FINAL PROJECT: BOAT DETECTION

The requested task is to deliver a program able to detect boats by a given input image.

To deliver this program it has been requested 90 hours of work, most of them requested to sample the provided dataset.

Before starting the report, I would like to point out that all my choices has been done by trying to avoid as much as possible because it was requested by the task, but also because my knowledge is very low due to the fact that I enrolled to university at late December, so I have skipped all the first semester (with the machine learning algorithm). This doesn't mean that I don't know anything about CNN, instead I really liked to try Keras and the CNN lecture.

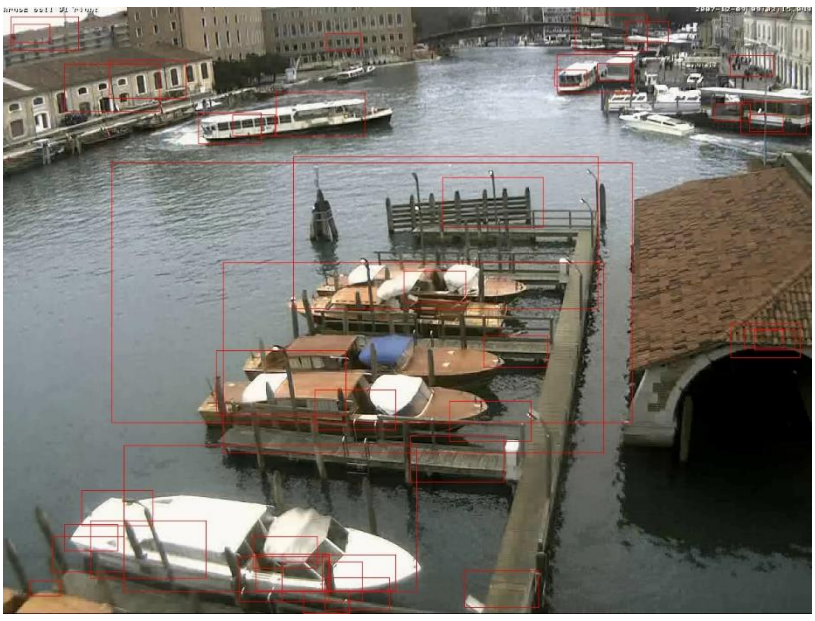
Due to the fact that it was requested to use as less as possible any kind of machine learning tool, my first approach was done with sliding window approach with a cascade of classifier, as we have seen during the face recognition lecture (Viola&Jones approach).

I have done this first choice because I was thinking that the best approach to solve a detection problem was to use a sliding window approach and because there were already some useful tools provided by OpenCV to train a cascade of classifiers with HAAR features.

This choice then imply to build a good and large dataset of very good boat samples and a much more larger dataset with negative samples. These samples must contain information about the general background, but, most important, they don't have to contain any boat.

The process to obtain and build a good dataset, by sampling the ones provided, took a lot of hours (about 5 days in total). After sampling and training the dataset with "opencv_traincascade" app (tool available with OpenCV 3.x versions) I have tested it with some already program that I have found on internet and the results best results obtained can be found in the folder: **`data/cascade_classifier_results`**

Here 2 examples:



The last classifier (the one that we can see in these picture) has been trained during a whole night with the goal to reach 20 stages.

As we can see, the results are really bad and the time to compute the classifier was too high, so I decided to change direction on the developing of the project.

Also, I think, that the bad results are due to the fact that Viola & Jones works very well with frontal faces, then with “objects” that are always on the same pose and that are very similar among them. Instead boats are very different (a gondola has vey few common things with a cruise boat) and also the provided dataset was full of boat with different shapes ad, in particular, poses. And this is another reason that took me to discard this idea.

Instead I kept the idea to use a sliding window approach and to use another kind of features, the SIFT features.

I have chosen the SIFT features because are very robust to scale, rotation and other image transformation. This solves the problem of different poses and the problem of different scales that they can be detected, so I can reduce the number of sliding window and have a faster approach.

My first test was done by creating a sliding window function and for each window extracting a the SIFT feature to compare them with a boat image.

The requested time to compute a single row of an image was too high, so I immediately gave up after the first test.

Convinced that the use of SIFT features was a good approach to classify boats (this because, as said before, they are very different among them and they are represented in a large different poses), I have decided use a different approach that is the one that at the end I have chosen.

BOAT DETECTOR

To detect the boats from the input image I have decide to preprocess the image with some low-mid level operations to obtain some regions that could identify a boat (localization process) and then pass these regions to a classifier to evaluate them. At the end, the IoU it's computed with the ground truth to obtain a score.

The standard used to compute the ground truth is the one provided by using the OpenCV application "opencv_annotation", that was already used for the cascade classifier stage, then every image is saved in a txt file outside the directory containing the images and each represents the ground truth of a single image. The ground truth is saved as following:

`folder\image.png N x1 y1 width1 height1`

where:

1. **`folder\`** is the name of the folder containing the image
2. **`image.png`** is the name of the file
3. **N** are the number of regions of interest (box of boats in our case) presented in the image
4. **`x1 y1 width1 height1`** is the tuple of 4 elements that represent a region in our image with by using the OpenCV class Rect(x1, y1, width1, heght1); also there are many tuples as the regions of interest indicated with **N**

The preprocess stage has been done with some experimental results on the Kaggle dataset by applying different filters in cascade, some techniques to improve the images and some techniques to cluster the regions.

In particular the best results is obtained by applying the following pipeline:

- Bilateral Filter
- Canny Filter (where only edges are considered)
- Morphological Gradient Operation
- Morphological Opening Operation
- Clustering of the regions by applying an algorithm based on IoU

The experiments were done by creating a specific file with all the useful function connected with one or more trackbars to have a live response of the process. This is implemented in the FilteringTest source file.

The experiment results were than implemented in the class Localization.

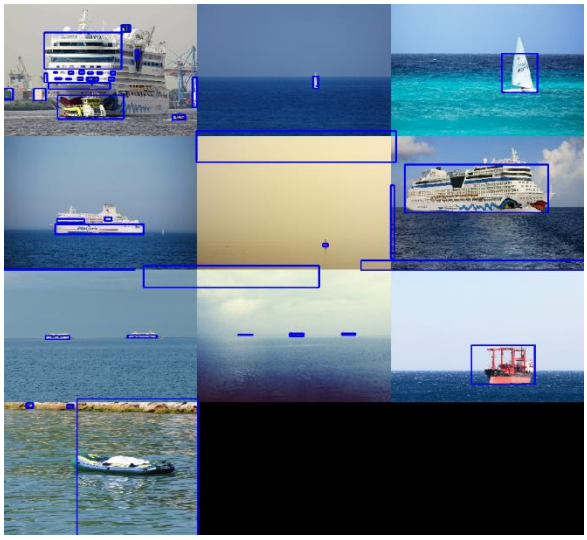


Figure 2 Gaussian Filtering

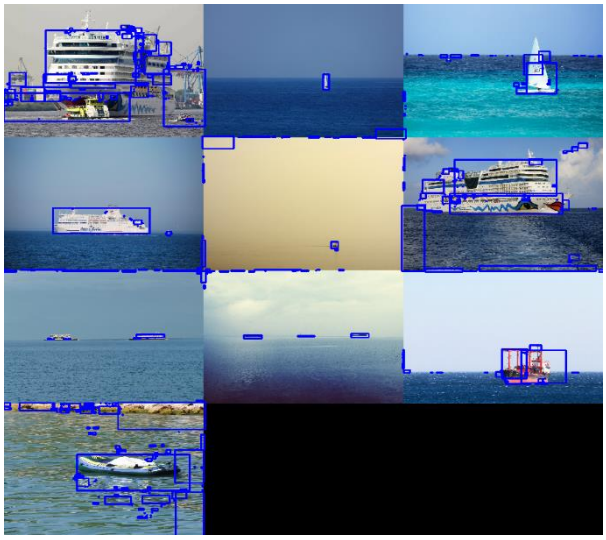


Figure 1 Bilateral Filtering

After that it is applied a simple Bag of Words/Features (from now on BoW) to classify the image.

The BoW is realized by using the tools provided by OpenCV, in particular the BoW has been "trained" by using BOWKMeansTrainer by computing the SIFT features in the collected dataset and clustering them in 10, 100 and 1000 words (it has been created 3 BoW). The classification process is done by BOWImgDescriptorExtractor class, that takes a Mat object as input and returns a vector containing an histogram, where in each bin there are the features that belong to that cluster. After this no SVM or any kind of other machine learning algorithm are used, but it computed the distance among each point and its cluster and then only the cluster with the minimum distance is kept. A classification score is the computed by counting all the remaining keypoints and dividing them by the number of clusters. The same score is computed by passing the input image to the BoW "trained" with a dataset of negative samples. At the end, a region is a boat only if the positive score is greater or equal to the score from the negative set.

The classification step is done by the class ClassificationBoat with the an auxiliary class Dictionary, for a better handling of the dictionary (BoW) functions.

The training process is done by the program BoWCreator, developed by me and provided with the rest of the code.

Also, it has been noted, that number of clusters does not change the quality of the classification.

The last step of the program is to compute the IoU with the ground truth and it's show with:

- a red region all the false positive
- a blue region all the regions that has an IoU greater than zero
- a light blue region the best matches with the ground truth
- a green region the ground truth with its best score (computed with IoU)

At the end a final score of the whole image is computed by summing up all the scores (zero for false-positive) and dividing it by the number of detected regions.

At the end these are the result obtained by testing it on the Kaggle dataset.

In the following page the obtained score results.

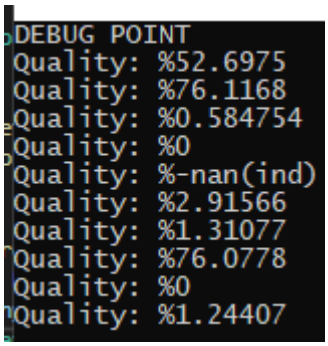


Figure 3 Bilateral FIltering with BoW 10 clusters

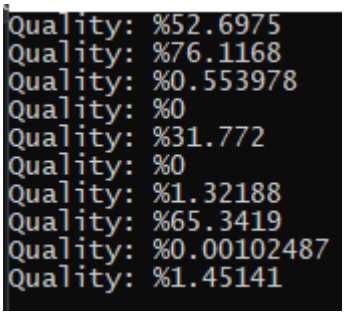


Figure 5 Bilateral Filtering with BoW 1000 clustes

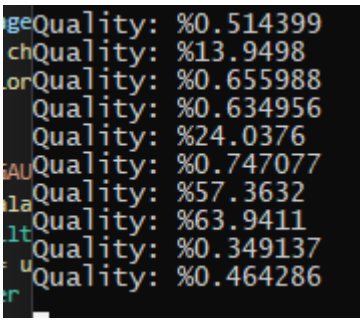


Figure 4 Gaussian FIltering with Bow 10 clusters

The Venice dataset has been trained and tested, but the results were very bad, because this kind of detection, unfortunately, is strongly dependent on the pre-processing step and the low quality images needs a total different approach on the preprocess.