Assignment 2: Filtering and edge detection

Francesco Done'
qwg586@alumni.ku.dk
Exchange Student

November 30, 2020

For this assignment the original image used was `lenna.jpg` that is a greyscale image, as shown below:

Figure 1: lenna.jpg

1 Gaussian filtering

The gaussian filtering has been implemented with `skimage.filters.gaussian` method, that permits to filter an image with a specified sigma σ that is the standard deviation for the Gaussian kernel. In particular, for this task, σ has been set to $1, 2, 4, 8$, as is it possible to see in the following figures:

(a) $\sigma = 1$ (b) $\sigma = 2$

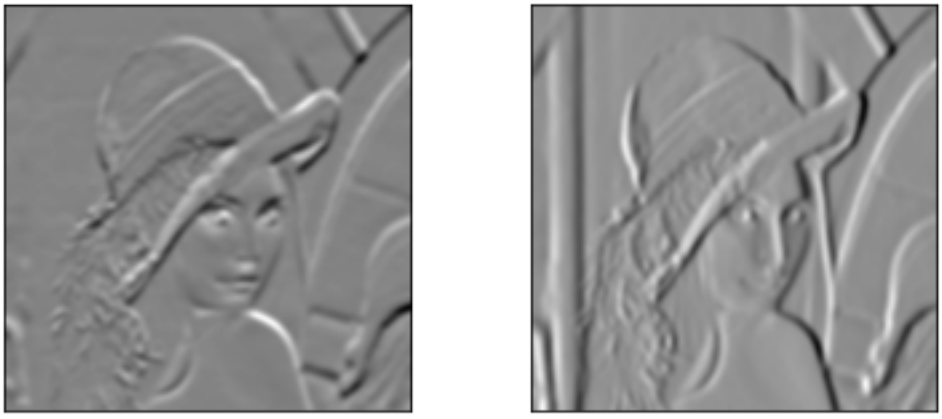
(c) $\sigma = 4$ (d) $\sigma = 8$

Figure 2: Gaussian filter applied with different σ values

For each pixel i in the original image, the filter computes the new image taking into account an area with with $3 \cdot \sigma$ pixels radius more or less, and with it, it will compute a new area A_i in which around the center of the Gaussian funciton (σ pixels radius with i as center) the weights are significantly distributed (that represents the 68.1% of A_i). In the area from σ pixels radius up to $3 \cdot \sigma$ pixels radius, the weights are not strongly distributed (the distribution with the covered area is available at this link).

2 Gradient magnitude computation

Computing the gradient magnitude with Gaussian derivatives has required first of all to calculate the x-derivative X' and the y-derivative Y' as shown here below with a Gaussian filter:



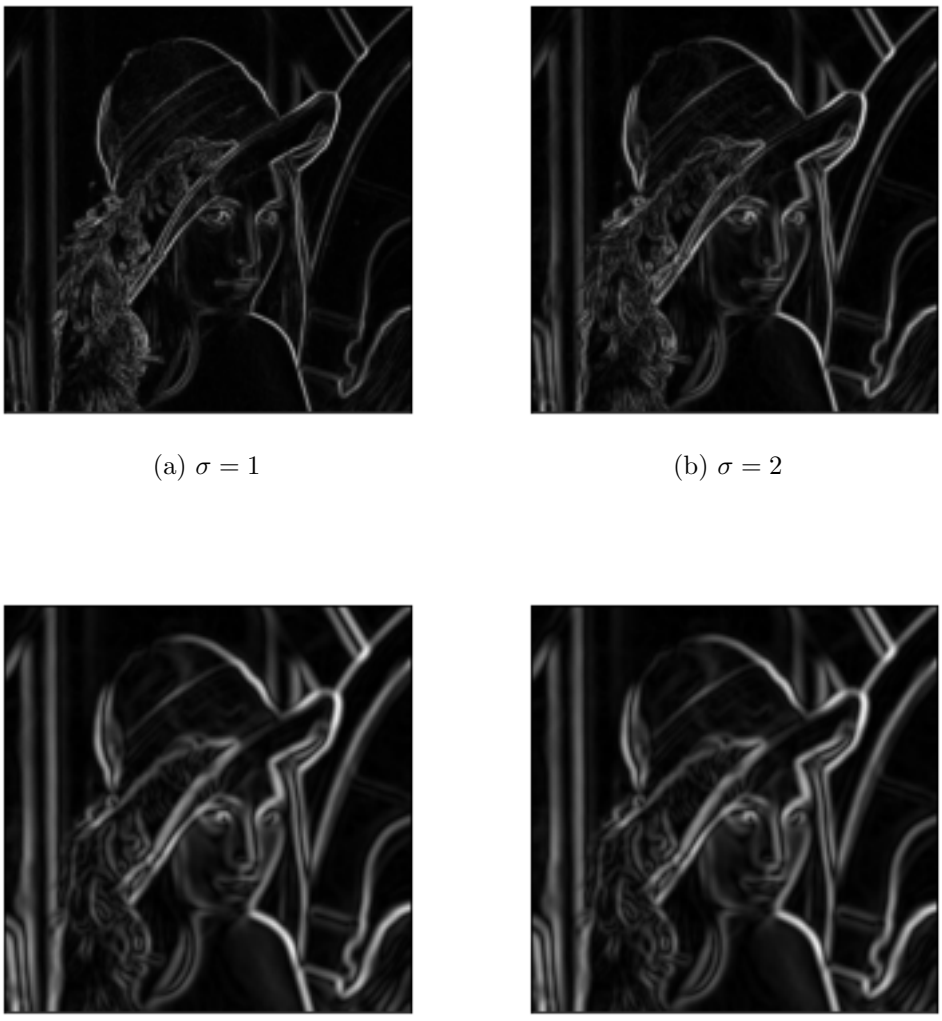
(a) (b)

Figure 3: Horizontal (a) and vertical (b) gaussian gradient calculated with $\sigma = 8$

These derivatives have been calculated for $\sigma \in \{1, 2, 4, 8\}$, then the magnitude $\| \bigtriangledown f \|$ has been computed with the formula:

$$\| \bigtriangledown f \| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2} = \sqrt{(X')^2 + (Y')^2}$$

The results are shown here below:



(a) $\sigma = 1$ (b) $\sigma = 2$

(c) $\sigma = 4$ (d) $\sigma = 8$

Figure 4: Gradient magnitude computation using Gaussian derivatives calculated with different σ values

As is it possible to see, combining the two derivatives (X' and Y') with the magnitude formula, makes possible to detect the shape inside the image, in this case, the woman face, the hat and part of the background. The results differ on their smoothness, that is with a lower σ the shape results detailed, meanwhile with a higher σ the border of the shape appears blurred and less

detailed. This is due to the Gaussian filter applied before detecting X' and Y', that is for higher σ values the filtered image is well blurred, not so much on the contrary, as explained in *chapter* 1.

3 Laplacian-Gaussian filtering

For this task, it was used the function `scipy.ndimage.gaussian_laplace` that takes in input an image, and a specific sigma value σ, it was chosen $\sigma \in \{1, 2, 4, 8\}$. This method is used for detecting rapid changes (also knows as edges) inside an image. The Gaussian filter is applied before the Laplacian since this kind of filtering (derivative based filtering) is influenced a lot by the noise, in this way it is possible to reduce that noise in order to make the entire process more accurate. As is it possible the more is the σ value the less is the accuracy in detecting edges, but if σ is too small (e.g. $\sigma = 1$) a lot of noise is captured and it compromises the final result.

(a) $\sigma = 1$

(b) $\sigma = 2$

(c) $\sigma = 4$ (d) $\sigma = 8$

Figure 5: Laplacian-Gaussian filtering applied with different σ values

4 Canny edge detection

The Canny edge detection has been implemented with `cv2.Canny` method. The parameters are: an image img, the high threshold value of intensity gradient $t1$, the low threshold value of intensity gradient $t2$ and the kernel size $ksize$ used in the Sobel filter, that is used to calculate the X' and Y' derivatives. Regarding $t1$ and $t2$, all the detected edges with an intensity gradient higher than $t1$, are for sure edges; all of those with an intensity gradient less than $t2$ are for sure non-edges, thus not considered; while the edges with an intensity gradient between $t1$ and $t2$ are considered or not based on the connection with edges or non-edges. Here below there are few examples with different thresholds and $ksize$ values:

(a) $t1 = 200 \quad t2 = 100 \quad ksize = 3$

(b) $t1 = 1500 \quad t2 = 800 \quad ksize = 3$

(c) $t1 = 1500 \quad t2 = 800 \quad ksize = 5$

(d) $t1 = 1500 \quad t2 = 800 \quad ksize = 7$

Figure 6: Canny edge detection applied

As is it possible to see, lowering the thresholds means that more edges are considered, also using a bigger kernel size with fixed thresholds values, means that more is the kernel size and more are the edges revealed.