***FINAL PROJECT – BOAT DETECTION***

In this report I will present my approach to the boat detection problem: in it based on Histogram of Gradients features and Support Vector Machines classifier. The task is quite difficult, since there are lots of problems in the images, like reflections, water wags, different scales and orientations for the ships, in addition to many different possible types and shapes of boats. Moreover, boats can appear at different distanced, and may be overlapping each other.

Considering also that the two test datasets are really different one from each other, 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 one on the Kaggle images, and viceversa. So, I had to accept a tradeoff, in order to get quite satisfactory output on both the datasets.

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

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

So, reading some papers, I found out that the combination of HoG features and SVM classifier gave really good results in object detection, also with different types of objects.

The first thing I did was preparing the datasets: training an SVM 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 misclassified by the classifier, for example windows, bridges, water, and so on. In total I used about 4200 positive images and 8400 negative ones.

Then, I applied a bit of preprocessing to the obtained images:

* resizing all of them so as 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 135x90, since those are the values with the best tradeoff between good performance of Venice dataset and good performance on Kaggle one.
* denoising, performed using a technique called “Non-Local Means Denoising”. This method considers a small window in the image, in searches in a small neighbourhood for similar patches, and the it replaces the pixel values with the average value computed from this set of similar prts of the image: in this way, I can gent better results with respect to simpler denoising techniques, like Gaussian blur or similar ones. Indeed, these 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 (check\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*), I think this approach is better than a simple Gaussian smoothing. The drawback of this method is the considerable computational time required, but it is still acceptable, considering the number of images to process.

The third step is the extraction of Histogram of Gradients feature from both the positive and negative training sets. The extracted gradient, 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 svm is trained on negative samples.

The next phase consists in testing the trained detector into te test datasets.