u195839 Umut Ekin Gezer
u204256 Ekaterina Erofeeva

Lab 4. Report

Task 1.1:

In this task, we understand whole the algorithms which were given. Accordingly, we implemented the functions in Task 1.2

Task 1.2

In this task, we implement the derivation of integral images through cumulative sum lookup (see GetIntergralImages.m). The script is implemented according to the original work of Viola and Jones and provided formulas. First, we sum over the rows of the image, then over the columns of the resulting cumulative matrix from the previous step. Then, we implement the auxiliary algorithm to reduce the number of detections (see ObjectDetection.m). We check if the detected areas overlap, and in that case, we choose only one area from the set of overlapping areas.

We tested the implemented code on several images (see *data* folder):

1: img1.jpg

u195839 Umut Ekin Gezer
u204256 Ekaterina Erofeeva

2: img1.jpg with an auxiliary algorithm

3: img2.jpg

u195839 Umut Ekin Gezer
u204256 Ekaterina Erofeeva

4: img2.jpg with an auxiliary algorithm

5: img3.jpg

6: img3.jpg with an auxiliary algorithm

As you can observe, sometimes the algorithm detects faces where there are none: for example, on img1.jpg on the uniform of spacemen. Another example of faulty detection is img2.jpg, where a large area containing multiple faces is detected as a single face.

Explanation of OneScaleObjectDetection.m:

Cascaded detection presented in the work of Viola and Jones can be viewed as a decision tree, where one classifier is viewed as the node of the decision tree. If the outcome of the first classifier is positive, the evaluation of the second classifier is set in motion, and so on. Effectively it creates a decision tree-like procedure. Code lines 35-38 “run” through the decision tree and sum over its nodes. For each stage of the classifier, the tree sum is added to the overall stage sum.

Task 1.3:

In this task, we go to the file MTCNN file and the demo code to test the method given in the paper “Joint Face Detection and Alignment using Multi-task Cascaded Convolutional Networks”.

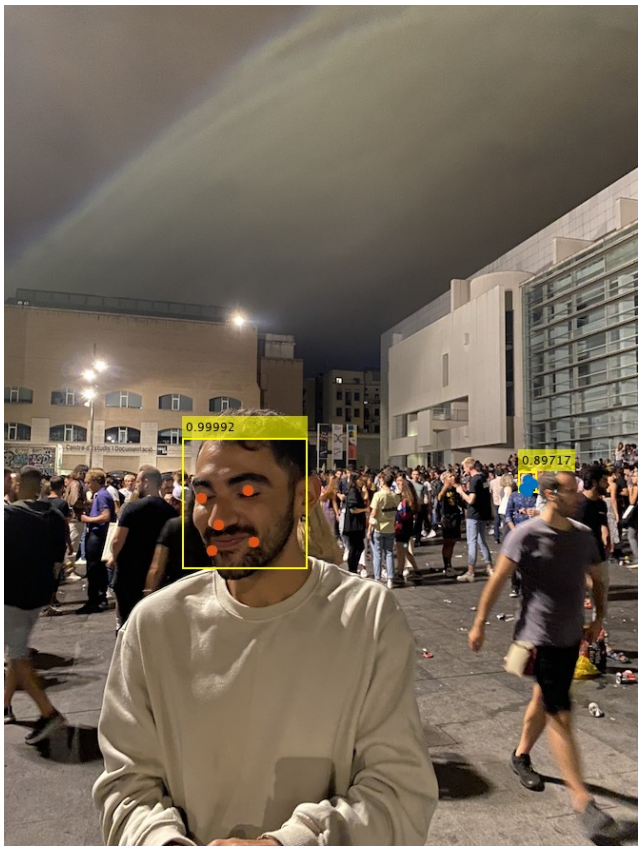
First, we would like to compare the result between the images that we take from my galleries or several internet sources. Our aim is to compare the efficiency of the algorithm in terms of some of challenges such as blurred part, not complete faces (face from right/left perspective, face with ffp2 mask) or not real human faces (paintings, printed face on t-shirts).

u195839 Umut Ekin Gezer
u204256 Ekaterina Erofeeva

Here in the first example, we chose the clearest faced image, which contains leaders of G20 countries during the G20 meeting in OSAKA. The face detection score of the leader in the rightest side is the minimum since he was not turning his face clearly on the camera like the other leaders. Nevertheless, the whole the scores are high as we can see in the example images in below. The algorithm found 10 faces in 0.185 seconds.

Secondly, we test an image of Umut in the MACBA. We chose the picture deliberately since there are a lot of people on the image, and we wanted to make challenge for this given algorithm. According to face detection algorithm, two faces were detected. The second face is detected, regardless of looking distance to camera, since the detecting five main organs (Left eye, Right eye, Nose, left mouth corner, Right mouth corner) in human face was one of the main principles behind the algorithm. Accordingly, we might assume that detecting the human organ parts on face carries more importance than distance of faces. The algorithm found 2 faces in 0.358 seconds.

u195839 Umut Ekin Gezer
u204256 Ekaterina Erofeeva



u195839 Umut Ekin Gezer
u204256 Ekaterina Erofeeva

Here is the third example to add a challenge for the algorithm. We will input the image, Umut with FFP2 mask and unmasked Mona-Lisa). I would like to compare the results between them accordingly, since my nose and two parts in mouth is not clear than Mona-Lisa's.

According to obtained score, Mona Lisa has greater score than Umut's masked face even she is not alive, and she has the face on the painting. Since her facial five components are more clear Umut's masked face. Nevertheless, we might assume that the algorithm detects the faces behind the mask whereas the mask decreases score of detection. In the Mona Lisa example algorithm detects two faces in 0.253 seconds. You can see also the second example with the famous painting "*La Liberté guidant le peuple »* and Umut's masked face again. We can derive similar assumption such like the Mona-Lisa example. In this example the algorithm detects 7 faces in 0.303 seconds.

In the last example we would like to input more challenging image, which includes masked faces and a blurred masked face.

The face detection algorithm has founded 7 faces in 0.711 second. Interestingly, the face detection algorithm detected even unclear faces. Whereas the algorithm could not detect the face in the left corner. Unfortunately, we could not detect the reason even we have checked the reference paper.

u195839 Umut Ekin Gezer
u204256 Ekaterina Erofeeva

Finally in the demo mode, in the last line we see some of parameters that we can apply our images in order to have quick and precise scores. These features are MinSize, MaxSize and PyramidScale. These features could be used for much precise results. For instance, in the Mona Lisa example if we tune the MinSize parameter from 24 to 30, the algorithm only detects Umut's masked face in 0.207 seconds, which is more effective than default function. The several detections could be done by retuning the three parameters.

```
detector = mtcnn.Detector();
tic();
[~, scores, ~] = detector.detect(im);
timeTaken = toc();
fprintf("Found %d faces in %.3f seconds.\n", numel(scores), timeTaken);
```

```
Found 2 faces in 0.253 seconds.
```

If you know the expected scale of your face there are also several Name-Value optional arguments which can be used to control the details of face detection

```
detector = mtcnn.Detector("MinSize", 30, "MaxSize", 48, "PyramidScale", 1.1);
tic();
[~, scores, ~] = detector.detect(im);
timeTaken = toc();
fprintf("Found %d faces in %.3f seconds.\n", numel(scores), timeTaken);
```

Detector(varargin1,varargin2,varargin3,varargin4,varargin5,varargin6)

Enter a value for **varargin** (Optional)

```
Found 1 faces in 0.207 seconds.
```