Assignment image processing

Fabrizio Casadei - 1952529

March-April 2021

Exercise 1

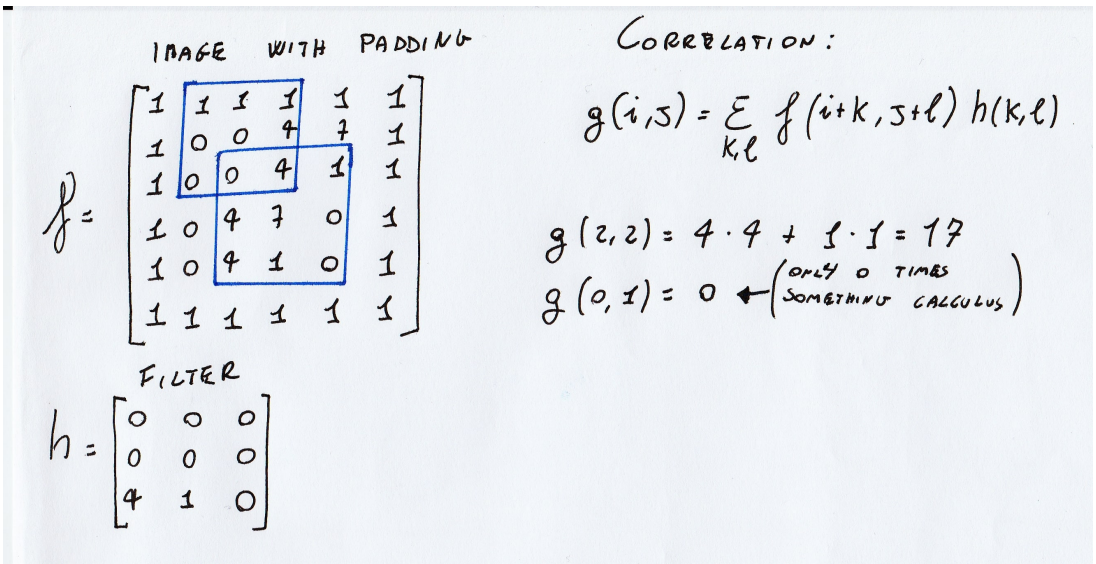
Data for the exercise:
Operation: Correlation
Image:

0	0	4	7
0	0	4	1
0	4	7	0
0	4	1	0

Kernel:

0	0	0
0	0	0
4	1	0

Paddings: Constant padding 1
First Coords: 2 2
Second Coords: 0 1

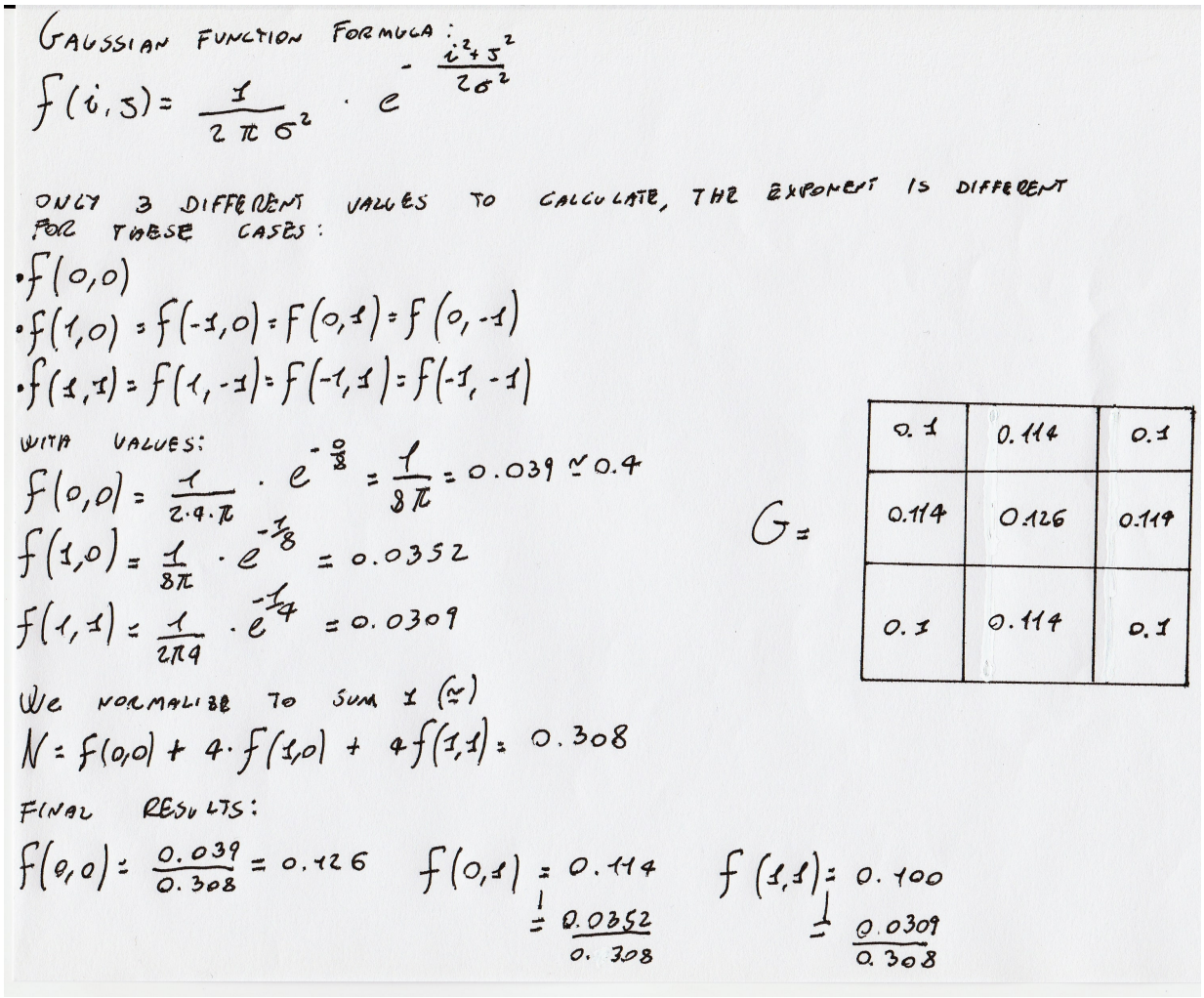


Discussion on padding:

The constant padding allows of having an output image with the same dimension of the original image as input in order to take into account boundaries.However doesn't add information from the image, like for example the *edge* or *symmetric* padding. This feature, in some cases, could be a drawback, because we are not using data distribution but a constant to move all the boundaries in that direction,

Exercise 2

Data for the exercise:
Gaussian filter dimension: 3×3, $\sigma = 2$



GAUSSIAN FUNCTION FORMULA:

$$f(i,j) = \frac{1}{2\pi\sigma^2} \cdot e^{-\frac{i^2+j^2}{2\sigma^2}}$$

ONLY 3 DIFFERENT VALUES TO CALCULATE, THE EXPONENT IS DIFFERENT FOR THESE CASES:

- $f(0,0)$
- $f(1,0) = f(-1,0) = f(0,1) = f(0,-1)$
- $f(1,1) = f(1,-1) = f(-1,1) = f(-1,-1)$

WITH VALUES:

$f(0,0) = \frac{1}{2 \cdot 4 \cdot \pi} \cdot e^{-\frac{0}{8}} = \frac{1}{8\pi} = 0.039 \simeq 0.4$

$f(1,0) = \frac{1}{8\pi} \cdot e^{-\frac{1}{8}} = 0.0352$

$f(1,1) = \frac{1}{2\pi 4} \cdot e^{-\frac{1}{4}} = 0.0309$

We NORMALISE TO SUM 1 ($\simeq$)

$N = f(0,0) + 4 \cdot f(1,0) + 4 f(1,1) = 0.308$

FINAL RESULTS:

$f(0,0) = \frac{0.039}{0.308} = 0.126$ $\quad f(0,1) = 0.114 \doteq \frac{0.0352}{0.308}$ $\quad f(1,1) = 0.100 \doteq \frac{0.0309}{0.308}$

$G =$

0.1	0.114	0.1
0.114	0.126	0.114
0.1	0.114	0.1

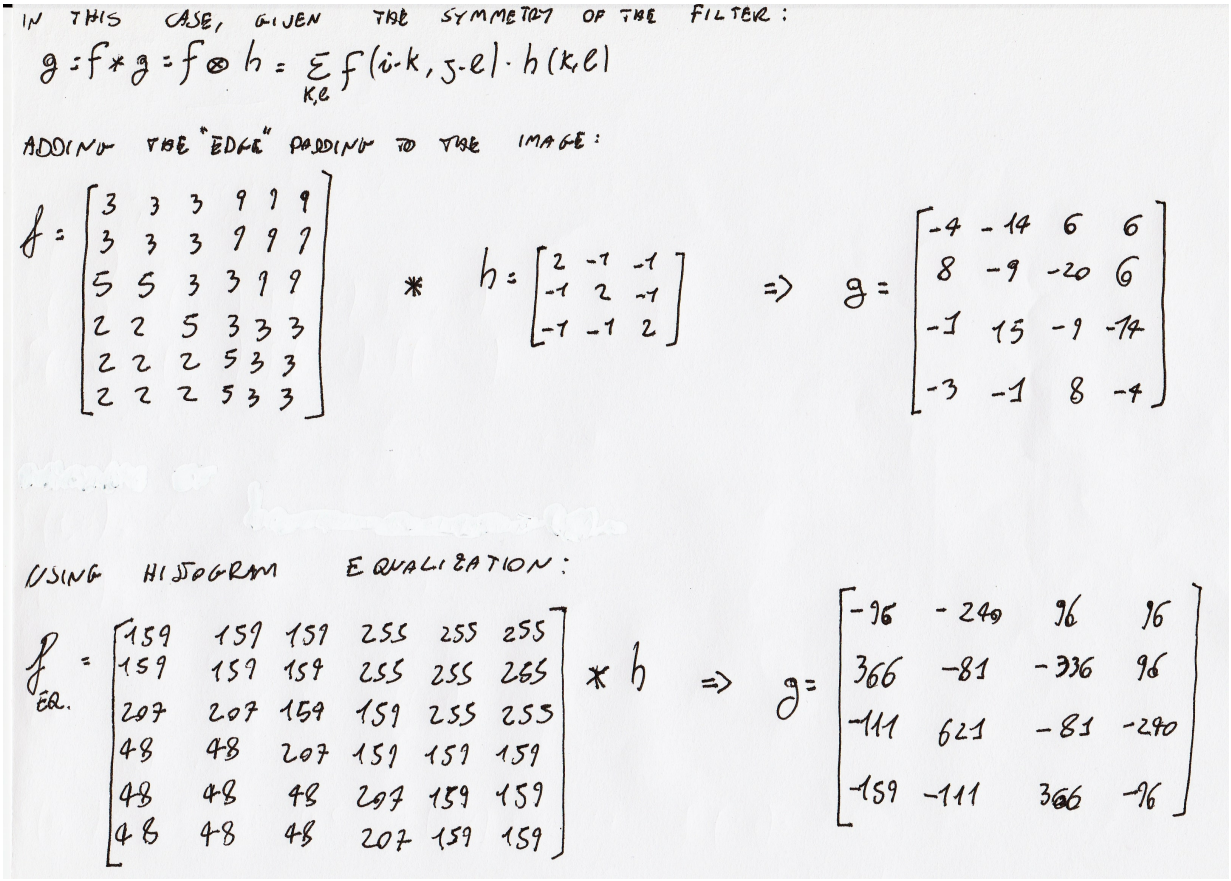
Exercise 3

Data for the exercise:
Image:

3	3	9	9
5	3	3	9
2	5	3	3
2	2	5	3

Kernel:

2	-1	-1
-1	2	-1
-1	-1	2



Discussion:

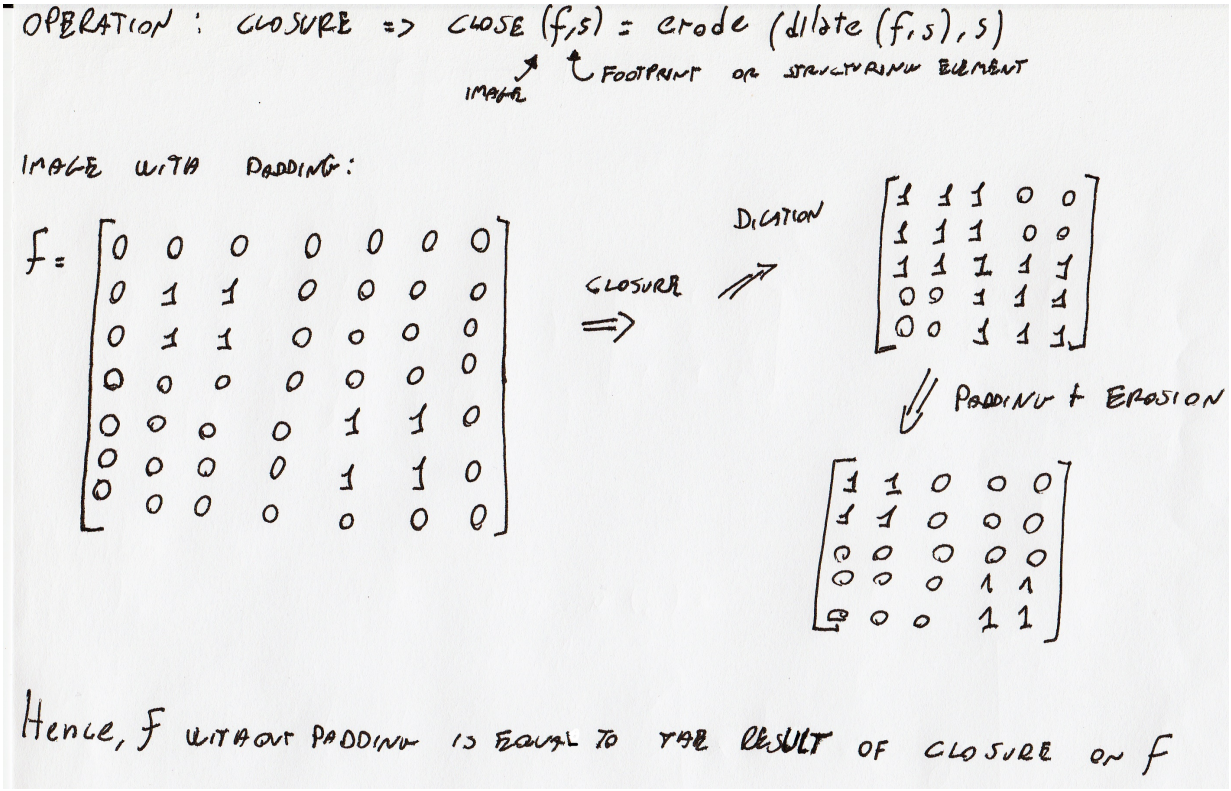
In this exercise, we have used the "edge" padding, in order to maintain data information from the image and having the same size as the input after the convolution. As it's possible to notice the summation of the values in the filter is equal to zero, therefore, we can establish that is a high pass filter, like the *Laplacian* for example. This one is used to detect line, more in the specific activates diagonal lines at $-45°$. this kind of filter highlight diagonal lines with higher values, for a better representation the histogram equalization process has been computed.

Exercise 4

Data for the exercise:
Image:

1	1	0	0	0
1	1	0	0	0
0	0	0	0	0
0	0	0	1	1
0	0	0	1	1

Operation: closure



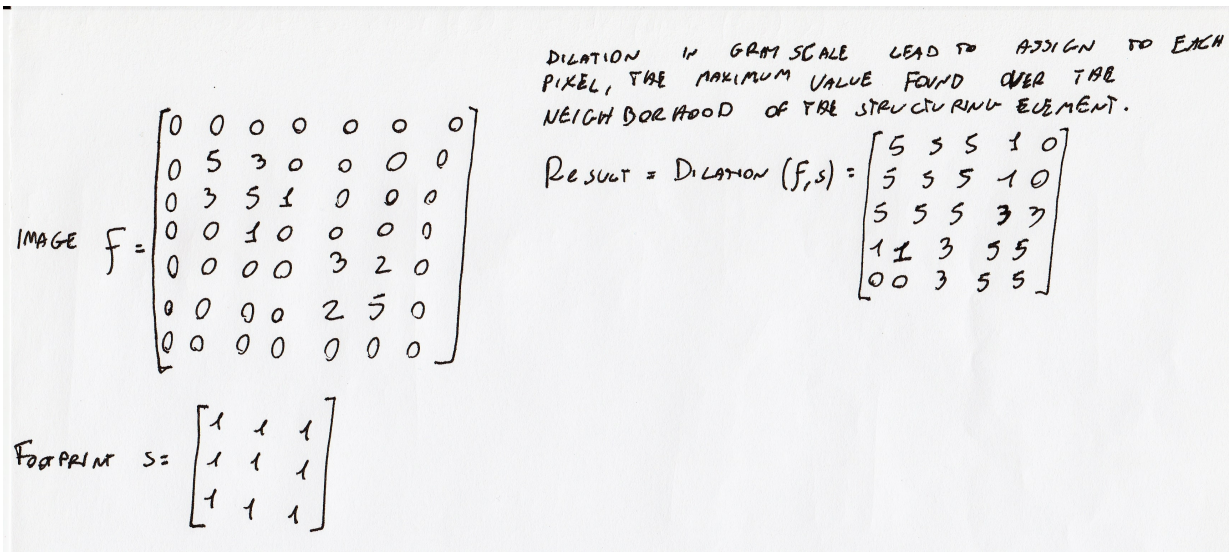
Discussion:
In this case, has been used a constant padding (both 1 and 0 lead to the same result), just to have equal output shape respect to the input.
After applying dilation and erosion for the closure operator, we have returned the same initial image, hence, we can conclude which doesn't lead to advantages. This because no noise can be noticed, two quadratic shapes are present in the top left and bottom right borders.

Exercise 5

Data for the exercise:
Image:

5	3	0	0	0
3	5	1	0	0
0	1	0	0	0
0	0	0	3	2
0	0	0	2	5

Operation: Dilation(3,3)

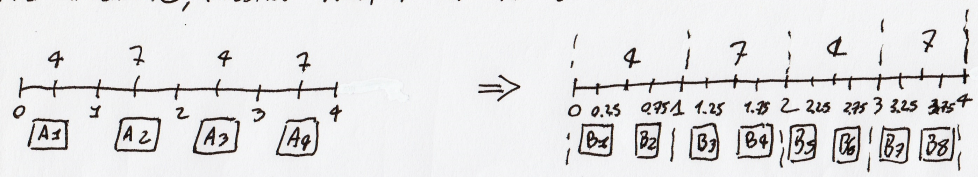


Discussion:
To maintain consistency with sizes of the input and the output we choose the constant padding, and zero as a constant, for not altering the final result. The dilation makes objects more visible, and fills small "holes", in the image. In this case, has connected 2 groups located on vertices (bottom right and top left), therefore this morphological operator can be used to create new lines and connections.

Exercise 6

Data for the exercise:
image: [4 7 4 7]
assignment: Upsample by a factor of 2 (resulting in a 1x8 vector) via linear Interpolation

USING LINEAR INTERPOLATION WE MAINTAIN THE RANGE BUT WE DOUBLE (FOR THIS CASE) THE INTERVAL, PASSING FROM 4 UP TO 8 VALUES.



FROM B_1 TO B_8 ARE THE NEW VALUES.

$B_1 = 4 \cdot 1$ (NO BETWEEN 2 VALUES)

$B_2 = 4\left(1 - d_{B_2\text{-}A_1}\right) + 7\left(1 - d_{B_2\text{-}A_2}\right)$ WITH $d_{B_2\text{-}A_1}$ AND $d_{B_2\text{-}A_2}$ DISTANCES FROM B_2 TO A_1 AND A_2.

$= 4(1 - 0.25) + 7(1 - 0.75) = 3 + 1.75 \simeq 5$

$B_3 = 4(1 - 0.75) + 7(1 - 0.25) = 1 + 5.25 \simeq 6$

$B_4 = 7(1 - 0.25) + 4(1 - 0.75) = 5.25 + 1 \simeq 6$

$B_5 = 7(1 - 0.75) + 4(1 - 0.25) = 1.75 + 3 \simeq 5$

$B_6 = 4(1 - 0.25) + 7(1 - 0.75) = 3 + 1.75 \simeq 5$

$B_7 = 4(1 - 0.75) + 7(1 - 0.25) = 1 + 5.25 \simeq 6$

$B_8 = 7 \cdot 1 = 7$

$\Rightarrow [4, 5, 6, 6, 5, 5, 6, 7]$

Exercise 7

Data for the exercise:
image:

10	11	10	4
10	11	10	1
9	5	2	5
2	9	5	2

HISTOGRAM EQUALIZATION FORMULAS:

$$cdf(i) = \sum_{j=0}^{i} P_x(x=j)$$

$$h(v) = \text{ROUND}\left(\frac{cdf(v) - cdf_{min}}{(M \times N) - cdf_{min}} \times (L-1)\right)$$

WITH $M = N = 8$ AND $L = 256$

FIRST WE COMPUTE THE LOOKUP TABLE

VAL	COUNT
1	1
2	3
4	1
5	3
9	2
10	4
11	2

CDF ⇒

VAL	CDF
1	1
2	4
4	5
5	8
9	10
10	14
11	16

h(v) ⇒

VAL	h(VAL)
1	0
2	51
4	68
5	119
9	153
10	221
11	255

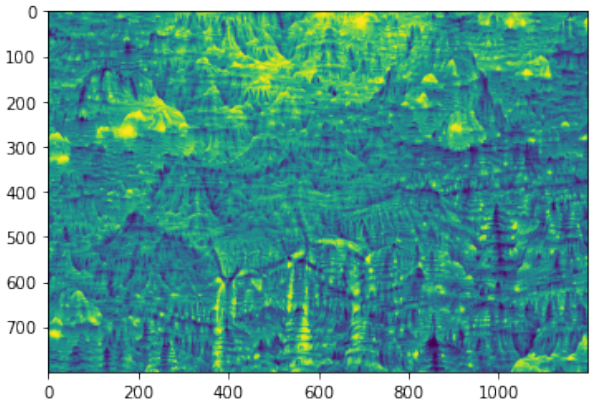
HENCE, CDF MIN IS 1, AND THE FINAL RESULT IS:

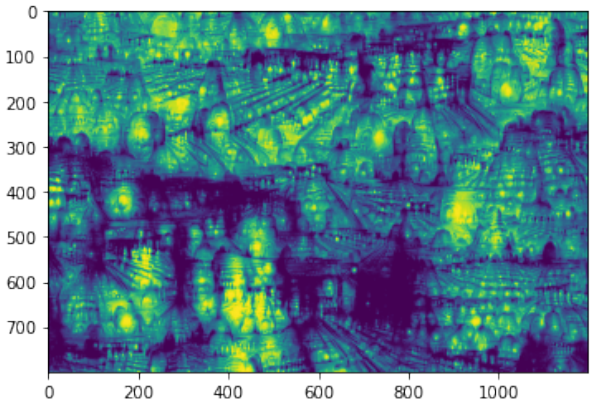
$$f_{EQ} = \begin{bmatrix} 221 & 255 & 221 & 68 \\ 221 & 255 & 221 & 0 \\ 153 & 119 & 51 & 119 \\ 51 & 153 & 119 & 51 \end{bmatrix}$$

Discussion:
We can appreciate how, after the histogram equalization, the values of the pixels are well spread into the whole grey-scale range used (256), which means a higher contrast between pixels. This method is very useful to have a sharper representation from images where pixel values are not well spread and are in a small range of values, leading to smooth regions.

Exercise 8

Data for the exercise:
images:





Both *Entropy* and *Fourier Transform* are useful tools to extract features, highlight objects, segmentation of the image, etc, and can be used on the provided images, let's see how.
Whether we want to extract high or low frequencies from an image, it's enough to transform in the frequency domain the image through the *Fourier Transform* or *Fast Fourier Transform*, centre the frequencies and multiplying by a filter. With a Low pass filter, we extract the lower frequencies, useful operation to smooth the image. Using a band pass filter in a certain frequency interval it's possible to define a kind of edge extractor, Instead, the high pass filter can be used to emphasize the details of an image, making sharper the image. Then *entropy* measures the grade of disorder or uncertainty and gives a clear value of how complex is to predict information or how much data in needed to represent a bunch of information. Well spread images reach a very high level of entropy while image defined in low range of intensity with a reduced set of spatial constraints leads to a low level of entropy. Shifting an entropy disk kernel over the image (similar to the convolution operation), we compute the entropy. Entropy can be used for example to segment images, can highlight the curvature of an object useful to extract features and clusters. these two powerful tools can be used for example in the following case:

- sharp details about colour separations on above images, computing the *entropy* with an appropriate disk size and then apply the *Gaussian High pass filter* to sharp the smoothed image from the first step, useful for *segmentation* tasks,
- it's possible to highlight or remove "homogeneous" parts of the image creating a mask using the entropy and normalizing values, after that, can be useful to smooth the contours using the Low pass filter. Finally the mask it's multiplied by the original image.
- Whether after the application of the entropy disk, especially with one having a too small size concerning the image, it's possible to reduce the noise using a *Low pass filter*.

Exercise 9

Noise is a very big problem in all the *computer vision* tasks, like for example the application of the *canny edge extractor*. The *Sobel* filter is used for the derivatives in order to retrieve the magnitude and orientation of the gradient at each pixel, and then the algorithm continues with the non-maximum suppression, the thresholding and the linking to connect edges. The *Sobel* filter doesn't act very well against the noise, and this can lead to bad results, especially for the real world images, cause the probability of having noise presence is high. Multiplication of an image in frequency by the high pass filter, like the derivatives, highlights also the noise, so you need to denoise the image first. But how get the best result from the *Sobel* line detection for example? we have to perform some denoising operations in order to clean first the image. We can use low pass filters to smooth the image, applying for example a *Gaussian* filter, with the standard deviation chosen accordingly with the noise level and the desired smoothing effect. In general, this kind of solutions called neighbourhood processing, are done using linear filters as *Gaussian* or the *Box* (not always well suited because leaves some artefacts on the resulting image). In addition, to add blurring, *neighbourhood operators* can be used to filter images in order to perform local tone adjustments, sharpen details, accentuate edges or as said remove/reduce noise.
in fact, smoothing kernels are often used to reduce high-frequency noise.
Also to conclude this first solution we highlight the possibility of using a filter called, *median filter*. This one can be used with a better result than the *Gaussian* filter in respect to the elimination of random noise. However in this case we are applying a non-linear filter.

One another possible manner to reduce the noise in an image is to use *morphological operators*, from the primitive ones like *erosion* and *dilation*, to the combined: *opening* and *closure*. To remove the so called *salt and pepper noise*, noisy holes distributed in the input image, the combination of *erosion* and *opening*, depending on the input image, can lead to a very well denoised image.
Opening and *closure* are inclined to leave large regions and smooth boundaries unaffected and at the same time they trying to remove small objects or holes and smoothing boundaries.

Exercise 10

Data for the exercise:
Features Extractor: Harris

In this exercise we have implemented *Bag of words model*, exploiting a bag of visual words as an occurrence counts of vocabulary about local image features. These features are extracted following these methods: *SIFT*, *Harris corner detector* and *FAST*. The images are first converted into greyscale and their sized reduced, then for the Harris we just extract the corners, and for both *SIFT* and *FAST* we also define descriptors. Below, we present visually the result obtained extracting features.

(a) Sampled image

(b) Features by SIFT

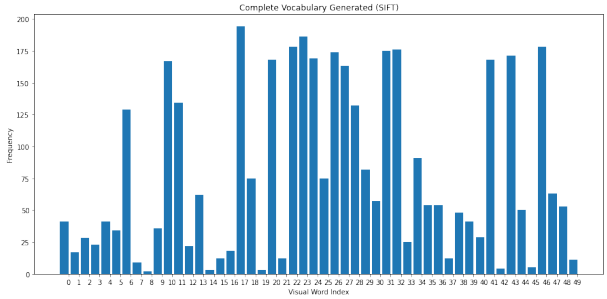
(a) Features by Harris

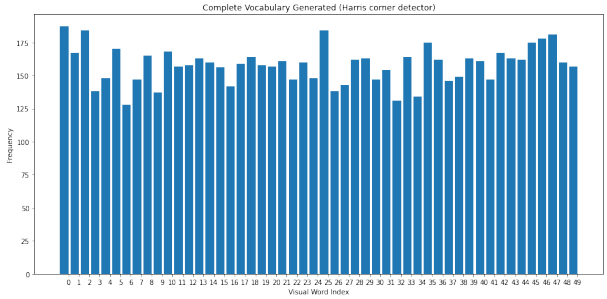
(b) Features by FAST

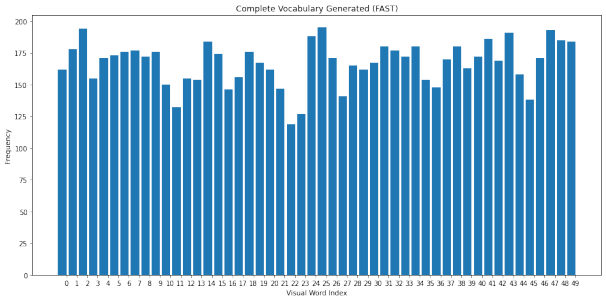
The number of these features depends also by several parameters used after in the detection process. For example, in the case of the Harris has been chosen a **non maximal suppresion** threshold equal to 0.1, used to filter probabilities come out. We can quantify these features as follow:

- *Harris corner detector*, 218
- *Scale Invariant Feature Transform*, 135
- *Features from Accelerated Segment Test*, 403

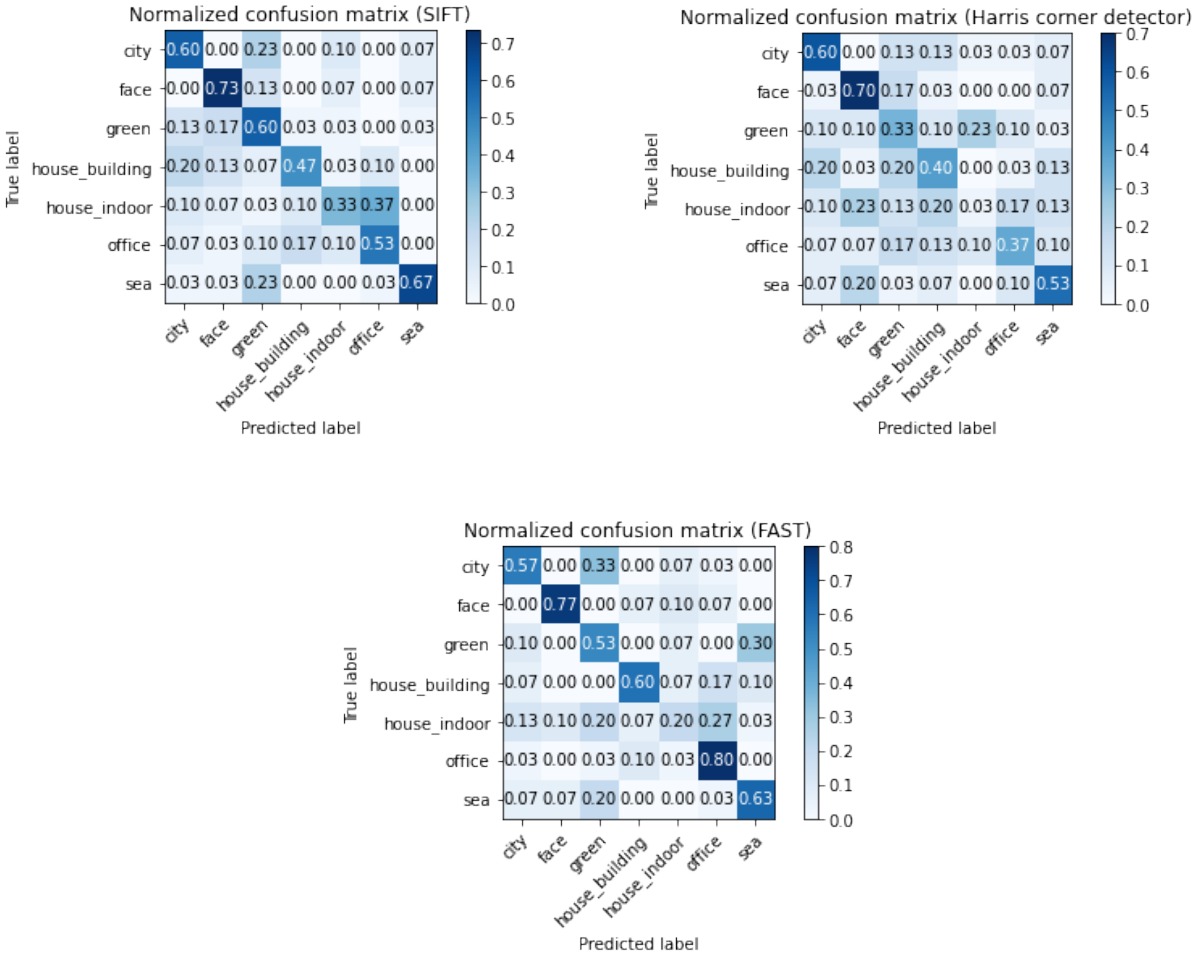
After this, we have clustered the data defining a **Kmeans** model for the image features, normalized those ones to present the histogram of word frequencies and train/predict results from an **SVM** model.
Histograms results are the following:







In the end, we have evaluated the 3 features extractors on a test-set, and plotted the corresponding**confusion matrix**.



The accuracy results that can be derived from the diagonal of these matrices and times spent on feature extraction are:

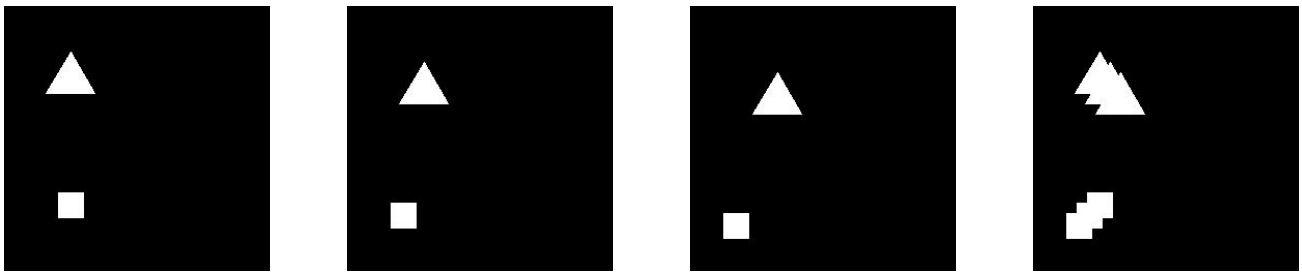
- *Harris corner detector* accuracy = 42.4% , time for features detection: 11.37 [s]
- *Scale Invariant Feature Transform* accuracy = 56.2%, time for features detection and building of descriptors: 26.07 [s]
- *Features from Accelerated Segment Test* accuracy = 58.6%, time for features detection and building of descriptors: 30.74 [s]

FAST can be seen as a kind of corner detector used for real-time application. In this execution has been set a low threshold for the detection of corners, and as result has been detected more than 3 times the ones found by *SIFT* with

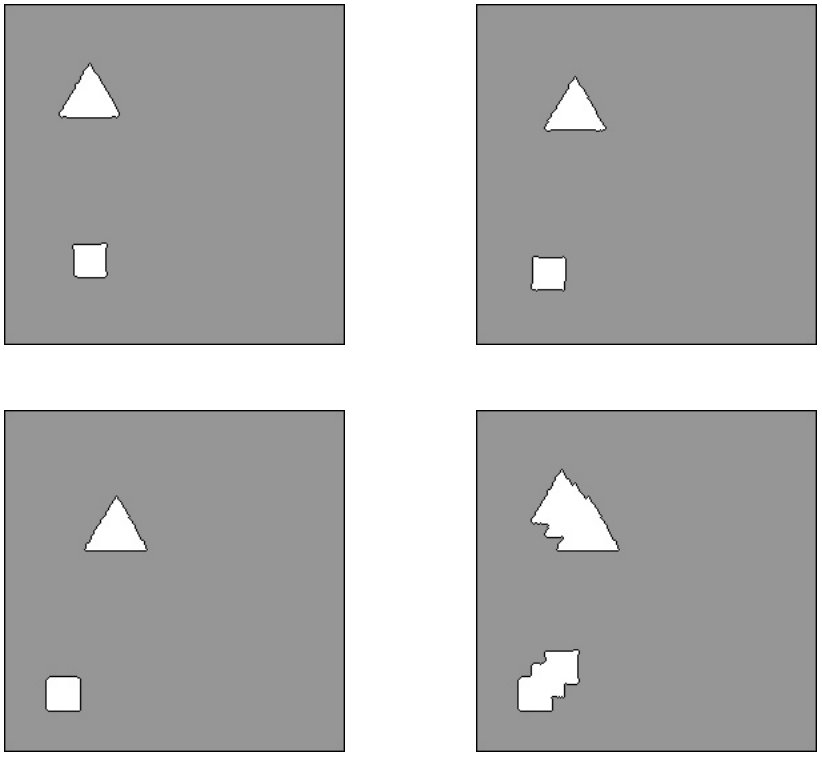
almost the same time. Moreover, the accuracy reach with this method is slightly greater than the accuracy from SIFT, Which can detect both corners and edges. This data gives a clear data of the optimization under the computations made by *FAST*, that consider circle patches of an image and the intensity values inside these.
The main clue is under which pixels inside the patch analyze for first, and if it's suitable to going in depth with the others, or passing to another patch.
If we consider the *Harris corner detector* instead, we reach even better results in terms of time, but no descriptor has been built, and using these "raw" features, also lead to have the worst result in terms of accuracy. *SIFT* reach better results detecting a fewer number of key-points used for the descriptors. *SIFT* it's not only invariant to rotation, like the *Harris*, but it's also unalterable to scale, intensity changes and it's robust to affine transformation and noise.

Exercise 11

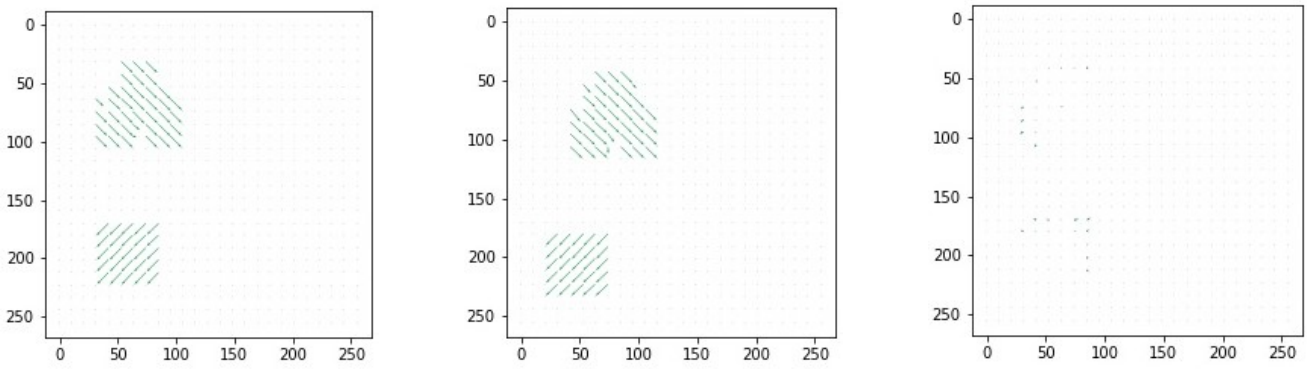
Data for the exercise:
Images:



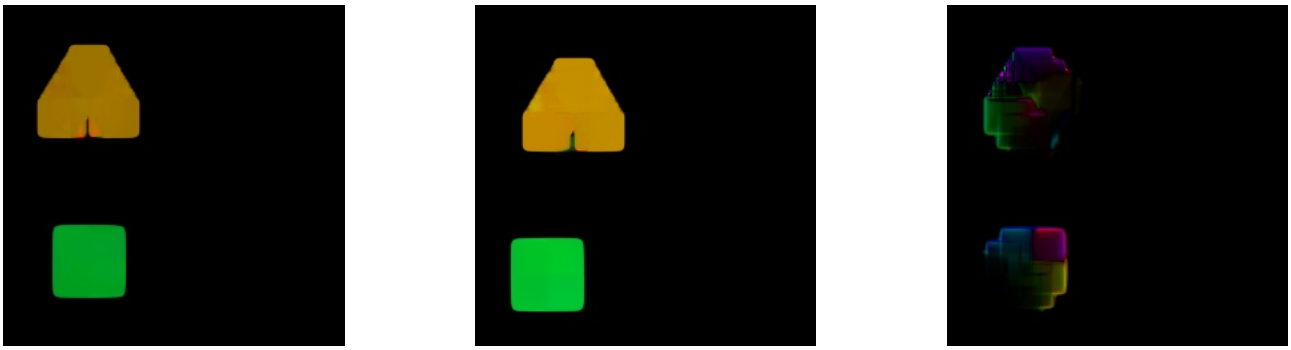
The *segmentation* is a very important task in the computer vision field, whose aim is to partition an image into subgroups of pixels in order to reduce the complexity of an image. For segmenting the given images, we have started using a threshold, the simplest method to segment, that partitions pixels based on the intensity. After this, are been used morphological operations to define the markers or labels, used to separate the most evident object from the rest of not important elements. In the specific, has been used the *erosion* to define the foreground of the image (quads and triangles), and the *dilation* for the background. Finally, the *Watershed* technique has been applied, which exploits the segmentation through topological reasoning.
Even if from the given images we have reached good results, defining an appreciable level of delimitation, this method suffers against real-world images. In fact, is very sensitive to noise, and pre-operations for denoising are sometimes required. Also irrelevant detailed, or smoothed images can cause *oversegmentation*. The performances are strictly connected to the definition of markers. On the other hand, if we define good markers for background and foreground features, this allows us to sharply segment images in a correct manner, with a very fast method.



The second part of this exercise we have performed the *optical flow* on the given images. the *optical flow* is used to recover image motion at each pixel from Spatio-temporal image brightness variations. From the results below, we can note that frame by frame the triangle is moving to the centre of the image, and the square toward the left-bottom corner. This *optical flow* has been computed using the textitGunnar-Farneback technique, which is a global method that evaluates a motion based on the variation of quadratic polynomial representations of frames.



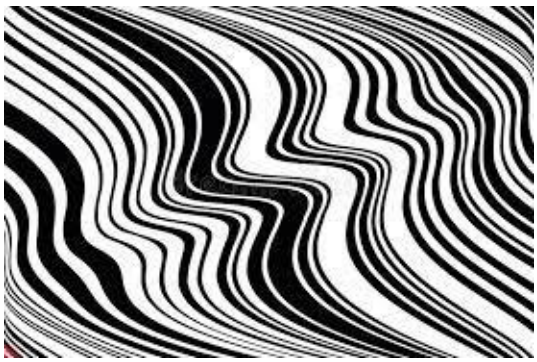
Vectorial optical flow

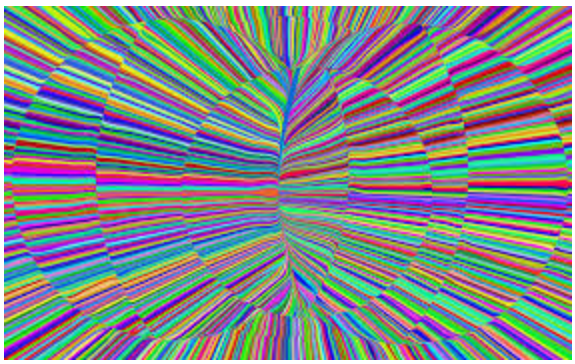


HSV encoded optical flow

Exercise 12

Data for the exercise:
images:



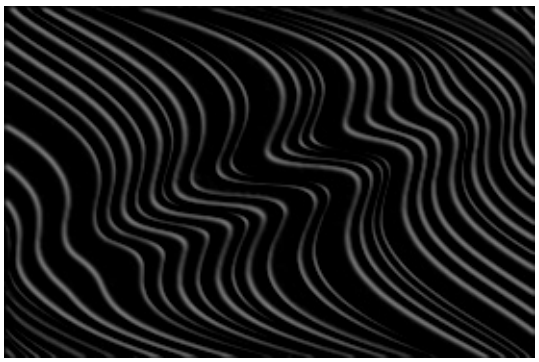


Given these two images, we have tested the following derivative filters to compute the texture gradients:

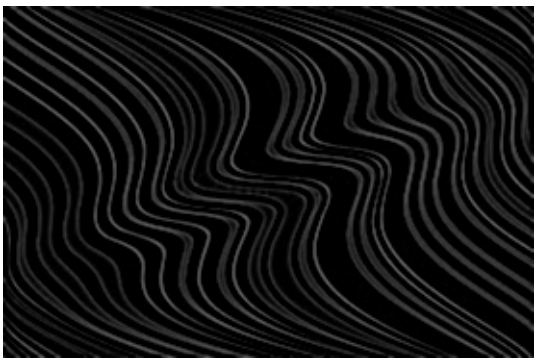
- *Laplacian*
- *Sobel*
- *Scharr*
- *Prewitt*
- *Roberts*

The *Laplacian* is the only second-order derivative filter. Derivative filters are very sensitive to noise, hence, could be a good practice remove the noise using a smoothing filter like the *Gaussian*. If we do this using The *Laplacian* filter is equivalent to apply the *Laplacian of Gaussian* filter to an image, which has good scale-space properties.

In the first image, we have a kind of wave lines pattern in a grayscale fashion. In this case, we are more interested in a filter which is able to give information about diagonal edges.
Both the *Prewitt*, *Sobel* and *Scharr* are good diagonal edges highlighters, but from the tests done, one of the most conclusive filters outcomes is from the *Roberts* filter in the vertical direction, which is able to extract the diagonal edges in a clear way also considering the wave pattern. The only drawback in this result it's the "elliptic" artefact positioned diagonally in the image. Also the *Laplacian* reach optimal results with this image:

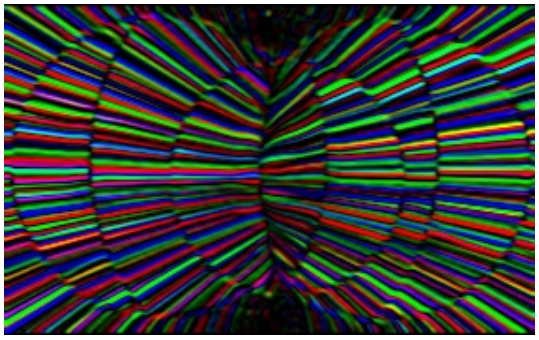


(w) Vertical *Roberts*

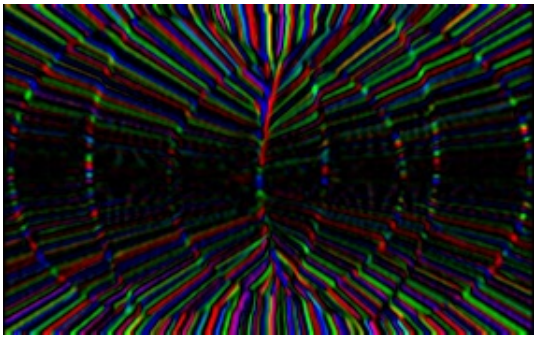


(x) *Laplacian*

In the second image, we have a mix of lines, diagonals, horizontals and few verticals, which are distorted by curves. The *prewitt* horizontal filter has shown a good response for this case, able to detect most of the edges, reducing effectiveness by moving towards vertical lines. Therefore we show both for the vertical and horizontal, which is the best for this case.



(y) Horizontal *Prewitt*



(z) Vertical *Prewitt*

Exercise 13

The *Downsampling* or *Decimation* it's the process of bandwidth reduction and sample-rate reduction and when it's applied to images this means reduction of resolution.
Three main algorithms are been used in this exercise:

1. *Laplacian Pyramid*
2. *Box filtering*
3. *Mipmap*

Before starting to describe them, it's important to define the technique used to evaluate the results of these methods. Even if it's possible to estimate the loss of information with the *mean squared error*, this in some cases can lead to very inaccurate results (one very different pixel is more important than many low different pixels). Therefore, has been used the *structural similarity index measure* or more simply, *SSIM*. This error function leads to estimate the absolute error, giving importance to the structure of objects in the visual scene, having an excellent estimation of similarity (in percentage).
For the evaluations are been used the below images:

from left to right: a simple geometric polyhedron (a), a medium complexity case (b) and a real world image (c).

The first algorithm consists of building the so-called "Pyramid", a sequence of subsampled images, by a certain *sampling rate* r. At each step, to attenuate the *aliasing* problem, the image is convolved with a Gaussian filter, to filter the high frequencies, then the residual, the difference between the original image and the blurred version, is computed. This step is crucial for the reconstruction phase. the last step is the deletion of rows and columns to reach the halve dimensions. These steps are then repeated based on r.
Even if this process it's lossy and we can't reconstruct the image as it was, with the residual information it's possible to descend the pyramid for obtaining the image at the original size with a good level of details.
The second technique implemented is the Box filtering. This method applies the same principles used for the downsampling phase in the Laplacian Pyramid. We filter the image with a low pass filter, this time is a 2×2 *Box filter* and we do this using the convolution with stride equal 2, to halve the dimensions. This process it's repeated until we reach the desired shape.
The last method is the *Mipmap*, this method allows to use of either a not even or not integer value for the *sampling rate*. This has been realized, generating the whole pyramid based on the ceil approximated value of r and then computing a double-weighted interpolation between the second last and the last layers, combined to form the final results. For this Mipmap implementation, the downsampling phase has been done using the Box filtering.

The Evaluation results are the following:

Image	Laplacian Pyramid	Box filtering	Mipmap
a	0.96	0.91	0.93
b	0.91	0.77	0.82
c	0.90	0.62	0.72

From these values, it's possible to have a clear idea of how much each method has performed on the images. But how it's possible to estimate the content loss of a downsample operation before applying it? One manner is appealing to the entropy value of each image. indeed the entropy represents not only the grade of disorder but also the data needed to represent each information, in this case, the images. Hence greater is the etropy and greater will be the loss downsampling. Using a disk of size 5 we have retrieved the following values about the average entropy:

Image	entropy
a	0.51
b	1.83
c	4.44

These values are consistent with the others found in the previous table and can be used for an estimation of content loss.