COMPUTER VISION COURSE FINAL PROJECT – BOAT DETECTION

Tonin Alessandra – ID 2027136

In this report I will present my approach to the boat detection problem and the obtained results on the assigned test images. For compiling instruction and all other technical information regarding the execution of the code, please refer to the README.md file provided with the source files. The only fact I would like to highlight is that, when running the code, you can decide either to execute it all and re-train from scratch the model, or just test the saved and already trained one: this is provided for convenience, since re-training the entire model and testing it, will take a considerable amount of time (at least 20 minutes on my PC).

My solution to the proposed challenge is based on Histogram of Oriented Gradients and a Support Vector Machines classifier [1][2]. In general, boat detection task is quite difficult, since there is a lot of variance among the images, due to different scales, orientations, types and shapes of boats and, most of the times, there is noise due to water waves and reflects. Moreover, ships can appear at different distance from the camera, and may be overlapping each other, so partially occluded.

Regarding in particular this work, we need to consider that the two provided test datasets are really different one from the other, so all the choices regarding the various parameters are a compromise between the results in each of them: this means that, in general, a parameter value which led to good results on Venice dataset, gave bad ones on the Kaggle images, and vice-versa. So, I had to accept a trade-off, to get a reasonable output on both the test sets.

Now, I will describe in detail the procedure I adopted, and some choices I made.

First, I decided to use classical computer vision techniques to carry out the project, avoiding deep learning methods, even if models like Convolutional Neural Networks or Single-Shot Detectors are currently the state of the art in this field.

Reading some papers, in particular [3], I found out that the combination of HoG features and SVM classifier gives really good results in object detection tasks, also when dealing with different types of items.

The first step I performed was preparing the datasets: training a Support Vector Machine requires a set of positive images and a set of negative ones. The set of positive samples have been obtained by cropping the Kaggle and Mar datasets' images in a way so that only the boats remained, with the minimum possible amount of water or general background. The set of negative images has been obtained by taking parts of the images that were not boats and that were wrongly predicted by the classifier, for example windows, bridges, water, and so on. Moreover, into the negative set I put the "buoy" category images from the Kaggle dataset [4], the "water" category images from the MAR dataset, and other negative samples provided and classified as false positive by the MAR dataset creators [5]. In total I used about 4200 positive images and 8400 negative ones. The code for the construction of the datasets is not provided, since I think it is not relevant for the purpose of this project.

Then, I applied some steps of pre-processing to the obtained images:

- resizing all of them to be of the same dimension, as required by the Support Vector Machine model. After a lot of tries, the size has been set to 130x90, since those are the values with the best trade-off between good performances of Venice dataset and on Kaggle one.
- denoising, performed using a technique called "Non-Local Means Denoising" [6]. This method considers a small window in the image and searches in a small neighbourhood for similar patches, and then it replaces the pixel values with the average value computed from this set of similar parts of the image: in this way, I get better results with respect to simpler denoising techniques, like Gaussian blur or similar ones. Indeed, these basic methods are not so robust against camera and scene motions, and since most of the images in the training set were taken using moving cameras or, in any case, during scene motions like, for example, waves caused by boat movements, I think this approach is better than the classical one. The drawback of this noise removing technique is the considerable computational time required, but it is still acceptable, counting the number of images to process.

The third phase is the extraction of Histogram of Oriented Gradients features from both the positive and negative training sets. The extracted gradients, moreover, need to be converted into a format that is accepted as training data by the OpenCV machine learning implementation.

After this, a SVM classifier for regression task is created using default values for parameters, and then it is trained using the data computed before. By default, the model is trained two times: the second one is a kind of hard-negative mining, since the classifier is trained on the negative samples.

The next phase consists in testing the trained detector on the images belonging to the test datasets: from the detection process, performed using the *detectMultiScale* method of the *HOGDescriptor* class of OpenCV, I get the predicted bounding boxes, together with the relative confidence scores.

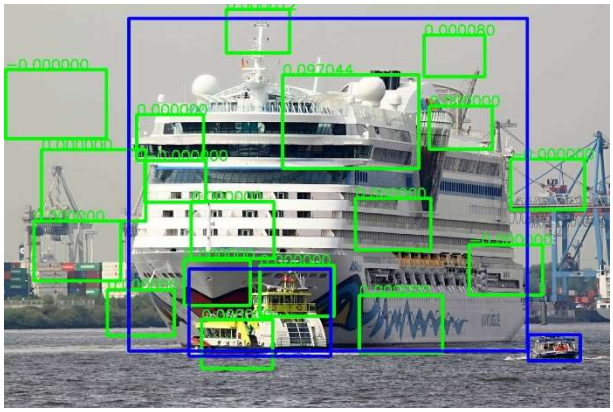
This data is used to perform the postprocessing phase, that aims to reduce the number of overlapping boxes and to evaluate the results obtained by the detector.

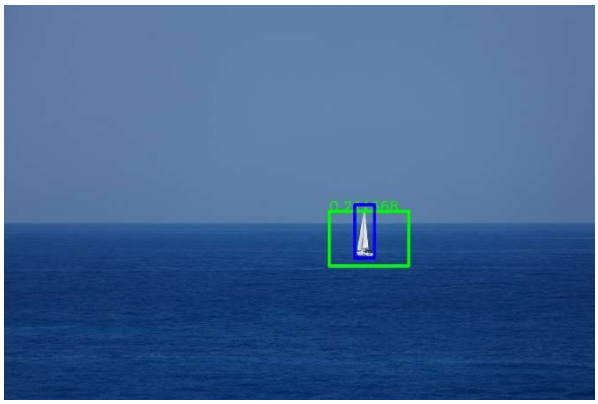
The number of overlapping boxes is reduced by using a Non-Maxima Suppression algorithm [7] [8] [9][10], based on confidence scores of each rectangle. The predicted bounding boxes are sorted by increasing confidence score and, at each iteration, the overlap between the highest confidence rectangle and the other predicted boxes is computed using the IoU metric (for a deeper explanation of this concept, refer to the next section, when analysing the performance evaluation). If the overlapping area is above a threshold, the predicted box is suppressed. For the choice of the threshold, I had the same problem as before, this value causes different behaviours in the various images, including some wrong suppression sometimes.

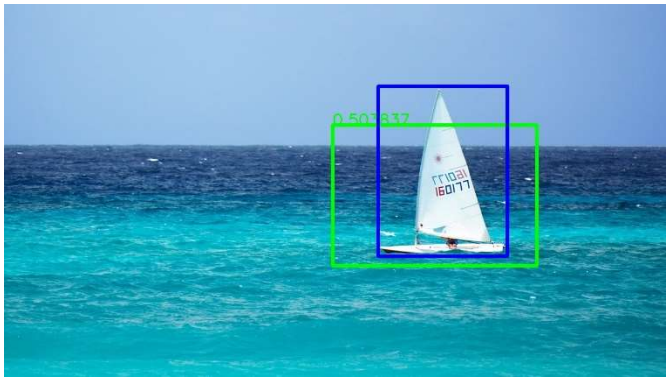
After this, we have the last part, that is the performance evaluation by means of the Intersection over Union metric [11][12]. This value is defined between a pair of bounding boxes, and it is computed as the intersection area of the two rectangles, divided by their union area. It is a commonly used metric in detection tasks, and it can be easily extended to other geometrical shapes. In this project, the IoU is computed between each predicted box and each ground truth box, and only the maximum IoU for each detected box is kept. As ground-truth for the test images, I used the .txt files provided in the forum: there's one file for each image and, in each file, a row for every real boat. The bounding box of each ship is stored as a 4-tuple composed by (x coordinate of top left

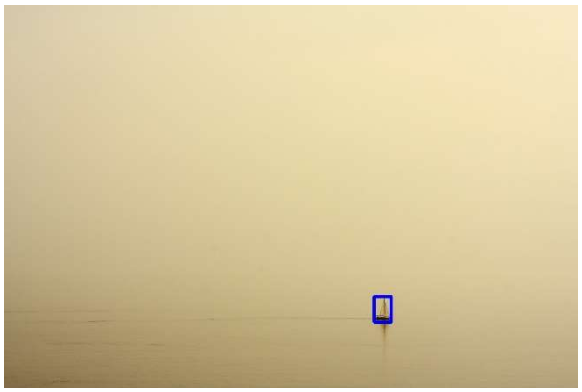
corner; x coordinate of bottom right corner; y coordinate of top left corner; y coordinate of bottom right corner). The .txt files are processed in order to obtain a vector, storing the mentioned coordinates in an easier-to-handle way in OpenCV.

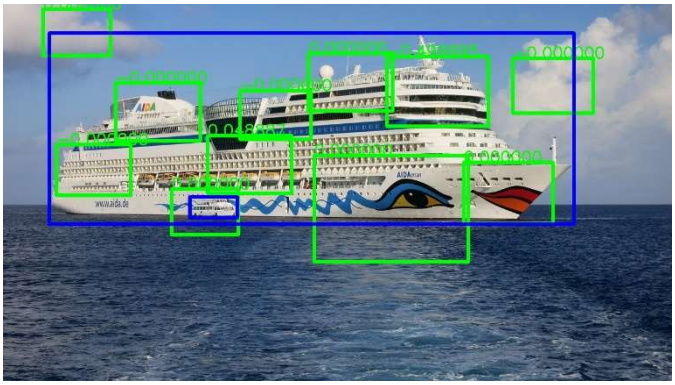
In the following section, you can see the results obtained by using this approach in the images belonging to the two assigned datasets (the same images are also saved in the results/ folder of the project, so you can better see the bounding boxes and the IoU scores). In each image, there are the ground-truth bounding boxes drawn in blue, while in green there are all the detected rectangles (both true and false positives). Next to each box, there's the IoU score.

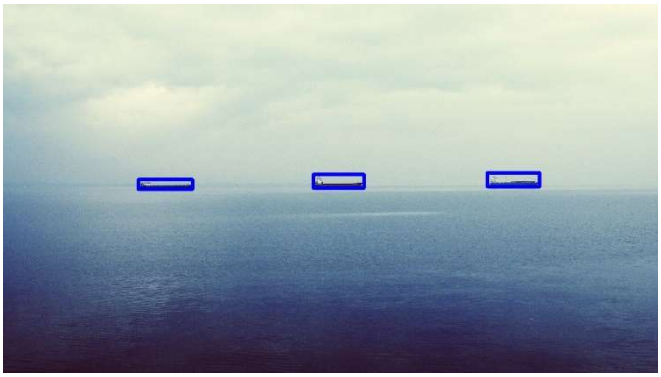


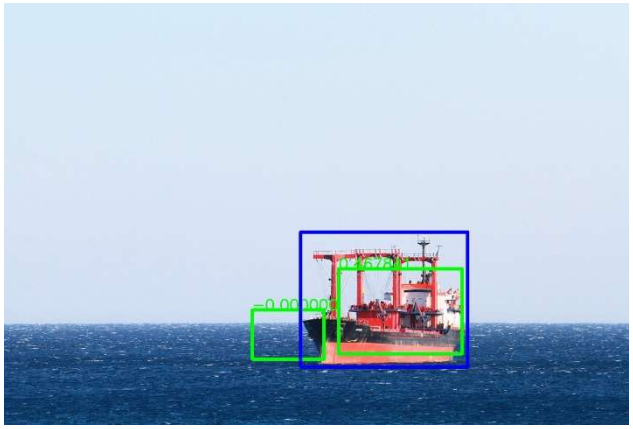


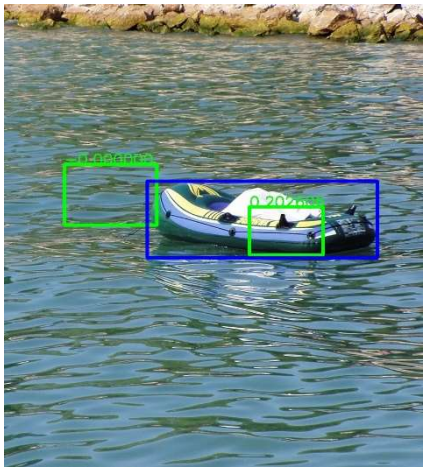


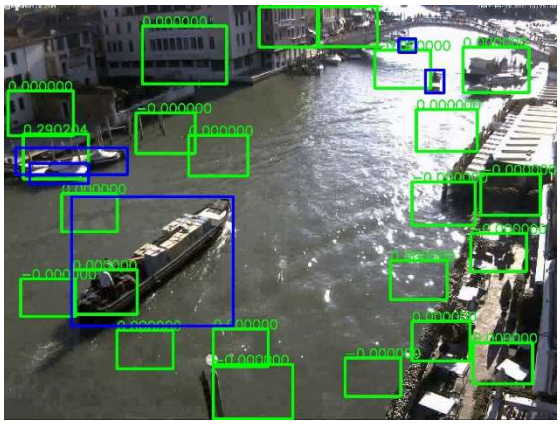


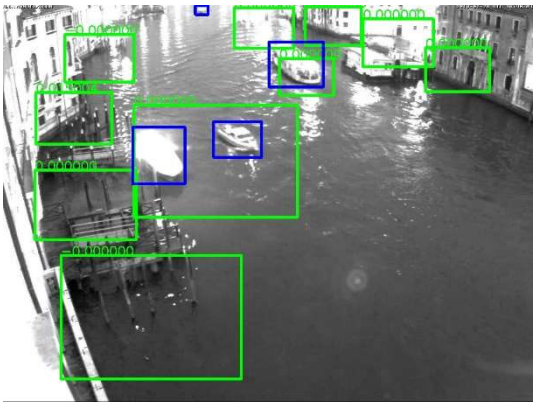


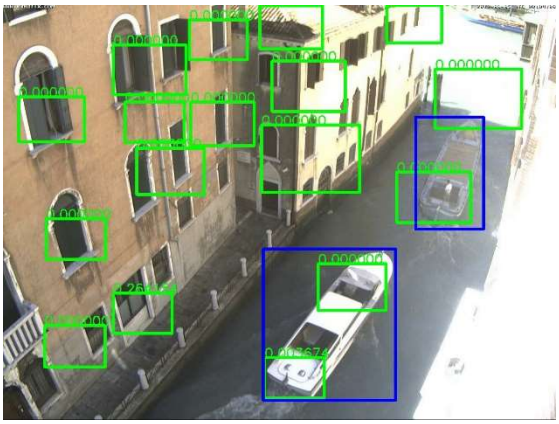


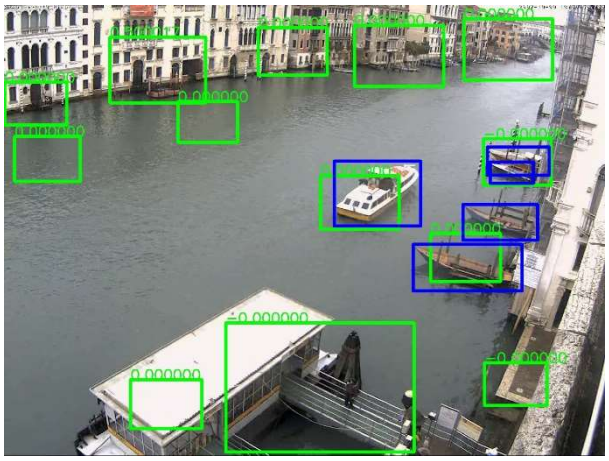
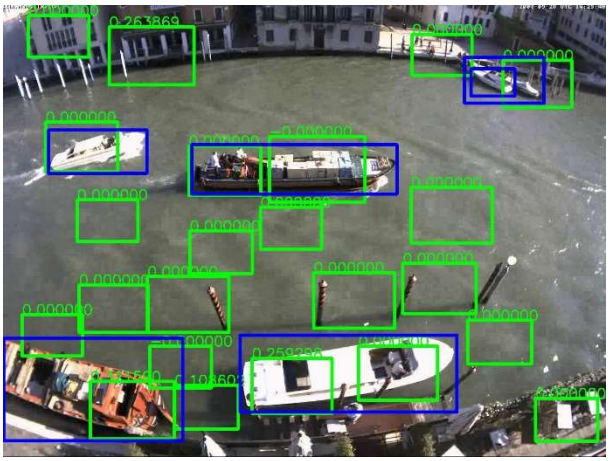
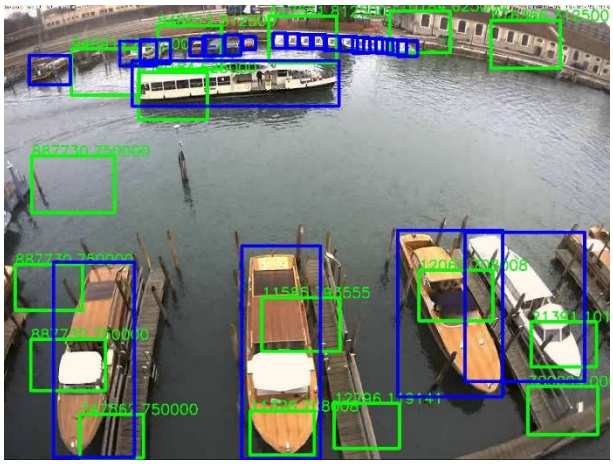
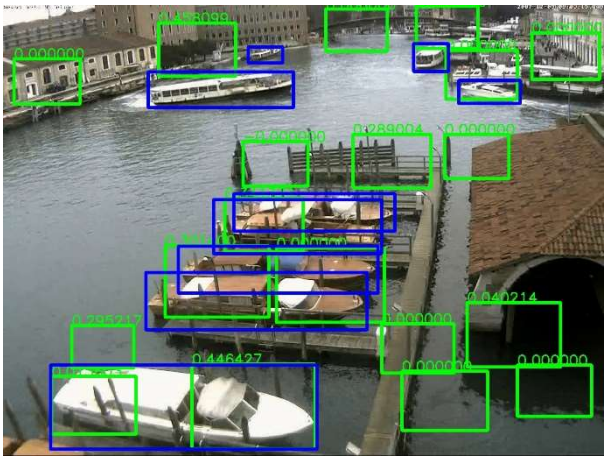
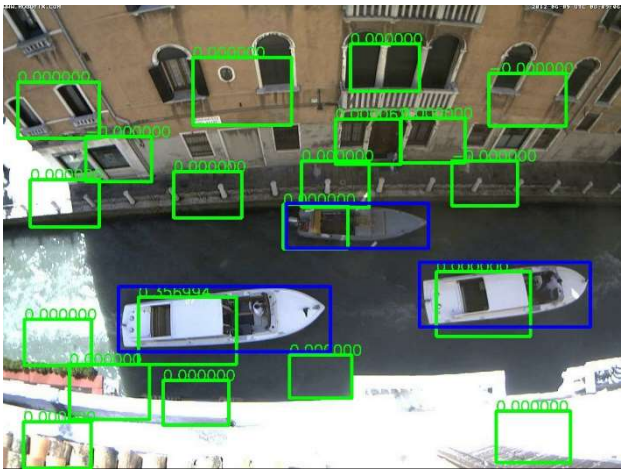
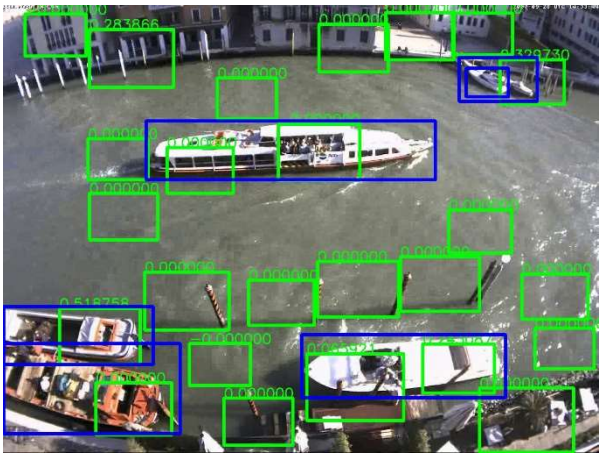
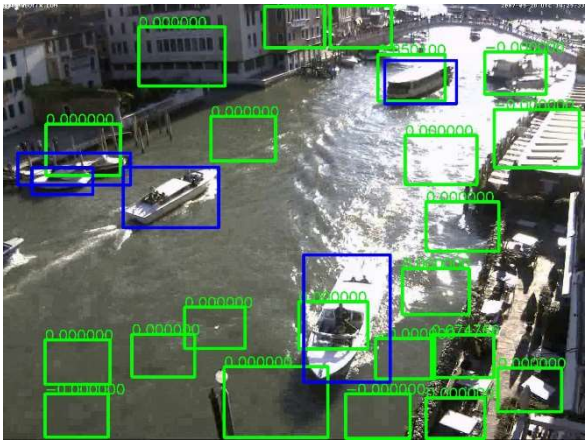
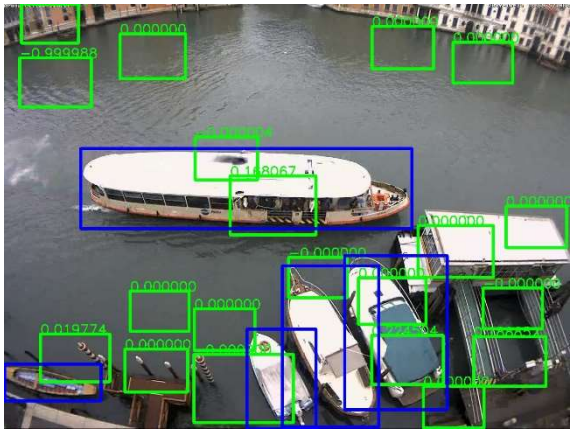










As you can easily see, the technique consisting in HoG features combined with a SVM classifier is not so good as expected, applied to this particular task: there are a lot of false positives, even if many examples of water or buildings are used to train the model, and a considerable amount of boats is not detected. Moreover, the IoU values are quite strange: this is probably caused by an error in the implementation of the intersection over union evaluation function.
I think these poor results are due to the particular characteristics of the two used datasets, which are really different one from the other. In Kaggle images, we have clean and easy to identify boats, in most cases just one per image, or more than one but clearly distinguishable one from the other and from the background. Moreover, most of these test images include lateral views of the ships, and quite big sizes for them. In the Venice dataset, instead, we have a lot of issues caused by buildings, bridges, wooden poles, reflects, ripples and waves, we have always many boats in each scene, and often ships are really close one to each other, or even partially occluded. Moreover, there are boats of a lot of different sizes, and the images are taken using cameras which are in an upper position with respect to the water: for this reason, in these images we have lot of top views of the ships.

All these differences in size, shape, environment and so on, also cause a big variance in the extracted features and have a negative impact on the potential results of the proposed method: for example, the choice of the size of the window to perform detection is not easy at all. I tried both with square windows and rectangular ones, and with multiple sizes for both cases, but I was not able to find a value which performed good in a sufficient number of images. So, the actual parameters used in the project are the result of many tries and of a trade-off, decided in order to have satisfactory results in at least some images of each dataset.

In conclusion, I think that this approach is not really suitable for this type of problem and for this data, maybe a region-based method, using some segmentation techniques to identify regions of interest, combined with a classification model, like SVM, or a neural network, would have led to better results.

REFERENCES

1. https://docs.opencv.org/4.5.1/d0/df8/samples_2cpp_2train_HOG_8cpp-example.html#a68
2. https://www.pyimagesearch.com/2014/11/10/histogram-oriented-gradients-object-detection/
3. http://lear.inrialpes.fr/people/triggs/pubs/Dalal-cvpr05.pdf
4. http://www.diag.uniroma1.it//~labrococo/MAR/detection.htm
5. https://www.kaggle.com/clorichel/boat-types-recognition/version/1
6. https://docs.opencv.org/4.5.1/d5/d69/tutorial_py_non_local_means.html
7. https://learnopencv.com/non-maximum-suppression-theory-and-implementation-in-pytorch/
8. https://www.pyimagesearch.com/2014/11/17/non-maximum-suppression-object-detection-python/
9. https://www.analyticsvidhya.com/blog/2020/08/selecting-the-right-bounding-box-using-non-max-suppression-with-implementation/

10. https://github.com/Nuzhny007/Non-Maximum-Suppression/blob/master/nms.h
11. https://www.pyimagesearch.com/2016/11/07/intersection-over-union-iou-for-object-detection/
12. https://towardsdatascience.com/metrics-to-evaluate-your-semantic-segmentation-model-6bcb99639aa2