Boat detection

Alessandro Canevaro

July 23, 2021

1 Introduction

1.1 Activity goal

The aim is to develop a computer vision algorithm that can detect boats in an image. Specifically, it should draw a rectangular bounding box around each founded boat, and it should work correctly even when the image has no boats in it. Note that boats can appear from any perspective, and they can be very different: from small dinghies to cruise ships. The intersection over union (IoU) metric is used on the provided test images to measure the algorithm's performance.

1.2 Adopted approach

Due to the great variety of boats and the availability of a large data set of images, a machine learning approach seems promising. Hence the use of a neural network in the proposed solution. In particular, the developed algorithm initially extracts features (key points and descriptors) from the image, then classifies the descriptors as boat/non-boat, and lastly, builds the bounding boxes. The last step is, in turn, composed of several sub-steps. The algorithm first generates a probability density function (PDF) of the potential locations of the boats. The PDF modes are then clustered using the mean shift algorithm, and finally, the bounding boxes are generated with a simple iterative procedure. The complete algorithm is here summarized:

1. Extract features key points and descriptors from the to-be-detected image.
2. Classify each descriptor as boat/non-boat using the trained classifier.
3. Generate the probability map of the possible positions of the boats.
4. Find and cluster the modes of the above probability density using the mean shift algorithm.
5. Prune and suppress "weak" clusters.
6. For each cluster, generate the bounding box.

2 Algorithm description

2.1 Features

Features descriptors are the only inputs required by the classifier. Therefore the choice of the features extractor algorithm is essential. Numerous algorithms exist, and they have different benefits and drawbacks (for instance, speed, number of extracted features, quality of the descriptors); however, as reported by these tests [TS18], SIFT features are the best overall. Hence are the ones used in this project.

2.2 Training the neural network

The goal is to train a feature classifier that distinguishes between features coming from boats and non-boats. The classifier's input will be the 128-dimensional feature descriptors vector, and the output will be a number encoding the boat/non-boat classes (i.e. 1 for boats and -1 for non-boats). A neural network was chosen for this task, particularly one with a fully connected architecture (multi-layer perceptron). After several trials, the best results were given by reducing the dimension of the input layer from 128 to 64 with PCA and using only one hidden layer with 42 neurons (2/3 of the input size). Furthermore, the chosen activation function is the sigmoid reported in equation 1, and RPROP [RB93] is used as the optimization algorithm as it quickens the learning process compared to the traditional backpropagation algorithm.

$$\sigma(x) = \frac{1 - e^{-x}}{1 + e^{-x}} \tag{1}$$

Since the whole project is built in c++ using the OpenCV library, the network is also built and trained in c++ with the machine learning module of OpenCV, allowing a better interface between the different parts of the proposed algorithm.
After extracting all the features from the 3000-images data set (and the ground truth labels), over 4 million features are available for training. They are divided into train and test data set using a 90%-10% split ratio. The train data set is in turn divided into 250 batches of approximately 15 thousand samples each. The `NO_INPUT_SCALE` flag is used in the training process since the data are normalized with zero mean and unit variance before splitting into batches. Moreover, after the first batch, the `UPDATE_WEIGHTS` flag is added to avoid resetting the weights. The training process took approximately 40 minutes.
Despite the large training data set, the error is approximately 37%[1] both on the train and test set. So it seems that the network is underfitting. However, it is not due to a lack of training data, and also an increase in the complexity of the network by adding more hidden layers does not improve the results.
Nonetheless, as shown by Figure 1, the network has roughly learned to correctly classify the main elements of the image, such as the sea and the ship (red features indicates "non-boat" whereas green stands for "boat"), still with some not negligible outliers like in the background shore.

[1] Defined as $error = 1 - accuracy$.

Figure 1: Classifier predictions

2.3 Building the probability map

The next step consists of building a two-dimensional discrete probability density function that describes how likely each pixel belongs to a boat. Consider the pixel p with coordinates (x, y), and let $\mathcal{F}_{p,r}$ be the following subset of features:

$$\mathcal{F}_{p,r} = \{f \in \mathcal{F} \,|\, dist\,(f, p) < r\} \tag{2}$$

Now consider equation (3), which is a weighted average of the predictions where the weights give more importance to features close to the evaluated pixel while also discarding features whose distance is greater than a threshold (r) from (x, y).

$$g\,(p) = \frac{\sum_{i=1}^{|\mathcal{F}_{p,r}|} pred\,(\mathcal{F}_{p,r}\,(i))\, w_{p,r}\,(i)}{\sum_{i=1}^{|\mathcal{F}_{p,r}|} w_{p,r}\,(i)}; \quad w_{p,r}\,(i) = \max\,(0, r - dist\,(\mathcal{F}_{p,r}\,(i)\,, p)) \tag{3}$$

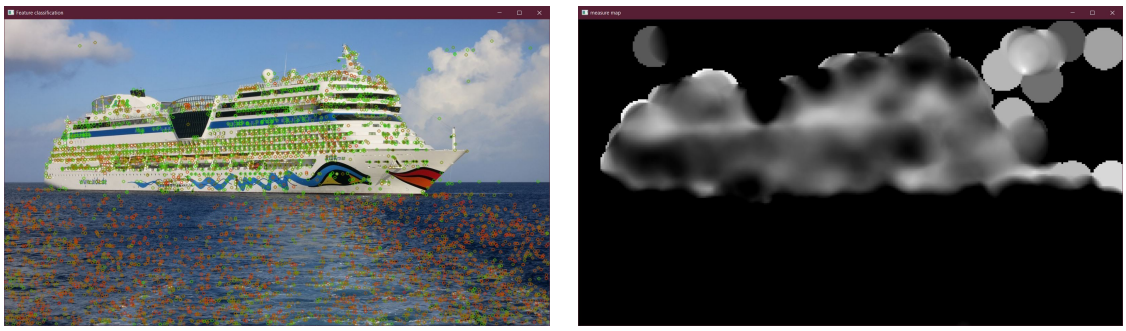
Formula (4) represents the sought probability density function[2].

$$h\,(p) = \frac{\max\,(0, g\,(p))}{\sum_{p \in image} \max\,(0, g\,(p))} \tag{4}$$

For example, consider Figure 2, where on the left the predictions are shown using green/red colour for boat/non-boat classes. While on the right side, the corresponding PDF (according to equation 4) is reported.

Note that high probability values are being assigned to the few features in the top right corner. Indeed they are being predicted as "boats". However, given the fact that boats are generally a reach source of features, it would be desired that areas with low features density are treated with less

[2]Note that only the numerator is needed since the denominator is a constant. In the code, function h will be normalized between 0 and 1 for plotting purposes.



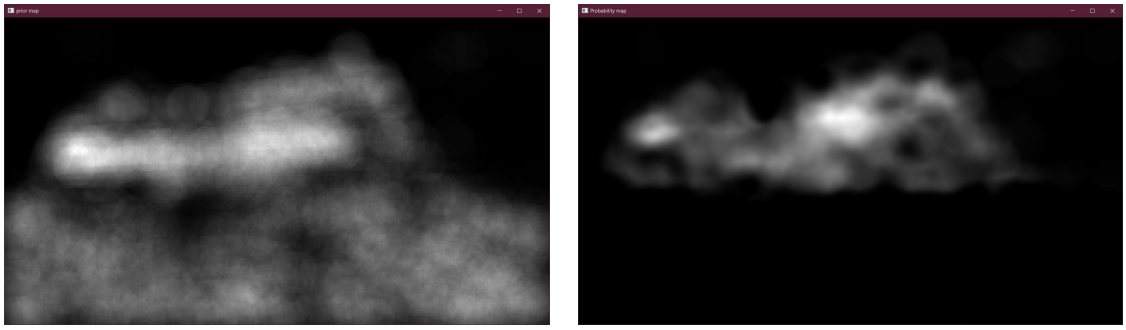
(a) Predictions (b) $h\left(p\right)$

Figure 2: From predictions to the PDF

importance. In virtue of this reasoning, consider the following term to be multiplied with h:

$$s\left(p\right) = \frac{\left|\mathcal{F}_{p,r}\right|}{\sum_{p \in image} \left|\mathcal{F}_{p,r}\right|} \quad \Rightarrow \quad q\left(p\right) = h\left(p\right) s\left(p\right) \tag{5}$$

Figure 3 (a) shows formula (5) applied to Figure 2 (a), and it can be noticed that the features in the top right corner are suppressed. Figure 3 (b) reports the product of $h\left(p\right)$ with $s\left(p\right)$ which gives more satisfactory results.



(a) $s\left(p\right)$ (b) $q\left(p\right)$

Figure 3: From predictions to the PDF

To improve computational speed, $h\left(p\right)$ and $s\left(p\right)$ are evaluated on a subset of pixels. For example, one of every four pixels, and the output value is assigned to the whole neighbourhood of the evaluated pixel (e.g. 4x4 rectangle). Lastly, a Gaussian blur is applied to smooth out the final probability map. This idea can be seen from a Bayesian perspective where $s\left(p\right)$ represents the prior, namely the probability of being a boat before seeing the classifier's predictions, $h\left(p\right)$ is the likelihood given by the predictions, and $q\left(p\right)$ is the posterior.

2.4 Clustering with Mean Shift

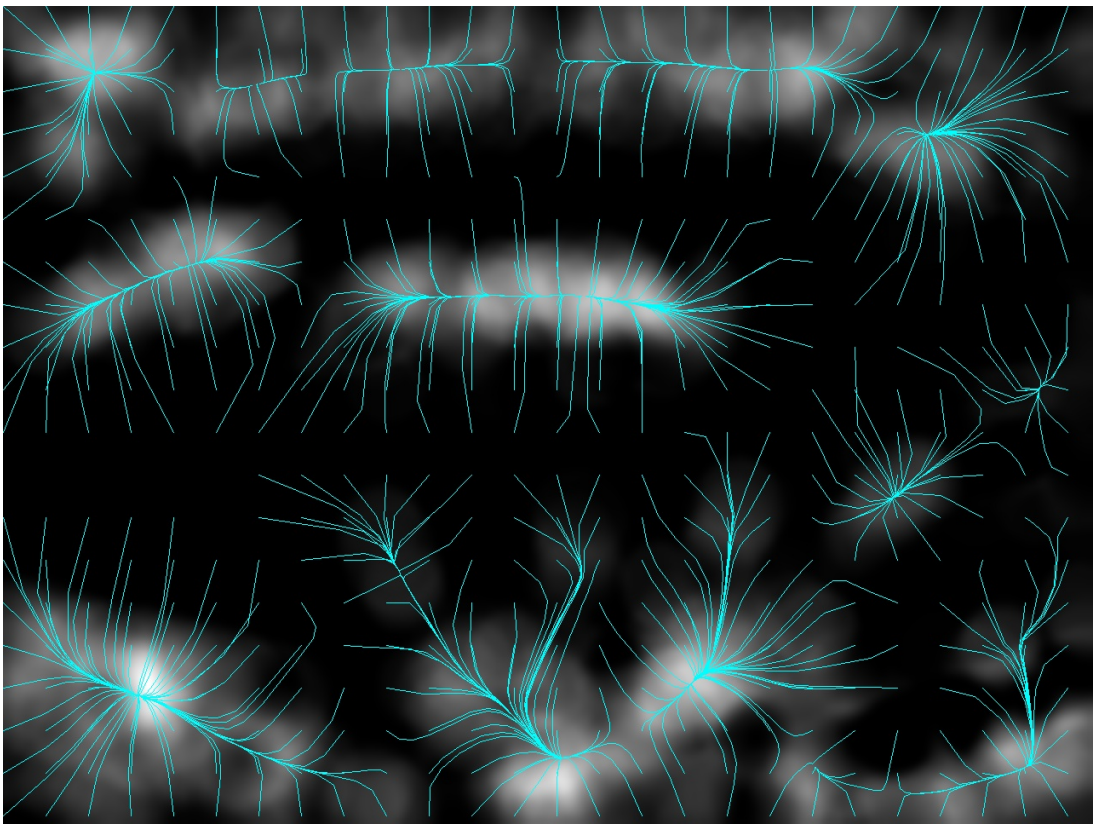
The aim is to find the modes of the previously defined density and obtain clusters. The number of clusters is not known in advance, and generally, they are not spherical shaped. Therefore a good option is given by the mean shift algorithm. Although usually mean shift seeks for modes starting from samples of the target density, in this case, since the density is known, it is better to adjust the mean shift algorithm to operate directly on the density. For this purpose, it is sufficient to change the definition of the center of mass as follows:

$$CoM(p, r_{ROI}) = \frac{\sum_{x_i \in r_{ROI}} dist\,(p, x_i)\, q\,(x_i)}{\sum_{x_i \in r_{ROI}} q\,(x_i)} \tag{6}$$

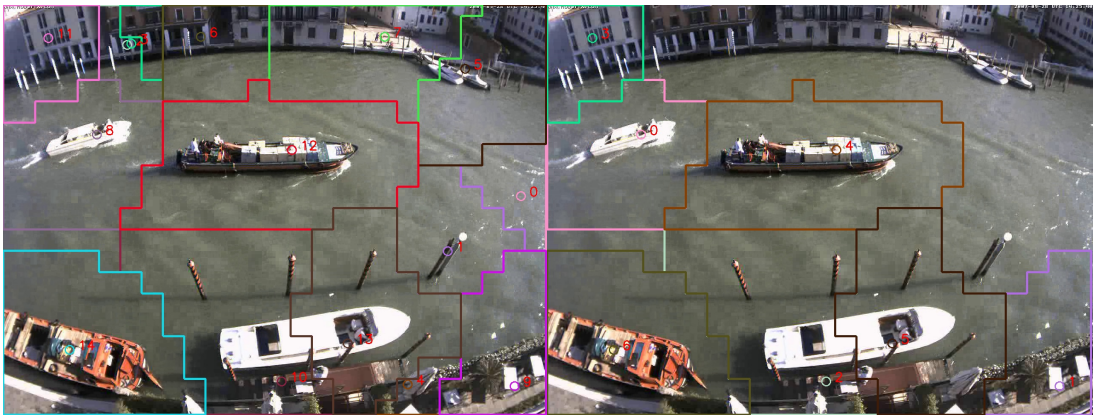
where r_{ROI} is the radius of the region of interest centred in p.

Moreover, since only a coarse segmentation is needed, the PDF support can be tessellated with few initial points.

Figure 4 (a) is an example of the attraction basins and the recovered clusters (b). A number that indicates the strongness of the cluster is assigned to each cluster. Namely, the clusters are ranked based on the average density in an area near the peak.



(a) Attraction basins



(b) Initial segmentation (c) Final segmentation

Figure 4: Mean shift clustering

Finally, clusters are suppressed and pruned (Figure 4 (c)). Suppression consists of erasing the weakest clusters (under a threshold). While pruning merges weak clusters to near strong ones. In particular,

the algorithm first performs a suppression, then the pruning operation and finally a second suppression with a higher threshold.
The output of this clustering process will be a set of masks that segment the probability density. Note that some areas will be left out from the segmentation, for instance, where the probability is zero.

2.5 Generating the bounding boxes

To create the bounding boxes, an iterative algorithm is developed. The bounding box should surround all the regions where the probability is high and, at the same time, include as few as possible areas where the probability is low. Each iteration consists of two steps: expansion and shrinkage. In the former, for each of the four sides of the rectangle, the average density in the rectangle adjacent externally to the side is evaluated (see Figure 5). If the average is bigger than a threshold, the rectangle is expanded in that direction. Conversely, the latter passage computation is done on a rectangle touching the inner side, and if the average is lower than a threshold, the rectangle is shrunk. These two steps are repeated for each cluster until either the box is no longer moving or the number of allowed maximum iterations is reached.

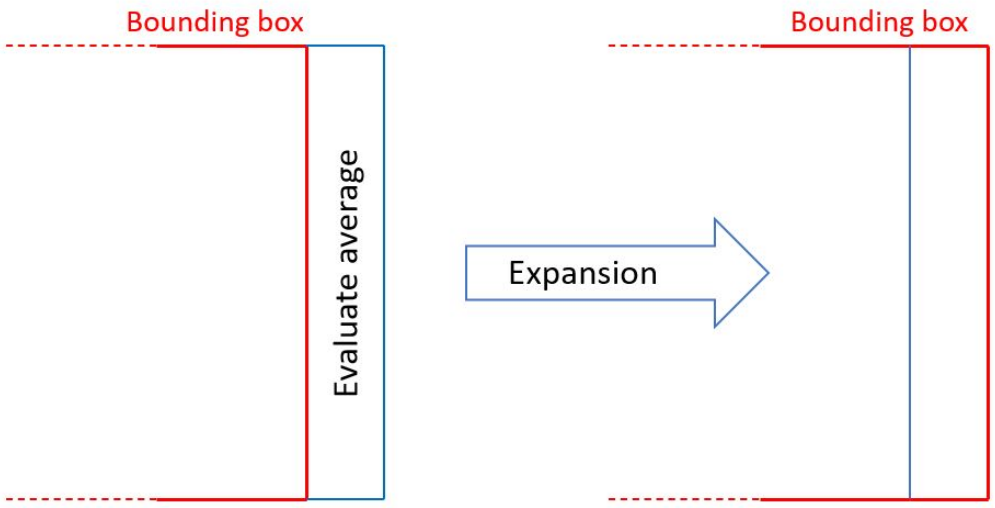


Figure 5: Expansion step on right side

The algorithm is initialized with a rectangle that has the same size as the whole image. It can also be initialized with a small rectangle at the peak of the distribution. However, if, after pruning the modes, the cluster is a multi-modal distribution, there is the possibility that one or more peaks are left outside the bounding box.
To increase time performance, the average is evaluated using the integral image of the PDF.

2.6 Algorithm parameters

Each step of the algorithm has many parameters that directly affect the quality of the results. Table 1 in the Appendix reports the main ones and their default value.

3 Results

The two most significant drawbacks of this algorithm are the classifier's poor accuracy and the fact that it uses only information from the features. The first weakness is the cause of a high false-positive rate, especially when the background is not uniform (as for the Venice data set). The second flaw causes to have multiple boxes for the same boat or a single box for multiple boats close together. Performance can be improved by fine-tuning the algorithm's parameters to each scenario. However, it would be desired to have an algorithm that works with the same parameters for every image.
The founded bounding boxes are drawn in red around the boats, while in green are represented the ground truth boxes. Moreover, for each green box, the IoU metric is reported in the top left corner as a percentage. For each green box, the IoU is computed for every predicted red box, and the highest value is chosen. Tables 4 and 5 (in the Appendix) report the average IoU value for each image. However, note that the computed value is not so significant for discriminating between a good and a bad detection algorithm. Suppose to have two detection algorithms: A and B. Both algorithms predicted boxes are the ground truth ones, but algorithm B has some extra false positive bounding boxes. For both algorithms, the average IoU will be one however, algorithm A is clearly better since it is not affected by false-positive boxes. Therefore Tables 4 and 5 report also the false positive ratio[3], and the missed ratio[4].

3.1 Kaggle data set

The algorithm attains the best results on these test images since, due to the uniform background, there are usually not many false-positive boxes. In Figure 6 an example of good detection is shown.

Figure 6: Example of good detection

[3]Defined as the number of predicted boxes that do not intersect with any ground truth box over the total number of predicted boxes.

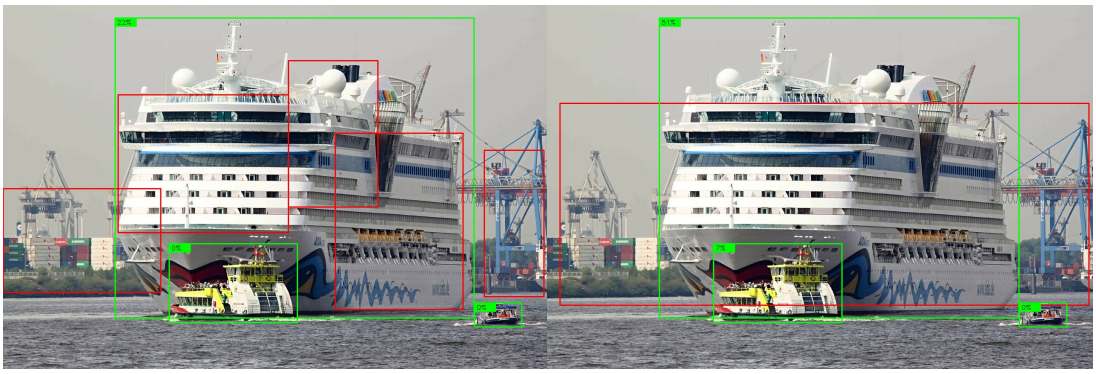
[4]Defined as the number of ground truth boxes that do not intersect with any predicted box over the total number of ground truth boxes.

However, if the boats are too small, not many features are found, and therefore, some boats could not be detected, as happens in Figure 7 (a). As already stated, the same parameters should be used for every image, although in this case, by adjusting some of them (see Table 2), results can be significantly improved (Figure 7 (b)).

(a) Default parameters (b) Tuned parameters

Figure 7: Example of missed box

Consider now Figure 8 (a), here both previously discussed issues are present. Indeed the cruise ship is split into three boxes, and there are two false positives (on the left and the right) coming from a neural network's misclassification. Boxes can be merged by increasing the window size of the mean shift clustering, but by doing so also the false positive boxes are incorporated (Figure 8 (b)).



(a) Default parameters (b) Tuned parameters

Figure 8: Example of multiple boxes for one object

3.2 Venice data set

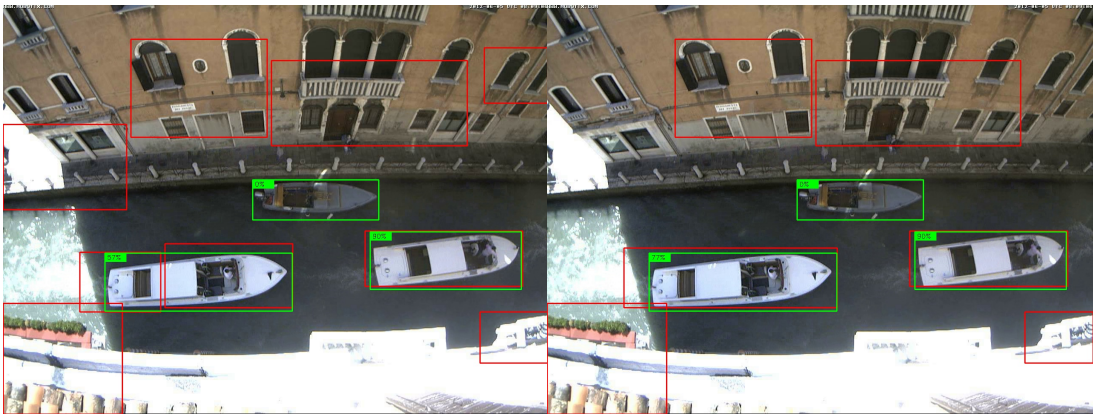
The performances on the Venice data set are generally lower since the images often include buildings on the sides of the canals that are often misclassified by the neural network. Furthermore, boats are often moored near these buildings or near each other (or both), making the clustering not work correctly.

For instance, in Figure 9 (a), there is a false positive box in the top left corner, and one in the bottom right one, the two boats in the top right corner are suppressed due to a wrong clustering (merged with the buildings), and the boat in the bottom (centre) is detected with two boxes. This last issue can be solved by increasing the pruning distance parameter as reported by Figure 9 (b).

A similar situation is depicted by Figure 10 (a), but hereby increasing also the sup2_thr parameter, it is possible to remove some of the false-negative boxes (Figure 10 (b)).

(a) Default parameters (b) Tuned parameters

Figure 9: Example of the main issues in the Venice data set



(a) Default parameters (b) Tuned parameters

Figure 10: Example of several false positives

Results are even worst in low light conditions, as shown in Figure 11. Besides the many wrongly classified features, the boats do not have many features; namely, the feature density is lower on the boat with respect to the buildings. Therefore the proposed algorithm is ineffective in these conditions.

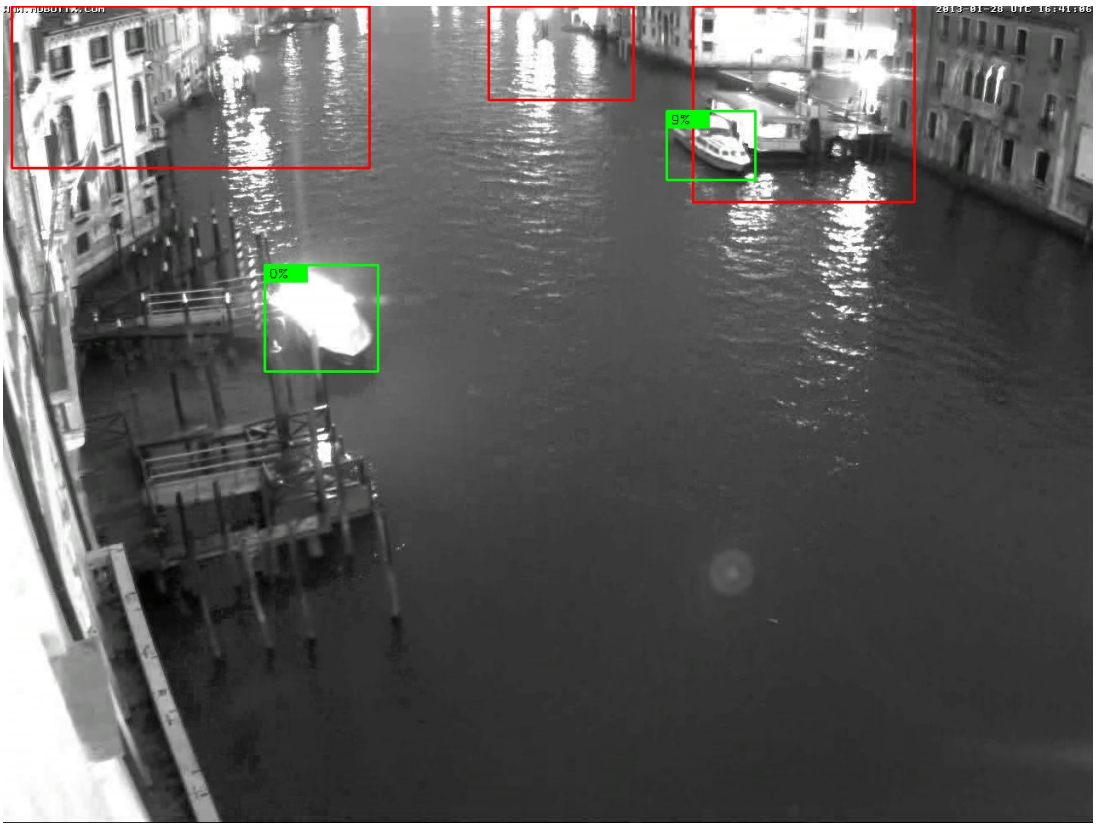
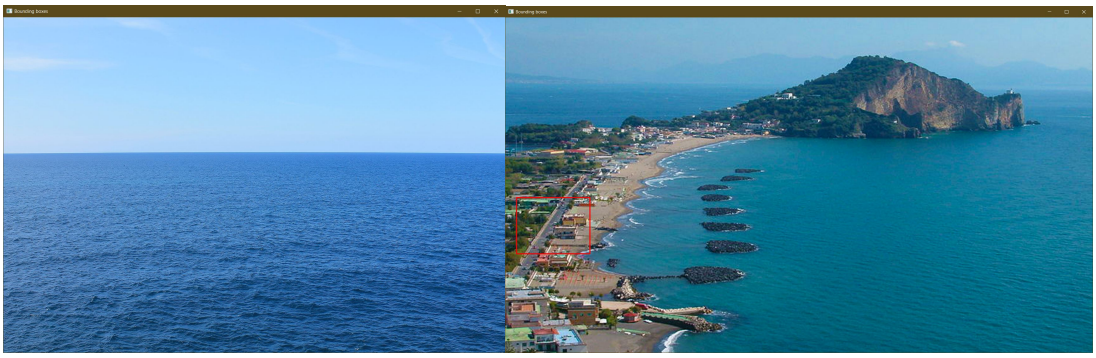


Figure 11: Example of ineffective detection

3.3 Images without boats

If the image has no boats in it, the algorithm should not draw any boxes. However, as Figure 12 shows, the algorithm tends to respond correctly only for simple scenes (as the ones found in the Kaggle data set), while false positives may occur in more cluttered scenarios.

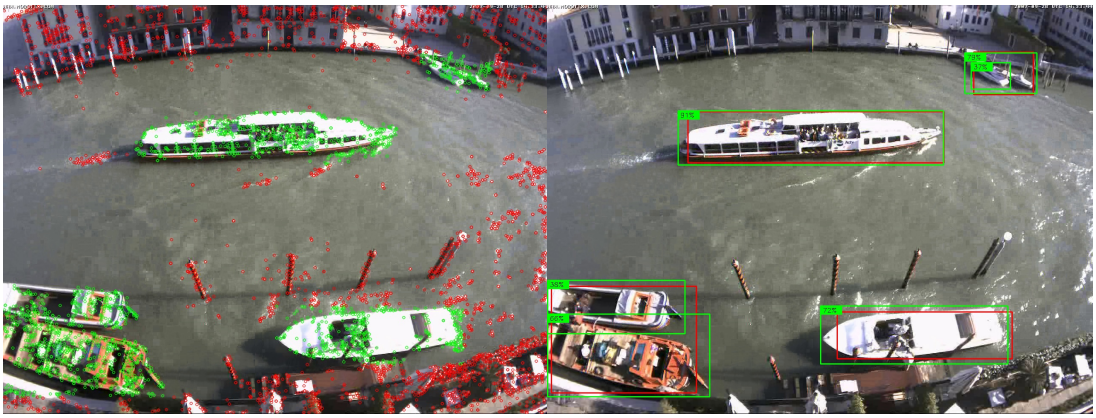


(a) Correct classification (b) Incorrect classification

Figure 12: Example: images with no boats

4 Conclusions

The proposed solution's results depend heavily on the classifier's accuracy, and the proposed one's performances are unsatisfactory. The data set is quite complex with a wide variety of boats; however, it would be suggested to use other machine learning libraries that implement more features such as dropout and softmax layers and better diagnostic tools. Despite the classifier being the weakest link of the algorithm, there are other issues that do not depend on the classifier. Even if perfect predictions are available, as shown in Figure 13, the algorithm can still confuse two adjacent boats as one.



(a) Ground truth classification (b) Bounding boxes

Figure 13: Example of detection with perfect classification

To solve this second problem, the algorithm should consider some other information of the image besides its features.

As already presented, performance can vary significantly by tuning the algorithm's parameters, and from Tables 2 and 3 some guidelines can be extracted. It is suggested to lower the thr2 parameter for offshore detection since the predicted bounding boxes are generally smaller than the actual boats. Whereas for scenarios near coastlines or harbours (or Venice), it is suggested to increase the sup_thr2 parameter to remove some of the false-positive boxes.

As far as computational time is concerned, the algorithm is not suited for real-time applications. In spite of the numerous stratagems been applied to speed up the computation, the total execution time to process an image ranges between 300 and 3000 milliseconds depending on the type of image. The pie chart in Figure 14 reports what time percentage each part of the algorithm requires[5].

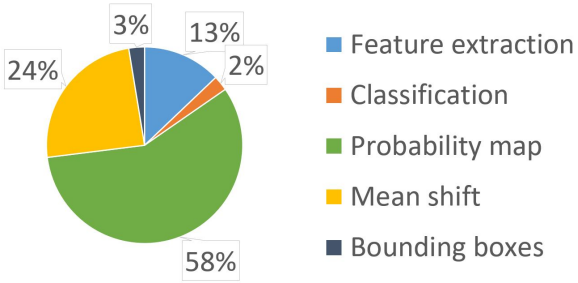


Figure 14: Subdivision of the execution time

[5]It is only indicative as it depends on the resolution of the image and on the chosen parameters of the algorithm.

5 Appendix

Name	Value	Description
r	50	Radius defined by equation 2
win_size	100	Mean shift window radius
prune_thr	100	Maximum distance for pruning
sup_thr1	0.2	First clusters suppression threshold
sup_thr2	0.5	Second clustering suppression threshold
bb_thr1	0.05	First bounding box placer threshold
bb_thr2	0.2	Second bounding box placer threshold

Table 1: Algorithm parameters

Image	r	win_size	prune_thr	sup_thr1	sup_thr2	bb_thr1	bb_thr2
Kaggle 01	d	300	d	d	d	d	d
Kaggle 02	20	d	d	d	d	d	0.05
Kaggle 03	d	d	d	d	d	d	0.05
Kaggle 04	d	d	150	d	d	d	d
Kaggle 05	40	d	d	d	d	d	d
Kaggle 06	d	d	d	d	d	d	0.05
Kaggle 07	d	d	150	0.1	0	d	0.05
Kaggle 08	15	d	d	0	0	d	0.05
Kaggle 09	d	d	d	d	d	d	0.05
Kaggle 10	d	75	d	d	0.8	d	d

Table 2: Kaggle test images tuned paramters (d = default)

Image	r	win_size	prune_thr	sup_thr1	sup_thr2	bb_thr1	bb_thr2
Venice 00	d	d	d	d	d	d	d
Venice 01	d	d	d	d	d	d	d
Venice 02	d	75	50	d	0.8	d	d
Venice 03	d	d	150	0.6	d	d	d
Venice 04	d	d	110	d	d	d	d
Venice 05	d	75	50	d	0.55	d	0.1
Venice 06	d	d	150	d	d	d	d
Venice 07	d	d	d	d	0	d	d
Venice 08	d	d	d	0	0.6	d	d
Venice 09	d	d	d	d	0.7	d	0.1
Venice 10	d	d	120	d	0.65	d	d
Venice 11	d	d	110	d	0.7	d	d

Table 3: Venice test images tuned parameters (d = default)

Image	Default parameters			Tuned parameters		
	Average IoU	False positive rate	Missed rate	Average IoU	False positive rate	Missed rate
Kaggle 01	0.073	2/5	2/3	0.195	0/1	1/3
Kaggle 02	0.371	0/1	0/1	0.560	0/1	0/1
Kaggle 03	0.347	0/1	0/1	0.609	0/1	0/1
Kaggle 04	0.175	0/1	0/2	0.343	0/1	0/2
Kaggle 05	0.189	0/1	0/1	0.218	0/1	0/1
Kaggle 06	0.236	0/1	0/2	0.357	0/1	0/2
Kaggle 07	0.126	0/1	1/2	0.440	0/2	0/2
Kaggle 08	0.132	0/2	1/3	0.538	0/3	0/3
Kaggle 09	0.799	0/1	0/1	0.876	0/1	0/1
Kaggle 10	0.777	5/6	0/1	0.456	0/1	0/1
Total	0.323	7/20	4/17	0.459	0/13	1/17

Table 4: Kaggle test images outcomes

Image	Default parameters			Tuned parameters		
	Average IoU	False positive rate	Missed rate	Average IoU	False positive rate	Missed rate
Venice 00	0.096	4/5	4/5	a.d	a.d.	a.d.
Venice 01	0.039	2/4	2/4	a.d	a.d.	a.d.
Venice 02	0.047	2/3	1/2	0.143	0/1	1/2
Venice 03	0.146	5/7	1/2	0.331	1/2	1/2
Venice 04	0.430	2/7	2/6	0.449	2/6	2/6
Venice 05	0.130	1/4	2/9	0.219	0/5	0/9
Venice 06	0.112	1/5	4/23	0.107	1/4	7/23
Venice 07	0.216	4/6	0/5	a.d	a.d.	a.d.
Venice 08	0.317	4/7	2/5	0.317	3/6	2/5
Venice 09	0.309	3/5	1/5	0.333	0/2	1/5
Venice 10	0.340	2/7	0/6	0.391	1/4	2/6
Venice 11	0.491	6/9	1/3	0.555	4/6	1/3
Total	0.223	36/70	20/75	0.266	22/51	23/75

Table 5: Venice test images outcomes (a.d. = as default)

References

[RB93] M. Riedmiller and H. Braun. "A direct adaptive method for faster backpropagation learning: the RPROP algorithm". In: *IEEE International Conference on Neural Networks*. 1993, 586–591 vol.1. DOI: 10.1109/ICNN.1993.298623.

[TS18] Shaharyar Ahmed Khan Tareen and Zahra Saleem. "A comparative analysis of SIFT, SURF, KAZE, AKAZE, ORB, and BRISK". In: *2018 International Conference on Computing, Mathematics and Engineering Technologies (iCoMET)*. 2018, pp. 1–10. DOI: 10.1109/ICOMET.2018.8346440.