CENTRO FEDERAL DE EDUCAÇÃO TECNOLÓGICA DE MINAS GERAIS

BRUNO NASCIMENTO DAMACENA

Problem Set 1

Problem Set proposed by Flávio Luis Cardeal Pádua for the Computer Vision course from Computer Department of CEFET-MG

Belo Horizonte
April 2018

1 Basic Acquaintance with Programmed Imaging

1.1

For loading an image using OpenCV and Python, it's basically use the cv command for loading the image from the root folder:

```
import cv2
image = cv2.imread('image.png')
```

1.2

To generate and display and histogram, it will be needed to use the MatPlotLib library. So, it's basically import the library, calculate the histogram for each color in the RGB channels, and show the plot.

```
for i, col in enumerate(('b','g','r')):
    histogram = cv2.calcHist([image], [i], None, [256],
    [0,256])
    pyplot.plot(histogram,col)
pyplot.show()
```

When the histogram is finished, it will be ploted like this:

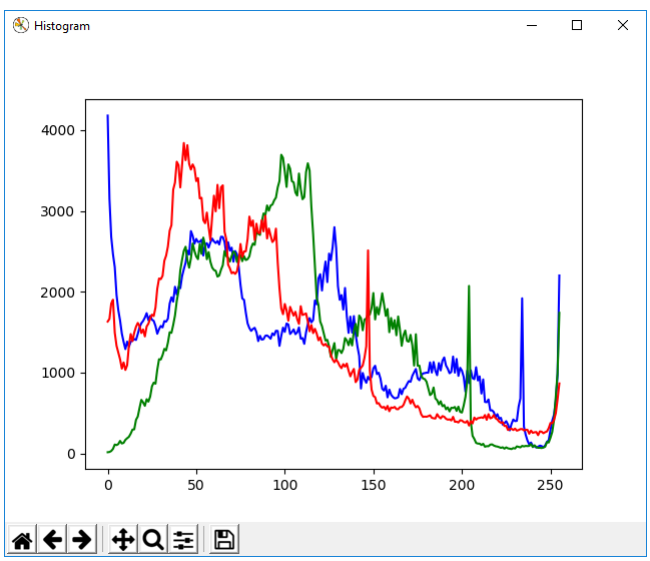

Figure 1: Histogram

1.3

For this section, it will be needed to set a callback for each time that an action happens. The chosen action was a mouse movement. When this action happens, I draw a square around the center of the mouse pointer, and calculate the position, color, intensity, mean and standart deviation of this square. It's basically like this:

```
def printRectangle(x,y,imgToShow):
    pt1 = (x-5, y-5)
    pt2 = (x+5, y+5)
    cv2.rectangle(imgToShow, pt1, pt2, (255,0,0), 1)
    cv2.imshow('Image', imgToShow)

def showParameters(x,y):
    color = img[y,x]
    intensity = numpy.average(color)
    mean = numpy.average(img[y-5:y+5, x-5:x+5])
    standartDeviation = numpy.std(img[y-5:y+5, x-5:x+5])
    print('Position:', (x,y))
    print('Color: ', color)
    print('Intensity: ', intensity)
    print('Mean: ', mean)
    print('Standart Deviation: ', standartDeviation)

def onMouseMovement(event,x,y,flags,param):
    if(event == cv2.EVENT_MOUSEMOVE):
        imgToShow = img.copy()
        printRectangle(x,y,imgToShow)
        showParameters(x,y)

cv2.setMouseCallback('Image', onMouseMovement)
```

Then, the image will load, and when you move you mouse over it, a blue square will show up:

For this 11x11 square, i'll have the following intensity, mean and standart deviation:

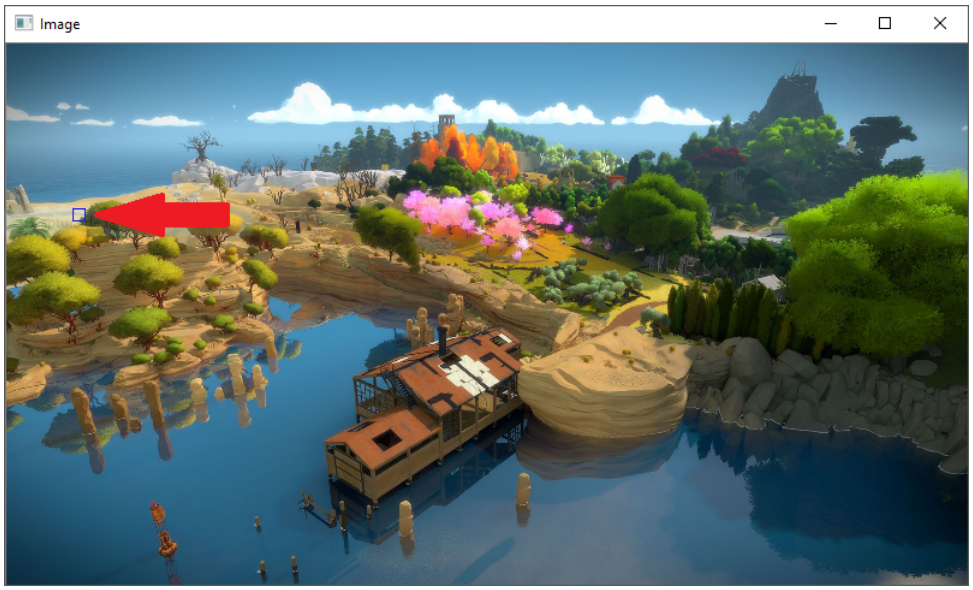

Figure 2: Image with the blue square

```
Position: (200 477)
Color:  [117 178 157]
Intensity:  150.66666666666666
Mean:  90.31666666666666
Standart Deviation:  39.633441967891486
```

Figure 3: Terminal Output for the previous 11x11 square

1.4

In terms of histogram, a homogeneous image have close values from all three channels. Thus, a inhomogeneous image will have peaks and valleys on all three channels of the histogram. In terms of mean and variance, a property of homogeneous images are a steady mean value, which implies lower variations.

3 Different Impacts of Amplitudes and Phase in Frequency Space on Resulting Filtered Images

3.1

First of all, I need to load the images and apply the Fourier Transform in each one of them:

```
img1 = cv2.imread('01_03_01.jpg', 0)
img2 = cv2.imread('01_03_02.jpg', 0)

fourierTransform1 = numpy.fft.fft2(img1)
fourierTransform2 = numpy.fft.fft2(img2)
```

Them, for each Fourier Transform, I can extract the amplitude and the phase of the images, like this:

```
amplitude1 = numpy.absolute(fourierTransform)
phase1 = numpy.angle(fourierTransform)

amplitude2 = numpy.absolute(fourierTransform)
phase2 = numpy.angle(fourierTransform)
```

Then, with the amplitude and phase, I can basically mess around with it. For instance, I can rearange the images, swapping the amplitude or phases values, like this:

```
img3 = amplitude1*(numpy.sin(phase2)*1j
+ numpy.cos(phase2))
img4 = amplitude2*(numpy.sin(phase1)*1j
+ numpy.cos(phase1))

img3 = numpy.ndarray.astype(numpy.absolute
(numpy.fft.ifft2(img3)), numpy.uint8)
img4 = numpy.ndarray.astype(numpy.absolute
(numpy.fft.ifft2(img4)), numpy.uint8)
```

Here is the result:

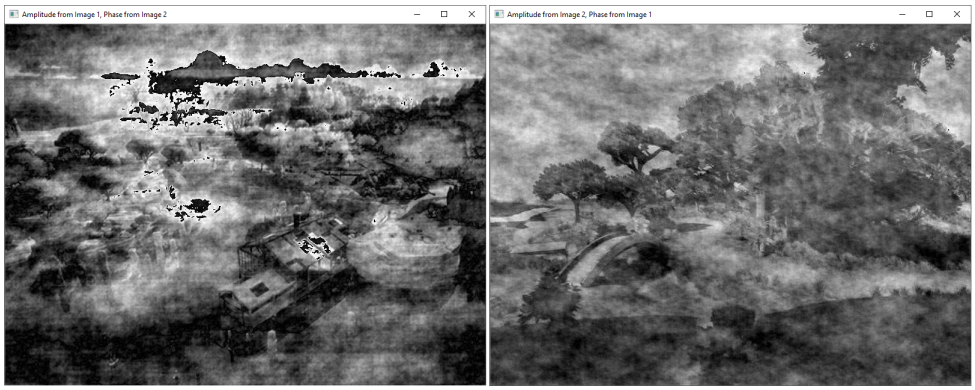

Figure 4: Mixed up images

3.2

We can also change the amplitude or the frequency of the image. Let's see what happens:

Figure 5: Original Image

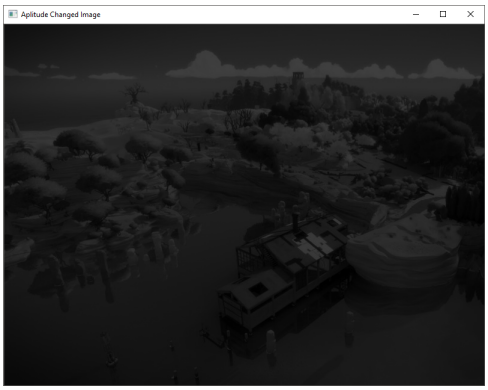

Figure 6: Amplitude Changed Image

Figure 7: Phase Changed Image

The more significative change is on the phase, which let the image all blurried. The amplitude messed with the brightness of the image.

3.3

I did the same measurements for human pictures. Here are some results:

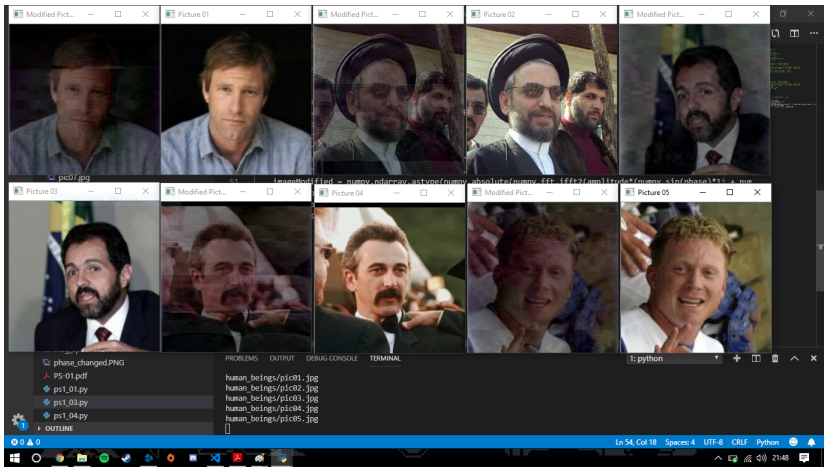

Figure 8: Amplitude and Phase linear changes on human pictures

4 Approximating the HSI Space by Planar Cuts

First of all, I created a 600x600 black image. Then, for each pixel of the image, I calculated the polar coordinate related to the center of the image. Then, I applied the reverse of this expressions: I reversed it like this:

$$H = \begin{cases} \delta & \text{if } B \leq G \\ 2\pi - \delta & \text{if } B > G \end{cases} \quad \text{with}$$

$$\delta = \arccos \frac{(R-G)+(R-B)}{2\sqrt{(R-G)^2+(R-B)(G-B)}} \quad \text{in } [0, \pi)$$

$$S = 1 - 3 \cdot \frac{\min\{R, G, B\}}{R+G+B}$$

Figure 9: Expressions for hue, saturation and intensity

```
def hsiToBGR(colorHSI):
    colorBGR = [0, 0, 0]
    if(HSI[1]==0):
        colorBGR = [HSI[1], HSI[1], HSI[1]]
    else:
        if(HSI[0]<2*math.pi/3):
```

```
            colorBGR[0] = 255*HSI[2]*(1-HSI[1])
            colorBGR[1] = 255*HSI[2]*(1+HSI[1]*math.cos
            (2*math.pi/3-HSI[0])/
            (math.cos(math.pi/3-HSI[0])))
            colorBGR[2] = 255*HSI[2]*(1+HSI[1]*math.cos
            (HSI[0])/(math.cos(math.pi/3-HSI[0])))
        elif(HSI[0]<4*math.pi/3):
            HSI[0] -= 2*math.pi/3
            colorBGR[2] = 255*HSI[2]*(1-HSI[1])
            colorBGR[0] = 255*HSI[2]*(1+HSI[1]*math.cos
            (2*math.pi/3-HSI[0])/(math.cos
            (math.pi/3-HSI[0])))
            colorBGR[1] = 255*HSI[2]*(1+HSI[1]*math.cos
            (HSI[0])/(math.cos(math.pi/3-HSI[0])))
        else:
            HSI[0] -= 4*math.pi/3
            colorBGR[1] = 255*HSI[2]*(1-HSI[1])
            colorBGR[2] = 255*HSI[2]*(1+HSI[1]*math.cos
            (2*math.pi/3-HSI[0])/
            (math.cos(math.pi/3-HSI[0])))
            colorBGR[0] = 255*HSI[2]*(1+HSI[1]*math.cos
            (HSI[0])/(math.cos(math.pi/3-HSI[0])))
    return colorBGR
```

Then, I must set a value for the intensity, in order to see the HSI Planar Cut. For instance, for I = 85:

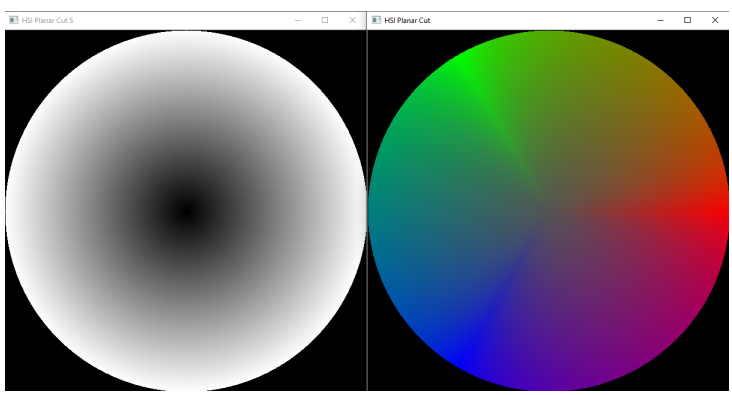

Figure 10: Intensity = 85