Assignment Template

Omar Bayoumi - 1747042

11 April 2021

1 Exercise 1

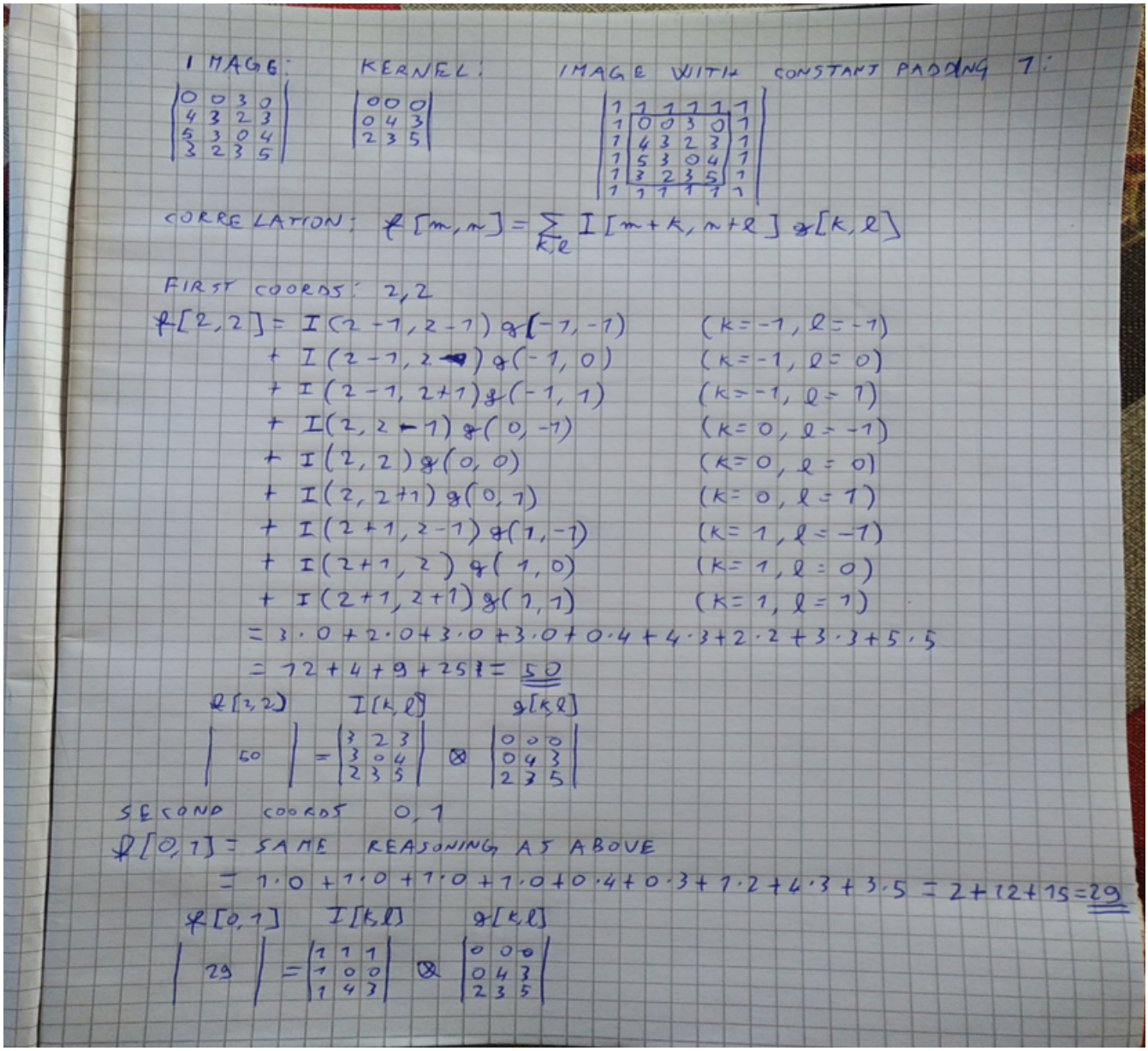


Figure 1: Exercise 1

Constant padding adds a border to the image with a constant colour in order to be able to apply a filter without

reducing the size of the image. This type of padding may create edges because it adds a constant value around the image that is not connected to the image.

2 Exercise 2

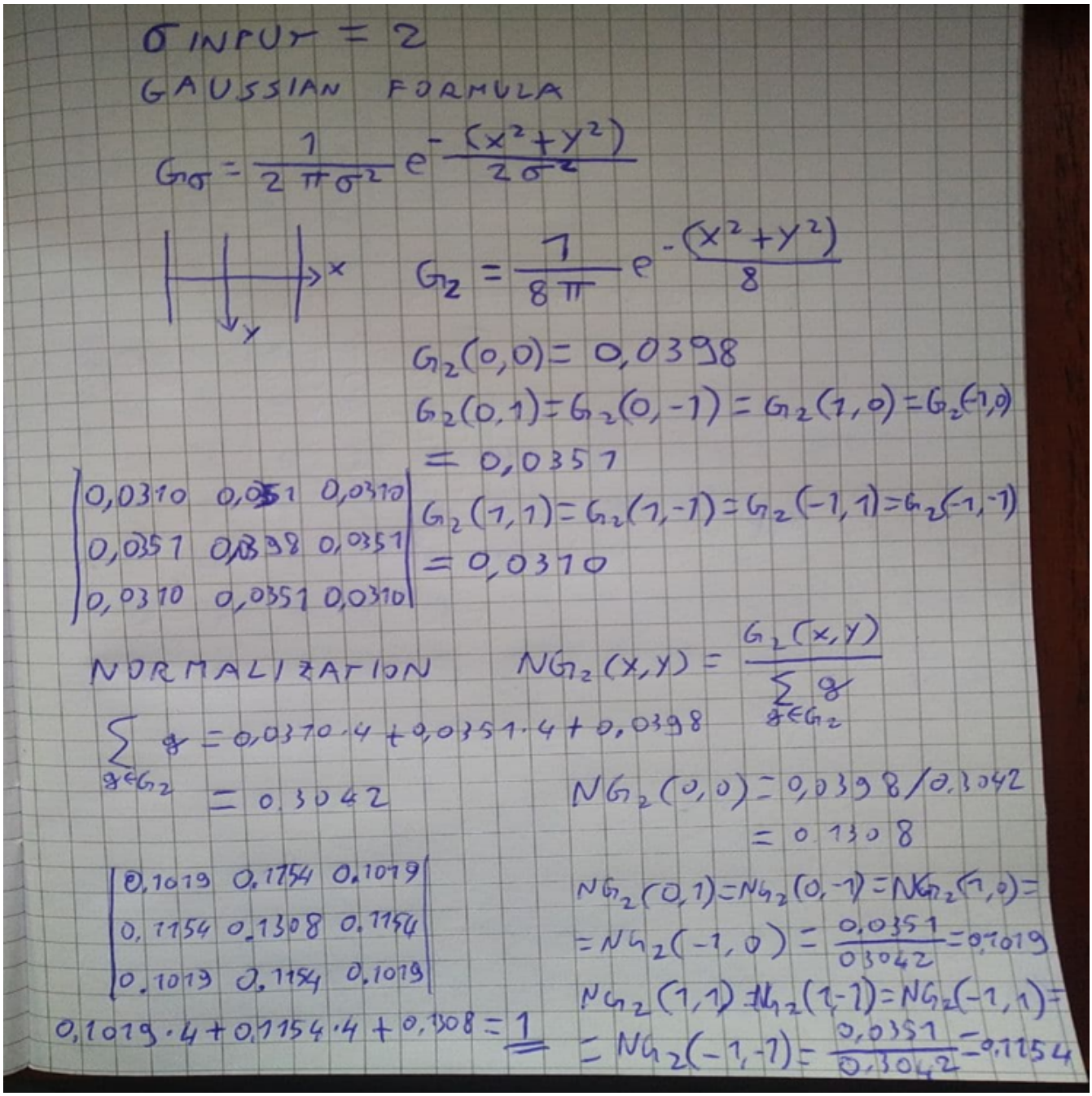


Figure 2: Exercise 2

3 Exercise 3

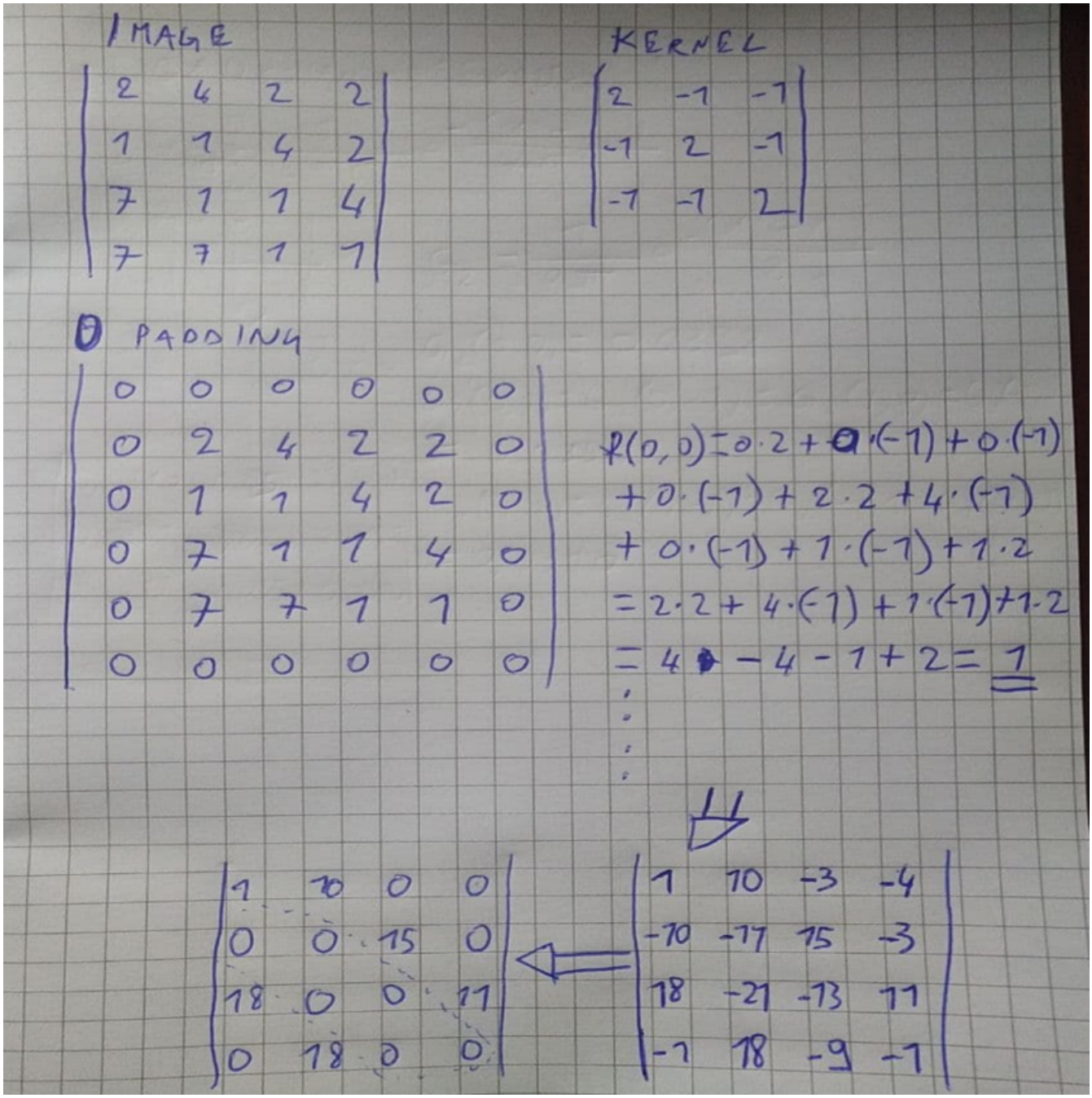


Figure 3: Exercise 3

0 padding has been used on the image to apply this kernel because it is simple. This kernel is detection of discontinuities and is used to detect lines. In particular, the one given detects lines at -45 degrees. At the end all negative values have been replaced with 0 to stay in the range [0, 255].

4 Exercise 4

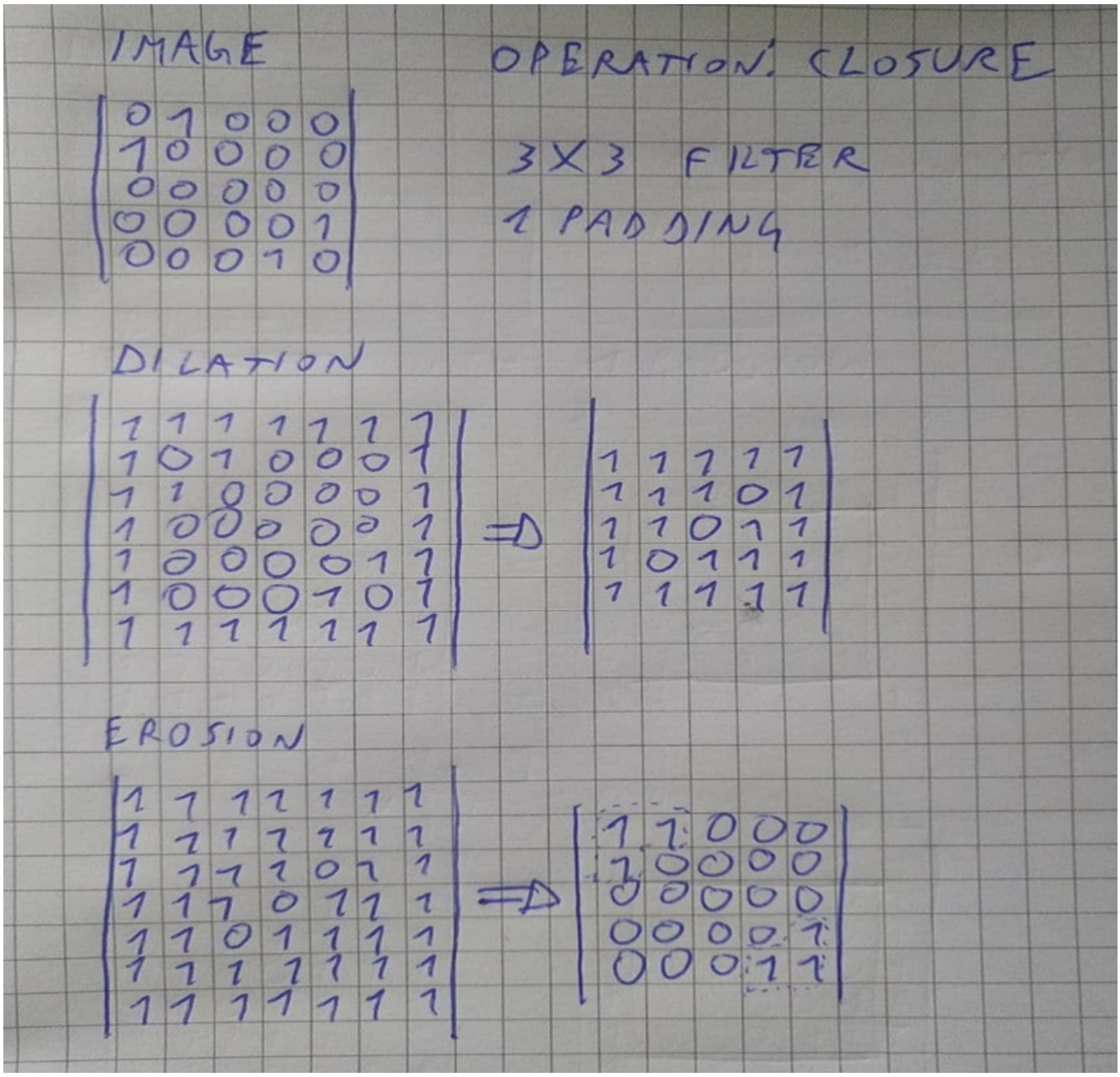


Figure 4: Exercise 4

The aim of the closure is to join separate lines. We first apply a dilation that increases the thickness to join points and lines close together, and then an erosion to return to the original thickness. A 3x3 filter and 1 padding were used: by adding ones to the border, in my case, it was possible to join the two points in the corners of the image.

5 Exercise 5

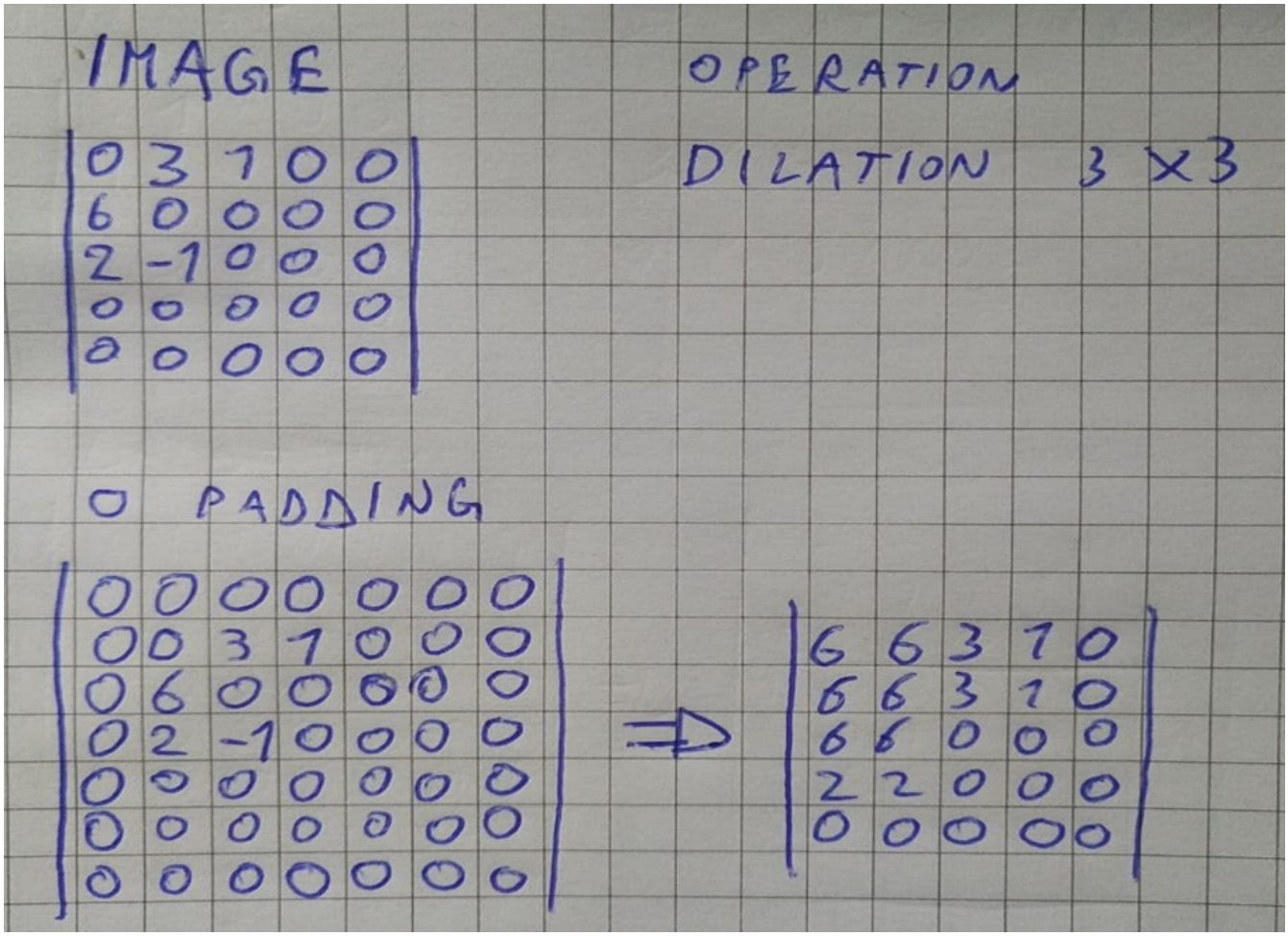


Figure 5: Exercise 5

Dilation on a gray-scale image calculates for each pixel the highest value among the neighbours with a maximum distance set by the filter size, 3x3 in this case. The effect it has on the image is to expand the areas with the highest intensity value. I have used 0 padding for simplicity and because it does not affect the choice of maximum.

6 Exercise 6

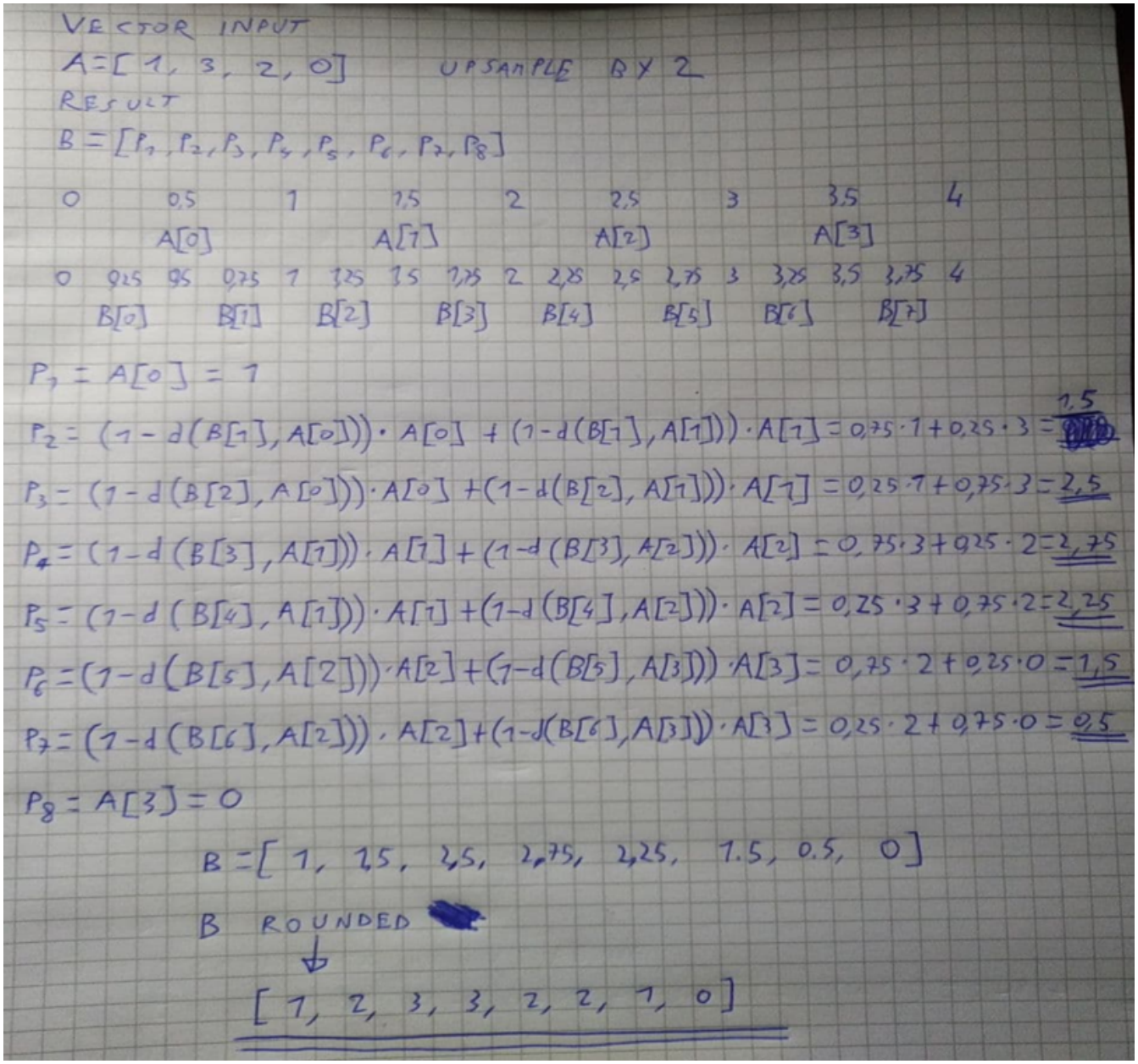


Figure 6: Exercise 6

7 Exercise 7

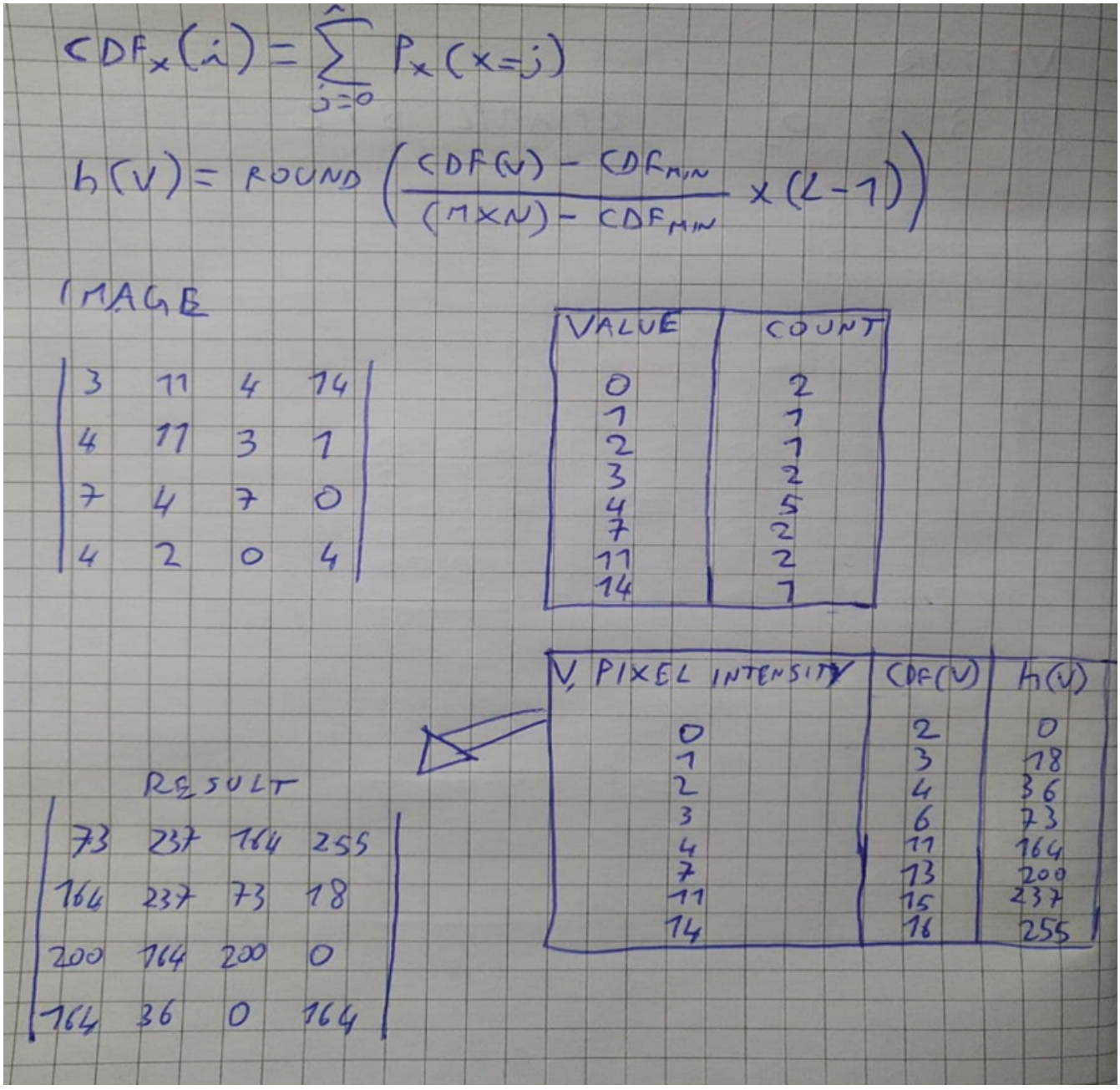
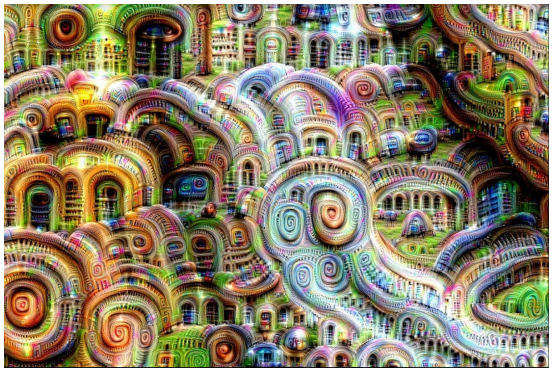


Figure 7: Exercise 7

Histogram equalization is used to increase contrast in images. I calculated for each pixel the cumulative distribution function (cdf(v)) to count the number of pixels with intensity less than or equal to the given intensity (v). Then, the equalization (h(v)) pixel is calculated. Finally, each intensity is replaced with the intensity calculated by h(v).

8 Exercise 8



(a) Image 1

(b) Image 2

Entropy. Entropy is a measure of chaos, i.e. the level of uncertainty of a random variable. For an image, the local entropy is related to the complexity contained in a given neighbourhood. Entropy can see, inside an image with a lot of noise, the figures contained within and highlight their textures. Entropy uses a circular kernel, and depending on the size of this kernel you can see a worse or better result. So the best way to use this method is to guess the size of the kernel. However, the final result will not have well-defined edges but an approximation.
Fourier Transform. It is based on the concept of the Fourier Series . The starting formula is this: $A\sin(\omega x + \phi)$ A is the amplitude of the signal, ω the frequency and ϕ the phase. If we think of these frequencies drawn in a 2D image, we can think of the centre as frequency 0, and the further away you go, the higher the frequencies get. If the function is not periodic it is not enough, but non-periodic functions, which have a finite area under the curve, can be expressed as an integral of sines and cosines multiplied by a weight function. This is the Fourier Transform. The particularity of the Fourier Transform is that it is possible to go from the frequency spectrum to the image and vice versa. If we only consider high frequencies, we get a filter called highpass. This filter acts as an edge detector on the image.
A combination of the two would make it possible to define the edges of the textures captured by the entropy. For image 1 the waves given by the image pattern are highlighted. For image 2, figures such as the bird in the centre, the animals at the bottom and the logs in the background are highlighted.

9 Exercise 9

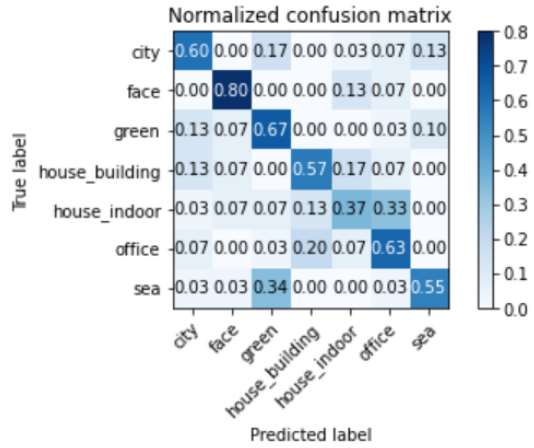
1. **Gaussian filter**

   Let's start with the Gaussian formula: $G_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}}$. The variable σ^2 represents the variance, i.e. the measure of how far the set of numbers is from their mean value. In addition, to apply the Gaussian filter, we also need to think about the size of the kernel, i.e. the window of pixels to be considered, x and y being the coordinates of adjacent pixels. The larger the kernel, the more neighbouring pixels to consider. The explained formula works like this: for each pixel it makes a sort of weighted average between the neighbours (taking into account the size of the kernel) so if the pixel has a lower intensity, then it is increased, vice versa if it has a higher intensity then it is decreased. If we think of a 2D function, we can think of this calculation as a flattening of the function which has a lot of noise (i.e. many small maximum and minimum peaks) in order to show about the real signal behind this function without noise. The same reasoning is applied to a 3D function and then to an image, which allows, through methods studied in class such as Sobel, to obtain interesting information that would otherwise have been impossible to obtain.

2. **Entropy**:

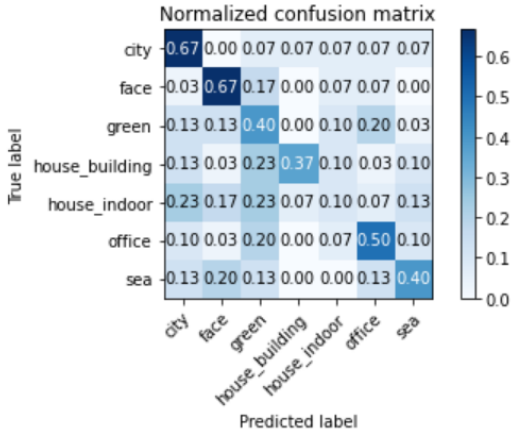
   Entropy is a measure of chaos, i.e. the level of uncertainty of a random variable. The entropy is formally defined as: $H(X) = -\sum_{i=1}^{\infty} P(x_i)\log(P(x_i))$. For an image, local entropy is related to the complexity contained in a given neighborhood. For example, if we create an image with a uniform distribution in the range [-15, +15] in the centre and [-14, 14] at the border, and both centred at a grey value of 128, entropy can see through the chaos and detect the central square's shape. To do this we calculate the local entropy using a disk structure of radius r in order to capture the local gray level distribution. This calculation is useful for finding information about texture areas within an image and then highlighting them with techniques such as sobel laplace and so on.

10 Exercise 10



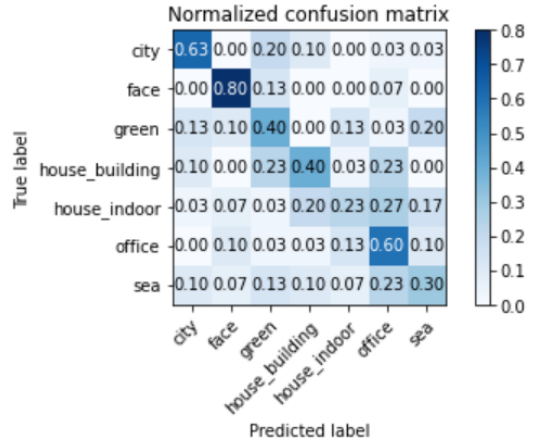

```
Confusion matrixes plotted.
accuracy score: 0.598
Accuracy calculated.
Execution done.
```

(a) SIFT confusion matrix




```
Confusion matrixes plotted.
accuracy score: 0.443
Accuracy calculated.
Execution done.
```

(b) HARRIS confusion matrix




```
Confusion matrixes plotted.
accuracy score: 0.481
Accuracy calculated.
Execution done.
```

(c) FAST confusion matrix

Figure 9: Confusion matrices

Harris: Harris corner detection is based on the idea that by looking through a small window at the image, and moving this window, we can see small variations in flat areas, variations in one direction in edges and variations in all directions in corners. Calculating gradients in a small area $I_x = \frac{\partial I}{\partial x}$ $I_y = \frac{\partial I}{\partial y}$; Subtract the mean of the gradient to the gradient for both; Obtain covariance matrix $\begin{bmatrix} \sum_{p \in P} I_x I_x & \sum_{p \in P} I_x I_y \\ \sum_{p \in P} I_y I_x & \sum_{p \in P} I_y I_y \end{bmatrix}$; Calculating eigenvalues and eigenvectors; Using a threshold on eigenvalues to determine angles (e.g. Harris & Stephens $R = det(M) - ktrace^2(M)$ with k empirical constant. If R>0 is a corner, if R<0 is a edge and if R<<0 is a flat area);

Sift: For sift we use the difference of gaussian (DoG) pyramid for choosing the keypoint. DoG uses two different σ (e.g. σ and kσ) to construct the octaves, at different resolutions. At this point if the pixel is a local extrema in the 26 neighbours (9 upper octave, 9 lower and 8 current) then this is a keypoint, otherwise we repeat the operation at lower resolution. Of these keypoints, those with little contrast and those representing edges are removed and to make the keypoint rotation invariance, we check the prevailing orientation to establish its orientation. If there is any

orientation in the 80% of the peak, then a new keypoint is constructed at that position with this second orientation;

Fast: For each pixel p in the image a classification is made to determine if this is a corner. Let I_p be the intensity of the pixel and t a value of threshold consider the 16 pixels in the circle around pixel p. If there are n pixels in succession in this circle that are either all brighter than $I_p + t$ or all darker than $I_p - t$, then this is a corner.

As can be seen from the confusion matrix: Sift and Fast confuse the sea with the green most likely because they identify corners that in the sea, being solid colour areas, are often not there but rather identify what is around as forests. Harris also identifies corners and confuses house building and city, probably because he identifies the corners of buildings in cities by associating them with those of houses. All 3 techniques fail with house indoors probably not because of the method but because of an unbalanced dataset (in fact there are few house building images).

11 Exercise 11

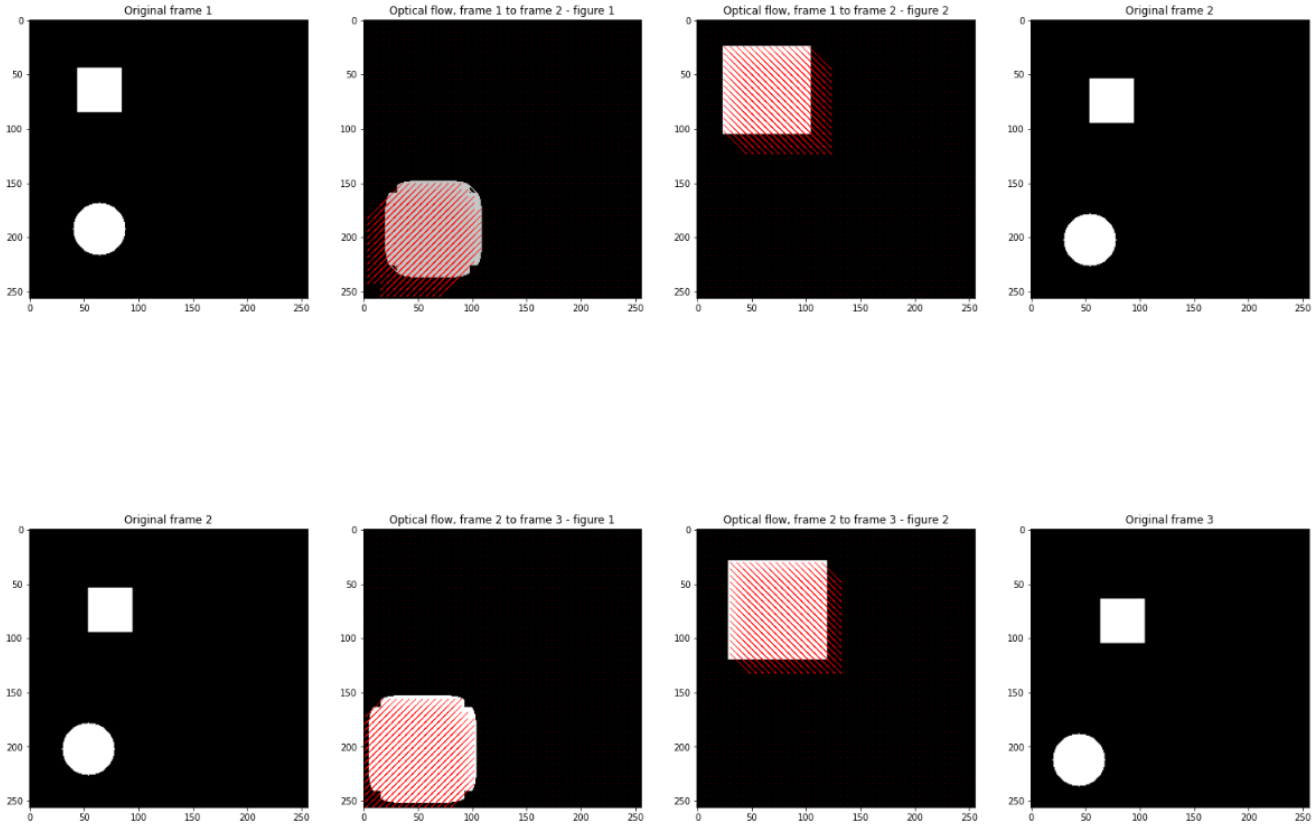


Figure 10: Optical flow

For each frame, I used a sort of simplified 1-loop kmeans to segment the two figures within the input image. First of all I took the image and made it binary with pixels > 250 white and < 250 black. Then I calculated the two most distant white points to use as clusters. At this point, for each white pixel, this was assigned to the nearest cluster, thus obtaining the coordinates of the figures separately (and then inserting white at these coordinates into a black image).

Considering the simple figures, only one cycle of kmeans was needed, so one strength is its accuracy and furthermore, if I had had to implement a version of kmeans up to convergence it would have been much slower, so another strength is speed. On the flip side, because it is a 1 cycle version, I don't have convergence and therefore the certainty of having the clusters in the centre of the figure. In general it is still a bit slow.

The method calculates the spatial derivatives using techniques such as Sobel, DoG and so on, for the first derivative with respect to x and with respect to y; the temporal derivative, i.e. the differences between the two consecutive frames; and the optical flow. The latter is calculated using the Lukas Kanade iterative method which returns the estimated vector field (u, v) that is plotted on the image with red arrows to show the movement of the figures: the square at the top left towards the centre and the circle at the bottom left towards the bottom left corner.

12 Exercise 12

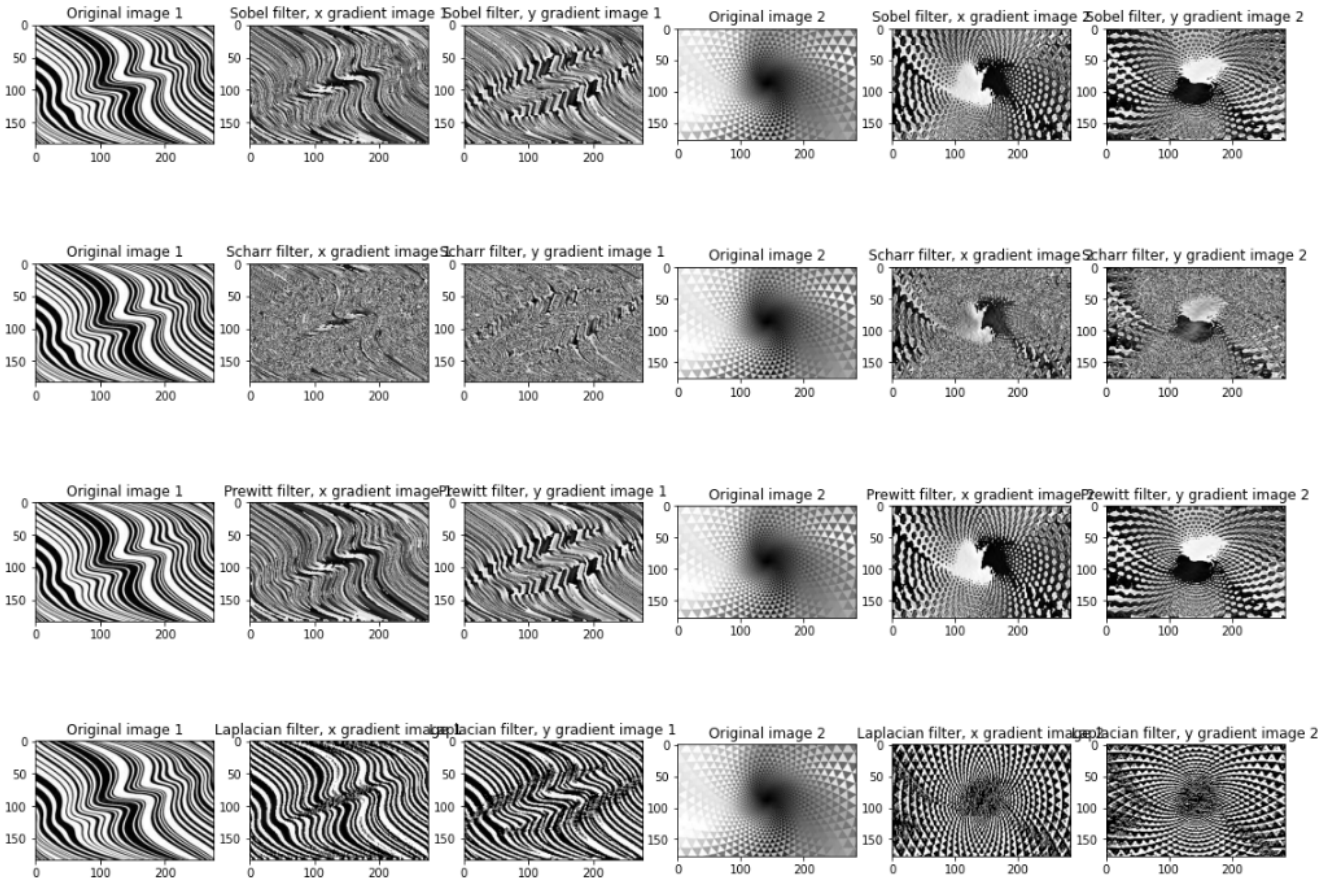


Figure 11: All texture gradients

Texture gradients are the result of calculating the derivative of the image with a discrete signal. The derivative is therefore the filter $\begin{bmatrix}1 & 0 & -1\end{bmatrix}$. If we multiply a filter by this derivative we finally get the filters used. The filters obtained can be two according to the order of the row-by-column multiplication, where the first is 3x1, the second 1x3 obtaining in the end a 3x3 matrix: derivative*filter=vertical filter and locates the horizontal edges; filter*derivative=horizontal filter and locates the vertical edges. However, the resulting filter applied to an image is very sensitive to noise, which is why gauss and then the gauss derivative were applied. Another filter used is the laplace filter which calculates the second derivative of gauss in order to obtain the zero crossing. The filters used are:

- **Sobel**: Horizontal filter $\begin{bmatrix}1 & 0 & -1\\2 & 0 & -2\\1 & 0 & -1\end{bmatrix}$ Vertical filter $\begin{bmatrix}1 & 2 & 1\\0 & 0 & 0\\-1 & -2 & -1\end{bmatrix}$
- **Scharr**: Horizontal filter $\begin{bmatrix}3 & 0 & -3\\10 & 0 & -10\\3 & 0 & -3\end{bmatrix}$ Vertical filter $\begin{bmatrix}3 & 10 & 3\\0 & 0 & 0\\-3 & -10 & -3\end{bmatrix}$
- **Prewitt**: Horizontal filter $\begin{bmatrix}1 & 0 & -1\\1 & 0 & -1\\1 & 0 & -1\end{bmatrix}$ Vertical filter $\begin{bmatrix}1 & 1 & 1\\0 & 0 & 0\\-1 & -1 & -1\end{bmatrix}$
- **Laplacian**: Horizontal filter $\begin{bmatrix}0 & 0 & 0\\1 & -2 & 1\\0 & 0 & 0\end{bmatrix}$ Vertical filter $\begin{bmatrix}0 & 1 & 0\\0 & -2 & 0\\0 & 1 & 0\end{bmatrix}$

For both image 1 and image 2 the ranking is: Scharr (the worst), Sobel, Prewitt and Laplace (the best). Scharr is probably the worst because of the 10 contained in the matrix which accentuates the central pixel a lot. Sobel and Prewitt both perform well and are very similar, probably because their matrices differ by 1. Laplace, on the other hand, due to zero crossing, detects the oriented edges of the figures better than the others.

13 Exercise 13

- **Metodo 1**: For the first method, the naive method was applied, which consists of eliminating even rows and even columns to obtain at the end an image downscaled by $\frac{1}{2}$. However, this method may cause aliasing, creating a loss of information and that effect known as "moire". If we take a signal, to discretize it we take points on this signal. If we now halve these points by simply taking one on one and one off, we will obtain a completely different signal, losing the information of the original one. There are antialiasing techniques to solve this problem. In my case, to reduce this effect we first applied a smoth effect, the Gaussian filter, and then proceeded with the algorithm described at the beginning.
- **Metodo 2**: The second method is the box filter. This method works by averaging the intensity of pixels in a window 2r x 2r where r is the radius of the box filter (in my case I used radius 1). $P = \frac{1}{(2r)^2}(F_{i,j} + F_{i,j+1} + F_{i+1,j} + F_{i+1,j+1})$ is the formula to obtain the pixel intensity, as said at the beginning, an average. The window is moved by 2r for each pixel in order to group all the pixels in the window into a single pixel and obtain a downscaling of $\frac{1}{2r}$. As with the first method, there may be a problem with pixellated effects and aliasing, so here too an image with a smoth effect, the Gaussian filter, has been used.

As mentioned in the two methods, before applying downsampling we create an image with a smoth effect to reduce the loss of information. One way to estimate how much information will be lost is to make the difference between the original image and the previously calculated smoth image, and the result of this subtraction, called the residual, is our estimate.