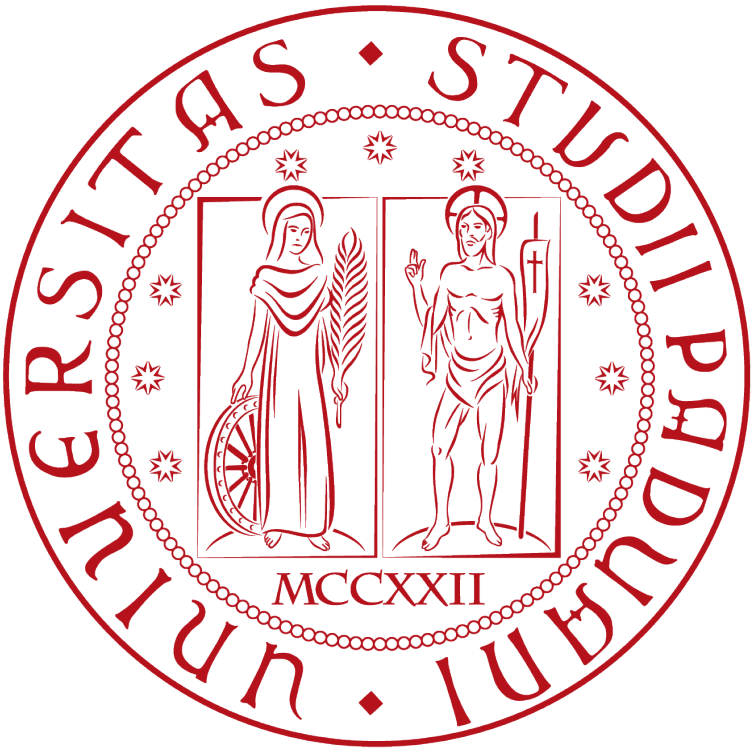
Computer Vision
Final Project Report

Roberto Carta , ID: 2025883

July 23, 2021



1 Automatic Boat Detection

The purpose of this project was to write a program able to detect boats in a given set of images.
This can be useful in real world scenarios such as maritime surveillance.
To accomplish this task the chosen approach was a cascade classifier which uses the method proposed by Paul Viola and Michael Jones in their paper, "Rapid Object detection using a Boosted Cascade of Simple Features" in 2001.
The whole work lasted 120 hours.

1.1 Training of the classifier

The used classifier was trained using the OpenCV tool called `opencv_traincascade.exe` and saved in a `.xml` file. A training dataset of 1000 positive sample images including boats of several types was used. Also a set of 2000 negative images was used to train the classsifier: this includes images of sea, of buoys and of buildings.
After some research it was found that a number of stages greater than 16 could lead to overfitting, so it was chosen to use 12 stages as a good trade-off.
The minimum height and width of the detected object was set to 24.
The classifier was trained using the HAAR features.

1.2 Preprocessing

It was chosen to work only with black and white images.
The classifier itself is not able to detect all the boats in the image so some preprocessing could improve the detection. A combination of Gaussian smoothing for denoising and edge sharpening with Canny algorithm was used, but sometimes improved the detection, sometimes not, so at the end it was discarded.
Since the model proposed by Viola and Jones is sensisitive to illumination changes, it was tried to enhance the brightness by summing the mean luminance of the image. With this approach better results were obtained, but they depended from image to image.
At the end it was chosen to not use any kind of preprocessing, since with the approaches described above the number of false negative was too high.

1.3 Detection

The classifier was loaded from the `.xml` file and a Cascade Classifier object was created. To perform the detection the function `detectMultiScale()` was used. If no boats are detected in the image, the text "No boats found" is returned.
The detection is not perfect, indeed many false positives were found. Above all, in the dataset containing Venice boats images, the classifier could detect also the windows of the surrounding buildings(see Figure below): this is due to the fact that those windows have a shape very similar to the one of a boat seen from above.
Without any kind of post processing the bounding boxes returned from the classifier would have resulted like in the Figure below.

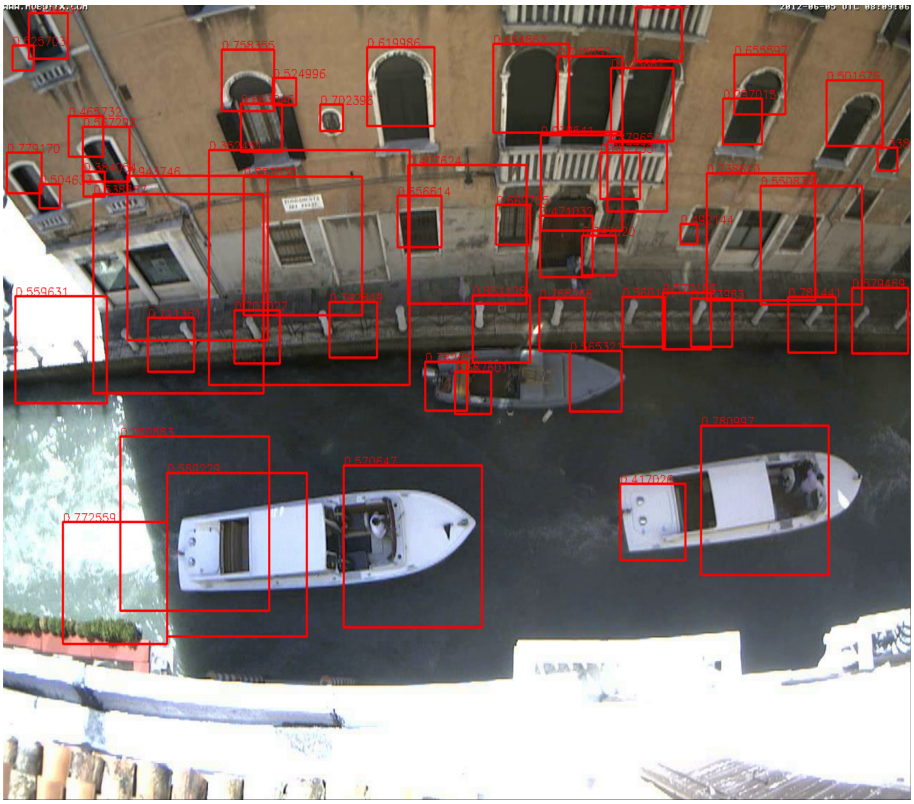


Figure 1: Many windows and other objects are detected and classified as boats. The number of returned bounding boxes is too high.

1.4 Post Processing

During the post processing stage, a clustering operation was performed over all the found bounding boxes. The function `partition()` has helped in doing so: boxes which overlapped and boxes whose centers were distant within a certain threshold were grouped in the same cluster. Some empirical threshold have been found and they worked fine. At the end all the boxes belonging to the same cluster have been merged.
However, doing so, there were still boxes containing false positives. To prune the false positives a few more steps were performed. Firstly an average smoothing was made by means of the function `blur()` to denoise the image and to delete some details. Then the edges were found with `Canny()` function: the choice was to set $lowThreshold = 100$ and $highThreshold = 200$. At the end the edges were subtracted to the smoothed image (to get a sharpened image) and the detection was made inside every box: if a box didn't contain a boat, then it was discarded. There were still some boats identified by 2 or more boxes, since this algorithm is not able to merge them all.
In the figures below some comparisons between the detection before and after the post processing are shown

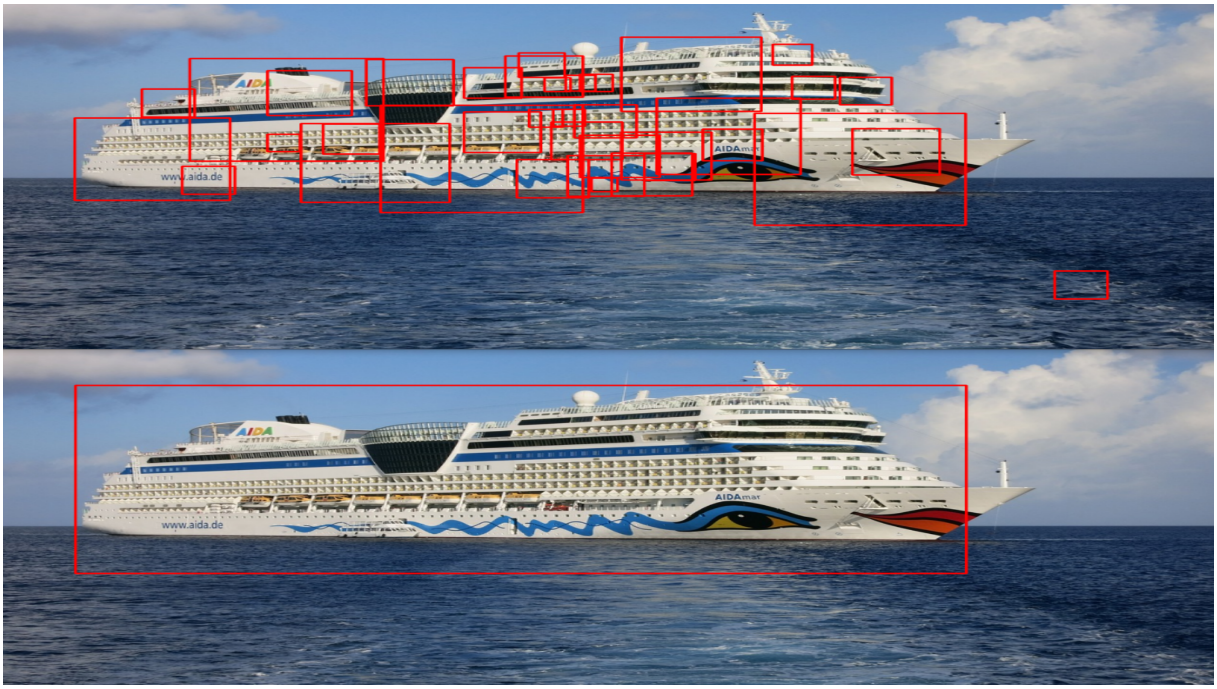


Figure 2: In the Kaggle dataset the post processing worked fine for most of the images

Figure 3: Before and after post processing of an image belonging to Kaggle dataset

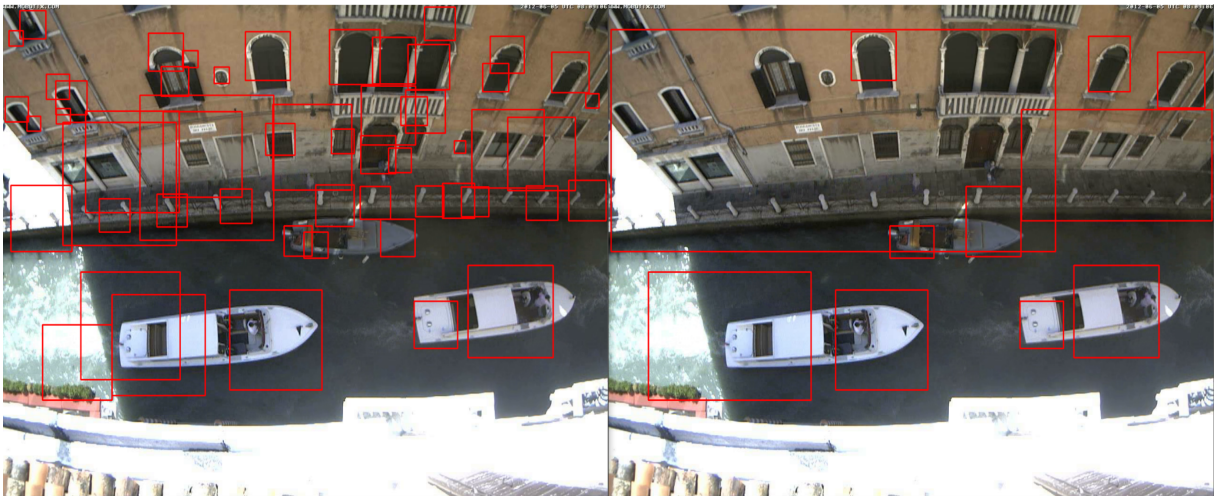


Figure 4: In the Venice dataset there are still false positives and boats detected by 2 or more boxes

1.5 Evaluation of the accuracy

The adopted evaluation method is the Intersection over Union metrics (IoU) between the bounding boxes returned by the algorithm for a single boat and the ground truth ones. The problem of having multiple boats in the image is how to associate each found box to the right one of the ground truth. To overcome this issue the IoU is computed in 3 steps: first check if the two boxes overlap; second check if the areas of the boxes are comparable (the smaller one must be at least 15% of the bigger one); finally calculate the IoU. Otherwise a found box has automatically IoU = 0. A final image containing the comparison between the ground truth boxes and the ones returned by the algorithm is shown.

1.6 Conclusions and Issues

In the Kaggle dataset the algorithm performs in general well while in the Venice dataset there are still many boats which can't be detected.
However many rotated boats, not seen from the side could not be detected since the Viola Jones algorithm is sensitive to rotations.
Since this algorithm is sensitive also to illumination changes, the detection could change drastically if some preprocessing on the brightness were done.

2 Results

Below the comparison between the found boxes(in red) and the ground truth ones (in green). The images reported are the only ones where at least one boat has been detected.
Also the IoU of every box is written.

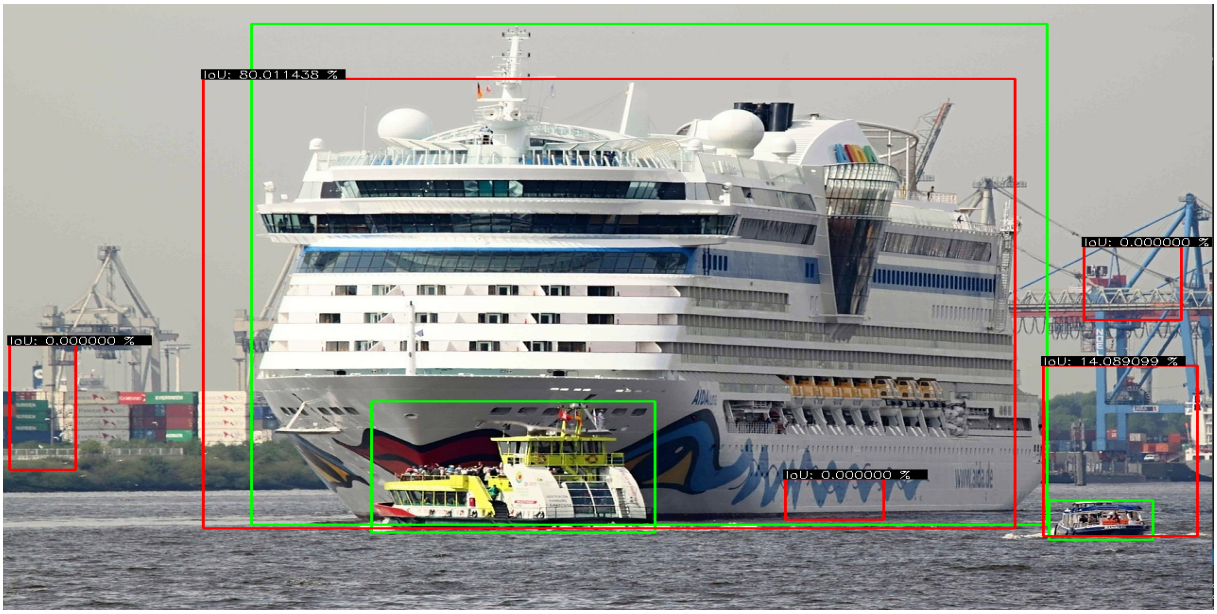


Figure 5: Found Boxes: 5. IoU : 0% 80% 0% 14% 0%

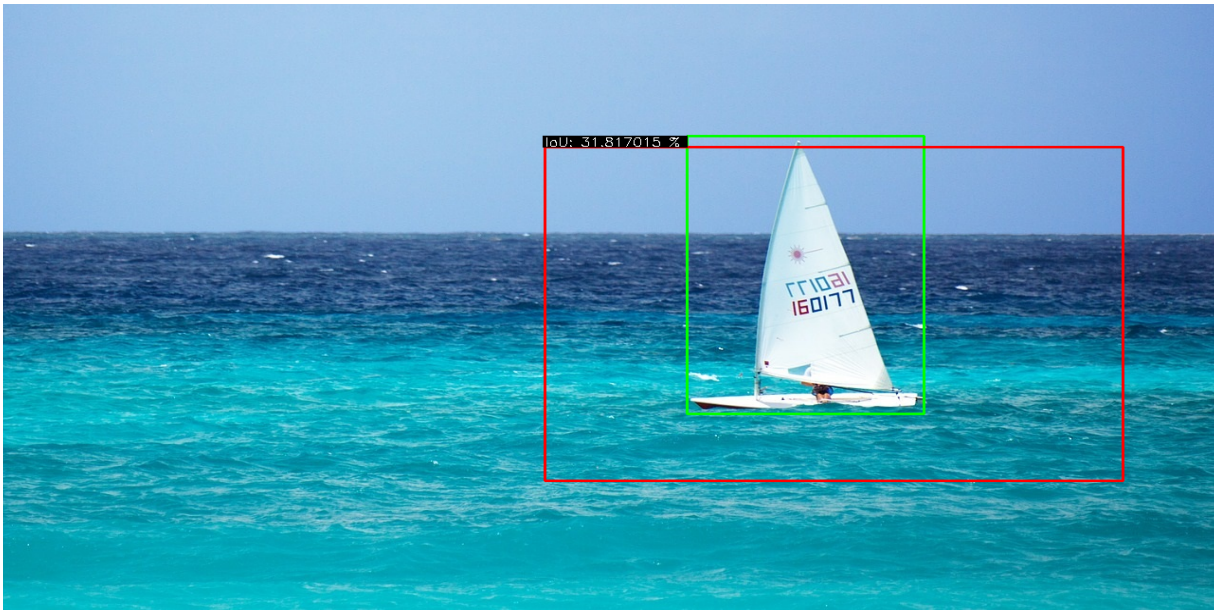


Figure 6: Found Boxes: 1. IoU : 31%

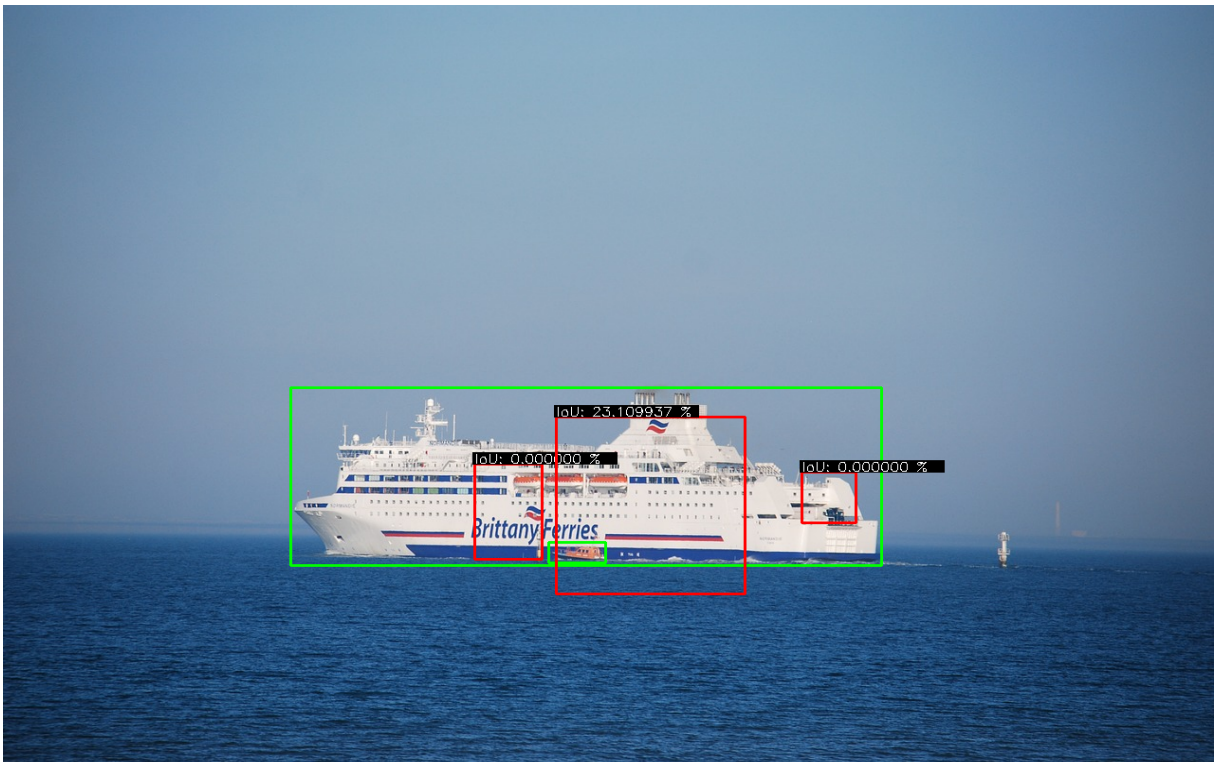


Figure 7: Found Boxes: 3. IoU : 0% 23% 0%

Figure 8: Found Boxes: 1. IoU : 60%

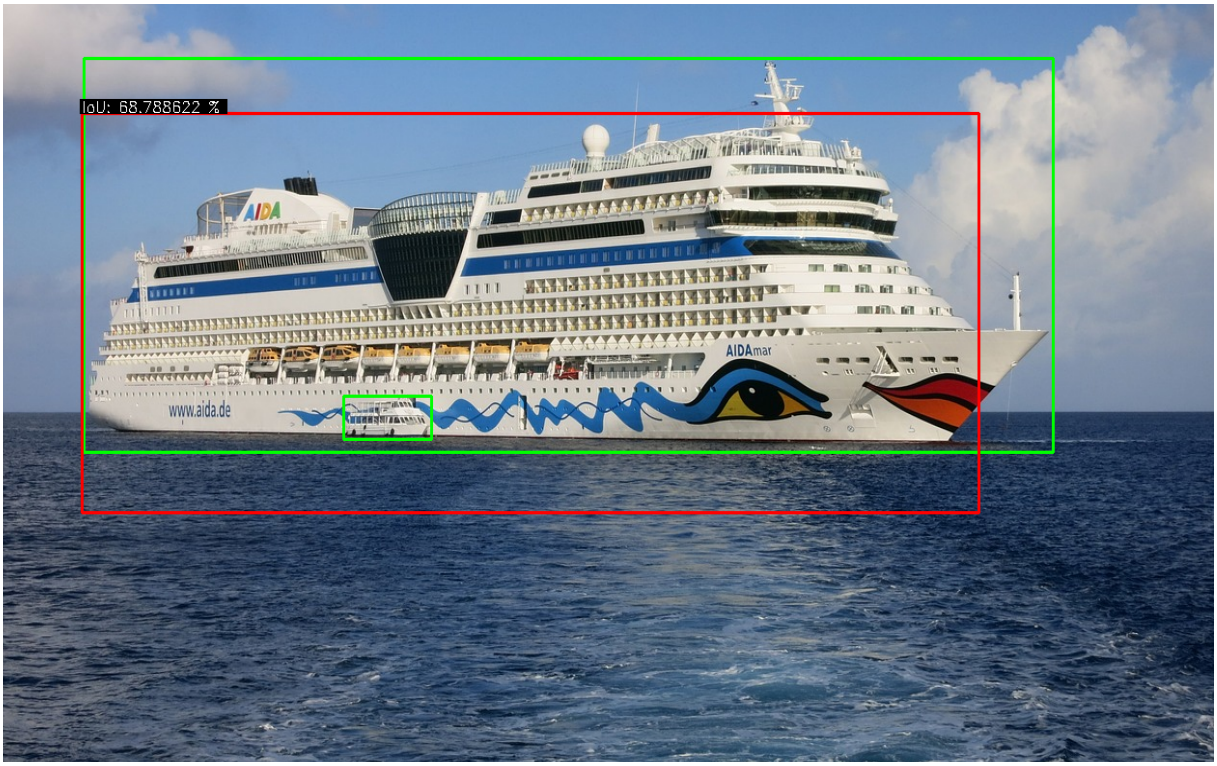


Figure 9: Found Boxes: 1. IoU : 68%

Figure 10: Found Boxes: 2. IoU : 45% 61%

Figure 11: Found Boxes: 1. IoU : 74%

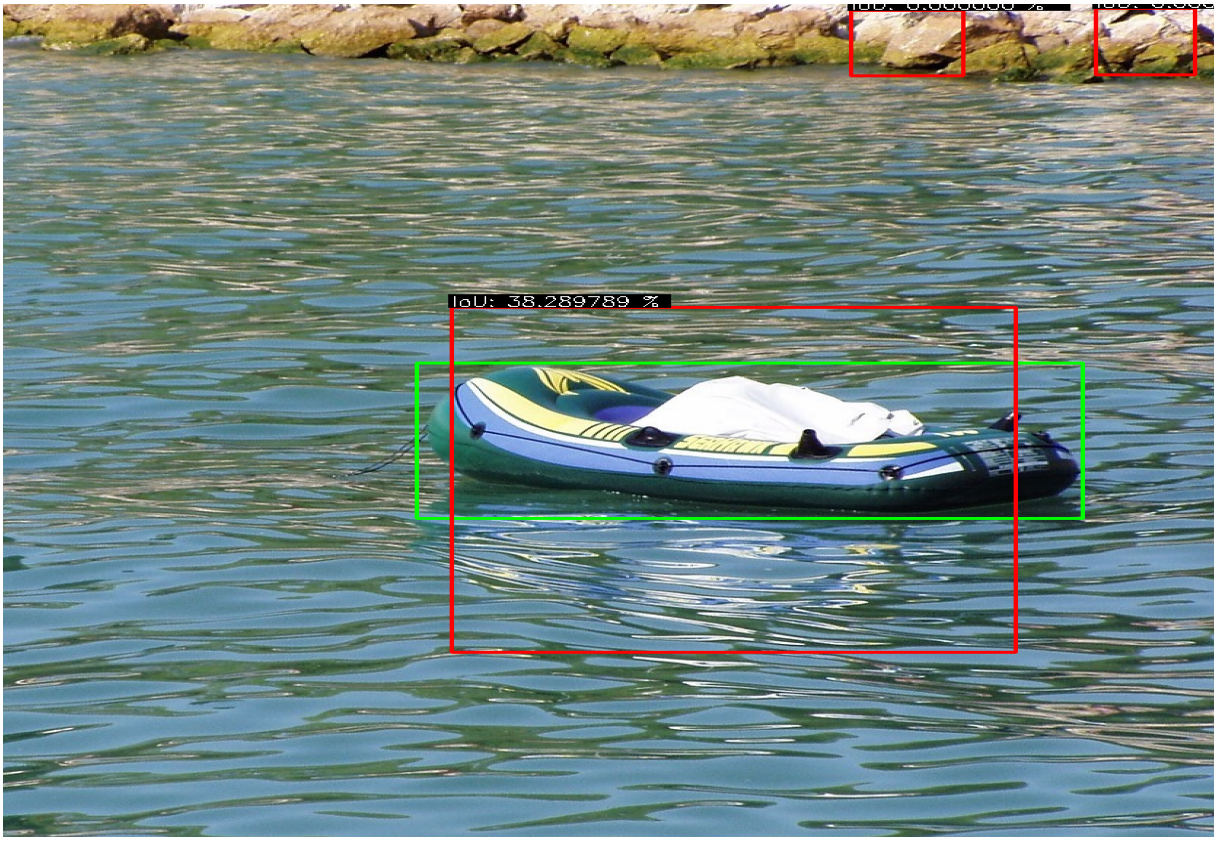


Figure 12: Found Boxes: 3. IoU : 0% 0% 38%

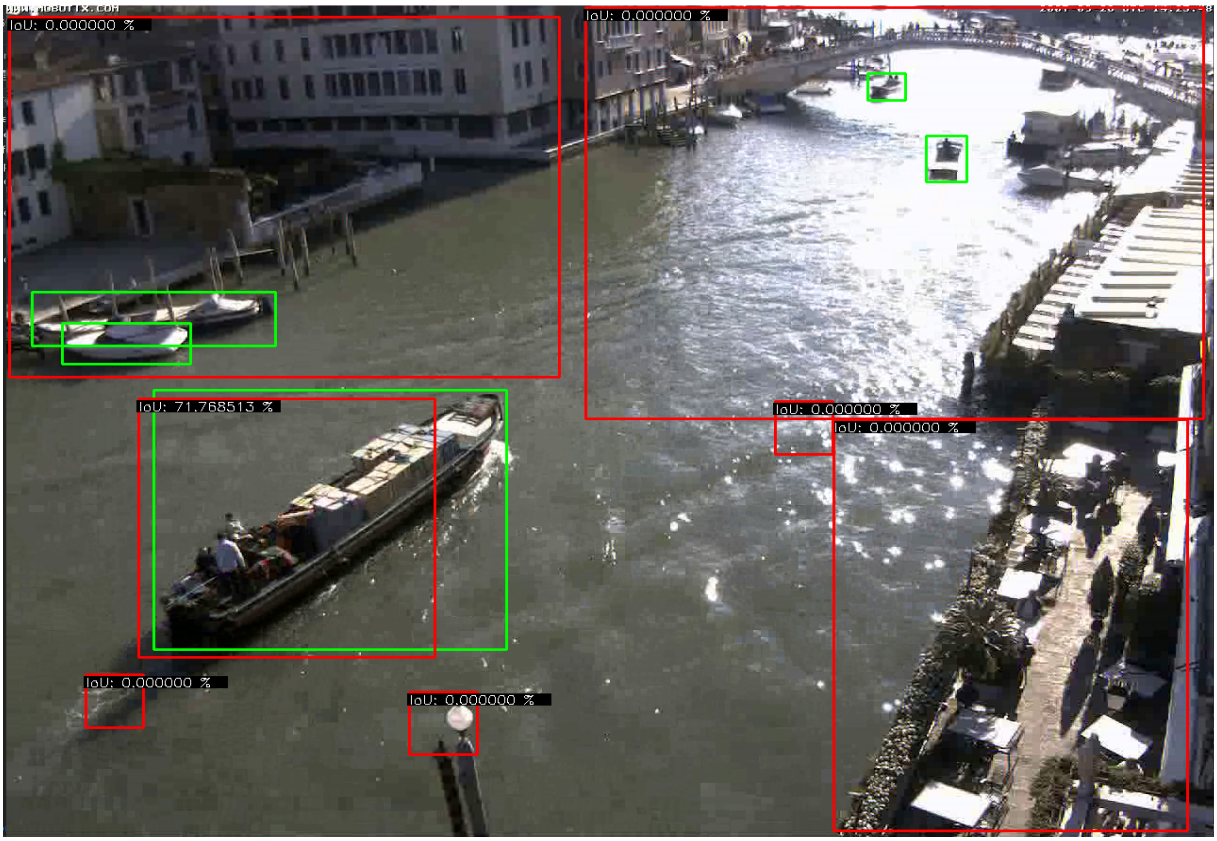


Figure 13: Found Boxes: 7. IoU : 0% 0% 71% 0% 0% 0% 0%

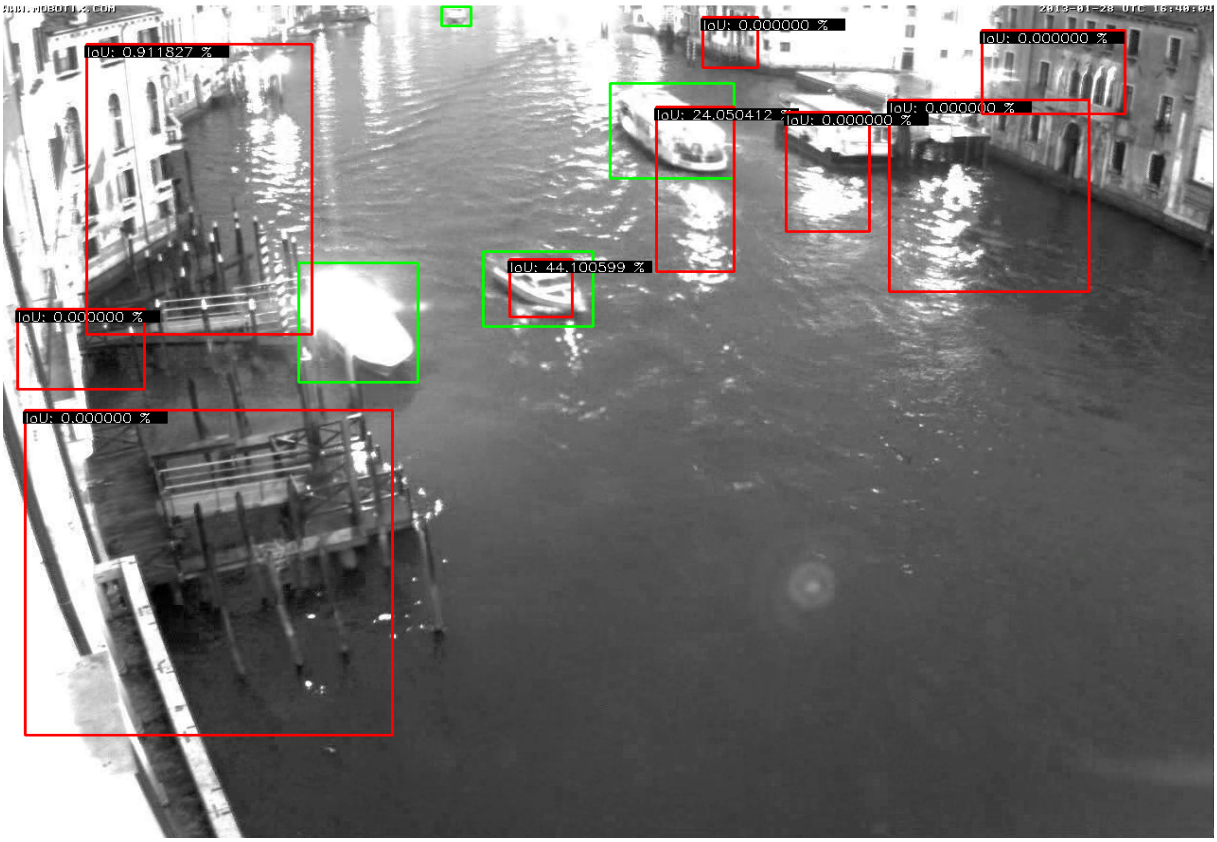


Figure 14: Found Boxes: 9. IoU : 0% 0% 0% 44% 24% 0% 0% 0% 0%

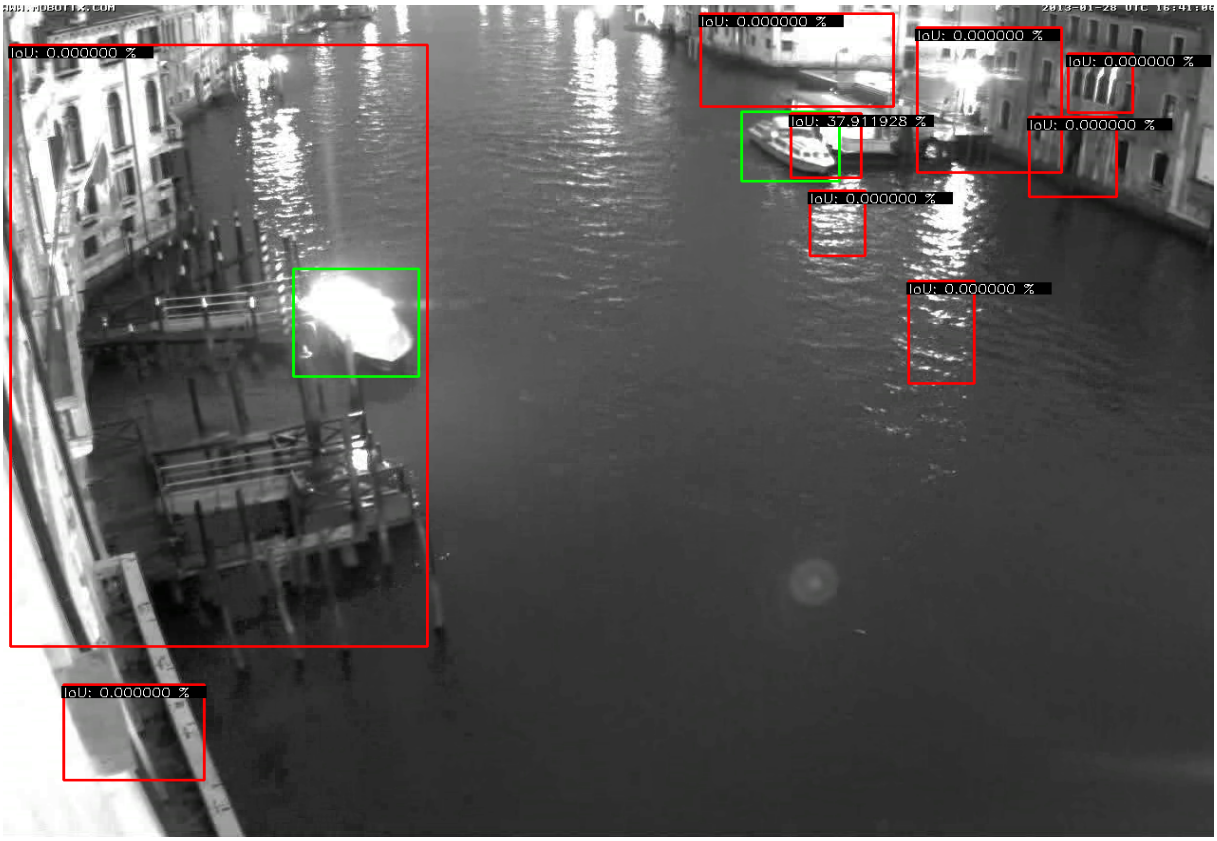


Figure 15: Found Boxes: 9. IoU : 0% 0% 0% 37.9% 0% 0% 0% 0% 0%

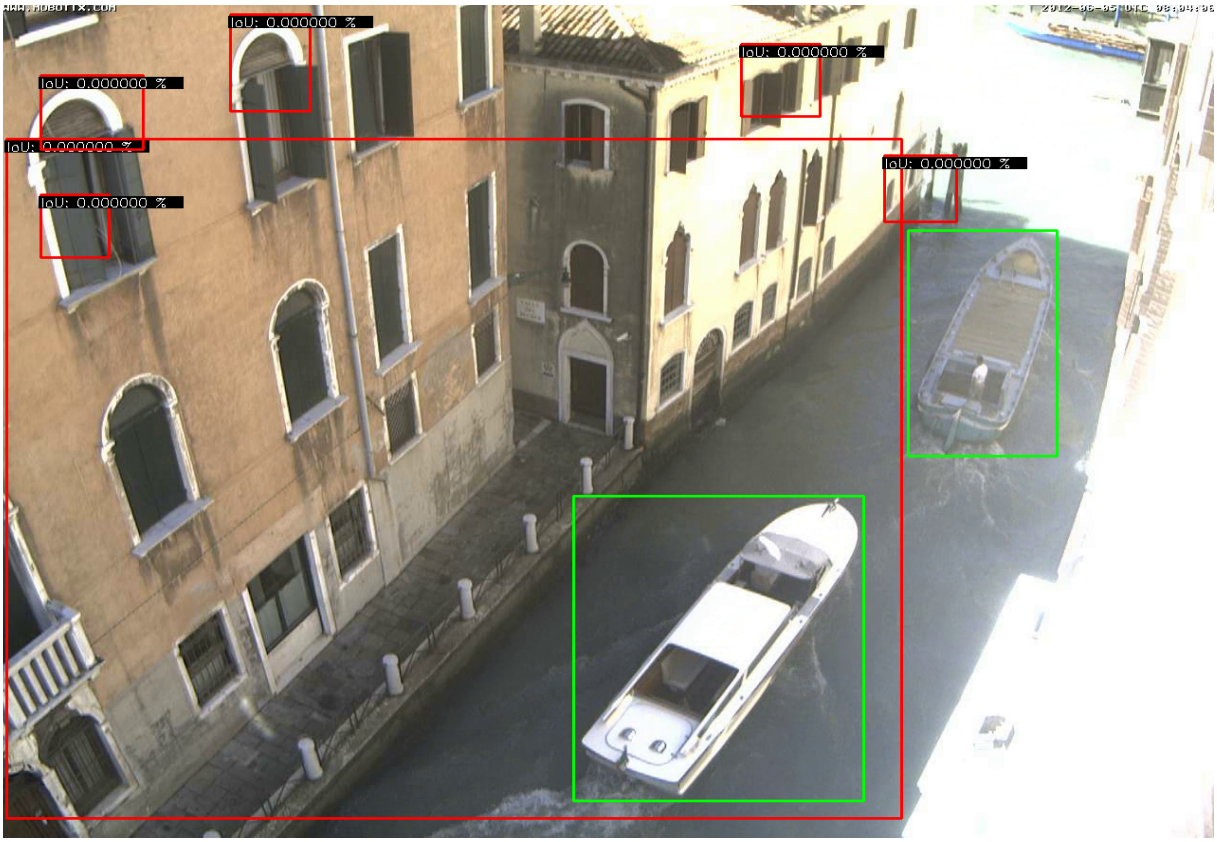


Figure 16: Found Boxes: 6. IoU : 0% 0% 0% 0% 0% 0%

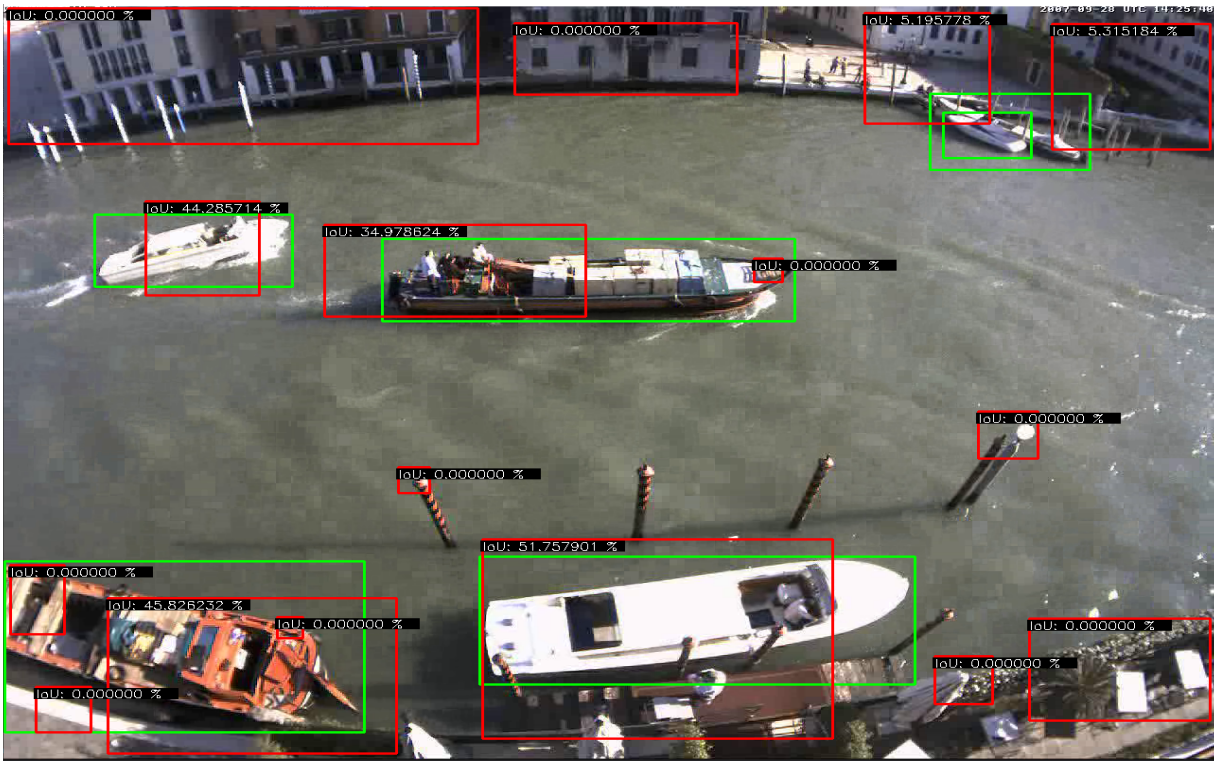


Figure 17: Found Boxes: 15. IoU : 0% 44% 0% 0% 45% 0% 34% 0% 51% 0% 5% 5% 0% 0% 0%

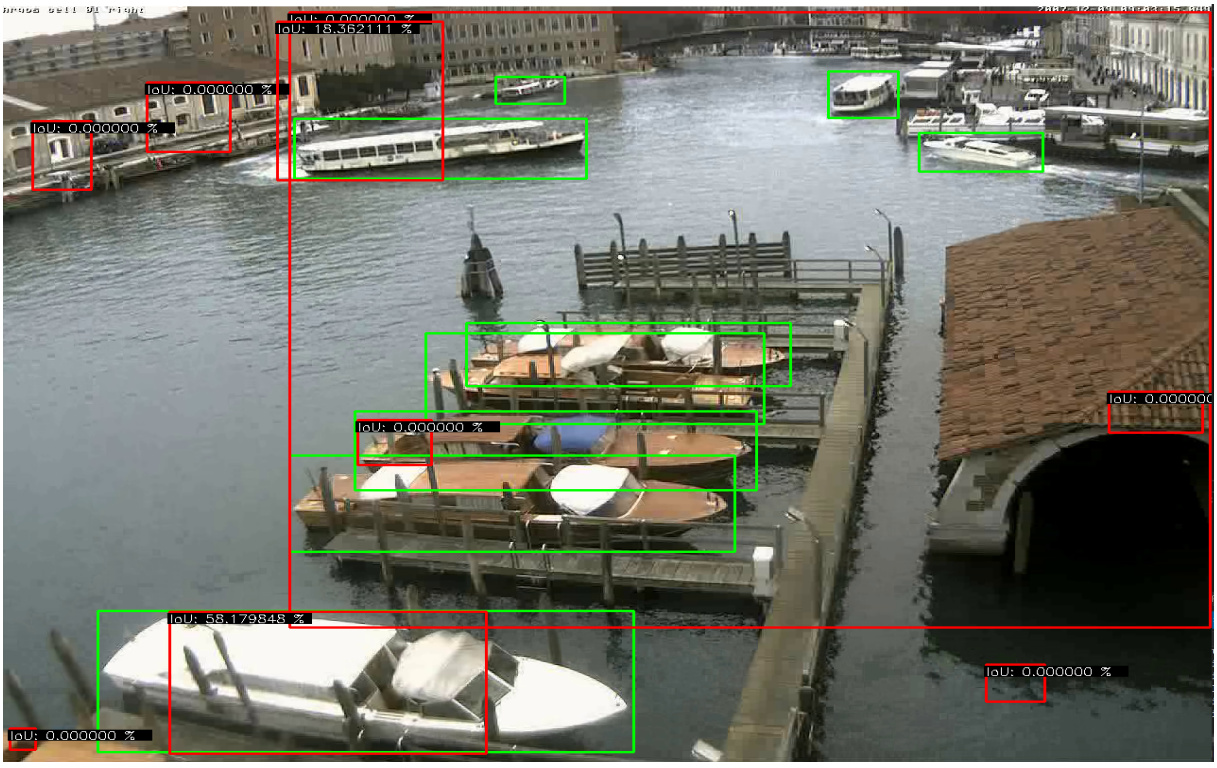


Figure 18: Found Boxes: 9. IoU : 0% 0% 0% 58% 0% 18% 0% 0%

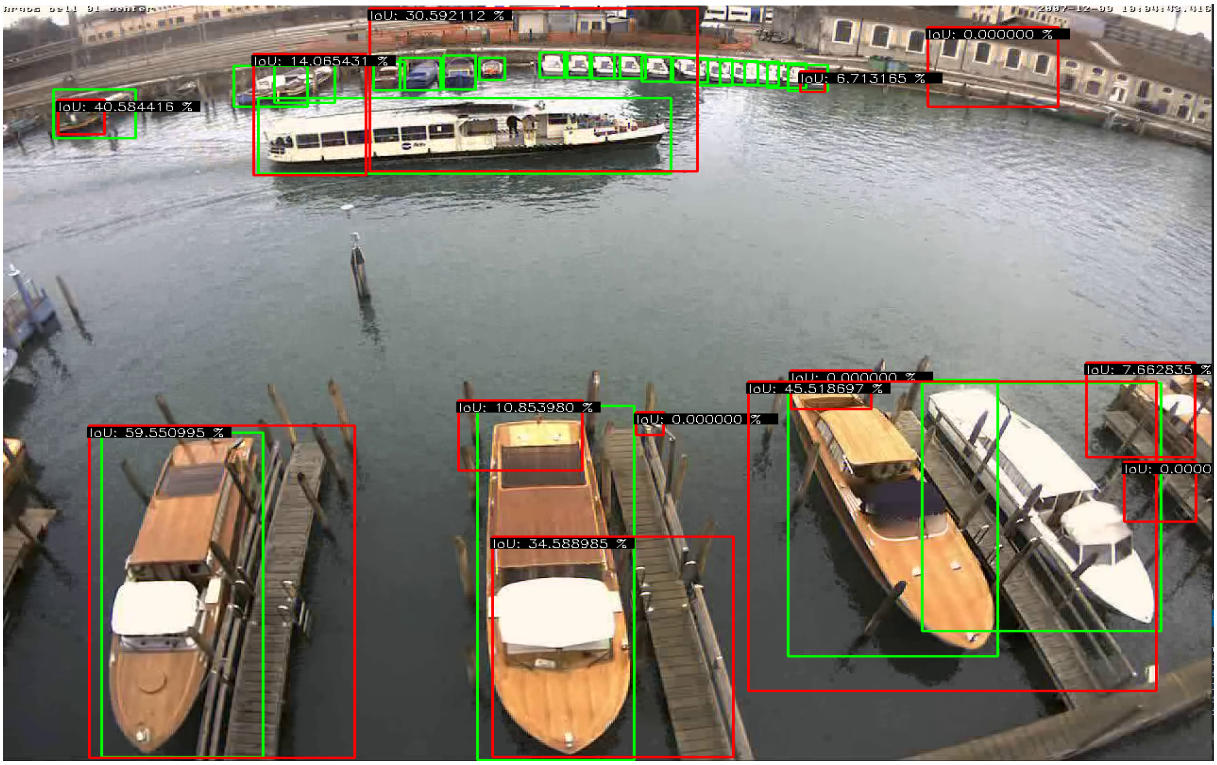


Figure 19: Found Boxes: 13. IoU : 40% 59% 14% 30% 10% 34% 0% 6% 0% 45% 0% 7% 0%

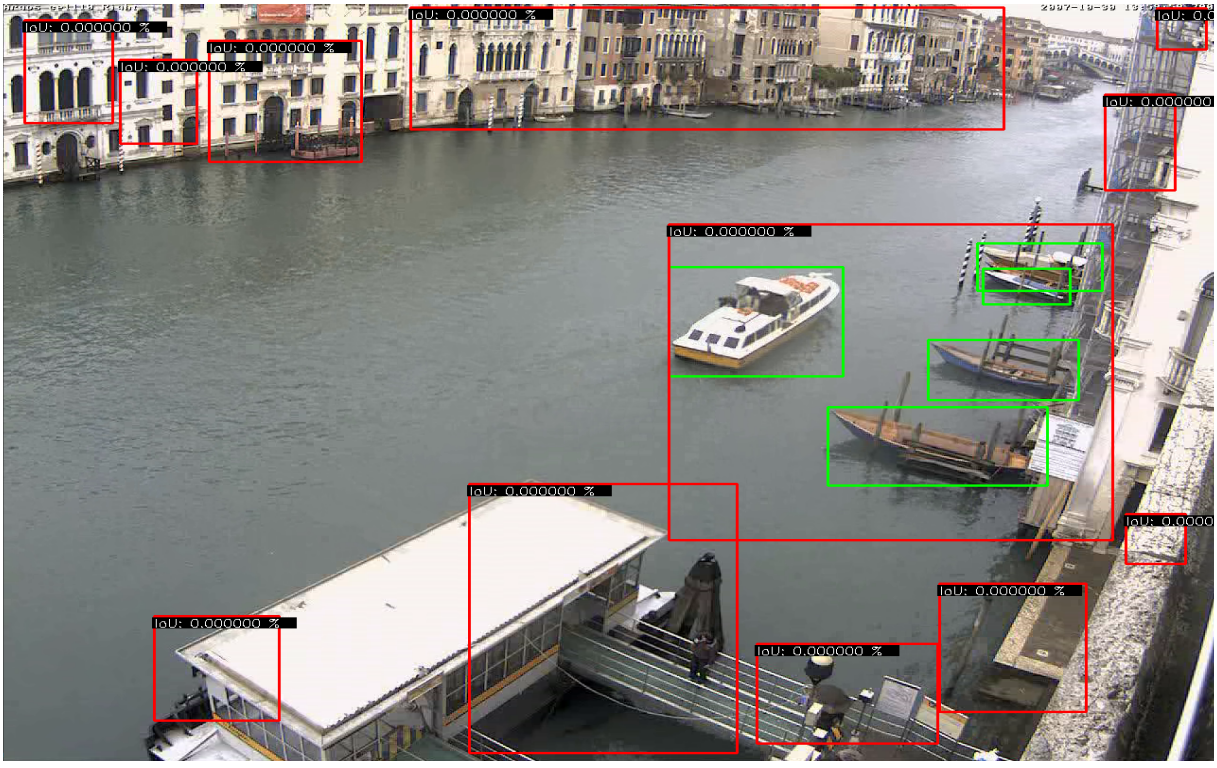


Figure 20: Found Boxes: 12. IoU : 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0% 0%

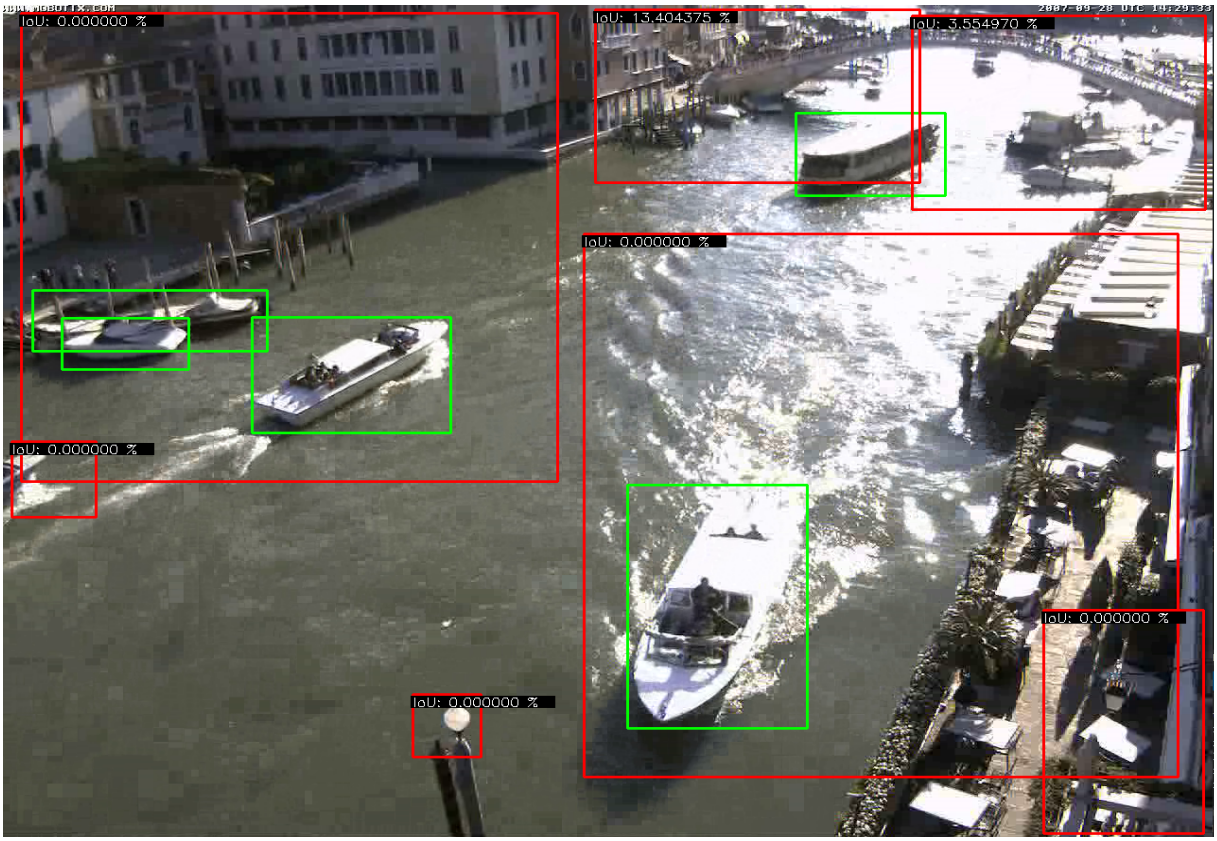


Figure 21: Found Boxes: 7. IoU : 0% 0% 13% 3% 0% 0% 0%

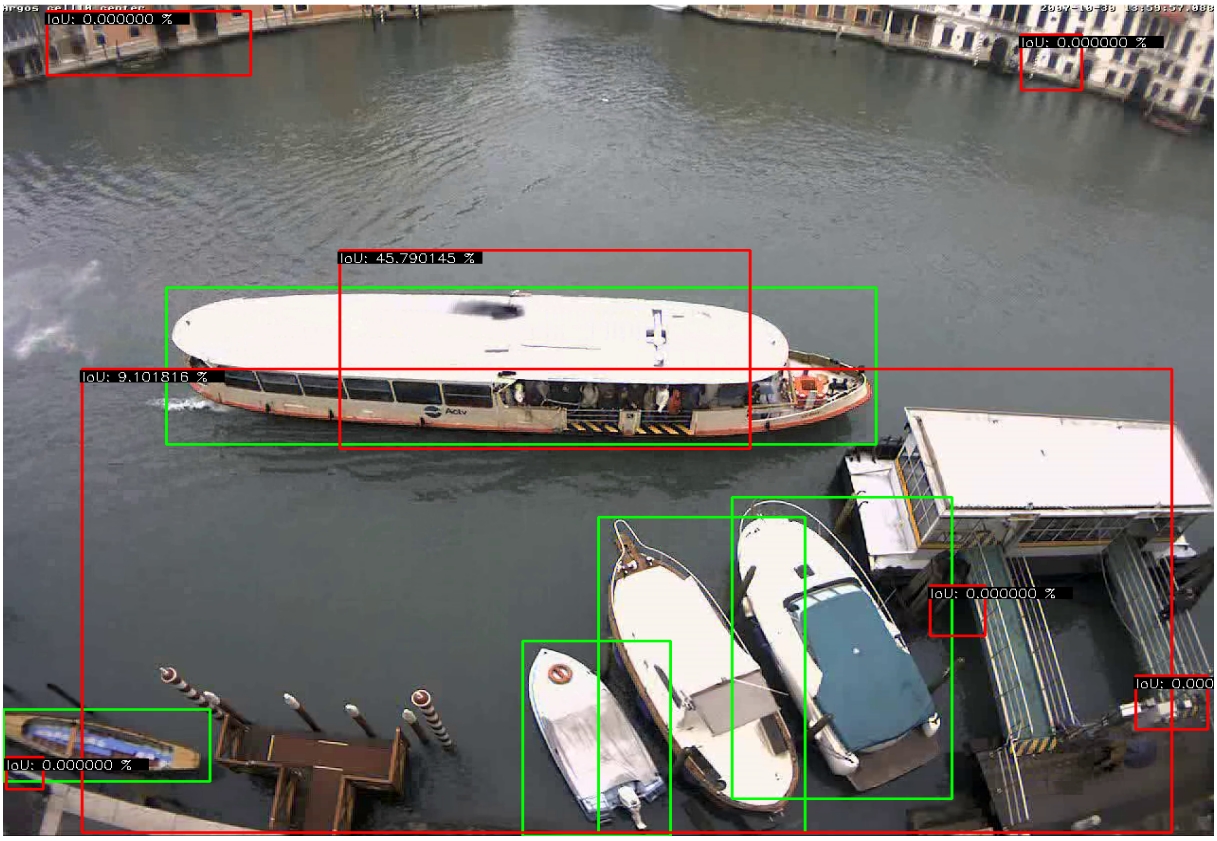


Figure 22: Found Boxes: 7. IoU : 45% 9% 0% 0% 0% 0% 0%

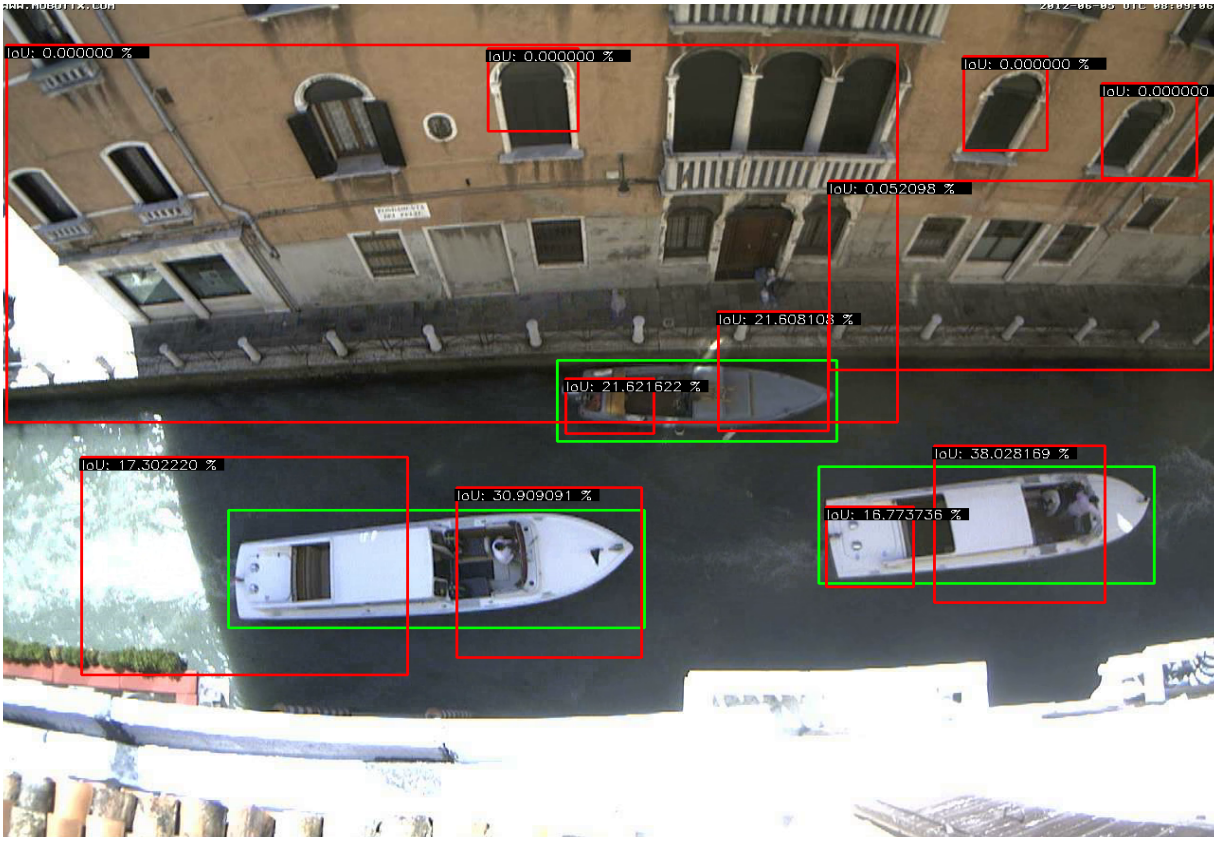


Figure 23: Found Boxes: 11. IoU : 0% 0% 0% 0% 0% 21% 21% 17% 30% 16% 38%