Exam II

Luis Eduardo Robles Jiménez || 0224969
Sara Carolina Gómez Delgado || 0226594

Part I

(a) When performing linear filtering, explain the difference between correlation and convolution. When are the two operations equivalent?

Convolution is just like correlation, except that we flip over the filter before correlating. Thus, both operations are equivalent when the kernel is symmetric.

(b) What is the purpose of normalizing a filter kernel to sum to 1? Is this always necessary?

The purpose of normalizing a filter kernel to sum to 1 is to smooth the kernel, by doing this, we reduce the effects of noise (low-pass filter).

On the other hand, when we normalize the filter kernel to sum to 0, we highlight all the image parts where the values change rapidly and this way, we extract information (high-pass filter).

It is not always necessary since it depends on the objective that we seek to achieve.

(c) Explain two of the possible solutions for dealing with image-boundaries when filtering.

1. Padding the original image with zeros.
    a. We can think about it like an infinite image where you always find zeros whenever you get out of boundaries.

Part II

a. How does one make a Gaussian image pyramid that gets narrower toward the top?

The necessary steps to obtain a Gaussian pyramid that gets narrower toward the top are:

1. Apply a filter to smooth the image.
2. Subsample the image (half width and height) by taking every other pixel.

This steps will repeat **n** times, where **n** is the number of layers-1 that we want to get in our Gaussian pyramid.

In order to get this, we can use the `pyrDown()` function from the cv2 library in python which is used to downsample images.

b. Describe in words the effect of convolving an image by the kernel $k = \begin{pmatrix} 0 & 0 & 0 \\ 0 & 0 & 1 \\ 0 & 1 & 0 \end{pmatrix}$

Each pixel affixes the value of its right and bottom neighbor.

As a result of convolving an image with this kernel, we obtain a brighter image.

2. Padding the image with the first and last values.
    a. This is a similar solution to the last one, except that here you have an infinite image full of the limit values.

5	5	6	3	3
5	5	6	3	3
0	0	2	9	9
4	1	0	0	0
1	1	0	0	0

(d) Design one example of a 3x3 smoothing filter that emphasizes vertical neighbors, and emphasizes the middle pixel even more.

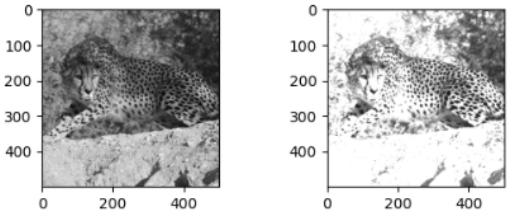
```
1/14 *  [[0, 3, 0],
         [0, 8, 0],
         [0, 3, 0]]
```

As we want to emphasize the vertical neighbors, we give them a greater value than the others (even better if we put zero to them). And as we want to emphasize the middle pixel even more, we give to it even a greater value.
Finally, we divide this matrix by 14 because the sum of all the elements is 14.

(e) Give three reasons that a 2D Gaussian makes for a good smoothing kernel.

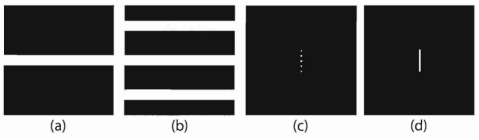
Using a Gaussian kernel is a good idea because it has several benefits, like:

1. It's a very well-known topic, which is good since there's a lot of information about it. Also,



Part III

In the subimages below, (a) and (b) denote two input images, while (c) and (d) are the magnitude of their Fourier transforms (not necessarily in that order). How are the images related, i.e., does $a \rightarrow c$ or $a \rightarrow d$? Please explain your reasoning.



The frequency domains have vertical lines since it's in that direction where we can find more change.

> a → d

The frequency corresponding to the letter **d** belongs to the image **a** since a single line in an image seems like a line as well in its frequency domain facing 90 degrees apart.

> b → c:

On the other hand, something similar happens with the frequency domain **c** but looking at its corresponding image, there are multiple changes, thus, there's less continuity and converts the line into a series of dots. The higher the frequency of changes, the more spread will the dots be.

as it's a common and effective strategy, this filter is part of other algorithms like edge detection and others.

This means it has theoretical and practical use cases.

2. Another benefit of this kernel is that it is a 2D Gaussian mask in the frequency domain, thus, it's useful in both spaces.
3. The smoothing degree can be easily controlled with the standard deviation and it's independent of the size of the kernel. That's a feature not so easy to find in other alternatives.

(f) (i) You are given only the Discrete Fourier Transforms (DFTs) of an image f and a filter-kernel g: F[f] and F[g], respectively. Explain how you would generate the filtered image version of f.

The steps to apply a kernel to an image given only Discrete Fourier Transforms are as follows:

- Go back to the space domain.
  - This is achieved by applying the `Inverse Discrete Fourier Transform`, which takes the **DFT** and converts it to an image where each cell is the intensity of a pixel.
  - And so the kernel has to be converted by using the `Inverse Discrete Fourier Transform`.

  Note: Depending on the given **DFTs**, *unshifting* might be necessary, this can be easily done with the `fft.ifftshift` function provided by numpy.
- Apply the kernel.

  Now that we have both our image and kernel in the space domain, the kernel has to be used to

Part IV

What is meant by Nyquist sampling rate?

It's the rate given by the Nyquist theorem and it's not image processing focused but that's one of its application fields. What this theorem basically states is:

> 💡 A continuous signal can be perfectly reconstructed from its discrete version using linear interpolation, if sampling occurred with frequency $f_s \geq 2f_{max}$

In this matter, we can conclude that aliasing would be avoided when downsampling an image if we take the Nyquist frequency or higher.

filter the image. And the steps to perform that task are:

- Place the center of the filter on each pixel of the image that we just generated.

  Note: If the kernel exceeds the image borders, several workarounds can be applied. See **Part I Question C**.

- In another image, store the result of a weighted sum, that is: the sum of intensities of each pixel multiplied by the values that the kernel has.

- Once those steps are done, the final image should be somewhat similar to the original one but with changes that vary depending on the filter.

  A noise reduction could have been applied, border detection, blur or whatnot.

(g) If F[G] looks like the following 2D image, what effect would this filter have on f?

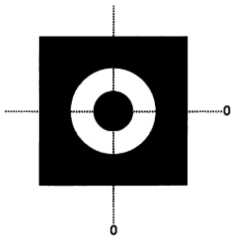


Figure F[G]. DFT of filter g, centered on frequency coefficients (0, 0).

A band-pass filter allows frequencies within a given range, so applying a filter like this would reject low and high values. The final result depends on the values of the band, but it’s commonly used to enhance the borders of an image, since it would reject the blurred version of the image and suppress noise at the same time.