Coral Reef Image Annotation. In: CLEF2020 Working Notes. CEUR Workshop Proceedings, CEUR-WS.org (2020)
4. Efron, B., Tibshirani, R.: Improvements on cross-validation: the 632+ bootstrap method. Journal of the American Statistical Association **92**(438), 548–560 (1997)
5. Ghani, A.S.A., Isa, N.A.M.: Underwater image quality enhancement through composition of dual-intensity images and Rayleigh-stretching. SpringerPlus **3**(1) (2014). https://doi.org/10.1186/2193-1801-3-757
6. He, K., Gkioxari, G., Dollar, P., Girshick, R.: Mask R-CNN. In: 2017 IEEE International Conference on Computer Vision (ICCV). IEEE (2017). https://doi.org/10.1109/iccv.2017.322
7. Ionescu, B., Müller, H., Péteri, R., Abacha, A.B., Datla, V., Hasan, S.A., Demner-Fushman, D., Kozlovski, S., Liauchuk, V., Cid, Y.D., Kovalev, V., Pelka, O., Friedrich, C.M., de Herrera, A.G.S., Ninh, V.T., Le, T.K., Zhou, L., Piras, L., Riegler, M., l Halvorsen, P., Tran, M.T., Lux, M., Gurrin, C., Dang-Nguyen, D.T., Chamberlain, J., Clark, A., Campello, A., Fichou, D., Berari, R., Brie, P., Dogariu, M., Ştefan, L.D., Constantin, M.G.: Overview of the ImageCLEF 2020: Multimedia Retrieval in Lifelogging, Medical, Nature, and Internet Applications. In: Experimental IR Meets Multilinguality, Multimodality, and Interaction. Proceedings of the 11th International Conference of the CLEF Association (CLEF 2020), vol. 12260. LNCS Lecture Notes in Computer Science, Springer (2020)
8. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.: Microsoft COCO: Common Objects in Context. In: Computer Vision – ECCV 2014. pp. 740–755. Springer International Publishing (2014). https://doi.org/10.1007/978-3-319-10602-1_48
9. Pebesma, E.: Simple Features for R: Standardized Support for Spatial Vector Data. The R Journal **10**(1), 439–446 (2018). https://doi.org/10.32614/RJ-2018-009
10. Peng, Y.T., Cosman, P.C.: Underwater Image Restoration Based on Image Blurriness and Light Absorption. IEEE Transactions on Image Processing **26**(4), 1579–1594 (2017). https://doi.org/10.1109/tip.2017.2663846
11. Polyak, B.T., Juditsky, A.B.: Acceleration of stochastic approximation by averaging. SIAM journal on control and optimization **30**(4), 838–855 (1992)
12. R Core Team: R: A Language and Environment for Statistical Computing. R Foundation for Statistical Computing, Vienna, Austria (2020), https://www.R-project.org/
13. Ren, S., He, K., Girshick, R., Sun, J.: Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. In: Advances in neural information processing systems. pp. 91–99 (2015)
14. RStudio Team: RStudio: Integrated Development Environment for R. RStudio, Inc., Boston, MA (2019), https://www.rstudio.com/
15. Srividhya, K., Ramya, M.M.: Object classification in underwater images using adaptive fuzzy neural network. In: 2017 13th International Conference on Natural Computation, Fuzzy Systems and Knowledge Discovery (ICNC-FSKD). pp. 142–148. IEEE (2017). https://doi.org/10.1109/fskd.2017.8392973
16. Wang, Y., Song, W., Fortino, G., Qi, L.Z., Zhang, W., Liotta, A.: An Experimental-Based Review of Image Enhancement and Image Restoration Methods for Underwater Imaging. IEEE Access **7**, 140233–140251 (2019). https://doi.org/10.1109/access.2019.2932130