Project number: 1

Course name: Computer Engineering

Student's name: João Victor Félix and Caroline Braz

Date due: 13/05/2024

Date handed in: 13/05/2024

DISCIPLINA: PROCESSAMENTO DIGITAL DE IMAGENS

Technical discussion and results

(One to three pages - max).

The following report aims to primarily report on the development of a project to explore the initial concepts learned in the course of Digital Image Processing. The project consists of elaborating two algorithms/functions to extract pixel values from grayscale images. All the codes were developed using Python.

The first function, called pixVal4e, receives three parameters: the grayscale picture and two scalar numbers that represent the row and column of the pixel in the picture. The output of this function must be the value of the pixel. At this function, the first step is to check if the numeric numbers received, represented by **r** (rows) and **c** (columns) are valid values based on the size of the picture. For this, the function **len** on Python was used. This function receives the matrix of the image and checks if the values are less than the parameters received. And also check if the values are greater than 0, because our function cannot receive negative numbers.

```
def pixVal4e(f, r, c):
      # check if the row and column are within the limits of the image
      if 0 <= r < len(f) and 0 <= c < len(f[0]):
            # if it's a valid value, return the pixel value
            return f[r][c]
      else:
            # if it's an invalid value, return nothing
            return None
```

To test this function, 2 different parameters were passed: the first one to find the pixel value on the origin and on the middle of the image.

```
v_origin = pixVal4e(f,1,1)
print('Valor do pixel na origem da imagem: {}'.format(v_origin))
v_middle = pixVal4e(f, 300, 269)
print('Valor do pixel no meio da imagem: {}'.format(v_middle))
```

Output:

Valor do pixel na origem da imagem: 133

Valor do pixel no meio da imagem: 60

The second function receives the picture as a parameter and waits for a click event on the image. Then, it returns the value of the pixel at the selected coordinates on the click. It uses the function pixValue4e created by passing the image and the coordinates as parameters. Then, it returns the value **v**, which is the value of the selected pixel.

```
import matplotlib.pyplot as plt
from pixVal4e import pixVal4e

def cursorValues4e(f):
      # listening for a click event
      def onclick(event):
            global r, c, v
            # coordinates r(row) and c(column)
            r, c = int(event.xdata), int(event.ydata)
            # pixel value
            v = pixVal4e(f, r, c)
            plt.close()
      # creates a new image and an ax axis object
      image, ax = plt.subplots()
      # show the image
      ax.imshow(f)
      #connects the onclick function to the button_press_event on the figure screen
      image.canvas.mpl_connect('button_press_event', onclick)
      # displays the previously created chart window
      plt.show()
      # checks the global variables r, c and v were defined by the onclick function
      if 'r' in globals() and 'c' in globals() and 'v' in globals():
            return (r, c, v)
      else:
            return None
```

This function show the image and wait for the click, then it return the values of the row, the column and the pixel clicked. To test this function, is necessary to set the picture **f** as parameter.

```
tupla = cursorValues4e(f)
print('r = {}\nc = {}\nv = {}'.format(tupla[0], tupla[1], tupla[2]))
```

Output:

r: 454
c: 10
v: 27

The other function is used to calculate the average grayscale value. It essentially sums the values of all the pixels in the picture and then divides by the number of pixels in the image.

```
def calculate_average_grayscale(f, r, c):
      sum_pixels = 0
      for row in f:
            for pixel in row:
                  sum_pixels += pixel
      # total number of pixels
      total_pixels = r * c
      # average of the pixel values
      average_grayscale = sum_pixels / total_pixels
      return nivel_medio_cinza
```

```
      average_grayscale = calculate_average_grayscale(f, column, row)
      print('O nível médio de cinza é {:.2f}'.format(average_grayscale ))
```

Output:

O nível médio de cinza é: 99.91

Other information that can be extracted from the picture are the size in column x row values and also the bit depth. With that values we can infer other informations, for example: multiplying the number of rows and the number of columns, it is possible to find the total number of pixels at the picture, and, knowing that there is 8 bits at each pixel, calculate the range of intensity levels, considering k = 8, then the number of bits per pixel is $2^k = 2^8 = 256$, finally the conclusion is that the range of values is between 0 (for black) and 255 (for white).

```
      column, row = f.shape # get the size of the picture
      print('A dimensão da imagem é {}x{}'.format(column, row))
      depth_bit = f.dtype.itemsize * 8
      print('A profundidade de bits é {}'.format(depth_bit))
```

Output:

A dimensão da imagem é 600x469

A profundidade de bits é 8

Finally, we can convert the JPG image to a .tif file, which is an image type supported by various applications and platforms.

```
      cv2.imwrite('imagem_salva.tif', f)
      print('Imagem salva no formato tif!')
```

References

(submit here the references cited).

Python. "The Python Standard Library — Python 3.8.1 Documentation." Python.org, 2020, docs.python.org/3/library/index.html.

Guimarães, Marly. "UNIDADE II - Fundamentos de Imagem - Representação de Imagens Digitais." *Google Docs*, drive.google.com/file/d/1EW13lV46qw49tGG3pkGbHrkTReHF6R8T/view.