Signal and Image Processing - HW2

Noëlle Schenk

April 11, 2018, time invested : 9.5h

1 Comments

Please note that I was not able to run the line `from tools_template import *` and I don't know exactly why. As I think it will run on your system, I let this line in the code. However, the scripts also work if just `python3 tools_template.py` followed by `python3 <ex1-3>`, so I hope this counts as running code.

2 Exercise 1

Increasing filter size makes the dark border around the image to be thicker, as the code does handle borders by zero padding. If another way of handling borders would be chosen, this would change. Generally, a larger filter takes into account more neighbors and leads to a higher blurring intensity.
A large sigma gives more weight to far neighbors and therefore also leads to a higher blurring intensity.

3 Exercise 2

On Figure 1 shows two edge detection outcomes with different parameters. The filter size was the same, only σ 2 was changed. The difference in DoG are striking, on the left compared to the right, edges are much thinner and the whole image shows less standard deviation in grayscales (much less contrast). Probably, edges are detected better if the two sigma do not differ too much, not 25:1 as in the right image panel.
Smoothing removes small details and subtracting two smoothed images results in the DoG image that does not show small details any more, but traces the edges where pixel values change much.

4 Exercise 3.1 c

The created gaussian pyramid can be seen in Figure 3.

5 Exercise 3.4

Increasing α from 0.56 to a higher value accepts less values to be "high". That means, less pixels are 1 and the face detection is more restrictive. If α is increased too much, probably only the best matching face is detected.
Decreasing α allows more pixels to reach a high score and therefore detects more faces.
Depending on the situation, either a very restrictive or the contrary can be of use, e.g. to identify face candidates which are later classfied by hand, it makes sense to allow many pixels to be identified as faces to capture all faces. If no classification by hand is done and it is not incredibly important to find all faces on a picture, a restrictive search (with a high α) may fit better.
It is by the way also normal to not only detect the middle of the face (mostly the nose) at one given pixel but many pixels on the nose.

Figure 1: Gaussian pyramid.

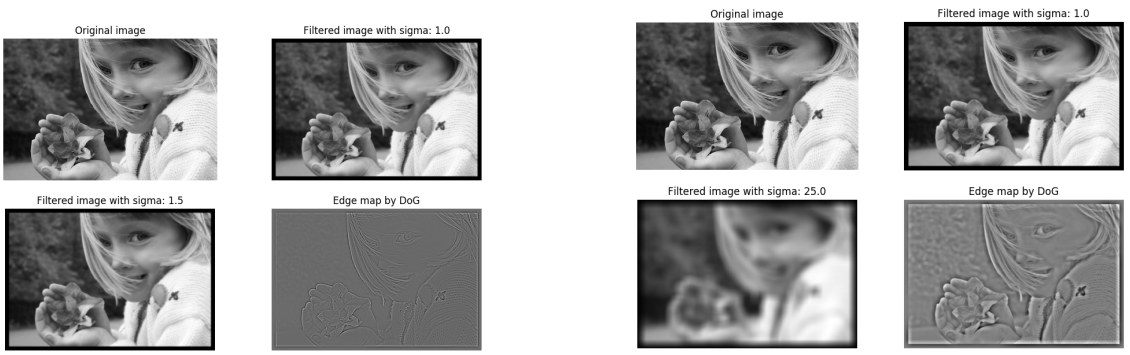

Figure 2: Edge detection with different parameters. Left : σ 1 = 1, σ 2 = 1.5. Right : σ 1 = 1, σ 2 = 25, filter size 30 for both.

Figure 3: Gaussian pyramid.

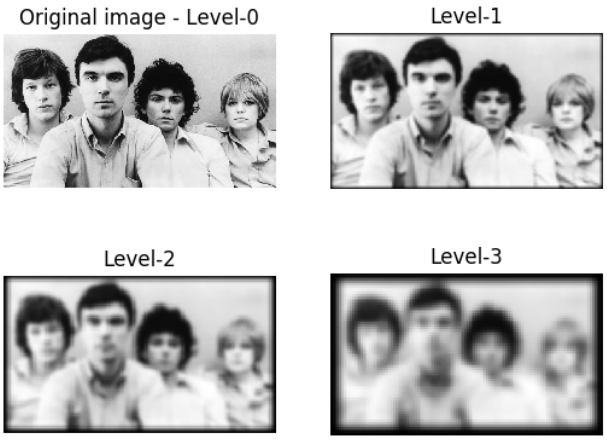

Figure 4: Parameters : $\sigma = 4$, scale $= 0.6$, filter size $= 8$, num levels $= 4$

Figure 5: Gaussian pyramid.

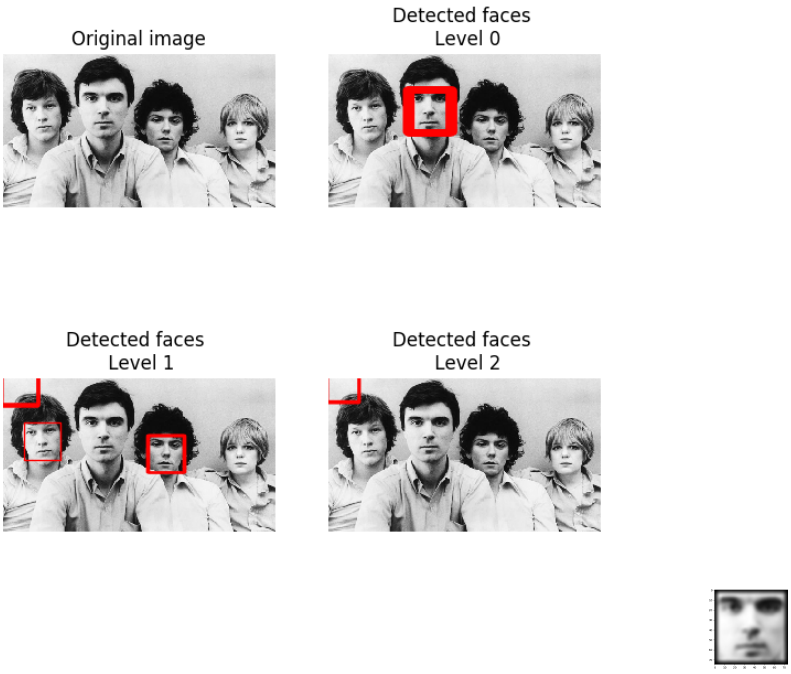

Figure 6: Detected faces (left) and the template which was used for level 1 (right).

6 Exercise 3.5

Figure 5 shows the detected faces with the rectangles around them. It is interesting that at level 0, when the template is of original size, the face where the template is taken from is detected. The other faces don't match, as they do not fit the size. At level 1, the two smaller faces of men are detected, but also a rectangle at the upper left corner. The template used for level 1 detection is shown below in Figure 5. Possibly, the dark borders match as well, however, this could easily be corrected by cutting away the dark corners or using another approach of dealing with the borders, e.g. Repetition or Reflection.
More possible pitfalls could be that only faces that look straight into the camera are detected. No faces from the side or from a given angle can be found. Further, only the faces that match a size of the gaussian pyramid can be detected. Then, only one template was used, which was the one detected at level 0. Of corse, this template matches best the same man. However, it would make sense at least using a template from a woman and a man, and if possible more templates. Further, this code only works for black and white images and I imagine that brighness also plays an important role. We only want to match the shapes, but what we do is also correlating the brightness which is useless in our case.

Anyway, I think the most important are the sizes, if we only work with 3 levels of sizes, we can only detect roughly three levels of face sizes. With the implemented algorithm, it would be very easy to make more sizes and probably also detect the smallest face. Further, the image could aso go through something like a brightness gaussian pyramid, so we would have different brighnesses of each template. Then, the dark borders need to be either cut away or omitted by another border approach. If we only want to detect faces from this image, we could aso take all faces as templates and test if the "right" match also shows the highest correlation (which was the case here). We don't have any smiling faces, so if we want to detect smiling faces or faces with other expressions, we need to include that also in our templates. I found it interesting that similarity between human does not play such a large role in this rough face detection, but more the general shape of a face.

An alternative could be to work with edges, that means first detecting all edges in the original image and also the template and do the same face detection with these images. This would prevent the problems with brightness. However, in colored images the skin color could probably be an important indicator for a face. And some parts of the face are not really detectable by edges, e.g. the cheeks and depending on brightness also the chin.
Possibly, it could be an approach to separately detect eyes, noses, ears and other parts of faces and then overlap the individual correlations. We could identify a score to e.g. count in a given area how many face-parts matched and identify the most scoring areas. This would allow us to search for faces from the side as well.