Lab 1 Report

Theoretical understanding

Convolution

Mathematical operation that consists of a weighted sum of the intensities of pixels in the surrounding area of the original image P. The weights are stored in a filter kernel F, which has entries $f_{m,n}$, where $m \in \{-M \dots M\}$ and $n \in \{-N \dots N\}$ being M the height and N the width (on pixels) of the image. This is convolving P with the filter F, where two-dimensional convolution is defined as $x_{i,j} = \sum_{m=-M}^{M} \sum_{n=-n}^{N} P_{i-m,j-n} F_{m,n}$ where $x_{i,j}$ is the new value for the pixel at our output image.

It is important to notice that F is flipped in both directions, this is done to meet the commutative mathematical property of convolution, ensuring that the order of functions does not affect the outcome. This is needed due to the kernel being moved over the image, aligning with a specific set of image pixels at each position. This led us to the need of ensuring that the positions of the elements in the kernel match the corresponding positions in the image; the flipping compensates for the shifting effect and aligns the kernel's elements properly with the image's pixel values. It will also allow us to apply efficient computational methods, such as Fast Fourier Transform.

When applying the kernel without further modification we will run into problems at the borders of the images since we would be trying to access points that are outside the image. To deal with this we could apply different types of padding, including rows above and below and columns at both sides of our images with standard values allowing us to increase to fully process the image. Padding could be achieved using one of the most common padding techniques:

- Zero padding: including or assuming the values of P is 0 outside the defined image region.
- Constant padding: including or assuming the values of P is a outside the defined image region, where a is a constant.
- Reflective padding: mirroring the pixels along the borders of the image, replicating them in a mirrored fashion.

Finally, for achieving a clean computational implementation of convolution, we usually consider odd-sized kernels to have a center where we can centralize our operation. However, it is possible to apply convolution with even-sized or not-squared kernels.

Linear Filtering

Linear filtering is the result of applying different kernels to an image using convolution operation. Doing linear filtering we want to remove unwanted sources of variation and keep the information relevant for whatever task we need to solve, therefore, the kernel used will entirely depend on the expected result we want to obtain from the process. Some of the most used kernels in linear filtering are:

- Blur
  - Average box: Low-pass filter that smooths the image by making each output pixel the average of the surrounding ones, removing details, noise and edges from images.
  - Gaussian: By convolving an image with a 2D Gaussian defined as $f(m,n) = \frac{1}{2\pi\sigma^2} \exp\left[-\frac{m^2+n^2}{2\sigma^2}\right]$ each pixel in the resulting image is a weighted sum of the surrounding

pixels, where the weights depend on the Gaussian profile: nearer pixels contribute relatively more to the final output. This process blurs the image, where the degree of blurring is dependent on the standard deviation σ of the Gaussian filter.

- Edges
    - Laplacian: discrete two-dimensional approximation to the Laplacian operator ∇^2 given by $F = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$. This will result in a response of high magnitude where the image is changing, regardless of the direction of that change: the response is zero in regions that are flat and significant where edges occur in the image. It is hence invariant to constant additive changes in luminance and useful for identifying interesting regions of the image.
    - Prewitt: Searching for places in the image where the intensity changes abruptly we try taking the first derivative of the image along the rows or the columns. This will give us a flat result when there are no changes across a direction in our image, negative when the image values are increasing and positive if they are decreasing. The kernels for each direction (horizontal and vertical) will have the shape $F_x \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$ and $F_y \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix}$
    - Sobel: The same intuition than Prewitt filters case, Sobel filters use the first derivative identifying abrupt changes on the image intensities but increasing more (when compared to Prewitt's) the final intensity of the edge, leading to shapes $F_x \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix}$ and $F_y \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$ horizontal and vertical respectively.

It is important to remark that any of these kernels could be increased in size following specific relations between the values contained on it; however, it will not be always a useful approach in some of those, especially the edges detection ones.

Template Matching

Different from convolution, cross-correlation do not flip the kernel, apart from that, the process is the same. Then, we use Normalized Cross-Correlation (NCC) to find in an image f the given template g by searching for the maximum values (representing the best matches of the template across the image) given by the NCC formula. The NCC formula recalls on the existing difference between g and the current region of the f, this is $\sum_{x=0}\sum_{y=0}[f_{x,y} - g_{x,y}]^2$, since the squared difference give us two constant terms, we don't care about them only considering the variable one resulting as $\sum_{x=0}\sum_{y=0}[f_{x,y} * g_{x,y}]$. Since we are searching for a maximum value, to obtain a result that will not be affected by the difference of intensities at different regions on the image when applying sliding window process, we will normalize (as the name of the formula says) constraining the obtained values to vary between -1 and 1, this normalization results in the NCC formula $NCC(f,g) = \frac{\sum_{x,y}((f_{x,y}-\mu_f)*(g_{x,y}-\mu_g))}{\sqrt{\sum_{x,y}(f_{x,y}-\mu_f)^2 * \sum_{x,y}(g_{x,y}-\mu_g)^2}}$. Our algorithm should calculate $NCC(f,g)$ for every point in f storing x and y coordinates from the maximum values finding the best matches of our template g on f.

Fourier Transform

The Fourier transform is a powerful tool in image processing and many other fields. It allows for the decomposition of an image into its frequency components, which can be useful for various tasks, including noise reduction and image quality improvement. To understand the theoretical concepts of Fourier transformations applied to image processing, we will explore the basic concepts.

The Fourier transform is a mathematical technique that allows one to move from one domain representation of information to another. In the context of image processing, two main domains are involved: the spatial domain and the frequency domain.

- Spatial Domain: This is the original domain of the image, where each pixel has a value corresponding to its brightness. The image is represented in two dimensions (2D), with x and y coordinates.

- Frequency Domain: This is the domain after the application of the Fourier transform. Instead of representing the image in terms of brightness intensities, it represents the image in terms of spatial frequencies. This means that the image is decomposed into different frequencies, including low frequencies (representing smooth patterns like homogeneous areas) and high frequencies (representing fine details like edges and textures).

The Fourier transform is useful in image processing for several reasons:

- Detection of Frequency Components: It allows for the detection of the frequency components of the image, which is essential for understanding its structure and content.

- Filtering Undesirable Frequencies: You can filter out undesirable frequencies, such as noise, by removing high frequencies. This improves image quality. This was the focus of our interest in the last part of Lab1.

- Compression Enhancement: In image compression, the Fourier transform is used to reduce the amount of information needed to store or transmit an image by focusing on important frequencies.

The key steps of the code that perform filtering of undesirable frequencies in the frequency domain are as follows:

- Fourier Transform: The Fourier Transform is applied to an altered image using the fft2 function, converting the image from the spatial domain to the frequency domain. This means that we are now analyzing the image in terms of frequencies rather than pixels, which is crucial for understanding and manipulating its spectral characteristics.

-Frequency Reorganization: The fftshift function is used to rearrange the frequency components calculated from the Fourier transform of the image. It translates the frequency components to place the zero frequency component (DC) at the center of the image in the frequency domain. The DC component represents the average values of the image, and placing it at the center makes it easier to analyze positive and negative frequency components with respect to this central point.

-Magnitude and Phase: After applying the Fourier transform, the image is represented in terms of magnitude (module) and phase. The magnitude contains information about the amplitude of different frequencies, while the phase contains information about the position of these frequencies.

-Filtering Aberrant Frequencies: In the code, we filter frequencies whose magnitude is above a certain threshold, which was visually determined by displaying the magnitude graphically. In our case, these aberrant frequencies were intentionally added for the exercise. However, in everyday situations, these aberrant frequencies are often due to noise or unwanted artifacts.

-Reconstruction of the Filtered Image: Once undesirable frequencies are filtered, we reconstruct the image in the spatial domain using the inverse Fourier transform (after also applying the inverse transformation of frequency reorganization). This yields an improved image, free of undesirable frequencies.

To assess the image improvement after filtering, two main metrics are used: PSNR (Peak Signal-to-Noise Ratio) and SSIM (Structural Similarity Index). These metrics help quantify how much better the filtered image is compared to the original image in terms of quality.

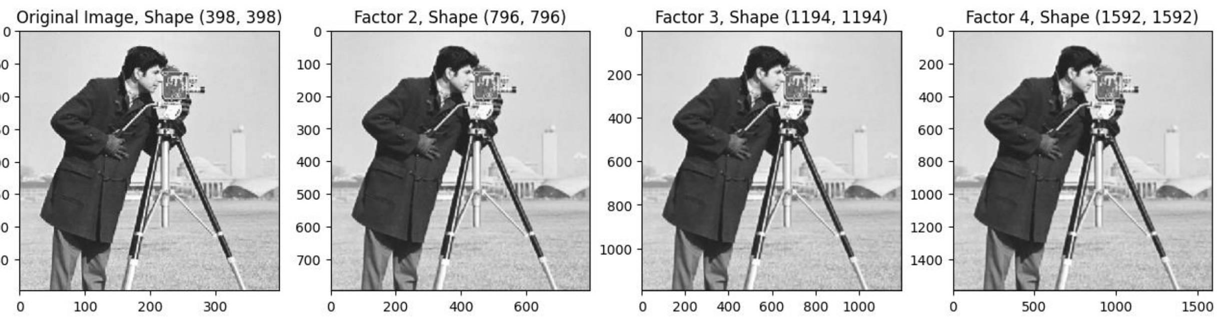
- PSNR: PSNR measures the fidelity of the filtered image compared to the original image in terms of noise. A higher PSNR indicates better image quality.

- SSIM: SSIM evaluates the structural similarity between the filtered image and the original image, taking into account not only noise but also the structure and details of the image.

Results & discussion

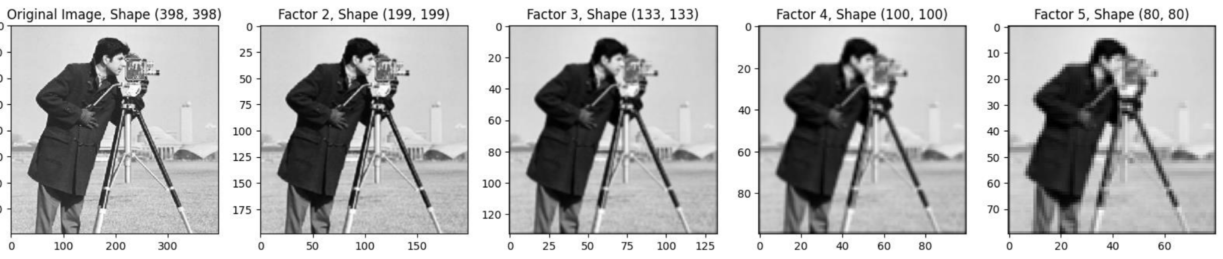
During the development of the practice, satisfactory results were obtained in all the developed modules. All of them are listed and commented below.

In the case of upscaling, four images are presented, the first (left) being the original input of the function, to the right, each of the images presents a larger scale factor starting with factor 2, which doubles the total amount of pixels in the image, passing through factor 3 and finally 4, increasing the dimensions of the image to 1592 x 1592 pixels, compared to the 398 x 398 that composed the image in the beginning. The graph is shown below.


Original Image, Shape (398, 398)
Factor 2, Shape (796, 796)
Factor 3, Shape (1194, 1194)
Factor 4, Shape (1592, 1592)

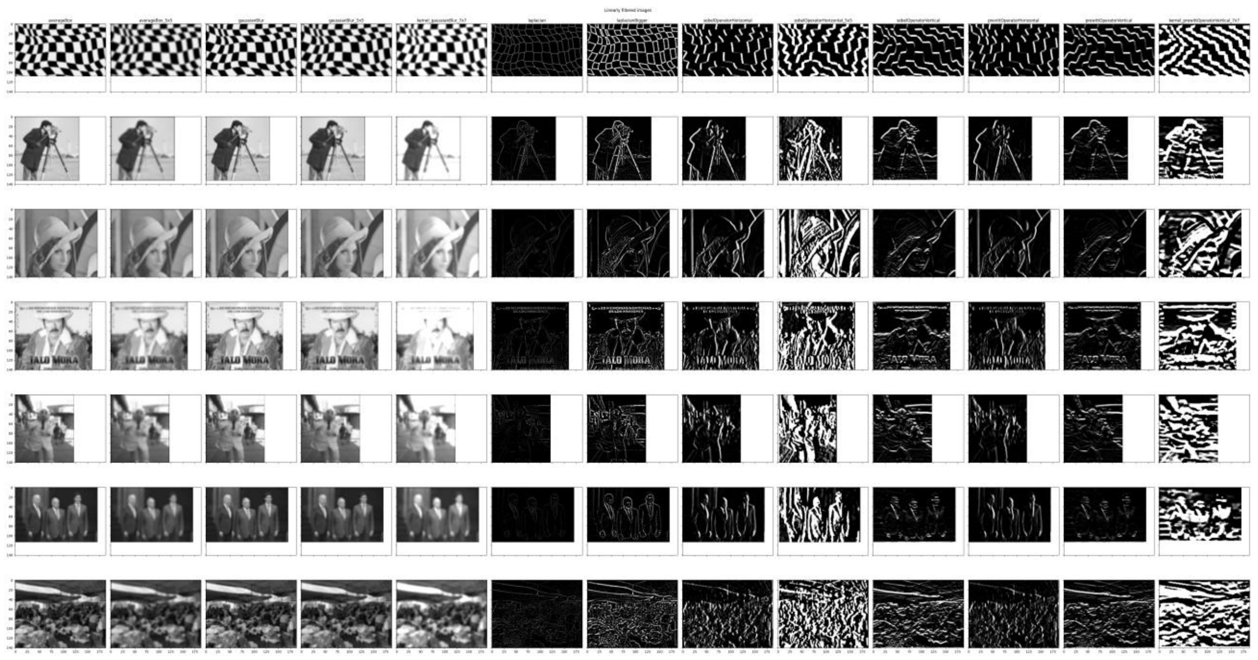
We can notice a correct and quite satisfactory implementation of the function because the image does not lose clarity or shape as it grows in dimension. The figures, intensities and textures are maintained, reducing the definition of the component pixels as their number increases in the image.

On the other hand, for the downscaling implementation we notice the same process as upscaling but inverted. Now, each of the images to the right of the original image (left) gradually decreases in dimension according to its scale factor, measuring only one third of its first ancestor. The following image demonstrates this:


Original Image, Shape (398, 398)
Factor 2, Shape (199, 199)
Factor 3, Shape (133, 133)
Factor 4, Shape (100, 100)
Factor 5, Shape (80, 80)

In this case, the smaller the number of pixels that make up the image, the sharper the image becomes. The pixels increase in dimension and the figure is shaped from a smaller amount of information. Here the images also decrease by a unit scale factor, becoming only a quarter of the size of the original image.

Possibly the largest output of the practice is a grid that shows the transformations suffered by each of the supplied images, filtered by each of the thirteen filters programmed for this purpose. This is visualized as follows:



From the above we can notice that in the case of the blur filters, the change is noticeable as the kernel size increases, with the 3x3 kernel diluting the image the least, and as the dimensions increase, the image becomes increasingly difficult to perceive.

In comparison, the Average Box filter shows a much stronger attenuation compared to Gaussian Blur, so more detail is lost. However, Gaussian blur appears to enhance the intensity of colors by increasing the contrast of dark areas versus light areas, a phenomenon that is not present in the Average Box filter.

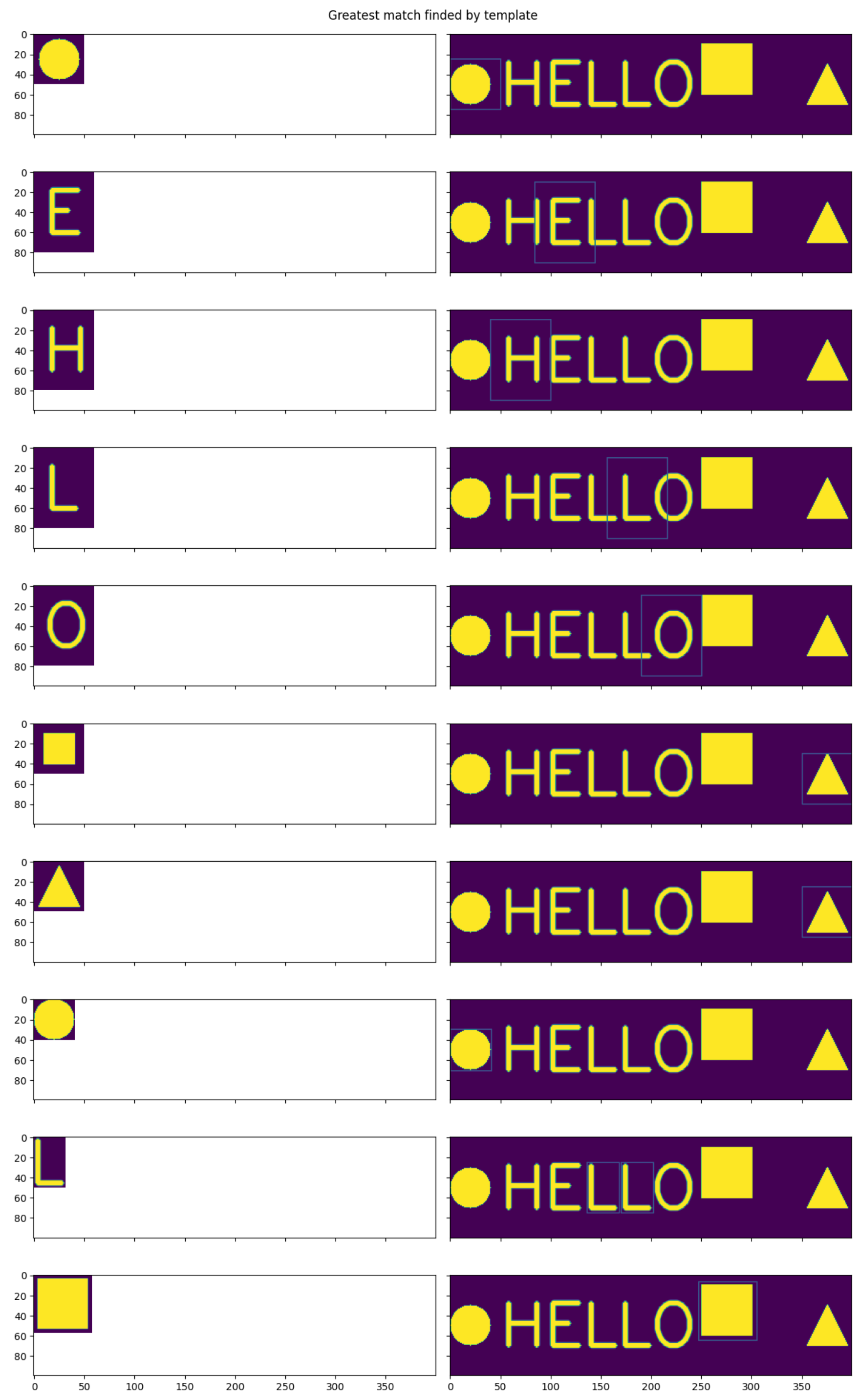
In the case of the contour detection filters, it is necessary to mention that the increase in size in the dimensions of the kernel used simply widens the strokes that are already perfectly perceptible in the 3x3 versions of each kernel, becoming detrimental when the image has many strokes and the width of the contours found is such that the correct segmentation of the image is lost, as happened with the 5x5 horizontal Sobel Operator or the 7x7 vertical Prewitt Operator.

About each filter we can say that the Laplacian kernel is very effective in detecting the contours of the main components of the image, perfectly attenuating backgrounds and textures. In the case of the directional filters, we can notice a great similarity in the results of the vertical and horizontal versions of the Sobel and Prewitt operator respectively, each doing a great job in the directional detection of contours, although, usually, the Sobel Operator highlighted more intensely those contours.

A peculiar phenomenon can be found in the plotting of the spots found through the NCC algorithm for pattern matching. At this point we should clarify the inclusion of three new templates

- A better adjusted adjusted to its limits.
- An “L” reduced in its original margins.
- A square of equal size than the one found in the base image.

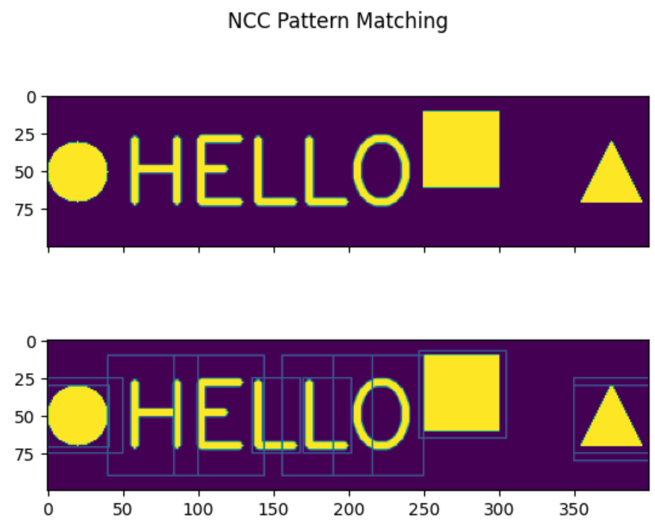
Below is displayed each of the filters and the area with the best fit to each of the templates, as well as their location within the image.



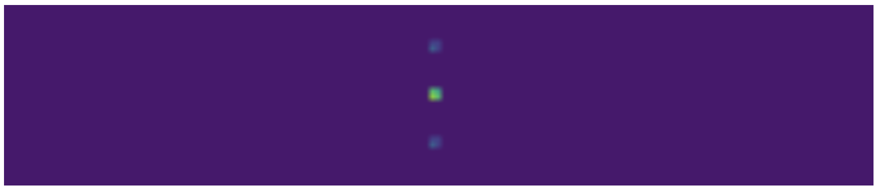
In all the literals of the original templates we found satisfactory fits, except for the case of the "L" that we expected to encompass both L's existing in the base image. However, we realized that these templates were very wide, so that our matches corresponded to only one of these L's. This assumption remained unchanged. This assumption was confirmed when, using a more suitable L, the system detected both letter repetitions.

Something somewhat similar (i.e., dependent on the clipping of the template) happened in the case of the square which, being much smaller than its counterpart in the image, the best match found corresponded to the triangle. After some lucidity, we understood the conflict between the reduced templates and the realization of the image we were looking for by including now our own template of the same size as the square in the image, so we were able to perfectly match the result.

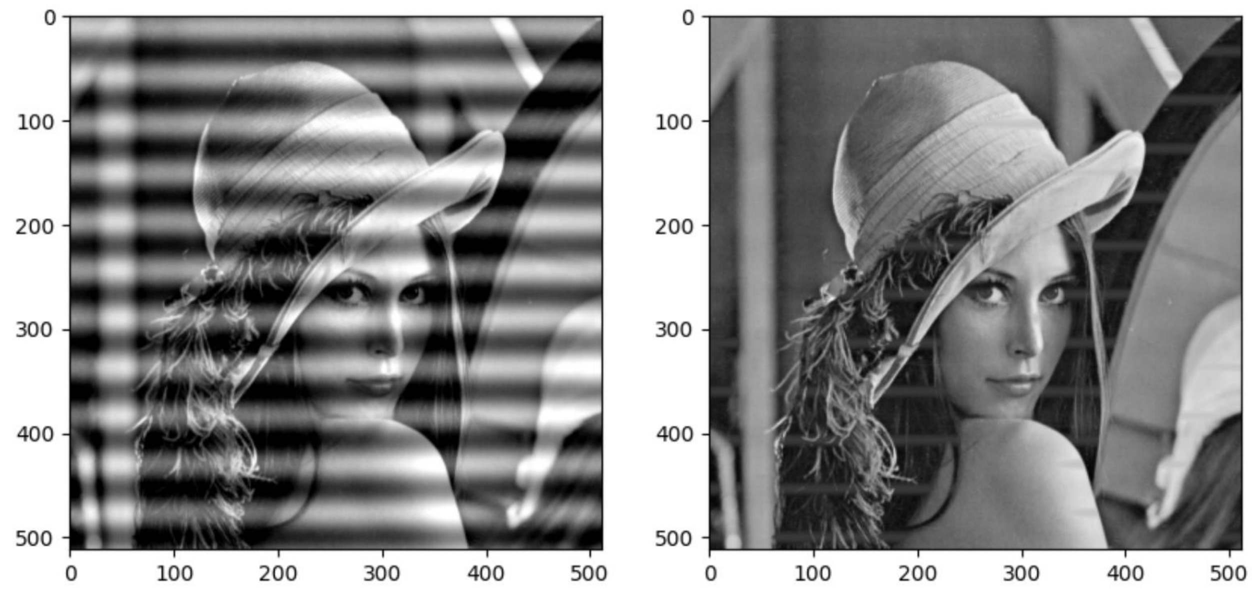
Below we can see, in a much more general outline, each of the matches found according to the 11 templates provided.



Below is the graphical representation of the Fourier transform module of the image 'fourier_1.jpg.' In this image, you can clearly distinguish the zero-frequency component as well as two symmetrical points that correspond to the same frequency. The outlier value of this frequency stands out prominently on this graph without the need to use a logarithmic scale. Thus, thanks to the colorbar on the side, we can identify the necessary cutoff frequency to filter out this frequency.



By filtering out this frequency using a low-pass filter (which filters out all frequencies except the zero frequency with a magnitude greater than the cutoff frequency of our filter), we obtain the image without this disruptive frequency. Consequently, when returning to the spatial domain, we obtain the corrected image.


0
100
200
300
400
500
0
100
200
300
400
500

Conclusion

Image processing allows us to explore possibilities through simple yet powerful operations. The union of mathematical and computational knowledge allows us to develop efficient and useful processes that transform input data into specific desired outputs.

Convolution starts as the basis of a wide range of transformations whose differentiating component is the kernels used, each of them with different compositions and backgrounds that are intended to have different utilities.

The sister operation of convolution is correlation, which in turn is also the basis of a more complex process called Normal Cross-Correlation, which is used for the detection of sub-images in a larger image. It will work by giving values that correspond to the amount of similarity found between the template and the evolved region at that time.

Finally, with the help of the fourier transform we convert our image to signals having a truly powerful tool for processing.