UNIVERSIDADE ESTADUAL DE CAMPINAS
INSTITUTE OF COMPUTING

INTRODUCTION TO DIGITAL IMAGE PROCESSING
MC920 / MO443

WORK REPORT Nº 01

OPERATIONS ON IMAGENS IN THE SPACE DOMAIN

Maria Tejada-Begazo (RA 197488)

April
2021

Contents

1 Introduction

In this work, we are going to perform operations on monochrome and colored images. So, the first thing that comes to mind is wondering what it means to have a color. Color is the visual perception that is generated in the brain when visually perceiving the world. The brain interpreted the nerve signals sent by the eyes' photoreceptors that generate visual perception [1]. In Figure 1, the reflection of light from an object is captured by the retina. The retina has different cones (red, green, and blue sensing). It should be noted that the choice of the primary red, green and blue colors displayed by an image are not based on the sensitivity of the cones. Instead, the most widely spaced spectra are selected, allowing a wide color gamut to be produced [2].

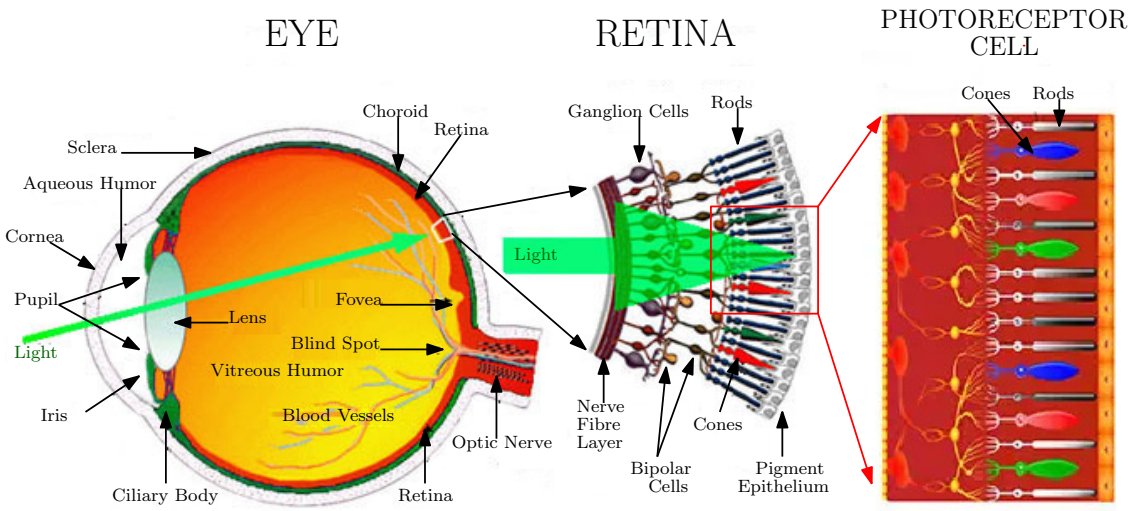


Figure 1: Parts of the Eye (Reproduced from [3, 4])

The attributes of color perception are brightness, hue, and saturation. The brightness represents the perceived luminance. Hue is the dominant color; for example, the difference between the hues of a monochromatic light source will be perceived by the difference in wavelength. Saturation is the purity of the color [5]. In the 1930s, the International Commission on Illuminance (CIE) standardized RGB representation by conducting color matching experiments using the primary color red (700 nm wavelength), green (546.1 nm), and blue. (435.8 nm) (Figure 2) [6].

It described a brief overview of RGB color space. However, the central objective is the operation in images. So, the convolution operator is the most used. The development of this work is based on this operator.

The convolution operator is based on a linear system, where an input signal $x(t)$ will give us an output $y(t)$ [7]. This relationship is determined by:

$$y(t) = \int_{-\infty}^{\infty} f(t,\tau) \times \tau \, d\tau \tag{1}$$

where the function f of two variables is valid for any linear system. Now imposing the displacement invariance constraint on an effort determines:

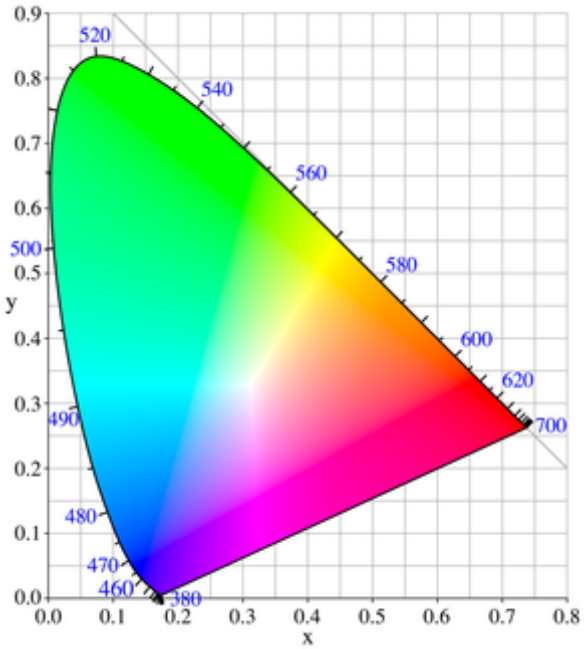


Figure 2: The CIE 1930 chromaticity chart (Reproduced from [2])

$$y(t-T) = \int_{-\infty}^{\infty} f(t,\tau) \times (t-T)\, d\tau \tag{2}$$

$$y(t) = \int_{-\infty}^{\infty} f(t+T,\tau+T) \times \tau\, d\tau \tag{3}$$

If we compare the Equation (1) and (3):

$$f(t,\tau) = f(t+T,\tau+T) \tag{4}$$

where these conditions are true for all T values. This means that $f(t,\tau)$ is constant as long as the difference between t and τ is constant. So, you can describe the function with the different $g(t-\tau) = f(1,\tau)$, as:

$$y(t) = \int_{-\infty}^{\infty} g(t-\tau) \times \tau\, d\tau \tag{5}$$

This stable integral that the output of a displacement invariant linear system is given by the convolution of the input signal with a function g which is characteristic of that system. When we need a discreet convolution. Therefore, the independent variables become an index, and the integral is replaced by a sum; that is determined by:

$$h(i) = f(i) * g(i) = \sum_j f(j) g(i-j) \tag{6}$$

In digital images, we use a two-dimensional convolution that is an extension of the convolution that we described previously. This convolution is also known as image filtering since it allows adjusting the local tone through the neighbors of a pixel. Convolution allows us to make soft blur, sharpen details, accentuate edges or eliminate noise. This operator determines the value of a pixel employing the

weighted sum of the values of its neighbors.

Figure 3 shows the convolution of $f(x,y)$ (image) with $g(x,y)$ (kernel), gives us output $h(x,y)$ (new image). The operation consists of passing the kernel throughout the image and storing the result in the central pixel. In this case, the strategy when the kernel is on the edge of the image is not to operate, so the final image has been reduced. The mathematical expression of the operation is determined by [6]:

$$h(i,j) = \sum_{k,l} f(i+k, j+l)g(k,l) \tag{7}$$

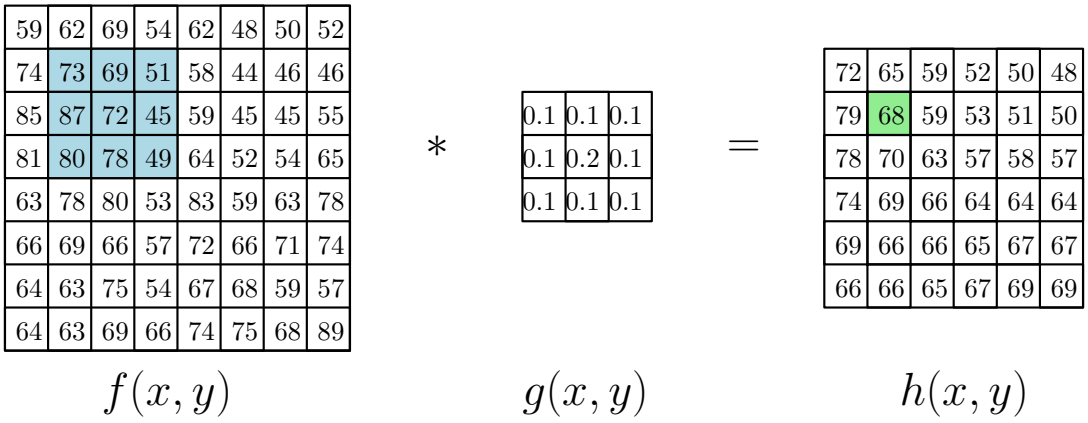


Figure 3: Convolution operator. The image on the left is convolved with the filter (kernel). Light blue pixels indicates the source neighborhood for the light green pixel [6]

2 Objectives

The objective of this work is to implement some operations in images, both monochromatic and colored, in the spatial domain. The masks and matrices presented below must be explicitly used in the codes, that is, any functions available in libraries should not be used in the implementation.

3 Development

This project develops in python 3.6 with the *OpenCV* and *skimage* libraries. The color and monochrome images must be copied in the **colorImages** and **grayImages** folders, respectively. The results of the exercises are saved in the **imgColor_result** and **imgGray_result** folder.

This project only accepts images in the PNG format. It allows to have different image dimensions in both types of images (color and monochrome).

3.1 Color Images

3.1.1 Color image alteration

When having a colorful image in RGB format, alter the image according to the following operation:

$$R' = 0.393R + 0.769G + 0.189B \quad (8)$$
$$G' = 0.349R + 0.686G + 0.168B \quad (9)$$
$$B' = 0.272R + 0.534G + 0.131B \quad (10)$$
$$(11)$$

The code made for the operation needs as input and output:

- **INPUT**:
  - img: The image is going to be tested.
  - filterR: The vector determined by $[0.393, 0.769, 0.189]$
  - filterG: The vector determined by $[0.349, 0.686, 0.168]$
  - filterB: The vector determined by $[0.272, 0.534, 0.131]$
- **OUTPUT**: New image

Code 2 allow us to perform this operation on the color image. It is observed that in line (2) to (4) the variables R, G, B. It was observed that the Blue and Red channels were inverted, so these variables allow us to correct this problem.

```
1  def filterRGB(img, filterR, filterG, filterB):
2      R = 2
3      G = 1
4      B = 0
5      row, col, bands = img.shape
6      r_colorImg = np.zeros((row, col, 3), np.uint8) #Only need 0-255
7
8      r_colorImg[:,:,R] = np.minimum(np.round(filterR[0]*img[:,:,R]) +
9                             np.round(filterR[1]*img[:,:,G]) +
10                            np.round(filterR[2]*img[:,:,B]), 255)
11     r_colorImg[:,:,G] = np.minimum(np.round(filterG[0]*img[:,:,R]) +
12                            np.round(filterG[1]*img[:,:,G]) +
13                            np.round(filterG[2]*img[:,:,B]), 255)
14     r_colorImg[:,:,B] = np.minimum(np.round(filterB[0]*img[:,:,R]) +
15                            np.round(filterB[1]*img[:,:,G]) +
16                            np.round(filterG[2]*img[:,:,B]), 255)
17     return r_colorImg
```

Code 1: Transformation of RGB to another color space (version 1)

3.1.2 Transformation of color to monochromatic image

If you have a color image in RBG, to change the image so that it only contains a single band of color, these values are calculated by the average weight:

$$I = 0.2989R + 0.5870G + 0.1140B \tag{12}$$

The code made for the operation needs as input and output:

- **INPUT**:
  - img: The image is going to be tested.
  - filterR: The vector determined by $[0.393, 0.769, 0.189]$
  - filterG: The vector determined by $[0.349, 0.686, 0.168]$
  - filterB: The vector determined by $[0.272, 0.534, 0.131]$
- **OUTPUT**: New image

Code 2 is observed in line (6), the output image is created only with the number of rows and columns of the original image. In addition, it is observed that the multiplication by the weights is rounded, since it should only be in a *uint8* type and verify that it does not go outside the limit of 255.

```
1  def RGB_to_Gray(img, weight):
2      R = 2
3      G = 1
4      B = 0
5      row, col, bands = img.shape
6      r_img = np.zeros((row, col), np.uint8) #Only need 0-255
7      r_img = np.minimum(np.round(weight[0]*img[:,:,R]) +
8                         np.round(weight[1]*img[:,:,G]) +
9                         np.round(weight[2]*img[:,:,B]), 255)
10
11     return r_img
```

Code 2: Transformation of color to monochromatic image

3.2 Monochromatic Images

It will focus on the filtration operations by means of operations of convolution of a mask by the image. This process is equivalent to running through the entire image, changing its values according to the mask weights and image intensities.

The Code 3 made for the operation needs as input and output:

- **INPUT**:
  - img: The image is going to be tested.
  - kernel: The mask. For example: $[[-1,0,1],[-2,0,2],[-1,0,1]]$ (h_1)
- **OUTPUT**: New image

Lines (6) and (7) establish a new image's end that depends on the mask size (Code 3). The new image is a copy of the original, that it is necessary to have consistency in the image's border. The *r_iter*,

c_iter are the point that saves the information to apply the mask. For example: we have a 3×3 mask, so the operation will start at position $(1, 1)$. And, If the image saves, then comment the lines (21) and (22); otherwise, it needs to rescale the pixel's intensity. This operation of pixels allows correcting the pixels negative and superior of 255.

```
1  def MyFilter(img, kernel):
2      row, col = img.shape
3      krow, kcol = kernel.shape
4
5      #The maximum row and col allows applied the kernel
6      max_row = (row - krow) +1
7      max_col = (col - kcol) +1
8
9      #Depend of the kernel size
10     r_iter = int(krow /2)
11     c_iter = int(kcol / 2)
12
13     r_img = np.zeros((row, col)) #np.copy(img)
14     r_img[:,:] = img[:,:]
15
16     for i in range(max_col*max_row):
17         r = int(np.floor(i/max_col))
18         c = int(i%max_col)
19         r_img[r+r_iter, c+c_iter] =np.sum(img[r:r+krow, c:c+kcol]*kernel)
20
21     #r_img = exposure.rescale_intensity(r_img, (0, 255))
22     #r_img = (r_img * 255).astype("uint8")
23     return r_img
```

Code 3: Operation of convolution on monochromatic images

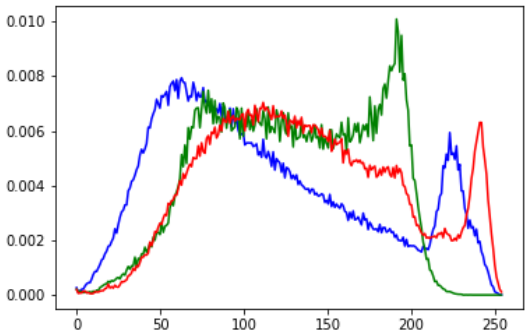
4 Results and Discussions

4.1 Color Images

4.1.1 Color image alteration

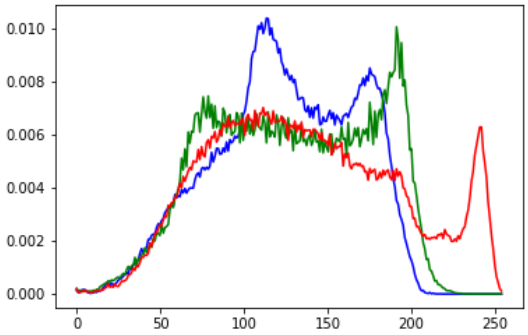
The results obtained from the color image alteration will be analyzed and discussed. It is observed in Image 1a is the original color image and 1c is the image resulting from the operation, which we observe that converts the image to sepia. The histogram of the original image (1b) shows a peak of the blue color of intensity 50. However, when it is transformed to sepia it is observed that the blue color extends from the intensity 75 to 200 (Image 1d).

a)



b)

c)

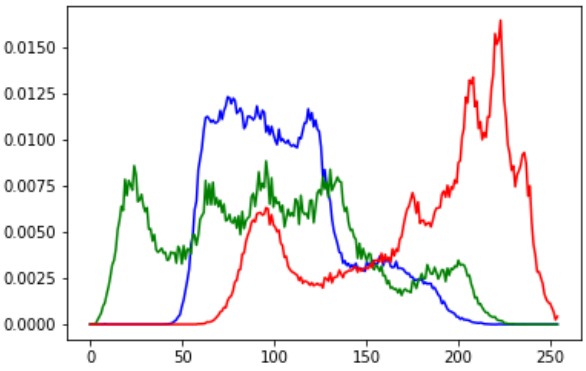


d)

Images 1: a) original baboon image, b) histogram of the original image, c) image resulting from the transformation, and d) histogram of the resulting image.

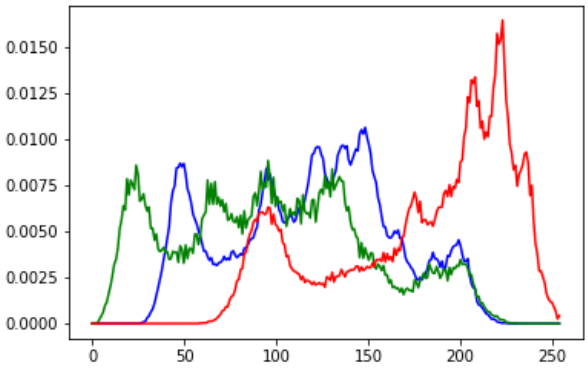
Image 32 shows that the image has been converted to sepia. However, the milestones of the image we see that the blue color is the one that expands. In image 2b the blue color is shown between $[50, 150]$, but in the 2d image it is shown that it has expanded $[40, 200]$.

a)



b)

c)

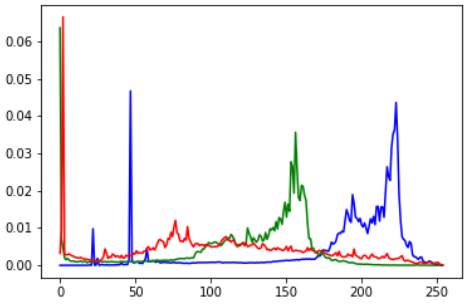


d)

Images 2: a) original Lenna image, b) histogram of the original image, c) image resulting from the transformation, and d) histogram of the resulting image.

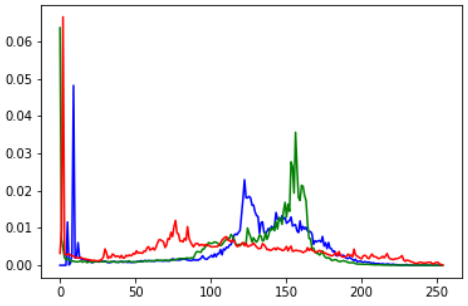
The interesting thing about Image 3 is the histogram, since Image 3b shows a peak of the blue color between the intensity 200 to 250. And after converting it to sepia, the peak of the blue color develops between the first intensities (Image 3d).

a)



b)

c)



d)

Images 3: a) original heaven image, b) histogram of the original image, c) image resulting from the transformation, and d) histogram of the resulting image.

This operation is to convert a color image to sepia. The most interesting are the histograms, since it is observed that the sepia image the color of blue is the one that changes with respect to the original histogram.

4.1.2 Transformation of color to monochromatic image

This operation allows you to change from a color image to monochrome. Different modifications have been made to the original formula to observe different aspects. The experiments were carried out with the following changes:

$$I = 0.2989R + 0.5870G + 0.1140B \quad (13)$$

$$I = 0.1140R + 0.5870G + 0.2989B \quad (14)$$

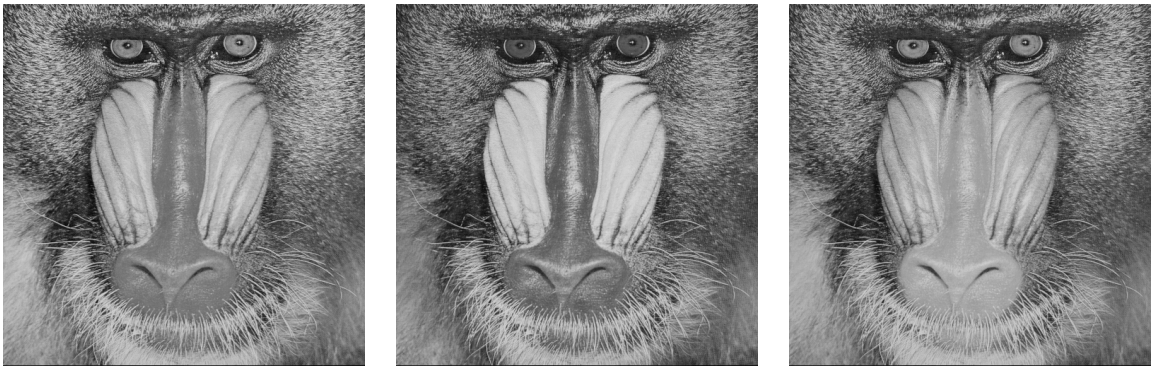
$$I = 0.2989R + 0.1140G + 0.5870B \quad (15)$$

$$I = 0.1140R + 0.2989G + 0.5870B \quad (16)$$

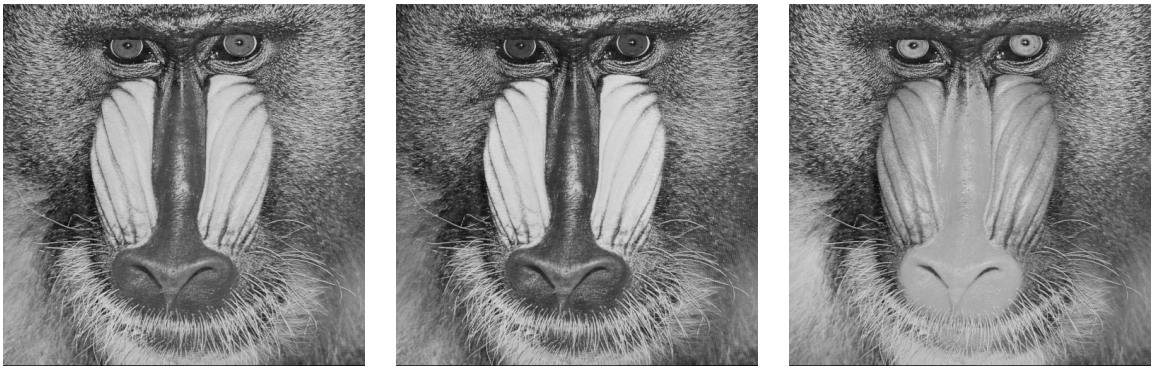
$$I = 0.5870R + 0.1140G + 0.2989B \quad (17)$$

$$I = 0.5870R + 0.2989G + 0.1140B \quad (18)$$

The original work was to observe the output of the Image 5a which we observed the image in monochromatic. Performing an exchange of the variable of band B with R, it is observed in Image 5d the eyes and nose darker than Image 5a. On the other hand, Image 5b presents a higher weighting for band B, a strong darkening is observed in the eyes, and by exchanging variables between G and R, we observe the darker edges. Finally, a greater weighting was made for the R band, and we observed that details of the central part of the nose have been lost (Image 5c and d).



a) b) c)



d) e) f)

Images 4: Result of the transformation of baboon: a) Experiment with Equation (13), a) Experiment with Equation (13), a) Experiment with Equation (13), b) Experiment with Equation (15), c) Experiment with Equation (17), d) Experiment with Equation (14), a) Experiment with Equation (16), and f) Experiment with Equation (18)

This operation allows us to make a correlation of the three bands to a single band. Also, some experiments have been proposed to obtain response because the green band is weighted more; It was shown in these experiments that Equation 13 is a good correlation to have the details without exaggeration.

4.2 Monochromatic Images

It will observe the results compared between my filter and *cv2.filter2D* to ensure that the results are correct. Image 5b has the result of my filter, and Image 5c is the result with *cv2*.

a) b) c)

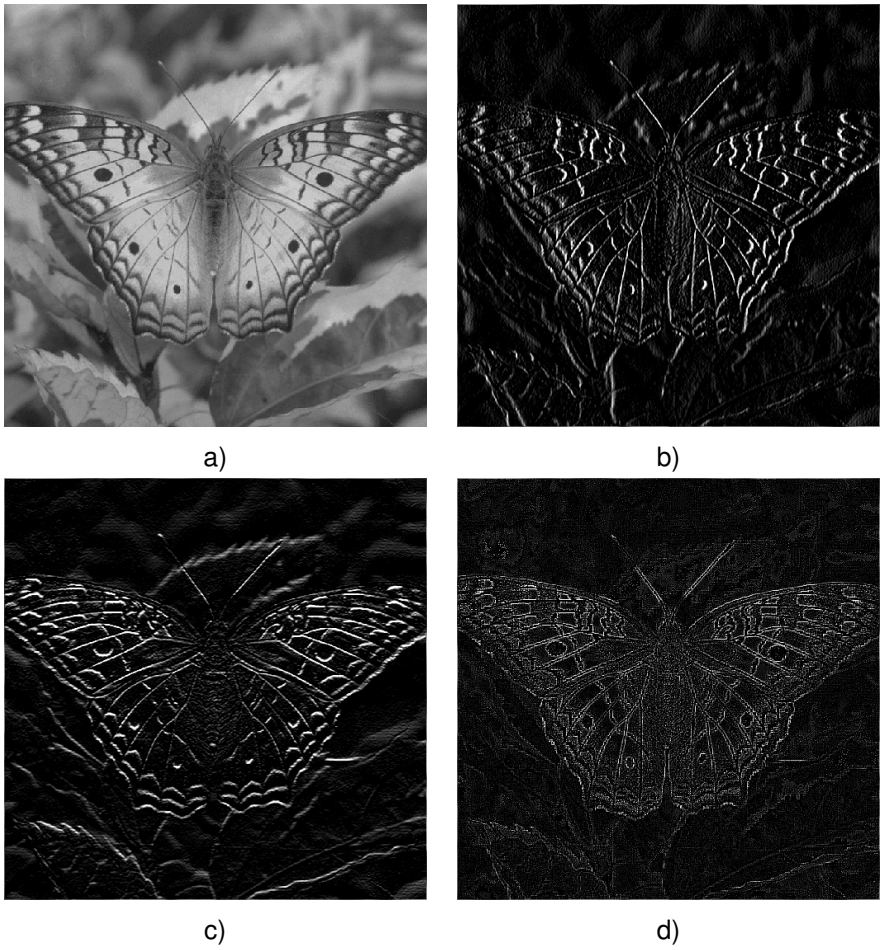
Images 5: a) is the image original, b) the result of my filter, and c) the result using *cv2*

Then, the results will be observed and analyzed with different masks. The masks are:

Image 6 shows us the results of the masks h_1 (Image 6b), h_2 (Image 6c), and h_3 (Image 6d), where

$$h_1 = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad h_3 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

they give us the detection of edges. The differences are in the procedures; in Image 6b it is obtained by scrolling the image vertically. Image 6c is obtained by scrolling horizontally. Finally, d) is edge detection without showing scrolling.



a) b) c) d)

Images 6: a) is the image original, b) the image with h_1 mask (Edge per column), c) the image with h_2 mask (Edge per row), and d) the image with h_3 (Edge detection)

$$h_4 = \frac{1}{9} \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

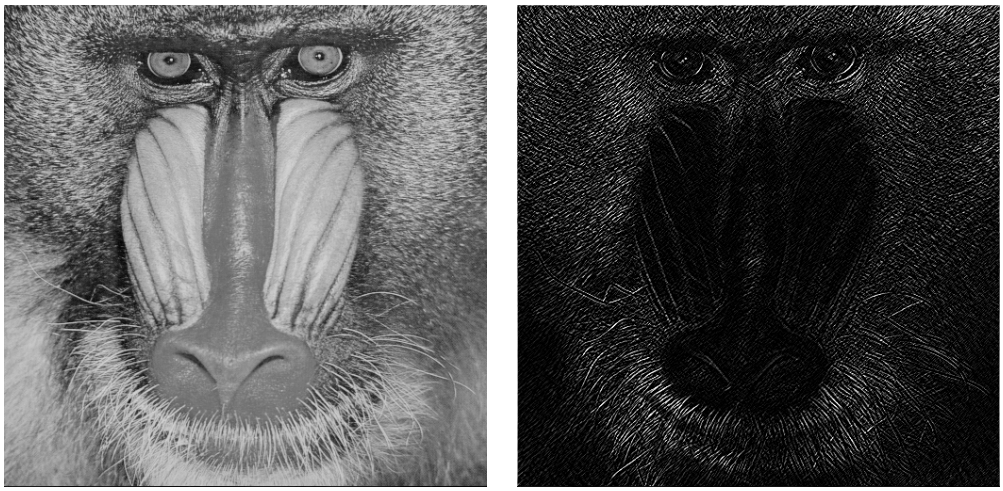
Image 7a the details of the bricks of the house. On the other hand, 7b) we observe that the bricks are not easily perceived. Therefore, h_4 is a mask that blurs the image.

a) b)

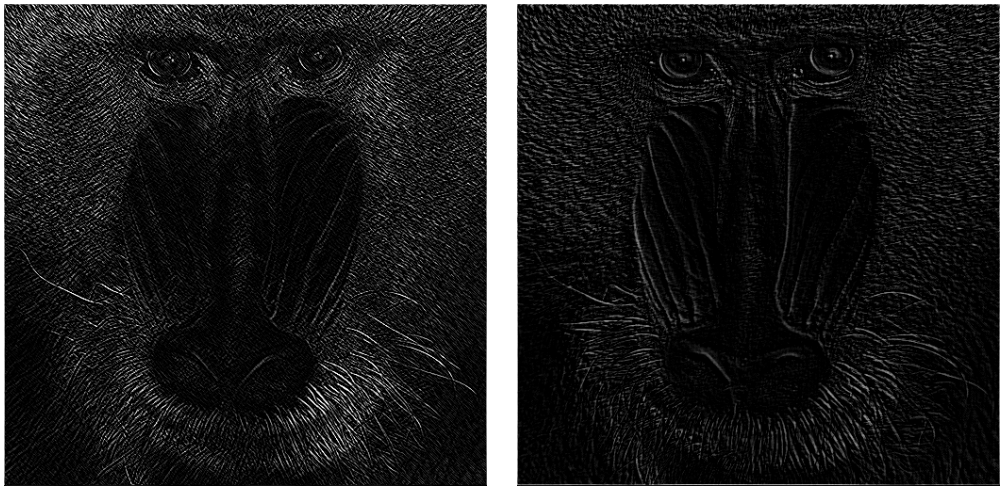
Images 7: a) is the image original, and b) the image with h_4 mask (Blurs filter)

$$h_5 = \begin{bmatrix} -1 & -1 & 2 \\ -1 & 2 & -1 \\ 2 & -1 & -1 \end{bmatrix} \qquad h_6 = \begin{bmatrix} 2 & -1 & -1 \\ -1 & 2 & -1 \\ -1 & -1 & 2 \end{bmatrix} \qquad h_7 = \begin{bmatrix} 0 & 0 & 1 \\ 0 & 0 & 0 \\ -1 & 0 & 0 \end{bmatrix}$$

Image 8a shows the original image. We can see 8b, 8c showing the darkest edges of the original image. For example, the nose has some more gray lines and is not easily seen in Images 8b and 8c. The most notable difference is that Image 8b (h_5) is shuffled through the main diagonal and 8c (h_6) is made through the secondary diagonal. The h_7 mask shows us an inversion of black with white.



a) b)

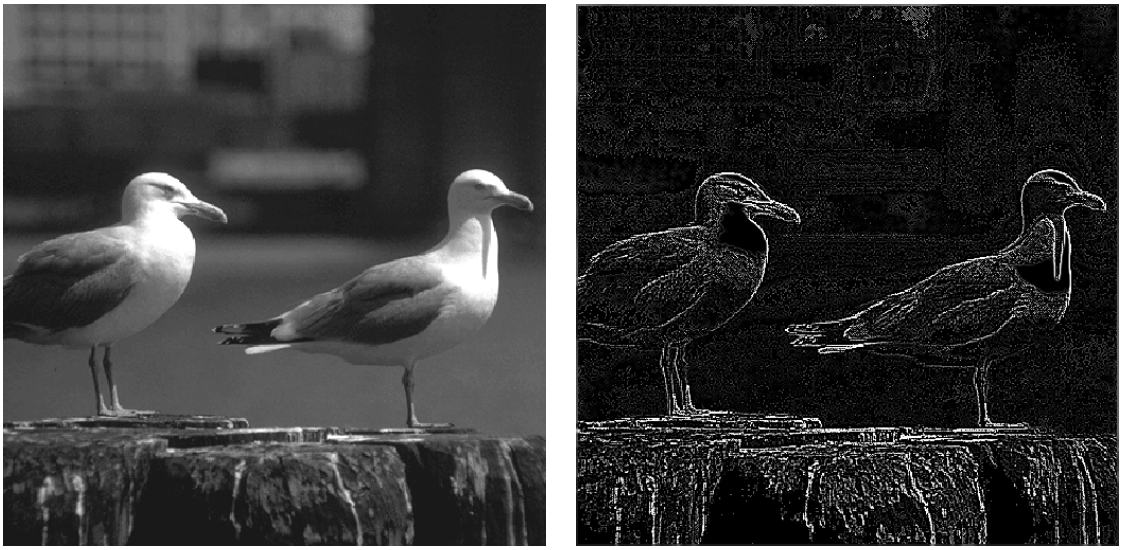


c) d)

Images 8: a) is the image original, b) the image with h_5 mask (Edge per principal diagonal), c) the image with h_6 mask (Edge per secondary diagonal), and d) the image with h_7 (Inversion of black with white)

$$h_8 = \begin{bmatrix} 0 & 0 & -1 & 0 & 0 \\ 0 & -1 & -2 & -1 & 0 \\ -1 & -2 & 16 & -2 & -1 \\ 0 & -1 & -2 & -1 & 0 \\ 0 & 0 & -1 & 0 & 0 \end{bmatrix}$$

h_8 highlights the edges of an image. Image 9b is observed details that simply have not achieved detect view. Therefore, this filter is important to obtain image details.



a) b)

Images 9: a) is the image original, and b) the image with h_8 mask

$$h_9 = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 16 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$

Image 10 shows the two masks h_4 and h_9. The h_9 image has a bit more blur than h_4; it is almost imperceptible.

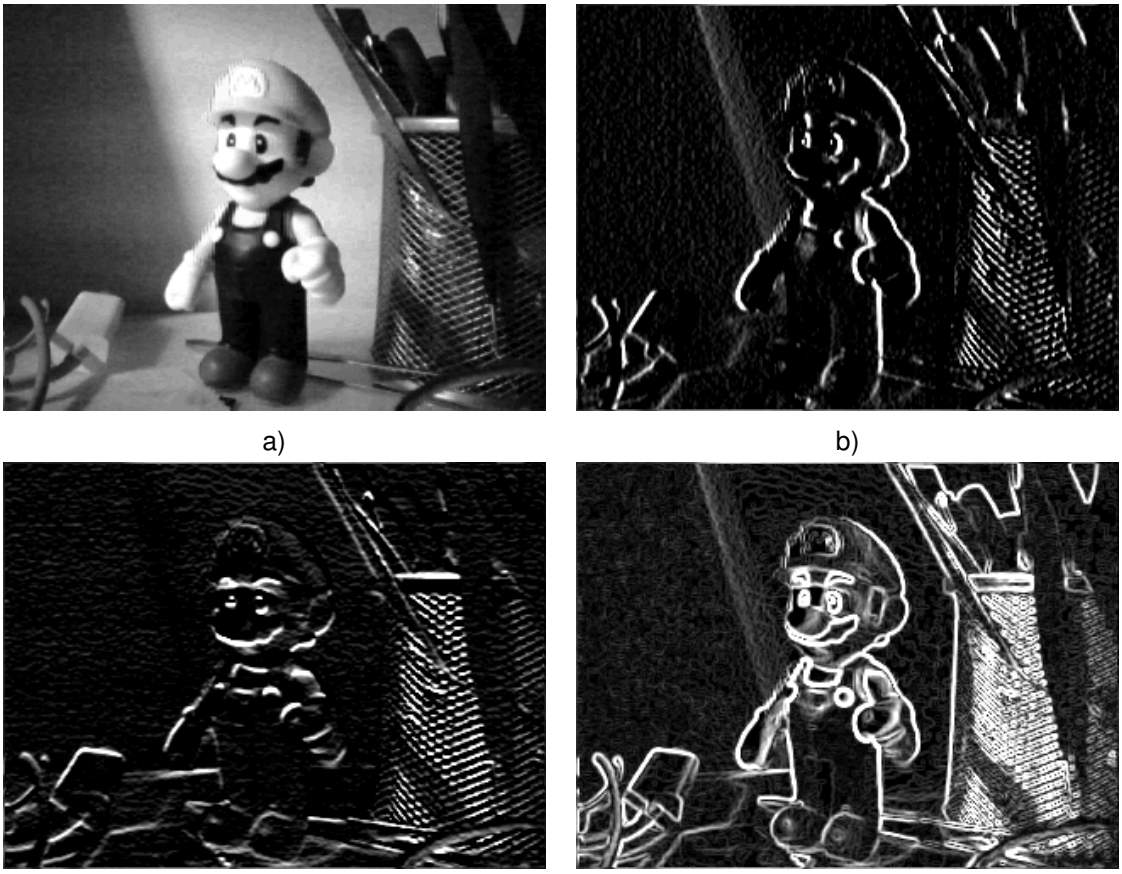
a) b) c)

Images 10: a) is the image original, and b) the image with h_4 mask (Blurs filter), c) the image with h_9 (Blurs filter)

Finally, the filters h_1 and h_2 are applied to the image individually. Then, they are combined by means of the expression:

$$sh_1h_2 = \sqrt{(h_1)^2 + (h_2)^2} \tag{19}$$

Image 11 shows the two images with the h_1 and h_2 masks. Image 11d shows the image after performing image Equation (19). Although the Images 11b, 11c failures observed contours of objects. Image 11d perfectly shows the contours of each object, that is, an enhancement of the edges of each object.



a) b)

c) d)

Images 11: a) is the image original, b) the image with h_2 mask, c) the image with h_2 mask, and d) the image with sh_1h_2

5 Conclusion

The implementation of the operations in the domain space has been successful. Throughout the project observed the matrix representation, that three-band vectors store the color for each pixel. Also, it understands that RGB is not the unique representation, since *OpenCV* represents the images in BGR, that is, the red and blue channels are interchanged.

The first exercise transforms a colored image into a sepia image. The second item will perform a transformation of the image from color to gray-scale. After making some modifications to the equation of the exercise, it was observed that the weighting of the green color must be higher to have a satisfactory result.

The first exercise transforms a colored image into a sepia image. The second item will perform a transformation of the image from color to gray-scale. After making some modifications to the equation, that the weighting of the green color must be higher to have a satisfactory result.

Finally, the convolution operation must be careful with the image's border, since the mask may come out of the image. Also, the masks give different results:

- h_1: Discover the edges of objects by scrolling columns.

- h_2: Finds the edges of objects by scrolling rows.
- h_3: Finds the edges of objects.
- h_4: Smooth an image.
- h_5: Finds the edges of objects by scrolling principal diagonal.
- h_6: Finds the edges of objects by scrolling secondary diagonal.
- h_7: Inverting the pixels, that is, the black pixels make them white.
- h_8: Enhance all the details in the image.
- h_9: Smooth an image.
- $\sqrt{h_1^2 + h_2^2}$: Finds and enhance the edges of objects.

References

[1] H.-C. Lee, *Introduction to color imaging science*. Cambridge University Press, 2005.

[2] D. Bull, *Communicating pictures: A course in Image and Video Coding*. Academic Press, 2014.

[3] H. Kolb, E. Fernandez, and R. Nelson, "The organization of the retina and visual system," *Webvision-The Organization of the Retina and Visual System*, 2005. [Online]. Available: https://webvision.med.utah.edu

[4] "Artificial retina," Lecture. [Online]. Available: https://www.dolcera.com/wiki/

[5] A. K. Jain, *Fundamentals of digital image processing*. Prentice-Hall, Inc., 1989.

[6] R. Szeliski, *Computer vision: algorithms and applications*. Springer Science & Business Media, 2010.

[7] K. R. Castleman, "Digital image processing," 1993.