Assignment Template

Edoardo Papa - 1962169

March-April 2021

1 Exercise 1

The data of the following exercise: Execution:

```
ex1(matricola)
```

```
Operation:
Convolution

Image:
[[0 0 5 7]
 [5 4 8 5]
 [0 5 7 5]
 [4 8 5 0]]

Kernel:
[[0 0 0]
 [0 5 4]
 [8 5 0]]

Paddings:
Reflection padding

First Coords
1 1
Second Coords
0 2
```

Figure 1: Data of the Exercise 1

Ex 1)

THE FIRST THING THAT I'LL DO IS A PADDING BECAUSE I WANT TO MANTAIN THE SAME SIZE OF THE IMAGE, AND IN PARTICULAR FOR THE SECOND COORDINATE I NEED TO ADD A PADDING.

$$\begin{bmatrix} 0 & 0 & 5 & 7 \\ 5 & 6 & 8 & 5 \\ 0 & 5 & 7 & 5 \\ 6 & 8 & 5 & 0 \end{bmatrix} \xrightarrow[(1,1)(1,1)]{\text{Adding Padding}} \begin{bmatrix} 6 & 5 & 6 & 8 & 5 & 8 \\ 0 & 0 & 0 & 5 & 7 & 5 \\ 6 & 5 & 6 & 8 & 5 & 8 \\ 5 & 0 & 5 & 7 & 5 & 7 \\ 8 & 6 & 8 & 5 & 0 & 5 \\ 5 & 0 & 5 & 7 & 5 & 7 \end{bmatrix}$$

I DON'T NEED TO APPLY THE PADDING FOR BOTH OF COORDINATE BECAUSE IT'S ENOUGH APPLY IT FOR THE SECOND COORDINATE.

$$\begin{bmatrix} 6 & 5 & 6 & 8 & 5 & 8 \\ 0 & 0 & 0 & 5 & 7 & 5 \\ 6 & 5 & 6 & 8 & 5 & 8 \\ 5 & 0 & 5 & 7 & 5 & 7 \\ 8 & 6 & 8 & 5 & 0 & 5 \\ 5 & 0 & 5 & 7 & 5 & 7 \end{bmatrix} \otimes \begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 6 \\ 8 & 5 & 0 \end{bmatrix}$$

$$f[m,n] = I \otimes g = \sum_{k,l} I[m-k, n-l] \cdot g[k,l]$$

COORDINATE 1 : (1,1)

COORDINATE 2 : (0,2)

I START ROTATING THE KERNEL IN ORDER TO REDUCE THE CONVOLUTIONS OF THE FORMULA AND FOR THINKING MORE EASILY.

Figure 2: Exercise 1 pt.1

FLIP TWICE :

$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 5 & 6 \\ 8 & 5 & 0 \end{bmatrix} \xrightarrow{1^a} \begin{bmatrix} 8 & 5 & 0 \\ 0 & 5 & 6 \\ 0 & 0 & 0 \end{bmatrix}$$

$$\begin{bmatrix} 8 & 5 & 0 \\ 0 & 5 & 6 \\ 0 & 0 & 0 \end{bmatrix} \xrightarrow{2^a} \begin{bmatrix} 0 & 5 & 8 \\ 6 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

COORDINATE 1 (1,1) :

$$\begin{bmatrix} 6 & 5 & 6 & 8 & 5 & 8 \\ 0 & 0 & 0 & 5 & 7 & 5 \\ 6 & 5 & 6 & 8 & 5 & 8 \\ 5 & 0 & 5 & 7 & 5 & 7 \\ 8 & 6 & 8 & 5 & 0 & 5 \\ 5 & 0 & 5 & 7 & 5 & 7 \end{bmatrix} \otimes \begin{bmatrix} 0 & 5 & 8 \\ 6 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$

$$= (0\times0)+(0\times5)+(8\times5)+(6\times5)+(6\times5)+(0\times8)+(0\times0)+(5\times0)+(7\times0) = 60+20+20+0 = 80$$

$$= 80$$

IN THE POSITION (1,1) AFTER EXECUTE THE FILTER WILL STAY $\boxed{80}$

Figure 3: Exercise 1 pt.2

COORDINATE 2 (0, 2):

$$\begin{bmatrix} 4 & 5 & 4 & 8 & 5 & 8 \\ 0 & 0 & 0 & 5 & 7 & 5 \\ 4 & 5 & 4 & 8 & 5 & 8 \\ 5 & 0 & 5 & 7 & 5 & 7 \\ 8 & 4 & 8 & 5 & 0 & 5 \\ 5 & 0 & 5 & 7 & 5 & 7 \end{bmatrix} \otimes \begin{bmatrix} 0 & 5 & 8 \\ 4 & 5 & 0 \\ 0 & 0 & 0 \end{bmatrix} =$$

$$= (0\times4)+(5\times8)+(8\times5)+(4\times0)+(5\times5)+(0\times7)+(0\times4)+(0\times8)+(0\times5) = 40+40+25+0 = 105$$

IN THE POSITION (0, 2) AFTER EXECUTE THE FILTER WILL STAY $\boxed{105}$

Figure 4: Exercise 1 pt.3

Exercise comment:

The reflect padding is a padding where the edge pixels are added into the outside copying the pixels reflected from the edge of the image. It's a good approach because we add to image something that isn't completely wrong considering that it's a part of the image. We can obtain a better performance than the other padding for example into the morphological image processing (where e.g. the zero padding could be worst). At same time, it could be worst in terms of time computation.

2 Exercise 2

The data of the following exercise:

```
ex2(matricola)

Sigma:
3
```

Figure 5: Data of the Exercise 2

Execution:

GAUSSIAN FORMULA:

$$G(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

$$\frac{1}{2\pi\sigma^2} = 0{,}01768$$

I WANT TO CREATE A 3×3 FILTER SO:

$G(0,0) = 0{,}01768 \cdot e^{-0} = 0{,}01768$

$G(0,1) = 0{,}01768 \cdot e^{-\left(\frac{0+1}{18}\right)} = 0{,}01672$

$G(0,-1) = 0{,}01768 \cdot e^{-\left(\frac{0+1}{18}\right)} = 0{,}01672$

$G(1,0) = 0{,}01768 \cdot e^{-\left(\frac{1+0}{18}\right)} = 0{,}01672$

$G(1,1) = 0{,}01768 \cdot e^{-\left(\frac{1+1}{18}\right)} = 0{,}01582$

$G(1,-1) = 0{,}01768 \cdot e^{-\left(\frac{1+1}{18}\right)} = 0{,}01582$

$G(-1,0) = 0{,}01768 \cdot e^{-\left(\frac{1+0}{18}\right)} = 0{,}01672$

$G(-1,1) = 0{,}01768 \cdot e^{-\left(\frac{1+1}{18}\right)} = 0{,}01582$

$G(-1,-1) = 0{,}01768 \cdot e^{-\left(\frac{1+1}{18}\right)} = 0{,}01582$

$$\begin{bmatrix} 0{,}01582 & 0{,}01672 & 0{,}01582 \\ 0{,}01672 & 0{,}01768 & 0{,}01672 \\ 0{,}01582 & 0{,}01672 & 0{,}01582 \end{bmatrix}$$

GAUSSIAN FILTER 3×3 $\sigma = 3$

Figure 6: Exercise 2 pt1

NORMALIZATION:

SUM OF ALL ELEMENTS: $0{,}01582 + 0{,}01582 + 0{,}01582 + 0{,}01582 + 0{,}01672 + 0{,}01672 + 0{,}01672 + 0{,}01672 + 0{,}01768 = 0{,}06328 + 0{,}06688 + 0{,}01768 = 0{,}14784$

$$\frac{1}{\text{SUM OF ALL ELEMENTS}} = 6{,}764$$

$$6{,}764 \cdot \begin{bmatrix} 0{,}01582 & 0{,}01672 & 0{,}01582 \\ 0{,}01672 & 0{,}01768 & 0{,}01672 \\ 0{,}01582 & 0{,}01672 & 0{,}01582 \end{bmatrix} =$$

$$= \begin{bmatrix} 0{,}107 & 0{,}113 & 0{,}107 \\ 0{,}113 & 0{,}1195 & 0{,}113 \\ 0{,}107 & 0{,}113 & 0{,}107 \end{bmatrix}$$

RESULT OF GAUSSIAN FILTER WITH NORMALIZATION

Figure 7: Exercise 2 pt2

3 Exercise 3

The data of the following exercise:

```
ex3(matricola)
```

```
Image:
[[9 9 9 9]
 [6 6 6 6]
 [1 1 1 1]
 [2 2 2 2]]

Kernel:
[[-1 -1 -1]
 [-2 -2 -2]
 [-1 -1 -1]]
```

Figure 8: Data of the Exercise 3

Execution:

Ex 3)

$$\text{IMAGE} = \begin{bmatrix} 9 & 9 & 9 & 9 \\ 6 & 6 & 6 & 6 \\ 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 \end{bmatrix} \qquad \text{KERNEL} = \begin{bmatrix} -1 & -1 & -1 \\ +2 & +2 & +2 \\ -1 & -1 & -1 \end{bmatrix}$$

I WANT TO MANTAIN THE ORIGINAL SIZE OF THE IMAGE SO I NEED TO ADD A PADDING. I DECIDE TO USE A REFLECT PADDING BECAUSE I HAVE ALREADY USED BEFORE AND I DON'T WANT TO ADD TO THE ORIGINAL IMAGE SOMETHING THAT COULDN'T BE "TRUE" FOR THE IMAGE.

The first thing is add padding (reflect)

$$\text{IMAGE + PADDING} = \begin{bmatrix} 6 & 6 & 6 & 6 & 6 & 6 \\ 9 & 9 & 9 & 9 & 9 & 9 \\ 6 & 6 & 6 & 6 & 6 & 6 \\ 1 & 1 & 1 & 1 & 1 & 1 \\ 2 & 2 & 2 & 2 & 2 & 2 \\ 1 & 1 & 1 & 1 & 1 & 1 \end{bmatrix} \quad \text{Apply kernel} \quad \begin{bmatrix} -1 & -1 & -1 \\ +2 & +2 & +2 \\ -1 & -1 & -1 \end{bmatrix}$$

double flip the kernel

Figure 9: Exercise 3 pt.1

Apply kernel

$f(0,0) = (6\cdot-1)+(6\cdot-1)+(6\cdot-1)+(9\cdot2)+(9\cdot2)+(9\cdot2)+(6\cdot-1)+(6\cdot-1)+(6\cdot-1) = 18$

$f(0,1) = (6\cdot-1)+(6\cdot-1)+(6\cdot-1)+(9\cdot2)+(9\cdot2)+(9\cdot2)+(6\cdot-1)+(6\cdot-1)+(6\cdot-1) = 18$

$f(0,2) = (6\cdot-1)+(6\cdot-1)+(6\cdot-1)+(9\cdot2)+(9\cdot2)+(9\cdot2)+(6\cdot-1)+(6\cdot-1)+(6\cdot-1) = 18$

$f(0,3) = (6\cdot-1)+(6\cdot-1)+(6\cdot-1)+(9\cdot2)+(9\cdot2)+(9\cdot2)+(6\cdot-1)+(6\cdot-1)+(6\cdot-1) = 18$

$f(1,0) = (9\cdot-1)+(9\cdot-1)+(9\cdot-1)+(6\cdot2)+(6\cdot2)+(6\cdot2)+(1\cdot-1)+(1\cdot-1)+(1\cdot-1) = 6$

$f(1,1) = (9\cdot-1)+(9\cdot-1)+(9\cdot-1)+(6\cdot2)+(6\cdot2)+(6\cdot2)+(1\cdot-1)+(1\cdot-1)+(1\cdot-1) = 6$

$f(1,2) = (9\cdot-1)+(9\cdot-1)+(9\cdot-1)+(6\cdot2)+(6\cdot2)+(6\cdot2)+(1\cdot-1)+(1\cdot-1)+(1\cdot-1) = 6$

$f(1,3) = (9\cdot-1)+(9\cdot-1)+(9\cdot-1)+(6\cdot2)+(6\cdot2)+(6\cdot2)+(1\cdot-1)+(1\cdot-1)+(1\cdot-1) = 6$

$f(2,0) = (6\cdot-1)+(6\cdot-1)+(6\cdot-1)+(1\cdot2)+(1\cdot2)+(1\cdot2)+(2\cdot-1)+(2\cdot-1)+(2\cdot-1) = -18$

$f(2,1) = (6\cdot-1)+(6\cdot-1)+(6\cdot-1)+(1\cdot2)+(1\cdot2)+(1\cdot2)+(2\cdot-1)+(2\cdot-1)+(2\cdot-1) = -18$

$f(2,2) = (6\cdot-1)+(6\cdot-1)+(6\cdot-1)+(1\cdot2)+(1\cdot2)+(1\cdot2)+(2\cdot-1)+(2\cdot-1)+(2\cdot-1) = -18$

$f(2,3) = (6\cdot-1)+(6\cdot-1)+(6\cdot-1)+(1\cdot2)+(1\cdot-2)+(1\cdot-2)+(2\cdot-1)+(2\cdot-1)+(2\cdot-1) = -18$

$f(3,0) = (1\cdot-1)+(1\cdot-1)+(1\cdot-1)+(2\cdot2)+(2\cdot2)+(2\cdot2)+(1\cdot-1)+(1\cdot-1)+(1\cdot-1) = 6$

$f(3,1) = (1\cdot-1)+(1\cdot-1)+(1\cdot-1)+(2\cdot2)+(2\cdot2)+(2\cdot2)+(1\cdot-1)+(1\cdot-1)+(1\cdot-1) = 6$

$f(3,2) = (1\cdot-1)+(1\cdot-1)+(1\cdot-1)+(2\cdot2)+(2\cdot2)+(2\cdot2)+(1\cdot-1)+(1\cdot-1)+(1\cdot-1) = 6$

$f(3,3) = (1\cdot-1)+(1\cdot-1)+(1\cdot-1)+(2\cdot2)+(2\cdot2)+(2\cdot2)+(1\cdot-1)+(1\cdot-1)+(1\cdot-1) = 6$

FINAL IMAGE

$$\begin{bmatrix} 18 & 18 & 18 & 18 \\ 6 & 6 & 6 & 6 \\ -18 & -18 & -18 & -18 \\ 6 & 6 & 6 & 6 \end{bmatrix}$$

this filter is a filter that allow us to find a line, in particular a horizontal lines.

Figure 10: Exercise 3 pt.2

Exercise comment:

The filter that's provide me allow us to find a line, in particular the horizontal lines. This filter is a high pass filter because the sum of all the number inside the filter is equal to zero.

4 Exercise 4

The data of the following exercise:

```
ex4(matricola)

Image:
[[0 0 0 0 0]
 [1 0 0 0 1]
 [1 1 1 1 1]
 [1 0 0 0 1]
 [0 0 0 0 0]]

Operation:
Opening
```

Figure 11: Data of the Exercise 4

Note: in the follow pictures could appear a yellow line, don't consider this line because it's a IPad error. I'm sorry for the inconvenience
Execution:

Ex 4)

Method = opening

$$\text{IMAGE} = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 1 \\ 1 & 1 & 1 & 1 & 1 \\ 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

I'll use a footprint = (3x3)

In matematical morphology, opening is the dilation of the erosion of a set A by a structuring element B.

$$A \circ B = (A \ominus B) \oplus B$$

where $\oplus$ means dilation
where $\ominus$ means erosion.

It help us for delete the noise.

EROSION → min
DILATION → MAX

Appluing the formula, we need to do two operation: 1) EROSION
2) DILATION

EROSION:

(IF THERE IS AT LEAST ONE "0" THE RESULT IS "0")

The first thing is add padding because i want to mantain the same dimention of the image. i decided to apply reflection because the zero-padding would weight too much for the result, and i could obtain a wrong result.

Figure 12: Exercise 4 pt.1

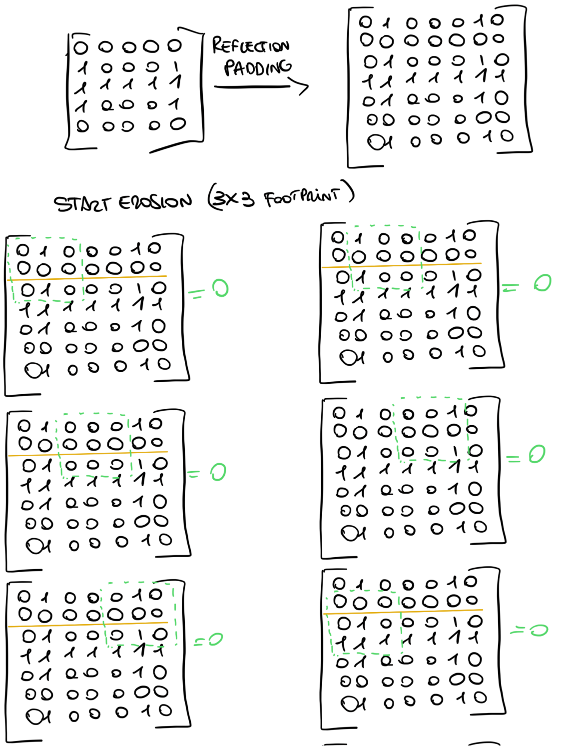

Figure 13: Exercise 4 pt.2

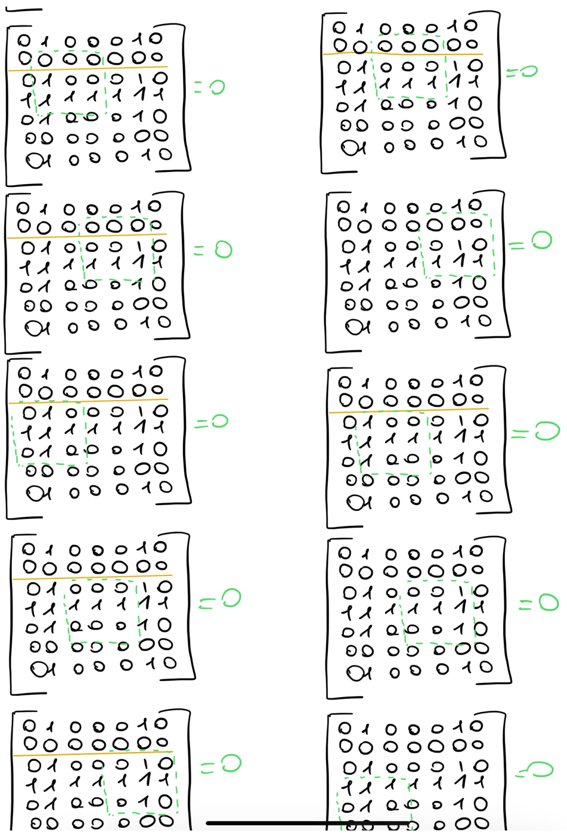

Figure 14: Exercise 4 pt.3

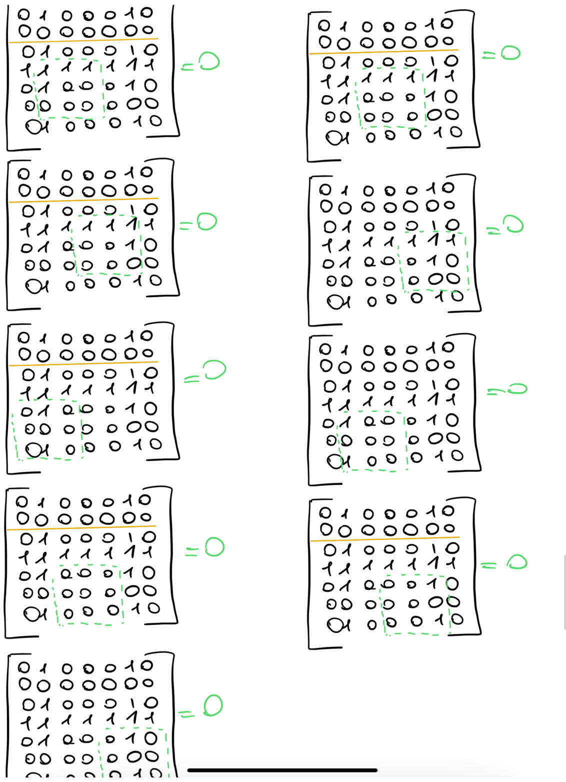

Figure 15: Exercise 4 pt.4

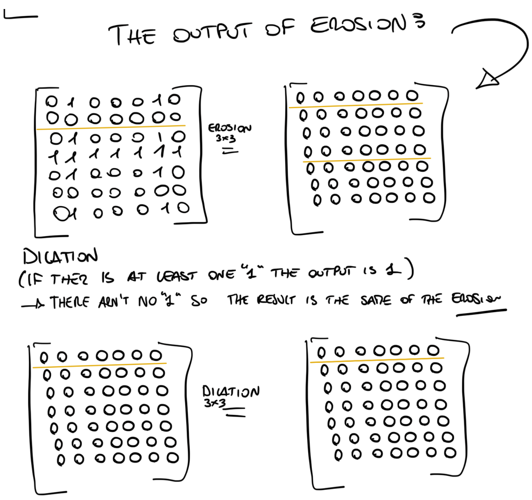

Figure 16: Exercise 4 pt.5

Exercise comment:

The opening morphology image processing consist into two different operation. The first is erosion and the second is dilation. For the proposed image with the footprint = 3x3 the output image is composed by only zero. We obtain this result because for the given footprint we not able to find a particular region such that there aren't zero pixel. So all the "1" are considered as "noise" (for this footprint). After the erosion we execute the dilation but due to all the image is zero the result will be the same (in the dilation the output is 1 if there is at least one "1").

5 Exercise 5

The data of the following exercise:

```
ex5(matricola)

Image:
[[0 0 0 0 0]
 [6 0 0 0 0]
 [1 1 0 0 1]
 [2 0 0 0 0]
 [0 0 0 0 0]]

Operation:
Erosion (3,3)
```

Figure 17: Data of the Exercise 5

Execution:

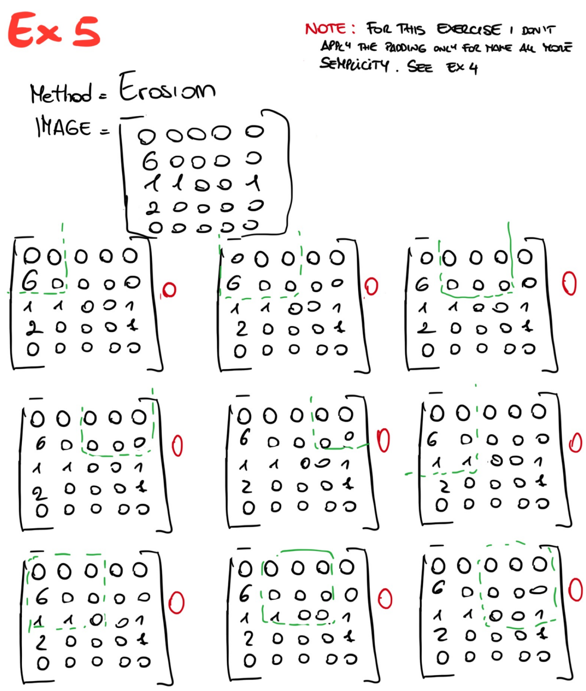

Figure 18: Exercise 5 pt.1

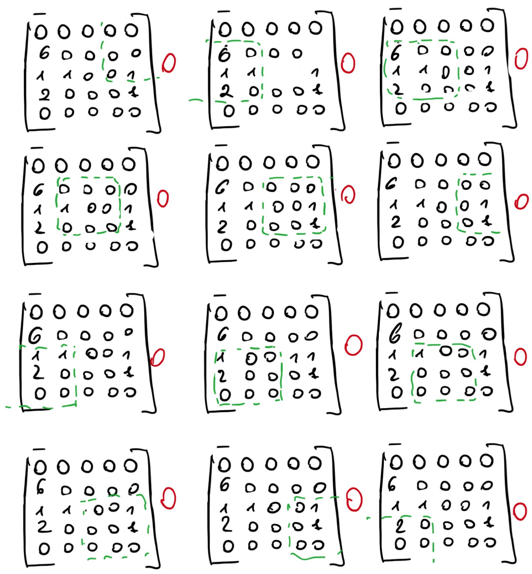

Figure 19: Exercise 5 pt.2

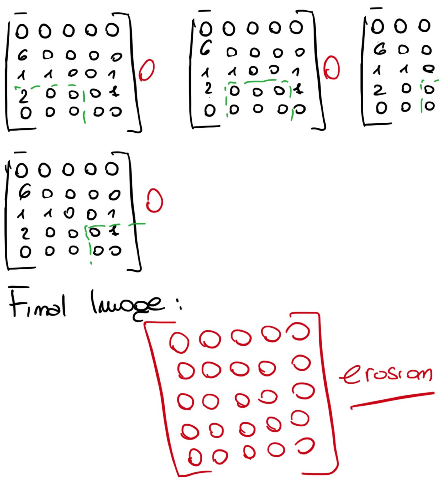

Figure 20: Exercise 5 pt.3

Exercise comment:

In the proposed image with the given footprint (3x3) the output of erosion is an image with all zero. That's happen because the erosion of a gray-scale image during the erosion process, come out the minimum value of the region that we are analyze. So with this footprint and this image, we always find a 0 as minimum.

6 Exercise 6

The data of the following exercise:

```
ex6(matricola)

Upsample by a factor of 2 (resulting in a 1x8 vector) via Bilinear Interpolation
[0 3 3 7]
```

Figure 21: Data of the Exercise 6

Execution:

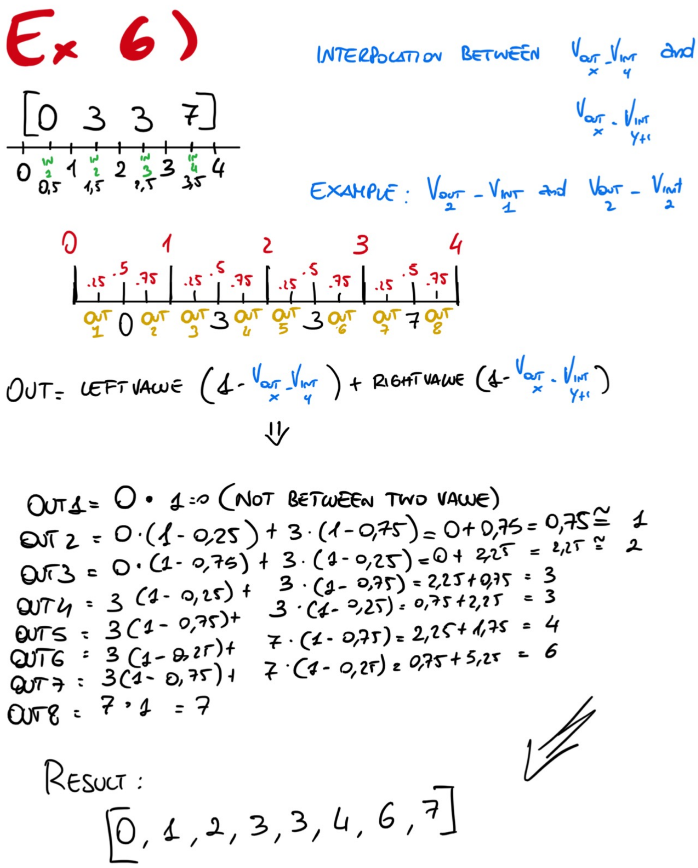

Figure 22: Exercise 6

7 Exercise 7

The data of the following exercise:

```
ex7(matricola)

Image Mat:
[[10 15  7  4]
 [ 7 15 10  1]
 [ 9  6  2  1]
 [ 6  9  1  6]]
```

Figure 23: Data of the Execise 7

Execution:

Ex 7

$$\text{IMAGE} = \begin{bmatrix} 10 & 15 & 7 & 4 \\ 7 & 15 & 10 & 1 \\ 9 & 6 & 2 & 1 \\ 6 & 9 & 1 & 6 \end{bmatrix}$$

$L =$ N° of max pixel intensity value : 256

$$P_n = \frac{\text{Num. pix with intensity } n}{\text{tot pix.}}$$

$n = 0, \dots, L-1$

$$P_1 = \frac{3}{16} = 0{,}1875$$
$$P_2 = \frac{1}{16} = 0{,}0625$$
$$P_6 = \frac{3}{16} = 0{,}1875$$
$$P_4 = \frac{1}{16} = 0{,}0625$$
$$P_7 = \frac{2}{16} = 0{,}125$$
$$P_{10} = \frac{2}{16} = 0{,}125$$
$$P_9 = \frac{2}{16} = 0{,}125$$
$$P_{15} = \frac{2}{16} = 0{,}125$$

rounds down to the nearest integer.

APPLY THE FORMULA :

$$T(k) = \text{Floor}\left[(L-1)\sum_{n=0}^{k} P_n\right]$$

Figure 24: Exercise 7 pt.1

$$P_1 \cdot 255 = 47{,}8125$$
$$P_2 \cdot 255 = 15{,}9375$$
$$P_6 \cdot 255 = 47{,}8125$$
$$P_4 \cdot 255 = 15{,}9375$$
$$P_7 \cdot 255 = 31{,}875$$
$$P_{10} \cdot 255 = 31{,}875$$
$$P_9 \cdot 255 = 31{,}875$$
$$P_{15} \cdot 255 = 31{,}875$$

$$T(1) = \sum_{k=0}^{1} P_k = 47{,}8125$$
$$T(2) = \sum_{k=0}^{2} P_k = 47{,}8125 + 15{,}9375 = 63{,}75$$
$$T(4) = \sum_{k=0}^{4} P_k = T(2) + 15{,}9375 = 79{,}6875$$
$$T(6) = \sum_{k=0}^{6} P_k = T(4) + 47{,}8125 = 127{,}5$$
$$T(7) = \sum_{k=0}^{7} P_k = T(6) + 31{,}875 = 159{,}375$$
$$T(9) = \sum_{k=0}^{9} P_k = T(7) + 31{,}875 = 191{,}25$$
$$T(10) = \sum_{k=0}^{10} P_k = T(9) + 31{,}875 = 223{,}125$$
$$T(15) = \sum_{k=0}^{15} P_k = T(10) + 31{,}875 = 255$$

THE RESULT IS :

$$\longrightarrow \begin{bmatrix} 223{,}125 & 255 & 159{,}375 & 79{,}6875 \\ 159{,}375 & 255 & 223{,}125 & 47{,}8125 \\ 191{,}25 & 127{,}5 & 63{,}75 & 47{,}8125 \\ 127{,}5 & 191{,}25 & 47{,}8125 & 127{,}5 \end{bmatrix}$$

Figure 25: Exercise 7 pt.2

Exercise comment:

The histogram equalization is a technique that try to adjust the image intensities in order to improve the contrast. In the given image, the value of the image is very lower (if we consider that the highest value is 255)! So, we apply the histogram equalization and the result is something that appear with more intensities and that include whole the possible value color range.

8 Exercise 8

The data of the following exercise:

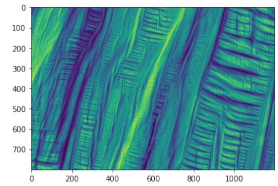

Figure 26: Data of the Exercise 8 pt1

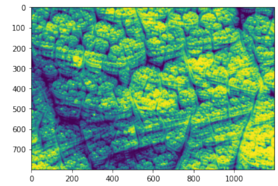

Figure 27: Data of the Exercise 8

Given the images, the Fourier transform allows us to distinguish figures in a very distinct way, for example lines, circles, etc. as these correspond to "patterns" in the frequency domain. These will be used to recognize the characters in the image. Furthermore, we can use the frequency domain to remove noise from the image. For example, if we have an image that has noises and these noises are "bright", we can use a filter to remove them. Applying the fft to the given images, we can notice two main frequencies: one horizontal and one vertical. These frequencies that we visualize correspond to vertical and horizontal lines present in our images. We can also note that these horizontal and vertical frequencies tend to "shift" by angle; in fact, in our images there are more vertical / horizontal lines with different inclinations. We can also use a low-pass filter to remove any noise. A low pass filter allows only low frequencies to pass. By low frequencies in an image we mean pixel values that change slowly. Since, in fact, this filter only allows the passage of low frequencies, therefore the contents of high frequencies such as noises, will be blocked. We can also use a high pass filter to identify changes in an image. Unlike the low pass filter, a high pass filter allows only high frequencies to pass through. By high frequencies in an image we mean pixel values that change drastically. For example, the edges of images or the edges of a figure etc. The output of this filter allows you to capture edges within an image which could then be used to sharpen the original image with the correct overlay calculation. This will increase the sharpness of the image by making the edges lighter.

Summarizing with the low pass filter we will preserve the general information of the image and we will eliminate the noise, with the high pass filter instead we will intensify the changes within the image.

Entropy, on the other hand, allows us to evaluate the amount of uncertainty about an event associated with a given probability distribution. As the noise in a given image increases, we will notice that the entropy increases and consequently the "events" will become less predictable. When processing images, entropy can be used for the classification of repeating textures or elements. Generally, an image with low entropy is more homogeneous than an image with high entropy.

9 Exercise 9

The first thing that I want to explain is a small introduction of the Noise. The noise is always present to every images due to image acquisition or quantization error or during the transmission ecc. We want to remove noise but keep up the details. We can have different type of filter. In particular I want to highlight the Gaussian Filter and Median Filter for noise reduction. Before starts to speak to these filters, I want to say that we can have different type of filter that operate into different type of noise. For example the linear filter like a Gaussian Filter try to eliminate the noise as "Gaussian Noise". This filter tends to blur the sharp edges, remove the lines and other image details. The Gaussian filter uses the Gaussian function for determine the value of the pixel. Specially the convolution of this filter brings the value of each pixel to converge harmonically with the values of the neighbors of that pixel. Due to this blur effect, the linear filter is using less for the noise reduction. Sometimes is used as a base for a non-linear noise reduction filter. The other Median Filter instead is a non-linear filter that is used for reduce a noise like salt and papper. This is a simple and good non-linear filter based on the order statistic. This filter could cause a small blur of the edges. With this filter the value of the output pixel is determinated by the median of the neighbors of that pixel. For this reason this filter is not sensitive to the extreme values (that have a high difference in terms of value between his neighbors). The first thing that we will do applying this filter is sorting all the pixel values in ascending order. After that substitute the pixel with the middle pixel value. This type of filter doesen't works well when there are a lot of impulse noise.

10 Exercise 10

The data of the following exercise:

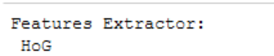

Figure 28: Data of the Exercise 10

Plot result of the following exercise:

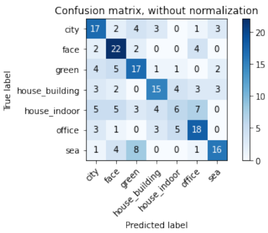

Figure 29: SIFT Confusion Matrix

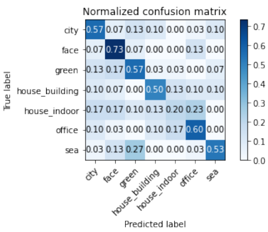

Figure 30: SIFT Normalized Confusion Matrix

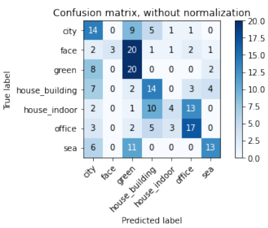

Figure 31: HOG Confusion Matrix

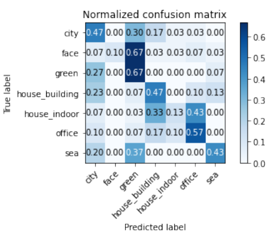

Figure 32: HOG Normalized Confusion Matrix

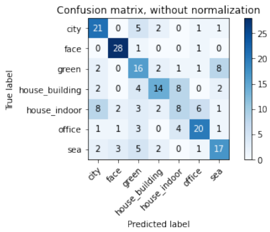

Figure 33: FAST Confusion Matrix

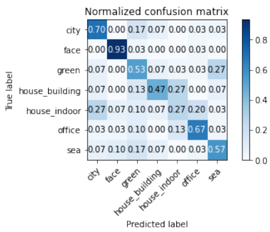

Figure 34: FAST Normalized Confusion Matrix

Plot the accuracy:

`accuracy score: 0.529`

Figure 35: SIFT Accuracy

`accuracy score: 0.405`

Figure 36: HOG Accuracy

`accuracy score: 0.590`

Figure 37: FAST Accuracy

The BoVW model can be applied to image classification or retrieval, treating image characteristics as words. To represent an image using the BoVW model, an image can be treated as a document. Likewise, the ”words” in the images must also be defined. To achieve this, the following three steps are generally performed: feature detection, feature description, and code register generation. After feature detection, each image is extracted from several patches. Feature representation methods deal with how to represent patches as numeric vectors, called feature descriptors. A good descriptor should have the ability to handle intensity, rotation, scale, and related variations to some degree. For this exercise, 3 different descriptors have been chosen: **SIFT, HOG and FAST**.

SIFT:
The SIFT key points of the objects are first extracted from a series of reference images and then stored in a “database”. An object is recognized in a new image by individually comparing each feature from the new image in this database and finding candidate matching features based on the Euclidean distance of their feature vectors. From the complete set of matches, subsets of key points are identified, which agree on the object and its position, scale and orientation. A consistent clustering is performed and then the probability that a particular set of characteristics indicates the presence of an object is calculated.

HOG:
HOG represents the histogram of oriented gradients, is a descriptor of characteristics and is used for image processing for the purpose of detecting objects. This method is similar to that of edge orientation histograms, scale-invariant

transformation descriptors, and shape contexts, but differs in that it is calculated on a dense grid of evenly spaced cells and uses superimposed local contrast normalization for better accuracy.

FAST:
Features from accelerated segment test (FAST) is a method of detecting angles, to extract characteristic points etc ... The most promising advantage of the FAST corner detector is its calculation efficiency. It is actually faster than many other well-known feature extraction methods, such as the Gaussian Difference (DoG) used by the SIFT and Harris detectors. One of the weaknesses of this method is that it is not rotation invariant (meaning it will not work if the target is rotated relative to the reference image). The key points extracted will then have to be extracted from some other descriptor as this method is only a detector and the robustness of the match will depend on it.

Comparing these three methods with the results obtained, we can see that:
with the SIFT method the accuracy score is 0.529, with the HOG method it is 0.405 and with the FAST method it is 0.590. Looking at this in more detail, we can see that the FAST method is efficient in detecting "faces", but not in "house indoor"; the HOG method performs well in detecting the green, but not in "house indoor" and "faces"; the SIFT method is performing in detecting "faces", but this too, like FAST, is not performing in detecting "house indoors". We can also note from these results that, as far as the "house indoor" category is concerned, none of the previous methods managed to extrapolate the features to the best. We have different performances for different methods and categories, this shows us how different methods are able to better "understand" certain categories and better extrapolate the features.

11 Exercise 11

The data of the following exercise:

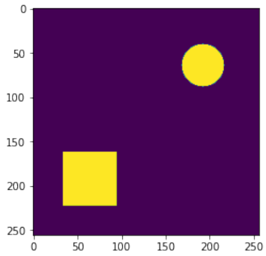

Figure 38: Data of the Exercise 11 pt1

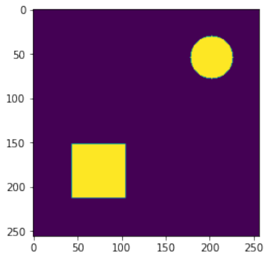

Figure 39: Data of the Exercise 11 pt2

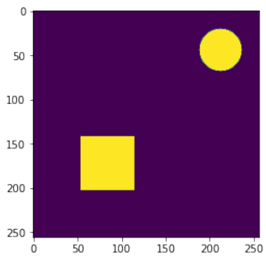

Figure 40: Data of the Exercise 11 pt3

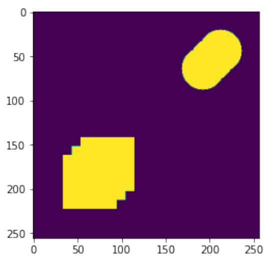

Figure 41: Data of the Exercise 11 pt4

Plot result of the following exercise:

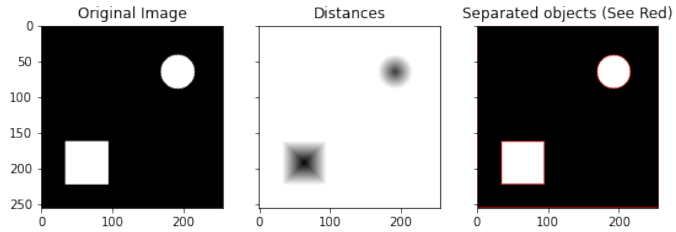

Figure 42: Segmentation

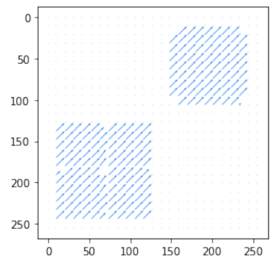

Figure 43: Optical Flow

(See the code) The segmentation of an image is a process where the image is divided into several segments by grouping regions of pixels with some characteristics. I decided to use the Watersheld method for this exercise. This method is often used for the separation of similar objects within the image (which may or may not touch each other). This algorithm is very fast and efficient, but not easy to get variety of regions for multiple segmentations.
In the implemented code, the first thing we are going to do is a conversion of the image to grayscale. Subsequently we will perform a "threshold" on the image in order to perfectly divide the white regions from the black regions. For the threshold, a minimum threshold value of 120 was used, in order to be sure to be immune to any image disturbances (such as, for example, an image with points that are not perfectly black). Subsequently, we will use an opening and a morphological dilate in order to eliminate all possible "white noises" that we might find on the image. Furthermore, I decided to expand the result of the previous operations in such a way as to increase the "boundary" of the elements, in order to divide even better what is considered "background" and what is considered "object".
Now let's apply a distanceTransform to the result of the "opening" operation performed previously and perform a threshold on the result with a threshold equal to 70 percent of the maximum distanceTransform value. At this point we have the foreground elements on the one hand and the background elements on the other. I'm going to apply a subtraction between these two arrays in such a way as to find those points that I couldn't distinguish, putting them in an array called "unknown". At this point we will apply the method we have chosen called "watershed" and as a result we will have an image that all has "-1" on the edges of the objects that have been identified.

Optical Flow (calcOpticalFlowFarneback):
After an analysis using the optical flow, it turned out that both the circle and the square move from bottom to top and from left to right.
The optical flow is known as the model of the movement of objects between two consecutive frames.
The Gunnar-Farneback was chosen as the model. This type of model, unlike the Lucas Kanade which works only on the corner points detected by the Shi-Tomasi algorithm, looks at all the points and detects the intensity changes of the pixels between the frames. First, this algorithm calculates the amplitude and direction of the optical flux and then displays the angle (direction) of the flux by hue and the distance (magnitude) of the flux.

12 Exercise 12

The data of the following exercise:

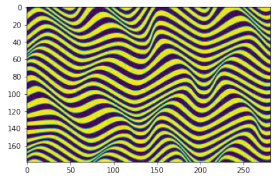

Figure 44: Data of the Exercise 12 pt1

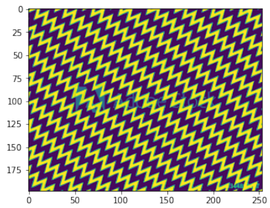

Figure 45: Data of the Exercise 12 pt2

Plot result of the following exercise:

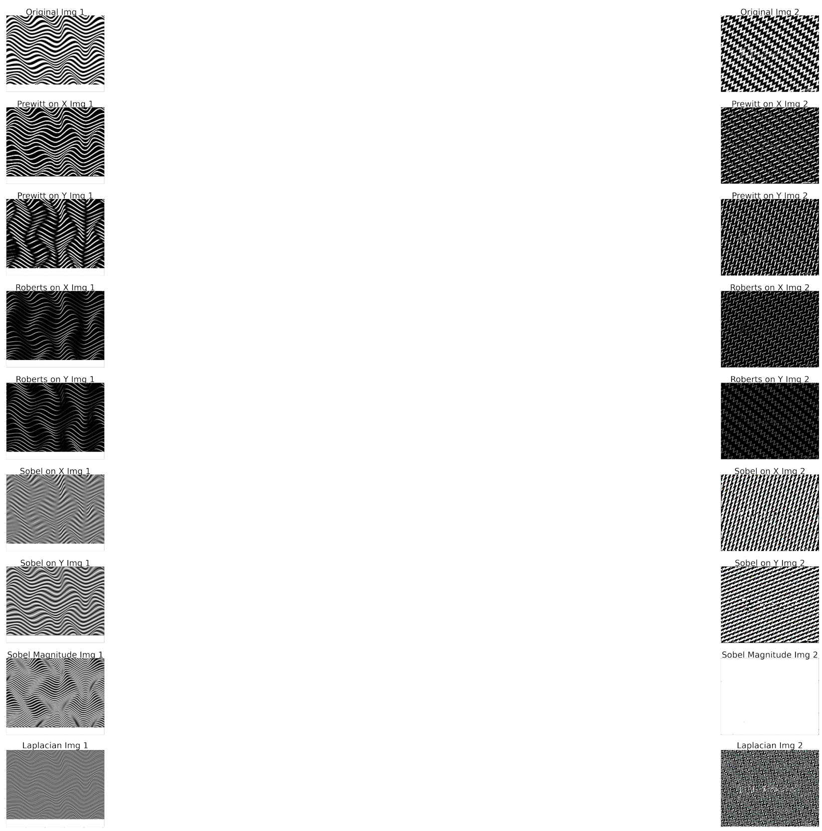

Figure 46: Filters

To answer this question I decided to use: Prewitt filter, Roberts filter, Sobel filter and Laplacian filter. All these filters have certain characteristics and produce different results to each other.
The Prewitt filter is widely used in the image processing process. In fact, this is used for edge detection. Prewitt is a discrete differentiation operator that calculates the approximation of the gradient of the image intensity function. The Roberts filter is also widely used for edge detection but, the idea behind the Roberts filter is to approximate the gradient of an image through a discrete differentiation obtained by calculating the sum of the squares of the differences between diagonally adjacent pixels. . The Sobel filter is also used for edge detection, but unlike the previous ones, an image is created here that emphasizes the edges. Finally we have the Laplacian filter, which is also generally used for edge detection since it tends to highlight regions of rapid intensity change.
The first thing that differentiates Laplacian from Sobel or Prewitt or Roberts is the use of only one kernel, in fact the laplacian like the Gaussian filter "needs" only 1 kernel unlike the other three which need a kernel for each axis.
We can also note from the images generated, that the Prewitt identifies the vertical and horizontal edges better than the Roberts which will tend to "make them smaller" and even better than the Sobel, in which they will be too emphasized.

13 Exercise 13

When you use a graphic image and want to perform ”resize” operations on it, it is necessary to generate a new image with a greater or lesser number of pixels as needed. In the event of a decrease in pixels, usually, there is a visible loss of quality.
The methods chosen for resizing the image to decrease the number of pixels are shown below: lanczos, bilinear, gaussian pyramid and mipmap.

Lanczos:
The Lanczos method provides better properties when it comes to preserving details and minimizing artifacts and aliasing. The process is very complex. The input samples are filtered through a Lanczos kernel to reconstruct them once the process is finished. The interpolation is implemented using a mathematical function known as the Sync function. This method has an interpolation result similar to the bicubic interpolation method.
The effect that this method produces is a sort of ring instead of a blur effect; This allows you to increase the perceived sharpness of the image.

Bilinear:
The Bilinear method is a simpler and faster method than the one discussed previously. This method allows you to reduce the visual distortion caused by resizing an image to a non-integral zoom factor. Bilinear interpolation takes into account the nearest 2x2 pixel values surrounding a given pixel. Then the weighted average of those pixels is calculated in order to obtain the final interpolated value.
By applying this method the result we will have will have a uniform appearance (this method is much more uniform than the closest neighbor method).

Figure 47: None

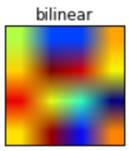

Figure 48: Bilinear

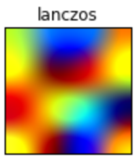

Figure 49: Lanczos

Gaussian Pyramid:

With this method we will first have to blur the input image with the GaussainBlur method and then perform the downsample. Usually the downsample has a reduction factor equal to 2. This blur and downsample procedure is repeated on the resulting image several times. This implies that with each application the image will become smaller and smaller, but will have greater uniformity. In addition, each pixel contains a local average corresponding to a neighborhood pixel at a lower level of the pyramid.

Mipmap:

The mipmap method corresponds to pre-calculated and optimized image sequences, each of which is represented at a progressively lower resolution than the previous one. The height and width of each image, or layer, in the mipmap is a power of two smaller than the previous layer.

This method is generally used for situations where the distance between an object and the camera can change. As the object moves away from the camera, the texture of the object will appear on the screen smaller than its actual resolution; in other words, there will be more than one texel (pixel of a texture map) per screen pixel. The texture will need to be resized in a process called minification filtering, which often requires the application to sample multiple texels to decide the color of a pixel.

Content loss:

In order to understand the content loss obtained from each method I decided to apply the mean-square-error. More precisely for the Bilinear and Lanczos methods I decided to perform a downscale and then a zoom and then go to find the difference between these two (original image and downsample image + zoom) by calculating the MSE. For the Gaussian pyramid instead, I went to save the residual (image - gaussianblur) before doing any downsample. Immediately after applying that, I applied the zoom adding it to the residual of the image. At this point I calculated the MSE between the original image and the downsample + zoom result.

For the Mipmap I performed a slightly different technique, in fact I performed the downsample and then a zoom to then subtract the result from the original image, then calculating the MSE.

Thinking instead from another perspective, in order to predict which image will lose more content information before applying any downsample method, I can first calculate the entropy on the image and analyze the result. In fact, it is possible that images with high entropy values tend to lose more information than images with lower entropy.