Visão computacional processamento de imagens

Luciana G. Fazan
Aluna do curso pós-graduação UFPR

1 Técnicas de pré-processamento de imagens

Para execução deste trabalho foi utilizado o GoogleColab. Código fonte está disponível no git: `https://github.com/ffaza/ta1` para clonar: https://github.com/ffaza/ta1.git

1.1 Converting image to grayscale

```
!pip install cv2_plt_imshow
import cv2
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
image= cv2.imread('drive/My Drive/IMAGENS/lena.png')
imageGray = cv2.cvtColor(image, cv2.COLOR_BGR2GRAY)
cv2_plt_imshow(imageGray)
cv2.waitKey(0)
```

Listing 1: Código fonte[1]

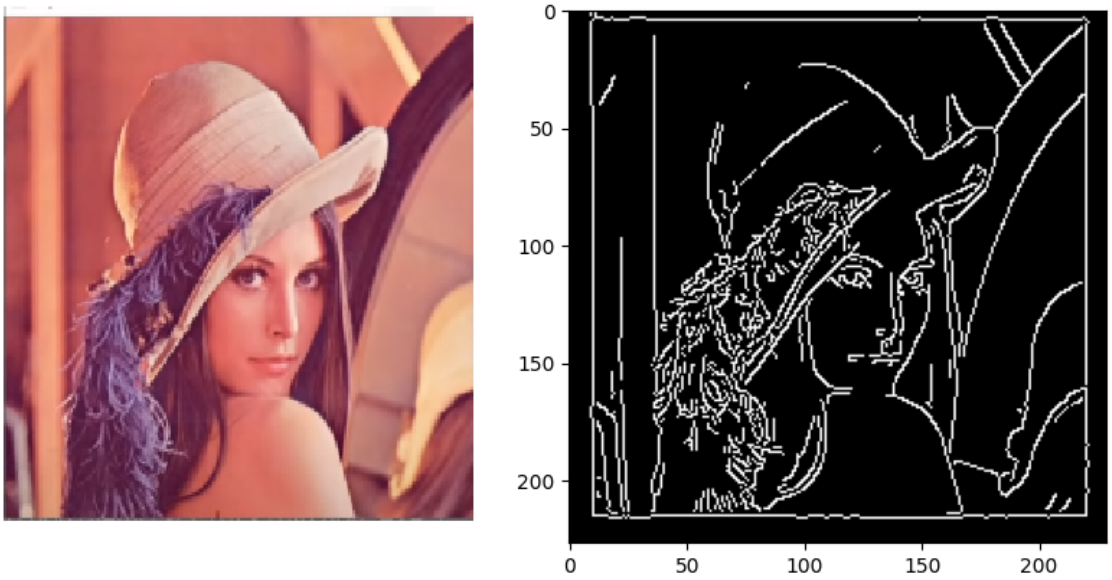
Figure 1: Grayscale

1.2 Edge detection

```
!pip install cv2_plt_imshow
import cv2
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
image = cv2.imread('drive/My Drive/IMAGENS/lena.png')
imageCanny = cv2.Canny(image, 150, 200)
cv2_plt_imshow(imageCanny)
cv2.waitKey(0)
```

Listing 2: Código fonte[1]

Figure 2: Edge



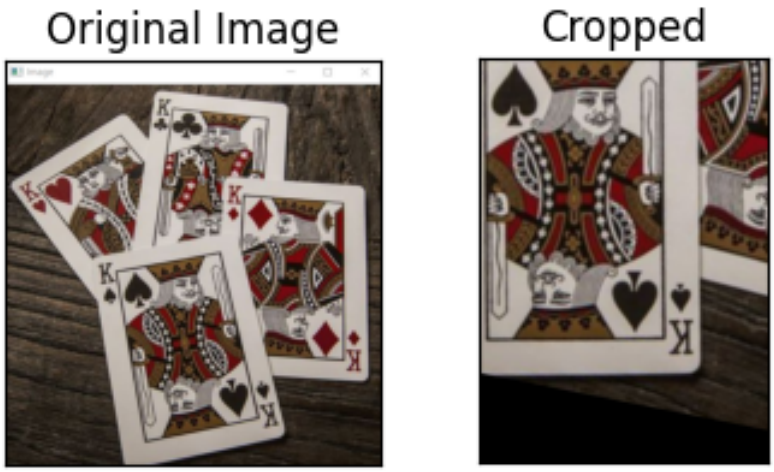
1.3 Cropping image

```
!pip install cv2_plt_imshow
import cv2
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
```

```
drive.mount('drive')
image = cv2.imread('drive/My Drive/IMAGENS/cards.png')
width, height = 250, 350
point1 = np.float32([[111, 219], [287, 188], [154, 482], [352, 440]])
point2 = np.float32([[0, 0], [width, 0], [0, height], [width, height]])
matrix = cv2.getPerspectiveTransform(point1, point2)
Output = cv2.warpPerspective(image, matrix, (width, height))
plt.subplot(221),plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(223),plt.imshow(cv2.cvtColor(Output, cv2.COLOR_BGR2RGB))
plt.title('Cropped'), plt.xticks([]), plt.yticks([])
plt.show()
```

Listing 3: Código fonte[1]

Figure 3: Cropping



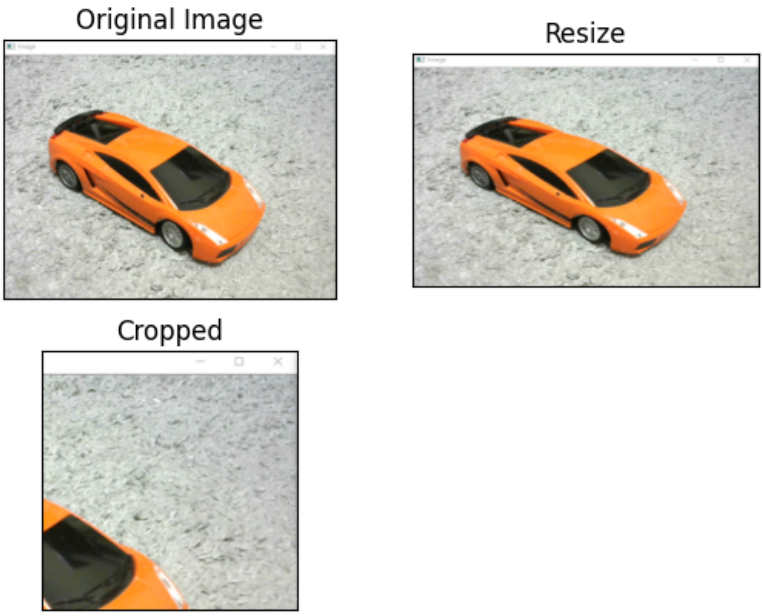
1.4 Resize and Crop

```
!pip install cv2_plt_imshow
import cv2
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
img = cv2.imread('drive/My Drive/IMAGENS/lamborg.png')
print(img.shape)
imgResize = cv2.resize(img,(300,200))
print(imgResize.shape)
```

```
imgCropped = img[0:200,200:400]
plt.subplot(221),plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(222),plt.imshow(cv2.cvtColor(imgResize, cv2.COLOR_BGR2RGB))
plt.title('Resize'), plt.xticks([]), plt.yticks([])
plt.subplot(223),plt.imshow(cv2.cvtColor(imgCropped, cv2.COLOR_BGR2RGB))
plt.title('Cropped'), plt.xticks([]), plt.yticks([])
plt.show()
```

Listing 4: Código fonte[2]

Figure 4: Resize and Crop



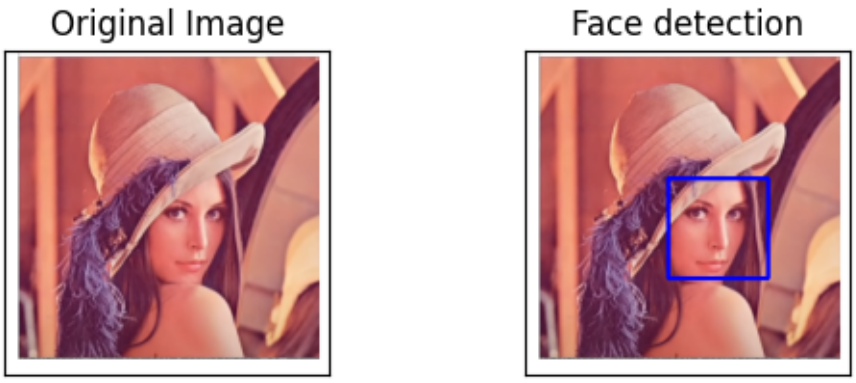
1.5 Face detection

```
#!pip install opencv-contrib-python
import cv2
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
image = cv2.imread('drive/My Drive/IMAGENS/lena.png')
image2 = cv2.imread('drive/My Drive/IMAGENS/lena.png')
faceCascade = cv2.CascadeClassifier(cv2.data.haarcascades + "
    haarcascade_frontalface_default.xml")
```

```
print(cv2.CascadeClassifier.empty(faceCascade) )
#imgGray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
faces = faceCascade.detectMultiScale(image)
for (x, y, w, h) in faces:
   nova=cv2.rectangle(image, (x, y), (x + w, y + h), (255, 0, 0), 2)
#cv2_plt_imshow(image)
#cv2.waitKey(0)
plt.subplot(221),plt.imshow(cv2.cvtColor(image2, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(222),plt.imshow(cv2.cvtColor(nova, cv2.COLOR_BGR2RGB))
plt.title('Face detection'), plt.xticks([]), plt.yticks([])
plt.show()
```

Listing 5: Código fonte[1]

Figure 5: Face detection



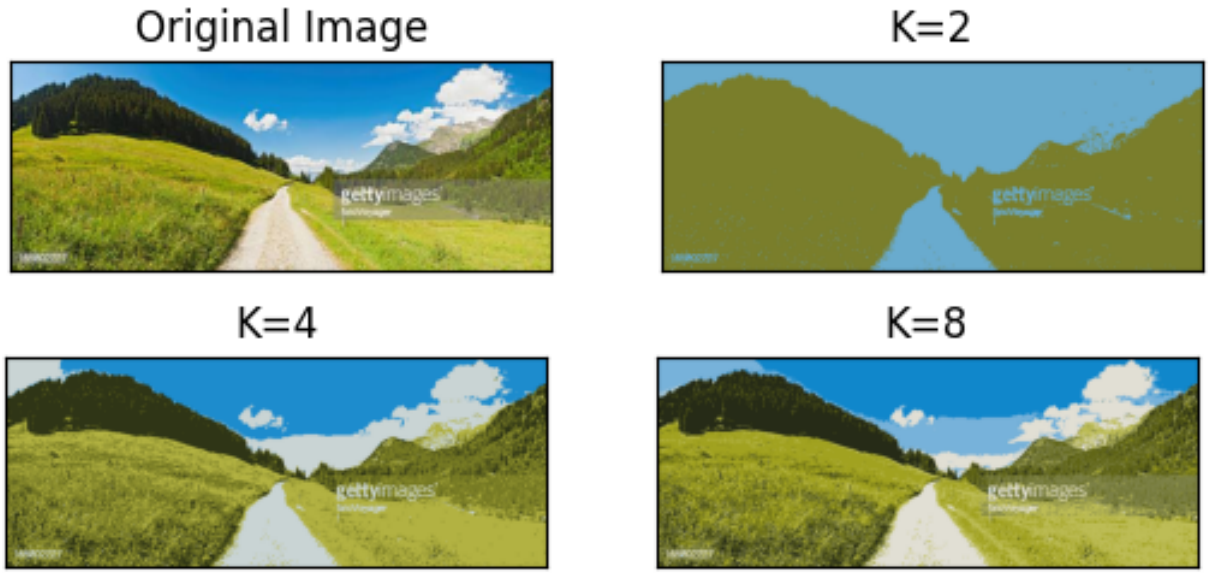
1.6 Quartization

```
# import required libraries
import matplotlib.pyplot as plt
import cv2
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
img = cv2.imread('drive/My Drive/IMAGENS/horizon.jpg')
Z = img.reshape((-1,3))
# convert to np.float32
Z = np.float32(Z)
# define criteria, number of clusters(K) and apply kmeans()
criteria = (cv2.TERM_CRITERIA_EPS + cv2.TERM_CRITERIA_MAX_ITER, 10, 1.0)
def colorQuant(Z, K, criteria):
```

```
    ret,label,center=cv2.kmeans(Z,K,None,criteria,10,cv2.
        KMEANS_RANDOM_CENTERS)
    center = np.uint8(center)
    res = center[label.flatten()]
    res2 = res.reshape((img.shape))
    return res2
res1 = colorQuant(Z, 2, criteria)
res2 = colorQuant(Z, 5, criteria)
res3 = colorQuant(Z, 8, criteria)
plt.subplot(221),plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(222),plt.imshow(cv2.cvtColor(res1, cv2.COLOR_BGR2RGB))
plt.title('K=2'), plt.xticks([]), plt.yticks([])
plt.subplot(223),plt.imshow(cv2.cvtColor(res2, cv2.COLOR_BGR2RGB))
plt.title('K=4'), plt.xticks([]), plt.yticks([])
plt.subplot(224),plt.imshow(cv2.cvtColor(res3, cv2.COLOR_BGR2RGB))
plt.title('K=8'), plt.xticks([]), plt.yticks([])
```

Listing 6: Código fonte[3]

Figure 6: Quartization com valor de K=2, 4 e 8



1.7 UpSampling

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
```

```
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
imageOriginal = cv2.imread('drive/My Drive/IMAGENS/potrait.jpeg')
image = cv2.imread('drive/My Drive/IMAGENS/potrait.jpeg')
image=cv2.pyrUp(image)
plt.subplot(221),plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('Original Image. Size (130, 130, 3)'), plt.xticks([]), plt.yticks
    ([])
plt.subplot(222),plt.imshow(cv2.cvtColor(image, cv2.COLOR_BGR2RGB))
plt.title('UpSample. Size (260, 260, 3)'), plt.xticks([]), plt.yticks([])
plt.show()
print("Size of image before pyrUp: ", imageOriginal.shape)
print("Size of image after pyrUp: ", image.shape)
```

Listing 7: Código fonte[4]

Figure 7: UpSampling

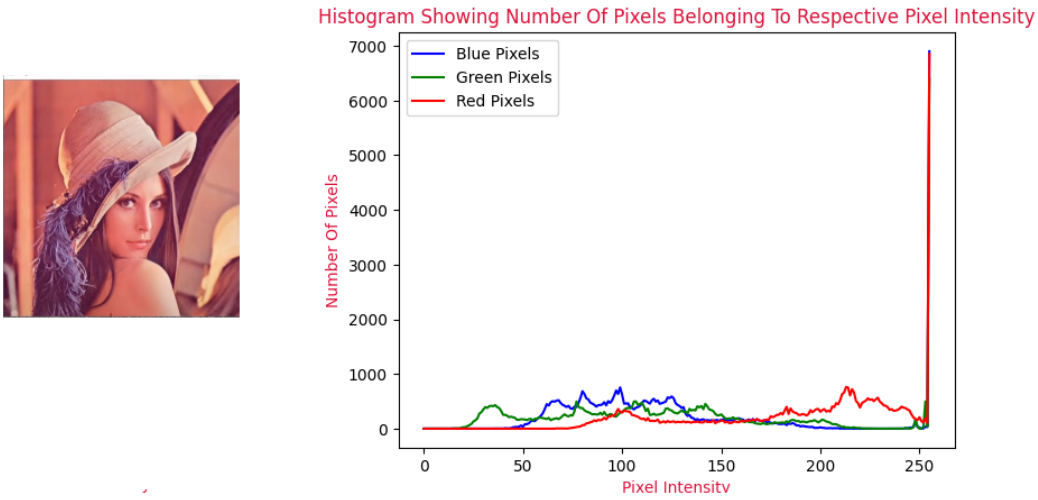
1.8 Intensity

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
image2 = cv2.imread('drive/My Drive/IMAGENS/lena.png')
colors = ('blue', 'green', 'red')
label = ("Blue", "Green", "Red")
```

```
for count,color in enumerate(colors):
    histogram = cv2.calcHist([image2],[count],None,[256],[0,256])
    plt.plot(histogram,color = color, label=label[count]+str(" Pixels"))
plt.title("Histogram Showing Number Of Pixels Belonging To Respective Pixel
     Intensity", color="crimson")
plt.ylabel("Number Of Pixels", color="crimson")
plt.xlabel("Pixel Intensity", color="crimson")
plt.legend(numpoints=1, loc="best")
```

Listing 8: Código fonteciteHistogramShowingNumber

Figure 8: Histograma de intensidade



1.9 Log transformation

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
image = cv2.imread('drive/My Drive/IMAGENS/arvore.jpg')
imageOriginal = cv2.imread('drive/My Drive/IMAGENS/arvore.jpg')
c = 255 / np.log(1 + np.max(image))
log_image = c * (np.log(image + 1))
log_image = np.array(log_image, dtype = np.uint8)
plt.subplot(221),plt.imshow(cv2.cvtColor(imageOriginal, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(222),plt.imshow(cv2.cvtColor(log_image, cv2.COLOR_BGR2RGB))
plt.title('Log transformation'), plt.xticks([]), plt.yticks([])
```

```
plt.show()
```

Listing 9: Código fonte[5]

Figure 9: Log transformation

1.10 Gamma transformation

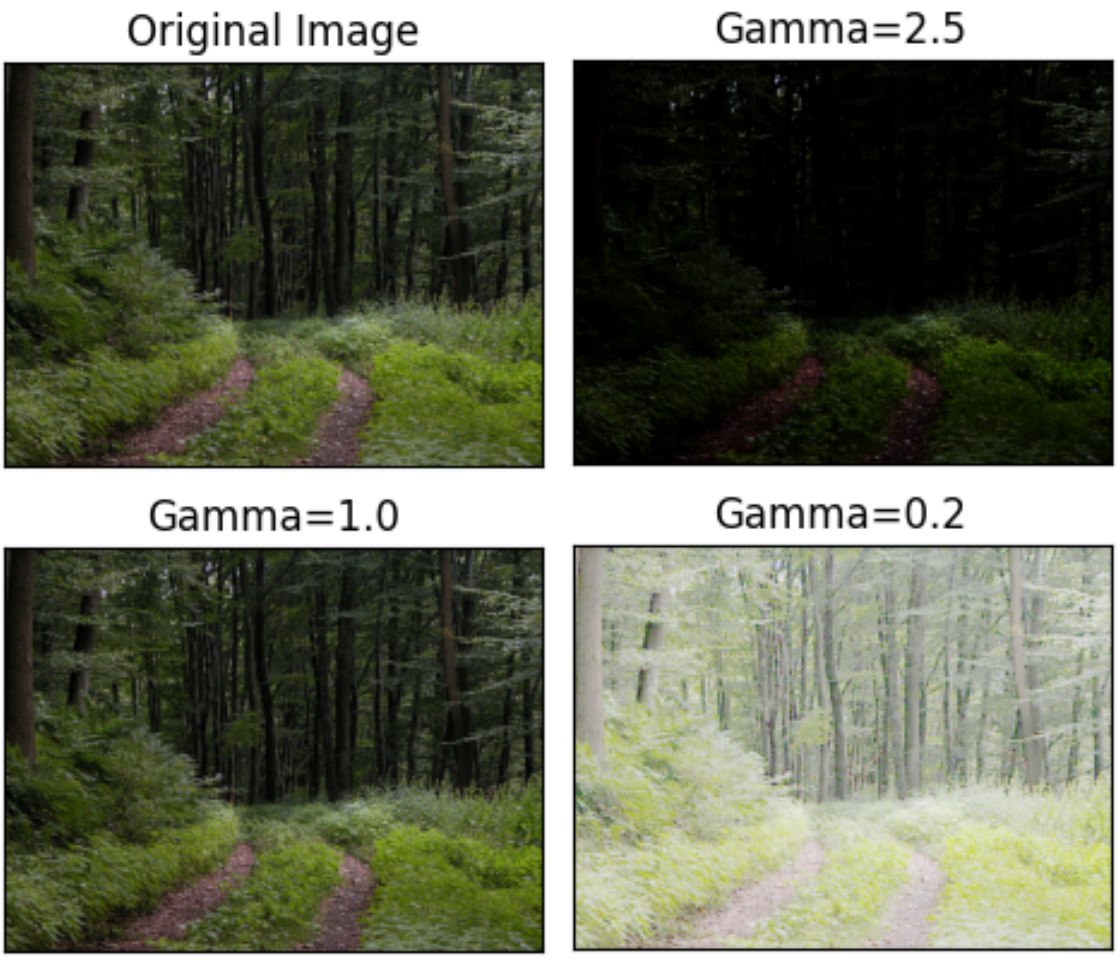
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
img = cv2.imread('drive/My Drive/IMAGENS/IMAGEM10.png')
imgoriginal = cv2.imread('drive/My Drive/IMAGENS/IMAGEM10.png')
#Gamma=2.5
gamma_a = np.array(255*(img/255)**2.5,dtype='uint8')
#Gamma=1.0
gamma_b = np.array(255*(img/255)**1.0,dtype='uint8')
#Gamma=0.2
gamma_c = np.array(255*(img/255)**0.2,dtype='uint8')
plt.subplot(221),plt.imshow(cv2.cvtColor(imgoriginal, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(222),plt.imshow(cv2.cvtColor(gamma_a, cv2.COLOR_BGR2RGB))
plt.title('Gamma=2.5'), plt.xticks([]), plt.yticks([])
plt.subplot(223),plt.imshow(cv2.cvtColor(gamma_b, cv2.COLOR_BGR2RGB))
plt.title('Gamma=1.0'), plt.xticks([]), plt.yticks([])
plt.subplot(224),plt.imshow(cv2.cvtColor(gamma_c, cv2.COLOR_BGR2RGB))
plt.title('Gamma=0.2'), plt.xticks([]), plt.yticks([])
```

10

```
24 plt.show()
```

Listing 10: Código fonte[6]

Figure 10: Gamma transformation



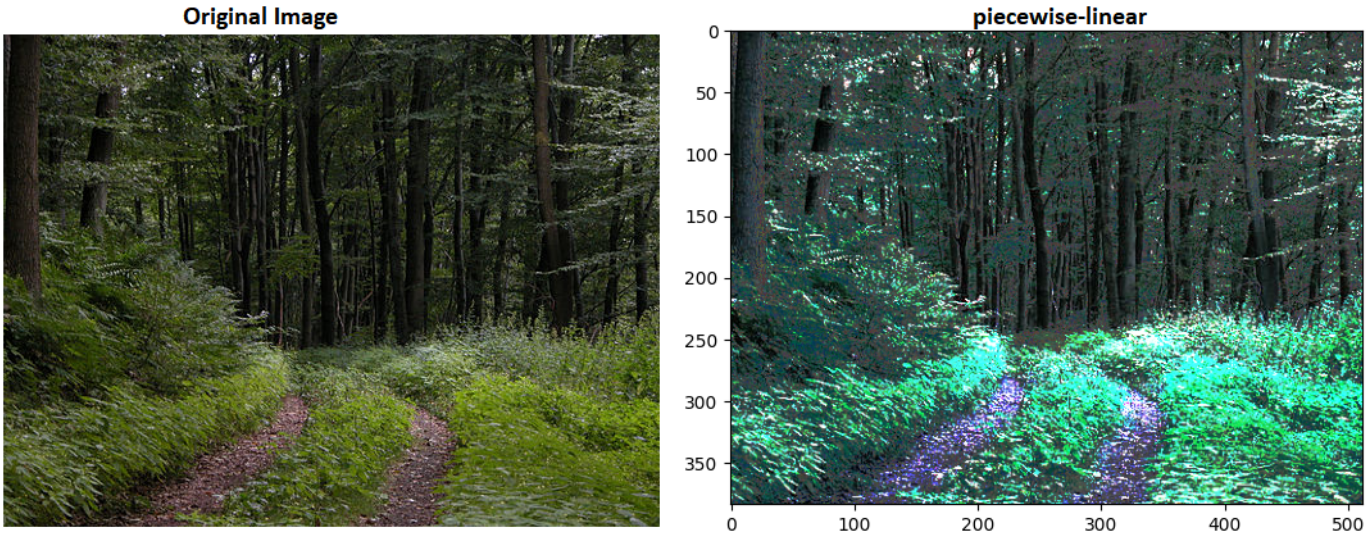
1.11 Piecewise-linear

```
1 import matplotlib.pyplot as plt
2 import time
3 import cv2
4 import numpy as np
5 import matplotlib.pyplot as plt
6 from cv2_plt_imshow import cv2_plt_imshow, plt_format
7 from google.colab import drive
```

```
#drive.authenticate_user()
drive.mount('drive')
img = cv2.imread('drive/My Drive/IMAGENS/IMAGEM10.png')
img = img/255
img = 1.0 * img
img[img<(50/255)] = 2 * img[img<(50/255)]
img[img>(100/255)] = 2 * img[img>(100/255)]
plt.imshow(img)
plt.show()
```

Listing 11: Código fonte[7]

Figure 11: Piecewise-linear



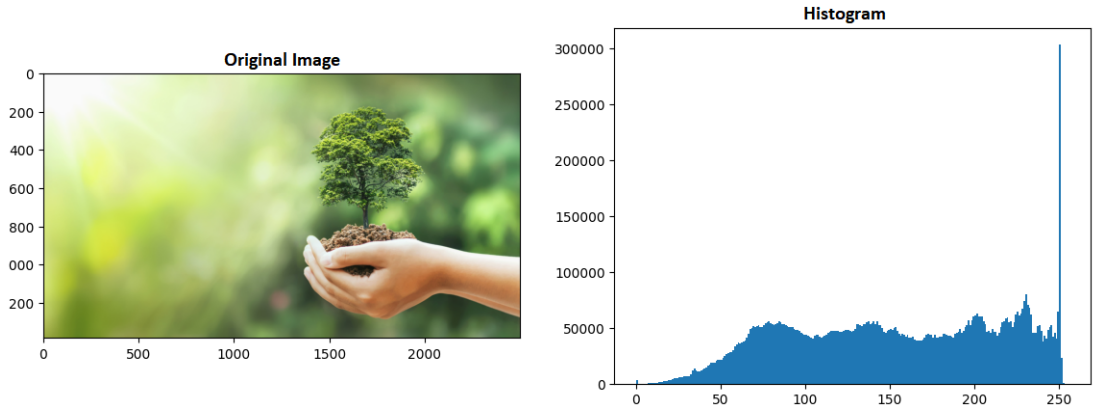
1.12 Histogram basic grey-level characteristics

```
import matplotlib.pyplot as plt
import time
import cv2
from matplotlib import pyplot as plt
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
img = cv2.imread('drive/My Drive/IMAGENS/arvore.jpg')
plt.hist(img.ravel(),256,[0,256]);
plt.show()
```

```
cv2_plt_imshow(img)
```

Listing 12: Código fonte[7]

Figure 12: Histogram basic grey-level characteristics

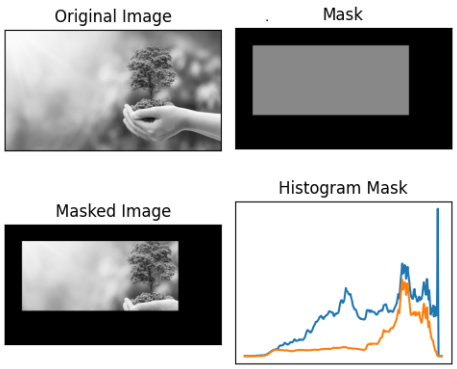


1.13 Application of Mask

```
import matplotlib.pyplot as plt
import time
import cv2
import numpy as np
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
img = cv2.imread('drive/My Drive/IMAGENS/arvore.jpg',cv2.IMREAD_GRAYSCALE)
mask = np.zeros(img.shape[:2], np.uint8)
mask[200:1000, 200:2000] = 5000
masked_img = cv2.bitwise_and(img,img,mask = mask)
hist_full = cv2.calcHist([img],[0],None,[256],[0,256])
hist_mask = cv2.calcHist([img],[0],mask,[256],[0,256])
plt.subplot(221),plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(222),plt.imshow(cv2.cvtColor(mask, cv2.COLOR_BGR2RGB))
plt.title('Mask'), plt.xticks([]), plt.yticks([])
plt.subplot(223),plt.imshow(cv2.cvtColor(masked_img, cv2.COLOR_BGR2RGB))
plt.title('Masked Image'), plt.xticks([]), plt.yticks([])
plt.subplot(224),plt.plot(hist_full), plt.plot(hist_mask)
plt.title('Histogram Mask'), plt.xticks([]), plt.yticks([])
plt.show()
```

Listing 13: Código fonte[8]

Figure 13: Application of Mask



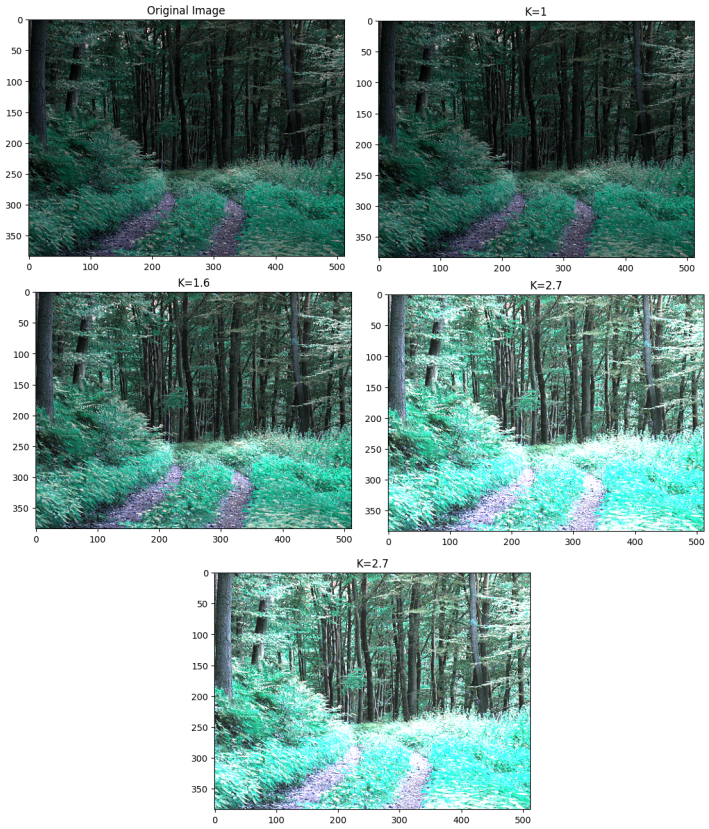
1.14 Lienar Filtering

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import matplotlib.pyplot as plt
import time
from matplotlib import pyplot as plt
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
img = cv2.imread('drive/My Drive/IMAGENS/IMAGEM10.png')
imgOriginal = cv2.imread('drive/My Drive/IMAGENS/IMAGEM10.png')
gray = cv2.cvtColor(img, cv2.COLOR_BGR2GRAY)
def point_operation(gray, K, L):
  gray = np.asarray(gray, dtype=np.float64)
  gray = img*K + L
  gray[gray > 255] = 255
  gray[gray < 0] = 0
  return np.asarray(gray, dtype = np.int64)
out1 = point_operation(gray, 1.0, 0)
out2 = point_operation(gray, 1.6, 10)
out3 = point_operation(gray, 2.7, 25)
out4 = point_operation(gray,4.0, 25)
#res = np.hstack([imgOriginal,out1, out2, out3,out4])
plt.imshow(imgOriginal)
plt.title('Original Image')
plt.show()
plt.imshow(out1)
plt.title('K=1')
plt.show()
plt.imshow(out2)
plt.title('K=1.6')
```

```
plt.show()
plt.imshow(out3)
plt.title('K=2.7')
plt.show()
plt.imshow(out4)
plt.title('K=4.0')
plt.show()
```

Listing 14: Código fonte[9]

Figure 14: Histogram equalization



1.15 Mean Filtering

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import matplotlib.pyplot as plt
import time
from matplotlib import pyplot as plt
```

```
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
img = cv2.imread('drive/My Drive/IMAGENS/brain.png')
median = cv2.medianBlur(img, 5)
#compare = np.concatenate((img, median), axis=1)
#cv2_plt_imshow(compare)
#cv2.waitKey(0)
plt.subplot(221),plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])

plt.subplot(222),plt.imshow(cv2.cvtColor(median, cv2.COLOR_BGR2RGB))
plt.title('Mean filtering'), plt.xticks([]), plt.yticks([])
plt.show()
```

Listing 15: Código fonte[9]

Figure 15: Mean Filtering

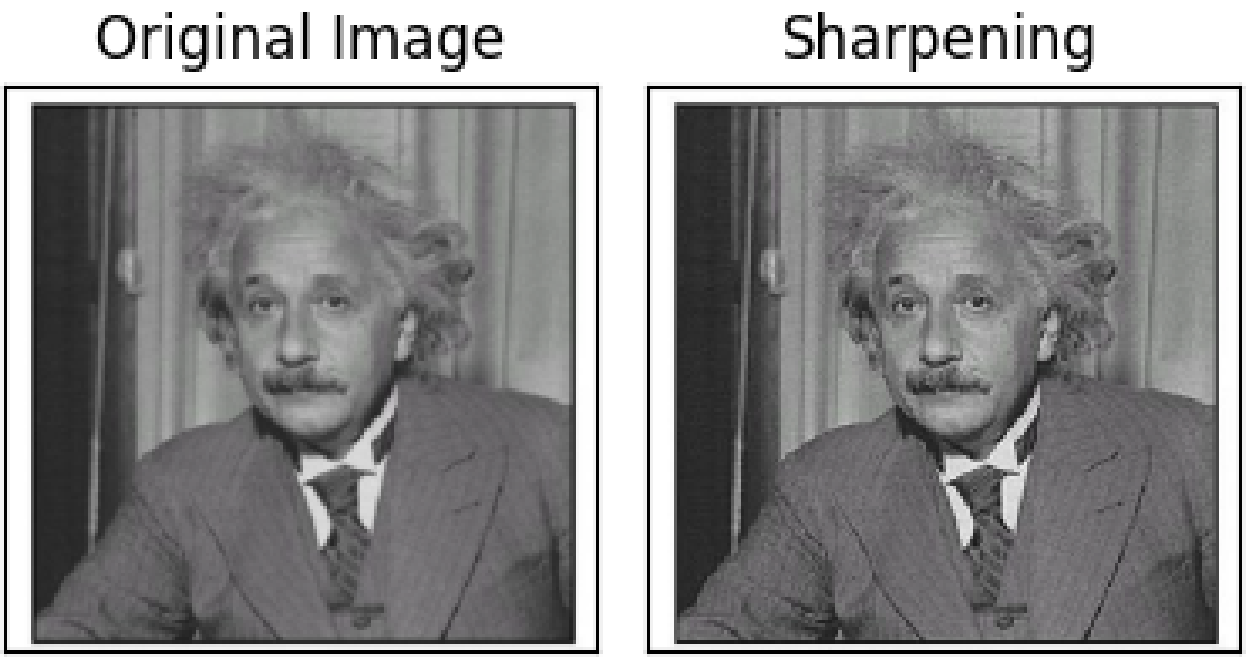
1.16 Sharpening

```
import numpy as np
import matplotlib.pyplot as plt
import cv2
import matplotlib.pyplot as plt
import time
```

```
from matplotlib import pyplot as plt
import matplotlib.pyplot as plt
from cv2_plt_imshow import cv2_plt_imshow, plt_format
from google.colab import drive
#drive.authenticate_user()
drive.mount('drive')
img = cv2.imread('drive/My Drive/IMAGENS/imagemHeisten.png')
kernel = np.array([[0, -1, 0],
                   [-1, 5,-1],
                   [0, -1, 0]])
image_sharp = cv2.filter2D(src=img, ddepth=-1, kernel=kernel)
compare = np.concatenate((image_sharp, img), axis=1)
#cv2_plt_imshow(compare)
#cv2.waitKey()
plt.subplot(221),plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(222),plt.imshow(cv2.cvtColor(image_sharp, cv2.COLOR_BGR2RGB))
plt.title('Sharpening'), plt.xticks([]), plt.yticks([])
plt.show()
```

Listing 16: Código fonte[10]

Figure 16: Sharpening



1.17 Gaussian Filters

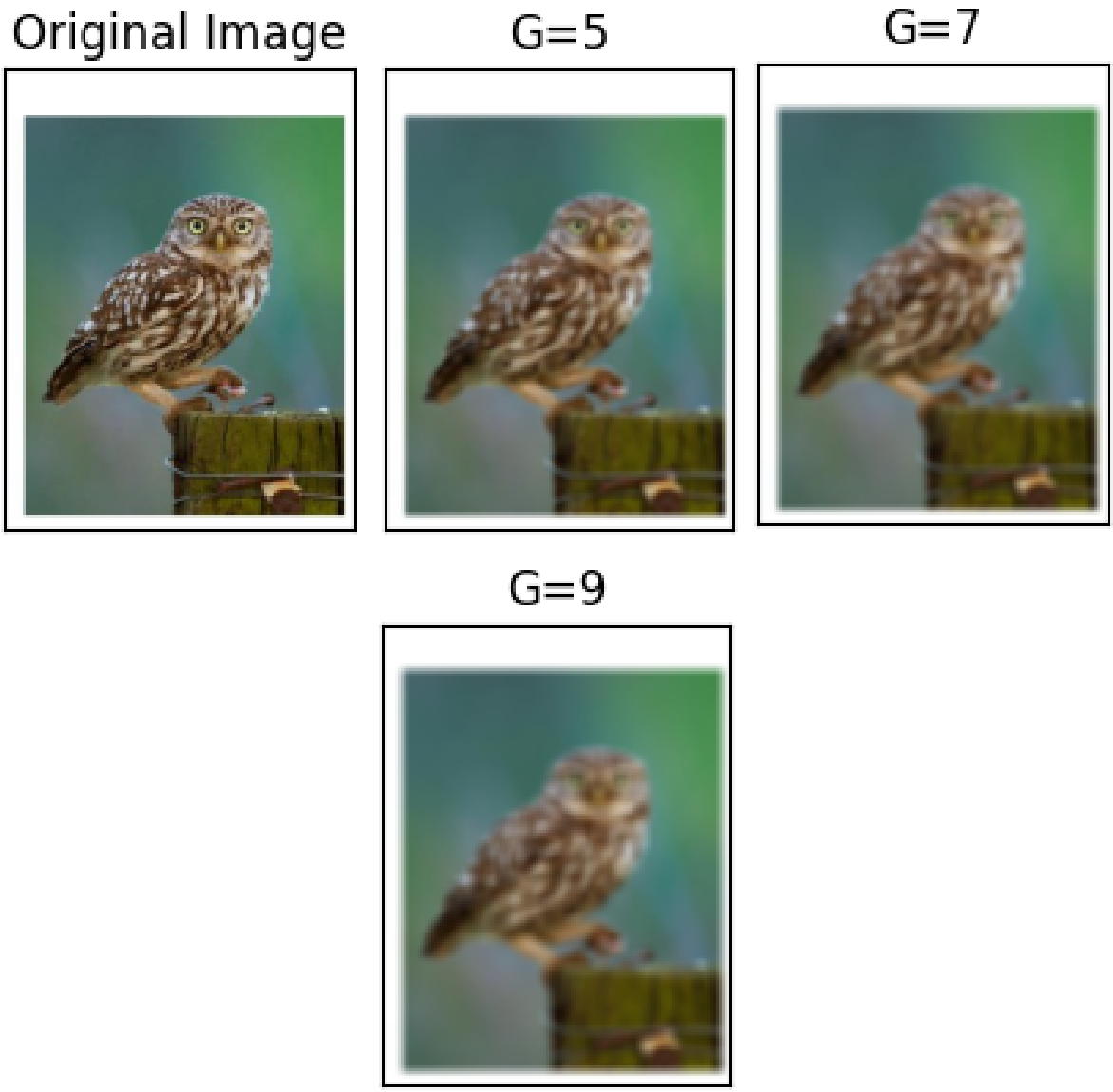
```
1
2  import numpy as np
3  import matplotlib.pyplot as plt
4  import cv2
5  import matplotlib.pyplot as plt
6  import time
7  from matplotlib import pyplot as plt
8  import matplotlib.pyplot as plt
9  from cv2_plt_imshow import cv2_plt_imshow, plt_format
10 from google.colab import drive
11 #drive.authenticate_user()
12 drive.mount('drive')
13 img = cv2.imread('drive/My Drive/IMAGENS/coruja.png')
14 dst1 = cv2.GaussianBlur(img,(5,5),cv2.BORDER_DEFAULT)
15 dst2 = cv2.GaussianBlur(img,(7,7),cv2.BORDER_DEFAULT)
16 dst3 = cv2.GaussianBlur(img,(9,9),cv2.BORDER_DEFAULT)
17 #compare = np.concatenate((img, dst1,dst2), axis=1)
18 #cv2_plt_imshow(compare)
19 #cv2.waitKey()
20 plt.subplot(221),plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
21 plt.title('Original Image'), plt.xticks([]), plt.yticks([])
22 plt.subplot(222),plt.imshow(cv2.cvtColor(dst1, cv2.COLOR_BGR2RGB))
23 plt.title('G=5'), plt.xticks([]), plt.yticks([])
24 plt.subplot(223),plt.imshow(cv2.cvtColor(dst2, cv2.COLOR_BGR2RGB))
25 plt.title('G=7'), plt.xticks([]), plt.yticks([])
26 plt.subplot(224),plt.imshow(cv2.cvtColor(dst3, cv2.COLOR_BGR2RGB))
27 plt.title('G=9'), plt.xticks([]), plt.yticks([])
28 plt.show()
```

Listing 17: Código fonte[10]. A imagem gerada pelo código pode ser vista na figura 17

1.18 Convolution

```
1  import numpy as np
2  import matplotlib.pyplot as plt
3  import cv2
4  import matplotlib.pyplot as plt
5  import time
6  from matplotlib import pyplot as plt
7  import matplotlib.pyplot as plt
8  from cv2_plt_imshow import cv2_plt_imshow, plt_format
9  from google.colab import drive
10 #drive.authenticate_user()
11 drive.mount('drive')
12 img = cv2.imread('drive/My Drive/IMAGENS/IMAGEM14.png')
13 imagem = cv2.cvtColor(img, cv2.COLOR_BGR2RGB)
14 kernel = np.array([[-2, -1, 0],
15                    [-1, 1, 1],
16                    [0, 1, 2]])
17 img = cv2.filter2D(imagem, -1, kernel)
18 #fig, ax = plt.subplots(1,2,figsize=(10,6))
```

Figure 17: Gaussian Filters

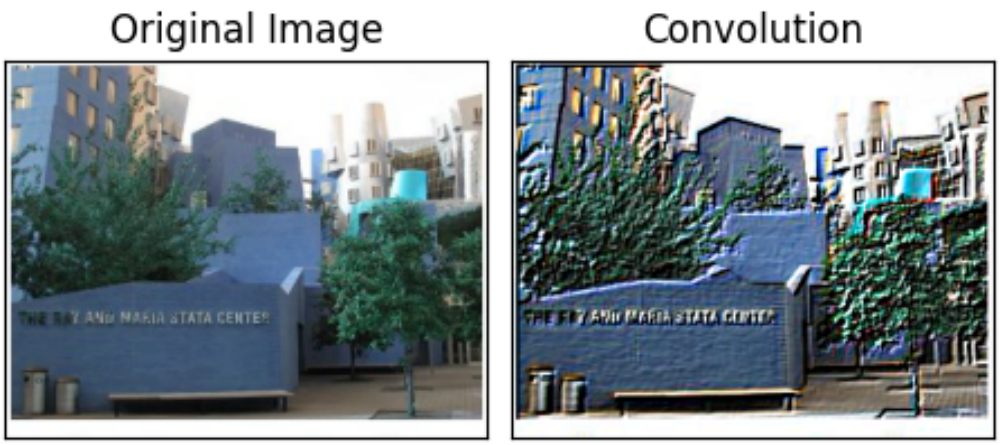



```
#ax[0].imshow(imagem)
#ax[1].imshow(img)
plt.subplot(223),plt.imshow(cv2.cvtColor(imagem, cv2.COLOR_BGR2RGB))
plt.title('Original Image'), plt.xticks([]), plt.yticks([])
plt.subplot(224),plt.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
plt.title('Convolution'), plt.xticks([]), plt.yticks([])
```

```
25 plt.show()
```

Listing 18: Código fonte[10]. A imagem gerada pelo código pode ser vista na figura 18

Figure 18: Convolution



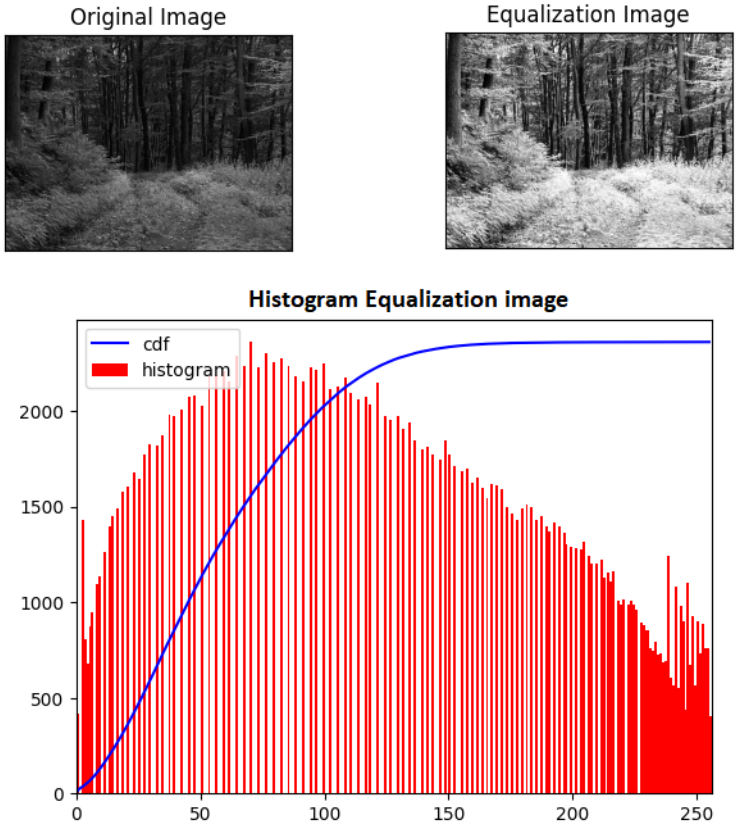
1.19 Histogram equalization

```
1  import matplotlib.pyplot as plt
2  import time
3  import cv2
4  import numpy as np
5  from matplotlib import pyplot as plt
6  import matplotlib.pyplot as plt
7  from cv2_plt_imshow import cv2_plt_imshow, plt_format
8  from google.colab import drive
9  #drive.authenticate_user()
10 drive.mount('drive')
11 img = cv2.imread('drive/My Drive/IMAGENS/IMAGEM10.png', cv2.
       IMREAD_GRAYSCALE)
12 imgOrginal = cv2.imread('drive/My Drive/IMAGENS/IMAGEM10.png',cv2.
       IMREAD_GRAYSCALE)
13 equ = cv2.equalizeHist(img)
14 res = np.hstack((img,equ))
15 cv2_plt_imshow(res)
16 plt.subplot(221),plt.imshow(cv2.cvtColor(imgOrginal, cv2.COLOR_BGR2RGB))
17 plt.title('Original Image'), plt.xticks([]), plt.yticks([])
18 plt.subplot(222),plt.imshow(cv2.cvtColor(equ, cv2.COLOR_BGR2RGB))
19 plt.title('Equalization Image'), plt.xticks([]), plt.yticks([])
20 plt.show()
21 cdf = hist.cumsum()
22 cdf_normalized = cdf * float(hist.max()) / cdf.max()
```

```
23 plt.plot(cdf_normalized, color = 'b')
24 plt.hist(equ.flatten(),256,[0,256], color = 'r')
25 plt.xlim([0,256])
26 plt.legend(('cdf','histogram'), loc = 'upper left')
27 plt.show()
28 cdf_m = np.ma.masked_equal(cdf,0)
29 cdf_m = (cdf_m - cdf_m.min())*255/(cdf_m.max()-cdf_m.min())
30 cdf = np.ma.filled(cdf_m,0).astype('uint8')
31 img2 = cdf[equ]
32 plt.plot(cdf, color = 'b')
33 plt.hist(equ.flatten(),256,[0,256], color = 'r')
34 plt.xlim([0,256])
35 plt.legend(('cdf','histogram'), loc = 'upper left')
36 #plt.show()
```

Listing 19: Código fonte[11]

Figure 19: Histogram equalization



References

[1] Prasad S. **Python for Image Recognition - OpenCV**. Online. Acessado em 22/04/2023, https://www.topcoder.com/thrive/articles/python-for-image-recognition-opencv.

[2] Woosung K. **opencv python**. Online. Acessado em 20/04/2023, https://wsstudynote.tistory.com/23.

[3] Shahid A. K. **Color quantization in an image using K-means in OpenCV Python**. Online. Acessado em 20/04/2023, https://www.tutorialspoint.com/color-quantization-in-an-image-using-k-means-in-opencv-python.

[4] Prasad N. **Downsampling an image using OpenCV**. Online. Acessado em 20/04/2023, #https://www.tutorialspoint.com/downsampling-an-image-using-opencv.

[5] Panka j. **Log transformation of an image using Python and OpenCV**. Online. Acessado em 20/04/2023, https://www.geeksforgeeks.org/log-transformation-of-an-image-using-python-and-opencv/.

[6] Shiv M. **Advanced OpenCV: BGR Pixel Intensity Plots**. Online. Acessado em 20/04/2023, https://www.analyticsvidhya.com/blog/2022/05/advanced-opencv-bgr-pixel-intensity-plots/.

[7] Pankaj P. **Piece-wise Linear Transformation**. Online. Acessado em 20/04/2023, https://www.geeksforgeeks.org/log-transformation-of-an-image-using-python-and-opencv/.

[8] OpenCV. **Histograms**. Online. Acessado em 20/04/2023, https://docs.opencv.org/3.4/d1/db7/tutorial_py_histogram_begins.html.

[9] Vijin B. **Image filtering techniques in OpenCV**. Online. Acessado em 24/04/2023, https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html.

[10] Shiv M. **Sharpening An Image using OpenCV Library in Python**. Online. Acessado em 24/04/2023, https://www.analyticsvidhya.com/blog/2021/08/sharpening-an-image-using-opencv-library-in-python/.

[11] OpenCV. **Histograms Equalization in OpenCV**. Online. Acessado em 24/04/2023, https://docs.opencv.org/4.x/d5/daf/tutorial_py_histogram_equalization.html.