Project Computer Vision: Boat Detection

Luca Scattolaro 2019157

20 July 2021

1 Introduction

The goal of this final project is to develop a system capable of detecting boats in the images. Automatic boat detection plays an important role in different fields for example maritime surveillance.
The general Idea of my solution is to use Cascade of Classifiers in order to detect all the boats and then process the results obtained using HOG descriptors and SVM in order to discard Non-Boat Bounding Boxes and use as post processing, to get better results if needed, the Canny edge detector.
In this report I will describe the reasons of the choices that I have done for my final model and at the end I will show qualitative and quantitative results on the test set.

2 Dataset

The dataset used in order to train my model is composed by images from:

- MAR Training Set: `http://www.diag.uniroma1.it//~labrococo/MAR/classification.htm`
- Kaggle Training Set: `https://www.kaggle.com/clorichel/boat-types-recognition/version/1`

I took part of a group in order to compute the annotation of the ground truth using Superannotate `https://superannotate.com/` but at the end I found some errors in the annotations so I worked with the ones shared on the forum studenti on the moodle page of the course so I used that dataset divided in Training and Test and I used as annotations the .txt files.

3 Models

To solve the Boat Detection task my final approach is to use a model composed by a combination of two Cascades of Classifiers and an SVM trained on HOG descriptors and a post processing based on Canny Edge detector in order to reach better performances on the test set.

3.1 Haar Cascade of Classifiers

In order to create my two cascades of classifiers for Boat detection I followed the guides at:
- `https://docs.opencv.org/3.4/dc/d88/tutorial_traincascade.html`
- `https://dikshit18.medium.com/training-your-own-cascade-classifier-detector-opencv-9ea6055242c2`.
The reason why I choose to use 2 different Cascade of Classifiers is basically to be sure that the union of their results contains all the boats present inside the image and also to see their different behaviour after a slightly different training phase that I will explain below.

3.1.1 First Cascade Model

I create this model using:

- Positive Images: 3116 images composed by a total of 5089 boats.
- Negative Images: about 2600 images (downloaded from internet) of different scenes like Houses, Mountains, Ocean (no boats), Buoys, Piers and so on...
- Haar features: all directions.
- 10 stages;

3.1.2 Second Cascade Model

I create this model using:

- Positive Images: 3116 images composed by a total of 5089 boats.
- Negative Images: about 2500 images.
- Haar features: all directions.
- 15 stages;

Figure : Example of 3 Negative images used for the Training of the Cascades

Training Phase
In details I created different directories and files:

- **pos** directory: contains all the positive images (images of the training set which contain boats)
- **boats.info** file: contains a line for each positive image that describe It's path and the number of Bounding Boxes of the Ground Truth and their positions inside the image.
- **neg** directory: contains all the negative images (images from internet which don't contain boats)
- **bg.txt** file: contains a line for each negative image with it's path

In order to compute the training of the second Cascade of Classifiers I used these command lines on the prompt:

- opencv_createsamples -info boats.info -num 3116 -w 48 -h 24 -vec boats.vec
- opencv_traincascade -data data -vec boats.vec -bg bg.txt -numPos 2000 -numNeg 2566 -numStages 15 -w 48 -h 24 -featureType HAAR -mode ALL

I tried to train different models for example I tried to apply some kind of pre processing for the images in the training set but the models that led to best results are the ones obtained using directly the images with no type of pre processing.

Test Phase
During the test phase I used the two Cascade of Classifiers with different hyperparameters in order to find boats of different sizes and shapes:
$boatClassifier.detectMultiScale(img, features, 2, 10, 0, Size(24, 16))$;
$boatClassifier2.detectMultiScale(img, features2, 1.7, 3, 0, Size(48, 24))$;
and I put together the Bounding Boxes found (represented by vectors features and features2).

3.2 SVM - HOG descriptors

The combination of the two cascades of classifiers led to results where often all the boats were founded but there were also too much false positives.
In order to reduce this problem I create and trained an SVM.
In order to create the dataset for the SVM I followed a precise pipeline:

1. run the first cascade of classifiers on the training set
2. for each bounding box found, check if it was good or not (contains a boat) by looking at the ground truth
3. if the Bounding Box was a false positive

- take the image inside the box, convert it into a Grayscale image
- Smooth the image and apply Canny Edge detector
- save the resulting image into a directory called **negForSVM**

4. for each image into the Real Bounding Boxes (of the ground truth) I followed the same processing as for negatives (Grayscale, Blur and Canny Edge detector) and I save the images into the directory **posForSVM**

After all these steps I obtained the training set composed by:

- Negative Images: about 20000
- Positive Images: about 15000

3.2.1 Preprocessing

To be more clear the preprocessing done to the images of my training set for SVM is composed by:

1. Convert to Grayscale: inputImgGray
2. Smoothing: I used the function $blur(inputImgGray, inputImgGray, Size(9,9))$;
3. Canny Edge Detector: $Canny(inputImgGray, inputImgGray, lowerthresh, lowerthresh * 3, 3)$;

Figure : Example of 3 Positive images of the training set used for SVM (Boats into directory posForSVM)

The only remaining thing was to perform the training.
In order to do that I calculated the HOG descriptor for each image in the training set resized to the dimension 48x48 in order to obtain 1500 features for each image. I saved all the descriptors of the images into a Matrix ($sampleFeatureMat$) and I saved all the Classification labels (1: boat, -1: No Boat) into a vector ($sampleLabelMat$).

After having computed those data I only needed to recall the training function for my SVM:
$Ptr < TrainData > td = create(sampleFeatureMat, SampleTypes :: ROW\_SAMPLE, sampleLabelMat)$;
$svm- > train(td)$;

With an SVM with the Parameters shown below:
$Ptr < SVM > svm = SVM :: create()$;
$svm- > setType(SVM :: Types :: C_SVC)$;
$svm- > setC(0.1)$;
$svm- > setKernel(SVM :: KernelTypes :: LINEAR)$;
$svm- > setTermCriteria(TermCriteria(MAX_ITER, 10000, 1e-6))$;

Then I saved the model obtained as **SVM_HOG_Canny.xml**.

3.2.2 HOG descriptor

In order to compute the HOG descriptor of an image I followed the Guide of OpenCV and I use it with this parameter:
$HOGDescriptor\ hog(Size(48,48), Size(16,16), Size(8,8), Size(8,8), 15)$;
So I specified, in order, window size, block size, block stride, cell size and number of bins.

The problem in the use of SVM is that, although I used a lot of images for the training, the model delete not only Bounding Boxes of the sea or houses and so on... but also of boats.
To reduce this problem I checked the number of Boxes Found by my Combined Model until now and If It's small then I use a Post Processing technique based on Canny edge detector.

4 Post Processing

At the end of the Machine Learning techniques applied one after the other I applied different techniques in order to get better results:

- Canny Edge detector for Object Detection. (Only if the model found until now a small number of Bounding Boxes)
- Union of similar Boxes: if they have a large intersection
- Elimination of Inner Boxes: eliminate Bounding Boxes completely inside other boxes.

4.1 Canny Edge Detector

In order to create new Bounding Boxes I followed this pipeline:

1. I blurred the image.
2. I computed Canny.
3. I found all the non black points inside the image (thanks to the threshold used they often represent boats).
4. I applied partition to the white points in order to divide different white regions.
5. I created Bounding Boxes around all the different regions found.

5 Example of the entire pipeline

Figure : Example of the entire pipeline used for Test Set

6 Results

In this section I will report the results obtained for all the Images on the Test set with both qualitative and quantitative performances.
The images below are composed by on the left the result obtained with my final Model and on the right the Ground Truth.
Into the results the colors shown for the boxes represent the goodness of the box with respect to the ground truth:

- Green: good box with IoU over 0.6
- Yellow: box with IoI from 0.3 to 0.6
- Orange: box with IoU under 0.3
- Red: False positive box

For the first three types you can also see the value of the IoU drawn at the top left of the box.

6.1 Kaggle Test Set

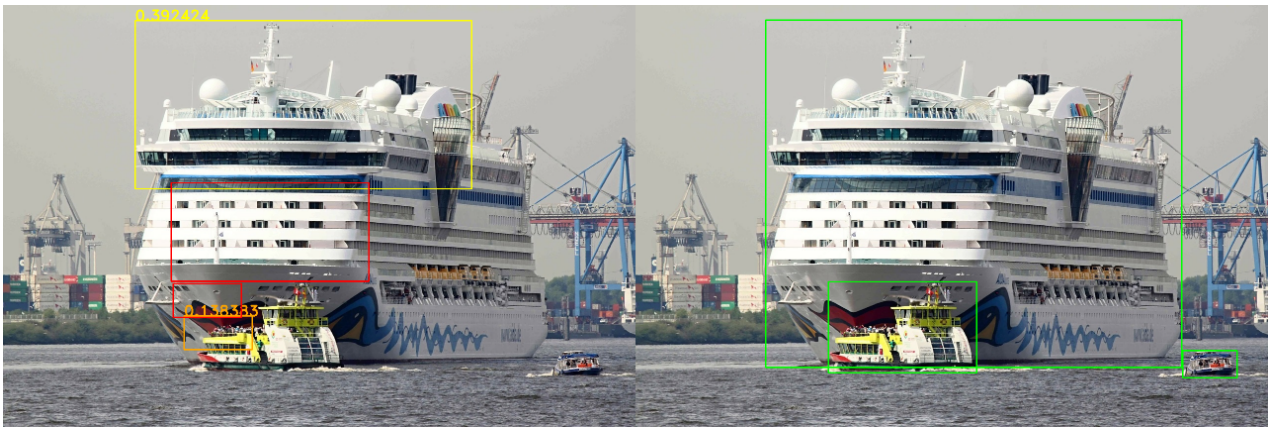

Image: 01.jpg
Bounding Boxes Found By Model: 4
Bounding Boxes Ground Truth : 3
Boats Found:
- IoU: 0.392424
- IoU: 0.138383
Missing Boats: 1
False Positive: 2

Image: 02.jpg
Bounding Boxes Found By Model: 1
Bounding Boxes Ground Truth : 1

Boats Found:
- IoU: 0.875588
Missing Boats: 0
False Positive: 0

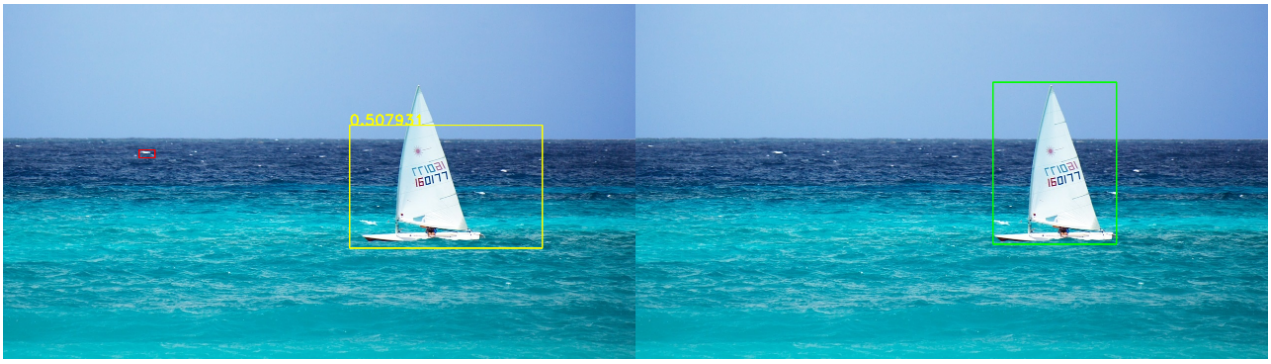

Image: 03.jpg
Bounding Boxes Found By Model: 2
Bounding Boxes Ground Truth : 1
Boats Found:
- IoU: 0.507931
Missing Boats: 0
False Positive: 1

Image: 04.jpg
Bounding Boxes Found By Model: 2
Bounding Boxes Ground Truth : 2
Boats Found:
- IoU: 0.828068
Missing Boats: 1
False Positive: 1

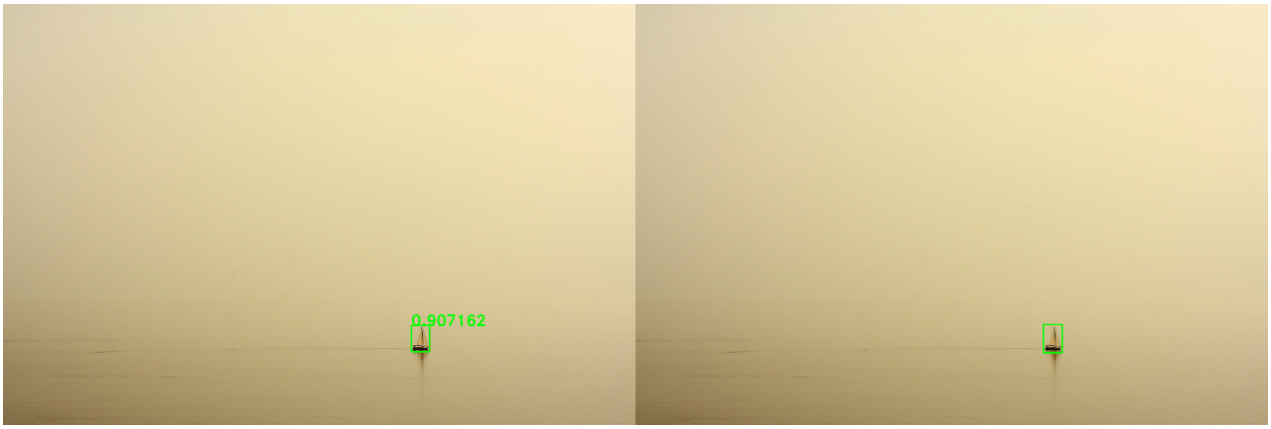

Image: 05.jpg
Bounding Boxes Found By Model: 1
Bounding Boxes Ground Truth : 1
Boats Found:
- IoU: 0.907162
Missing Boats: 0
False Positive: 0

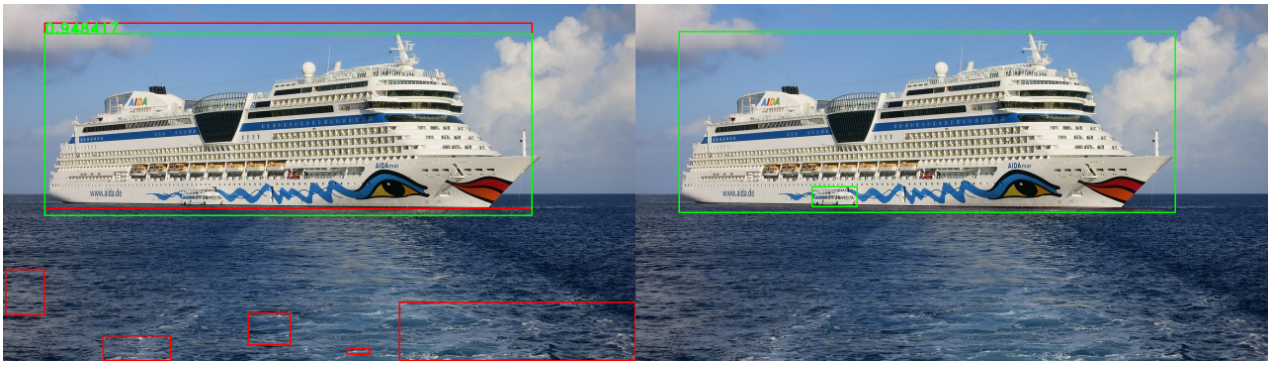

Image: 06.jpg
Bounding Boxes Found By Model: 7
Bounding Boxes Ground Truth : 2
Boats Found:
- IoU: 0.948417
Missing Boats: 1
False Positive: 6

Image: 07.jpg
Bounding Boxes Found By Model: 2
Bounding Boxes Ground Truth : 2
Boats Found:
- IoU: 0.865697
- IoU: 0.838026
Missing Boats: 0
False Positive: 0

Image: 08.jpg
Bounding Boxes Found By Model: 3
Bounding Boxes Ground Truth : 3
Boats Found:
- IoU: 0.744780
- IoU: 0.502244
- IoU: 0.492958
Missing Boats: 0
False Positive: 0

Image: 09.jpg
Bounding Boxes Found By Model: 4
Bounding Boxes Ground Truth : 1
Boats Found:
- IoU: 0.610255
Missing Boats: 0
False Positive: 3

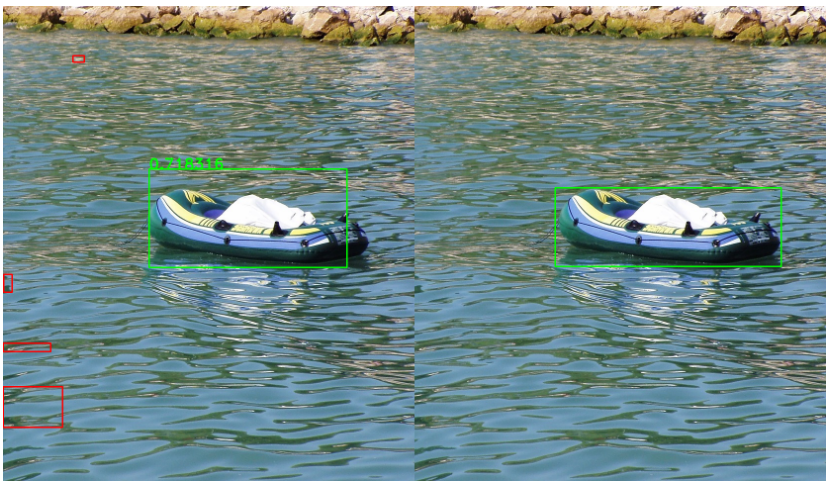

Image: 10.jpg
Bounding Boxes Found By Model: 5
Bounding Boxes Ground Truth : 1
Boats Found:
- IoU: 0.718316
Missing Boats: 0
False Positive: 4

6.2 Venice Test Set

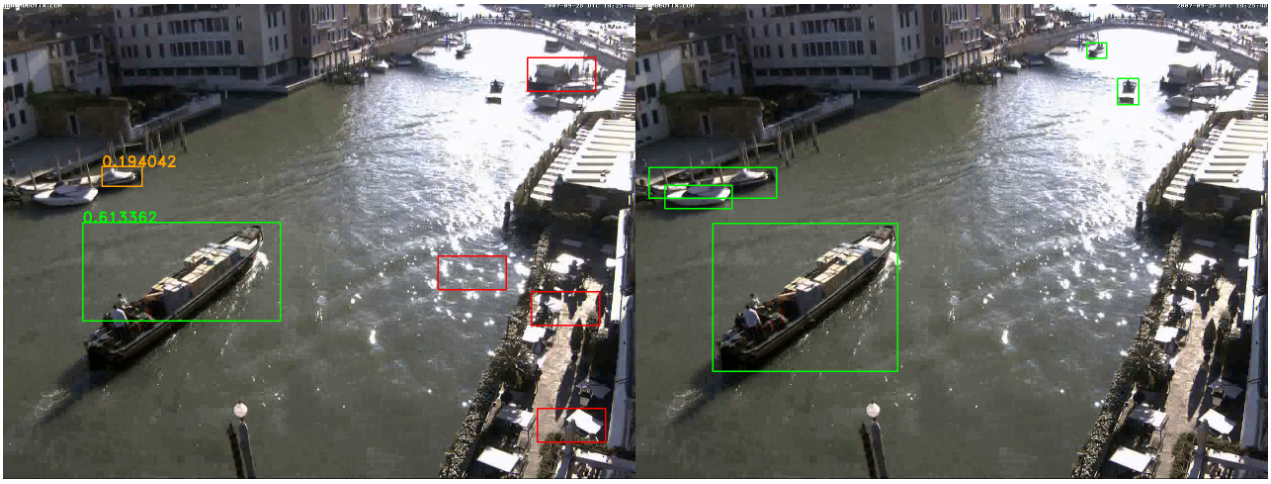

Image: 00.png
Bounding Boxes Found By Model: 6
Bounding Boxes Ground Truth : 5
Boats Found:
- IoU: 0.613362
- IoU: 0.194042
Missing Boats: 3
False Positive: 4

Image: 01.png
Bounding Boxes Found By Model: 2
Bounding Boxes Ground Truth : 4
Boats Found:
- IoU: 0.218529
Missing Boats: 3
False Positive: 1

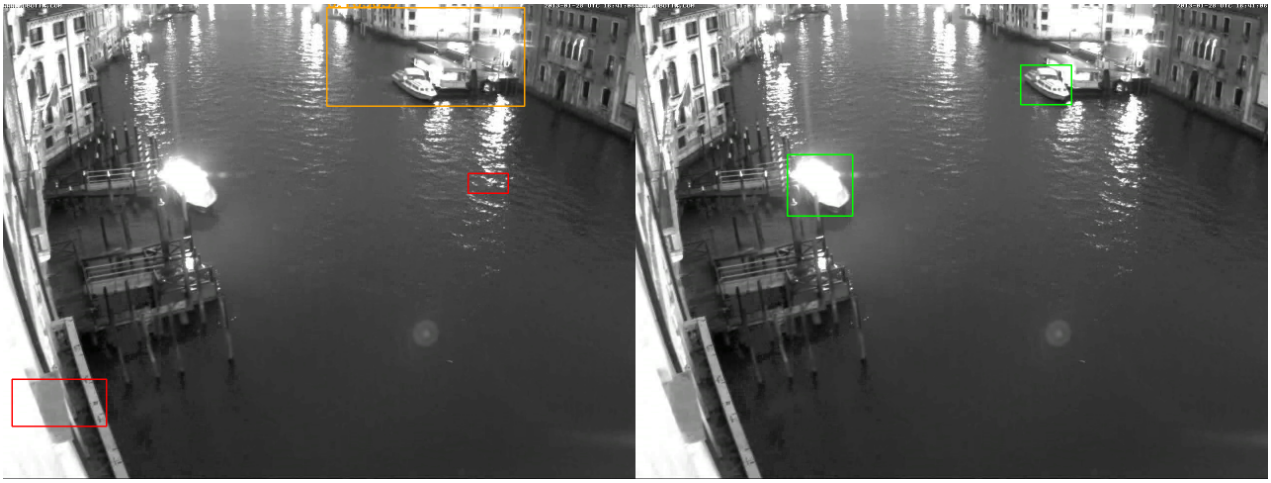

Image: 02.png
Bounding Boxes Found By Model: 3
Bounding Boxes Ground Truth : 2
Boats Found:
- IoU: 0.105037
Missing Boats: 1
False Positive: 2

Image: 03.png
Bounding Boxes Found By Model: 2
Bounding Boxes Ground Truth : 2
Boats Found:
- IoU: 0.309531
Missing Boats: 1
False Positive: 1

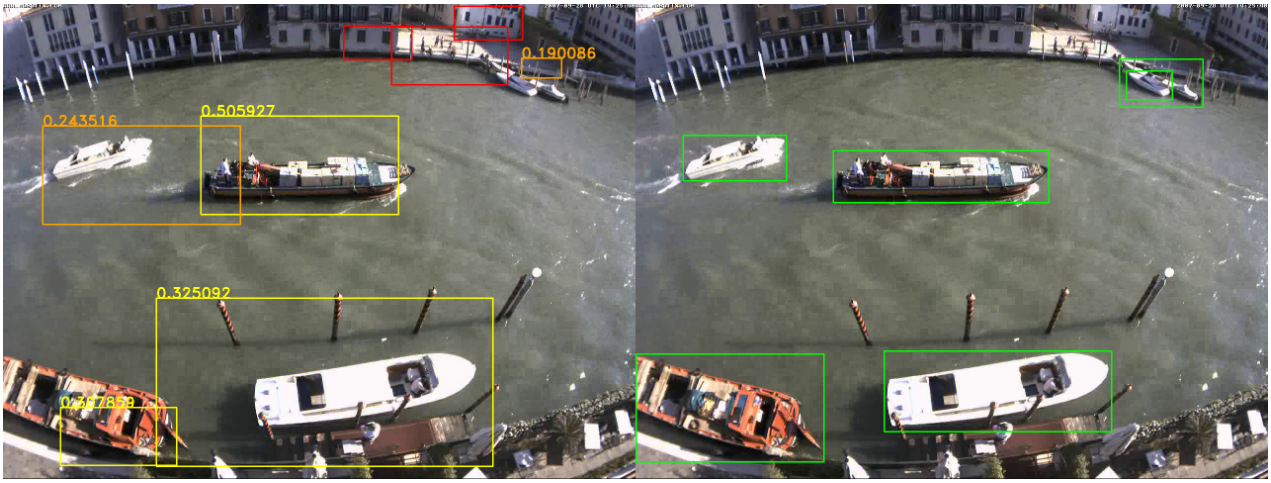

Image: 04.png
Bounding Boxes Found By Model: 8
Bounding Boxes Ground Truth : 6
Boats Found:
- IoU: 0.325092
- IoU: 0.307859
- IoU: 0.243516
- IoU: 0.505927
- IoU: 0.190086
Missing Boats: 1
False Positive: 3

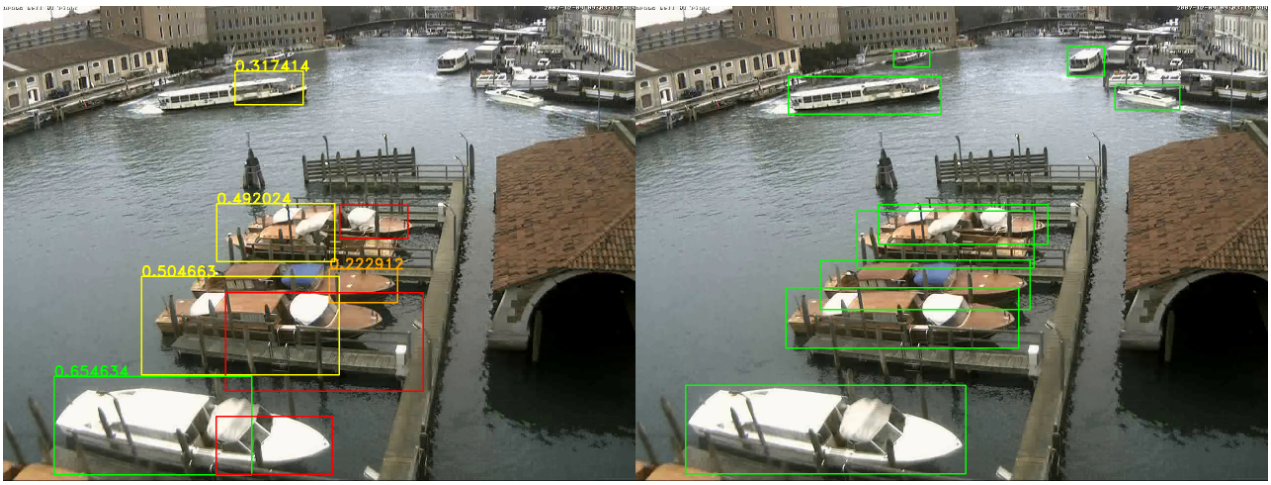

Image: 05.png
Bounding Boxes Found By Model: 8
Bounding Boxes Ground Truth : 9
Boats Found:
- IoU: 0.317414
- IoU: 0.654634
- IoU: 0.504663
- IoU: 0.222912
- IoU: 0.492024
Missing Boats: 4
False Positive: 3

Image: 06.png
Bounding Boxes Found By Model: 6
Bounding Boxes Ground Truth : 23
Boats Found:
- IoU: 0.037412
- IoU: 0.127342
- IoU: 0.402249
- IoU: 0.016508
- IoU: 0.030688
Missing Boats: 18
False Positive: 1

Image: 07.png
Bounding Boxes Found By Model: 4
Bounding Boxes Ground Truth : 5
Boats Found:
- IoU: 0.319850
- IoU: 0.626431
Missing Boats: 3
False Positive: 2

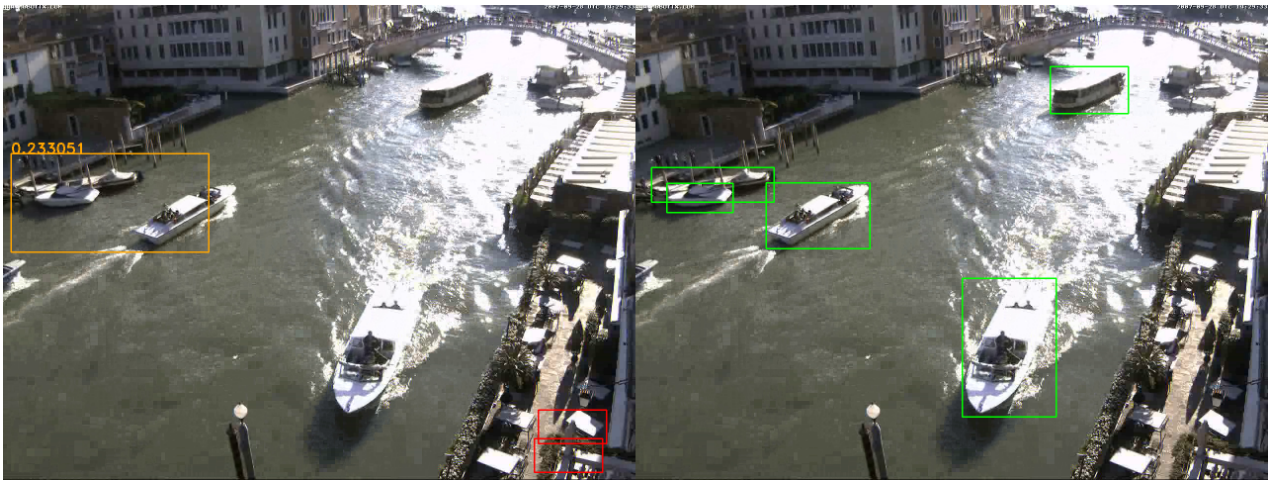

Image: 08.png
Bounding Boxes Found By Model: 3
Bounding Boxes Ground Truth : 5
Boats Found:
- IoU: 0.233051
Missing Boats: 4
False Positive: 2

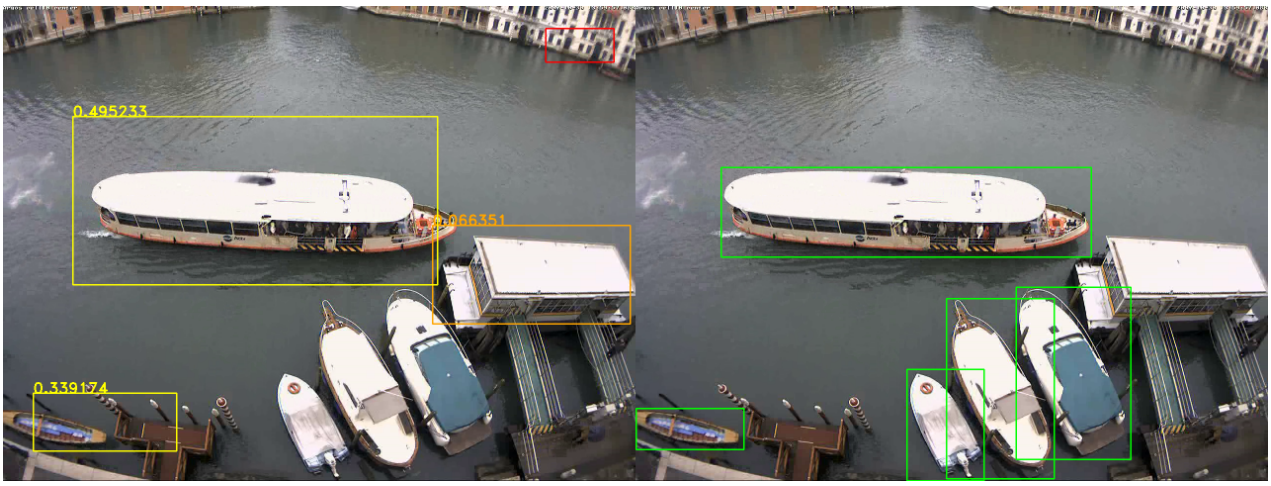

Image: 09.png
Bounding Boxes Found By Model: 4
Bounding Boxes Ground Truth : 5
Boats Found:
- IoU: 0.339174
- IoU: 0.495233
- IoU: 0.066351
Missing Boats: 2
False Positive: 1

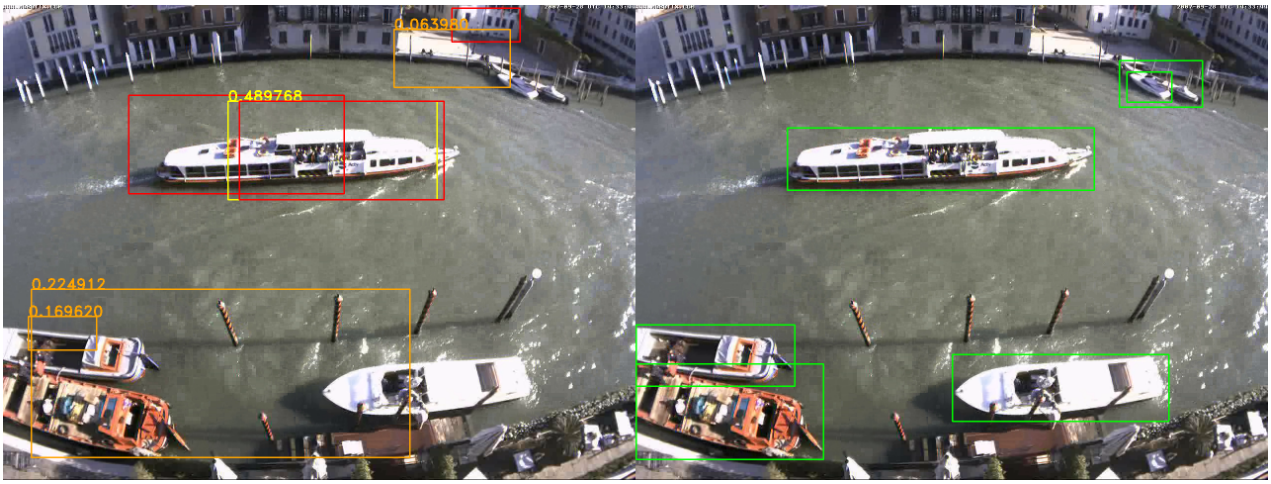

Image: 10.png
Bounding Boxes Found By Model: 7
Bounding Boxes Ground Truth : 6
Boats Found:
- IoU: 0.224912
- IoU: 0.169620
- IoU: 0.489768
- IoU: 0.063980
Missing Boats: 2
False Positive: 3

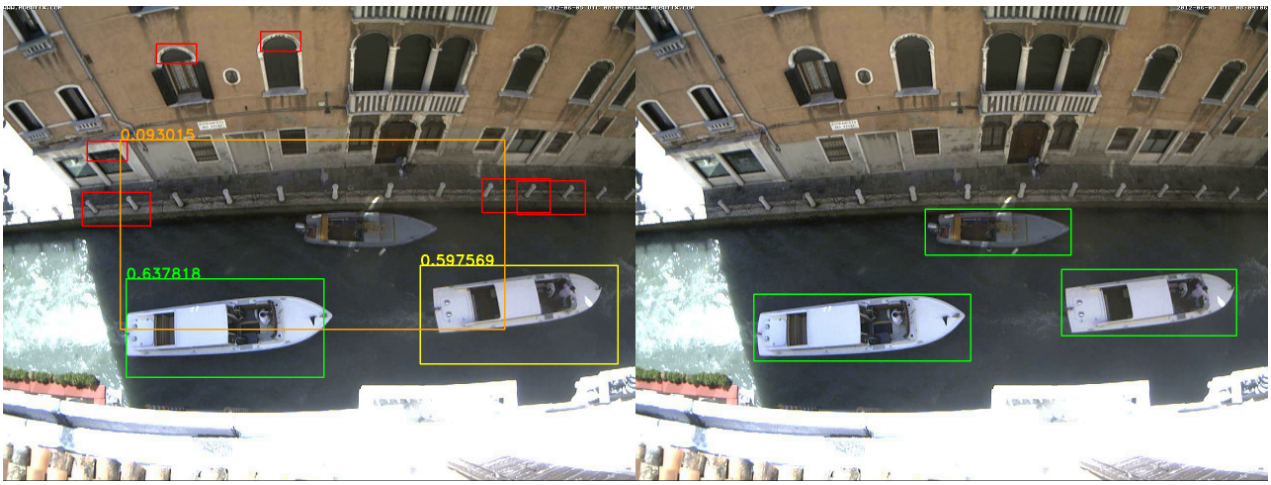

Image: 11.png
Bounding Boxes Found By Model: 9
Bounding Boxes Ground Truth : 3
Boats Found:
- IoU: 0.637818
- IoU: 0.597569
- IoU: 0.093015
Missing Boats: 0
False Positive: 6

7 Observations and Drawbacks

As we can see from the results above the final model works really good with some images (especially some of the Kaggle test set) and on the other hand It works really bad for example in some Venice test set images that have a lot of boats or they are not really visible or distinguishable with respect to houses and piers.
Another problem related with the Training phase of the Cascade of Classifiers is that are necessary some libraries that are no more present in the versions of OpenCV4 so I needed to change version and install OpenCV3.
Another problem is the high **Intra class Variance**: different size, shape, colors and structure of boats.
I think that in order to get better results It would be useful to train also a single cascade of classifiers but with a lot more images (both positive and negative) and for a lot more stages.
Unfortunately It has not been possible for different reasons:

- Lack of time: to create the annotation for a lot more positive images.
- Hardware limitations: for the training phase of the cascade of classifiers with only 10 stages and about 3116 positive and 2500 negatives it took 15 hours in my computer.

8 Program Organization

In this section I will explain the reason why of my code organization for this project.
The entire code is composed by:

- **project.cpp**: main where I organized the code to show the Menù to the user
- **datasetPreparation-utils** class: this is used just to store methods used for the preparation of the datasets for both the cascades of Classifiers and also the SVM (in this way the main is more clear)
- **BoundingBox** class: BoundingBox.h, BoundingBox.cpp
  I used this class that is similar to Rect but obviously with more functions and constructors useful for the Boat detection task.
- **BoatImage** class: BoatImage.h, BoatImage.cpp
  I used this class to contain an image and all the Bounding Boxes drawn on it.

I create It basically because It has some methods that made the code looks better and more clear.
It also enable the comparison between an output image of my model and the ground truth.

- **CombinedModel** class: CombinedModel.h, CombinedModel.cpp
  This class contains all the methods and constructors in order to be able to create and use a Combined-Model with the same composition of sub models described above.

9 Conclusions

I'm pretty happy with the results obtained during all the phases of the developing of project.
Obviously my model It's not comparable with the state of the art techniques based on deep learning but for sure It has been useful to learn more about some topics studied in the theory of the course for example HOG descriptor, Cascade of classifiers, Canny edge detector, Smoothing filters and SVM.