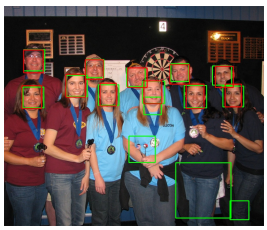
Image Processing Report

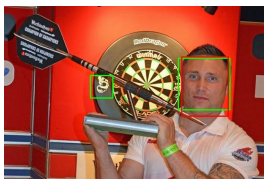
Theano Xirouchaki, Ruairi Fox

December 2, 2019

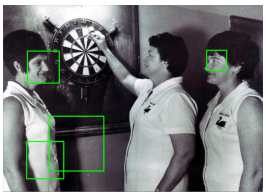
1 Ground Truth and Visualisation


dart4


dart5


dart13


dart14


dart15

Our first step was translating the code from c++ to python as it is a language we are more familiar with and enjoy more. We used the help of a tutorial [1] to do this. We then manually found the coordinates of the ground truth faces and stored them in a dictionary using the filename of each picture as the key. We ran the detector for each image and produced an output image with the required bounding boxes (see above) as well as the True Positive Rate and F1 score of each image (see below). It is worth noting we have not considered any of the faces in image 15 as ground truth as they are not frontal faces, which was the goal of the detection task.

Image	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TPR	0	0	0	0	1	1	0	1	0	1	0	1	0	1	1	0
F1	0	0	0	0	1	0.846	0	1	0	0.4	0	1	0	0.667	0.444	0

The True Positive Rate only gives a partial image of the data: it is always possible to get a perfect TPR score. This can be done by a "detector" that simply outputs a high number of bounding box coordinates covering parts of the image in a dense enough way. There is, essentially, a trade-off between true positives and false positives in a somewhat naive detector, we can maximise true positives, but false positives will also increase, or we can minimise false positives, but true positives will also decrease. That is why the F1 score is a better measure of the classifier's quality, as it accounts for precision as well as recall.

[1] https://docs.opencv.org/3.4/db/d28/tutorial_cascade_classifier.html

2 Building and Testing your own Detector

2.1 Training Performance

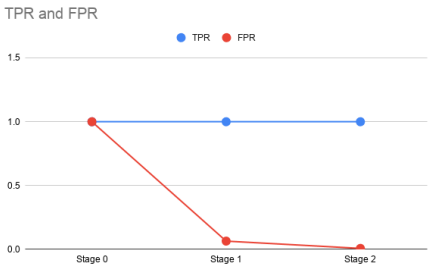
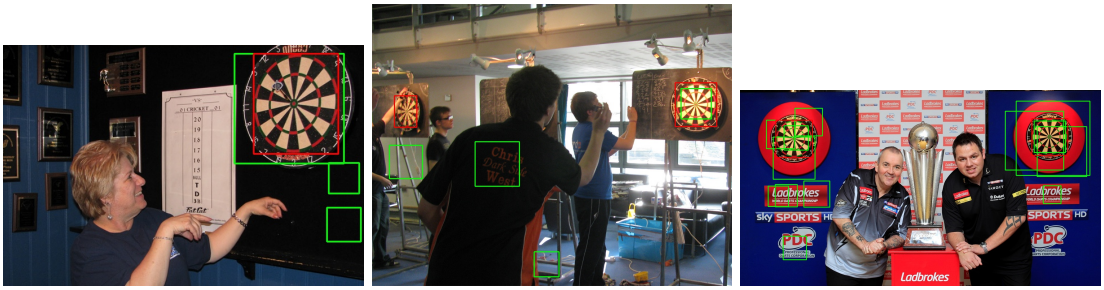


Figure 1: TPR and FPR against stages

In the above graph, we can see that at the start of stage 0, both the true positive rate and the false positive rate were 1. This means that it would detect every possible sample as a dartboard. After this, the TPR stays at the same value but the FPR goes down. This makes sense as the attentional cascade is meant to reject a portion of false positives at each stage.

2.2 Testing Performance



dart0 dart8 dart14

Image	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TPR	1	1	0	0	1	0	1	0	0	1	0.333	0	0	0	0.5	0
F1	0.5	1	0	0	0.5	0	1	0	0	1	0.222	0	0	0	0.133	0

Table 1: Average TPR = 0.36458, Average F1 = 0.27219 (IOU threshold 0.5)

Overall the cascade model performed poorly, failing to detect many of the dartboards across the images and having a lot of false positives. Show above are images 0,8,14. Image 0 is a relatively successful detection, picking up the dartboard with 2 false positives. Image 8 is almost a successful detection but sadly the IOU on the best box was 0.489 (just below the threshold). There are again some false positives around the image. Dart 14 is the image with the most false positives and only one true positive. This could be because of the circular plates that the dartboards are mounted on.

It is expected that the classifier does worse in testing than in training as training is creates a model that maximised the TPR across the training data while testing uses that fixed model on new and unseen data. It is very unlikely that a window will be a replica of a training image, explaining why the TPR is lower.

This could be because a poor set of samples was generated by opencv_createsamples or it could be that the cascade chose bad haar like features. Section 3 aims to establish a method that will get rid of many false positives. Section 4 aims to tune the training settings to build a stronger classifier, as well as investigate other approaches of identifying candidates

3 Hough Transform

For part three of the coursework we implemented line, circle, and ellipse detection and experimented with combining them with Viola Jones to improve detection. Seeing as running circle or ellipse detection on the whole image was very processing heavy we focused on running it in the windows already identified by Viola Jones with the intention of reducing false positives. This worked fairly well - the total number of false positives reduced from 35 to 10. Additionally, most of the remaining false positives include at least partial dartboards, and as such are harder to identify as false. Compared to part one,

Image	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TPR	1	1	0	0	1	0	0	0	0	1	0.3333	0	0	0	0.5	0
TPR Difference	0	0	0	0	0	0	-1	0	0	0	0	0	0	0	0	0
F1	1	1	0	0	1	0	0	0	0	1	0.3333	0	0	0	0.25	0
F1 Difference	+0.5	0	0	0	+0.5	0	-1	0	0	0	+0.111	0	0	0	+0.17	0

Table 2: Average TPR = 0.3021, Average F1 = 0.2865

We implemented line detection utilising gradient information to increase efficiency. For the detector, we check that a given Viola Jones box has at least 6 lines that are at angles at least 10 degrees different to one another. The idea behind this is that a dartboard has many intersecting lines, and we want to avoid double counting lines that are in reality one but have been detected as two very similar ones due to noise.

We implemented the method [2] proposed by Yonghong Xie and Qiang Ji for efficient detection although it was very slow so, to make it viable, we only considered one in thirty pixels of those that have passed the gradient magnitude threshold. This still gives accurate enough results without taking hours to run.

Overall positives: Our code eliminates a vast majority of the false positives that do not include dartboards, as the lack of lines and ellipses is noted. Figures 6-9 show an example of a complete false positive being eliminated.

Overall negatives: Our code does not eliminate false negatives that include partial darboards and as such does not deal with Viola Jones windows that are too big or small but do partly cover a dartboard. This shortcoming can be seen in figures 2-5. It also eliminates one true positive due to the picture being very angled.

Figure 2: Sobel image thresholded at 0.3

Figure 3: Line hough space

Figure 4: Ellipse hough space

Figure 5: The detected regions in image 10

Figure 6: Sobel image thresholded at 0.3

Figure 7: Line hough space

Figure 8: Ellipse hough space

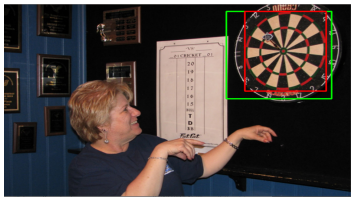
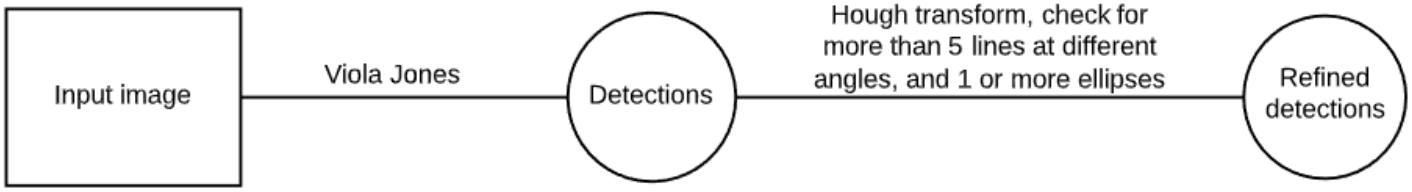


Figure 9: The detected regions in image 0



[2] Xie, Yonghong, and Qiang Ji. "A new efficient ellipse detection method." Pattern Recognition, 2002. Proceedings. 16th International Conference on. Vol. 2. IEEE, 2002

4 Improvement

4.1 IDEA: Speeding up hough space

Calculating the hough space takes a considerable amount of time, especially for ellipses. In order to speed up detection, we can pre-compute the hough space and turn what would be an O(N^3) problem into an O(1) problem. Unfortunately because of the way that we calculated the hough space it would have taken longer to apply the pre-computed results than to just recalculate for a given box. This method would need major adjustments to the code to work.

4.2 IDEA: Image segmentation

A dart board consists of a lot of strongly contrasting wedges. It should be possible to find a threshold that separates the dartboard effectively, and then it would be possible to count the objects. Unfortunately, when attempting to use this the results were often influenced heavily by noise and was unable to discriminate well between positives and negatives.

4.3 IDEA: Improving cascade TPR

The first method we will use to try and improve performance is tuning the options used to create samples and to build the classifier. We want a classifier that detects as many dartboards as possible. Once we have achieved that, then we can try to lower false positives, both in the cascade and the shape detection. The initial classifier used in section 2 was trained on 500 positive and negative images, so we trained another on 1000 of each. The results are shown below

Image	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
TPR	1	1	0	1	1	1	0	1	0.5	1	0.666	0	0	0	0.5	1
F1	1	0.666	0	0.5	1	0.666	0	0.5	0.4	0.666	0.266	0	0	0	0.222	1

Table 3: Average TPR = 0.6041, Average F1 = 0.4305 (IOU threshold 0.5)

This was a far more successful classifier than the original, having a significantly higher TPR and F1 score than the first classifier. This is likely due to the increased variety in samples that it got to learn off of.