Grupo de Pesquisa em Reconhecimento de Padrões e Otimização

DISCIPLINA: PROCESSAMENTO DIGITAL DE IMAGENS

Project number: 3

Course name: Computer Engineering

Student's name: João Victor Félix and Caroline Braz

Date due: 26/05/2024

Date handed in: 26/05/2024

WWW.PATTERNRECOGNITIONPLACE.UFAM.EDU.BR
PROFA MARLY GUIMARÃES F COSTA
MCOSTA@UFAM.EDU.BR

Technical discussion and results

(One to three pages - max).

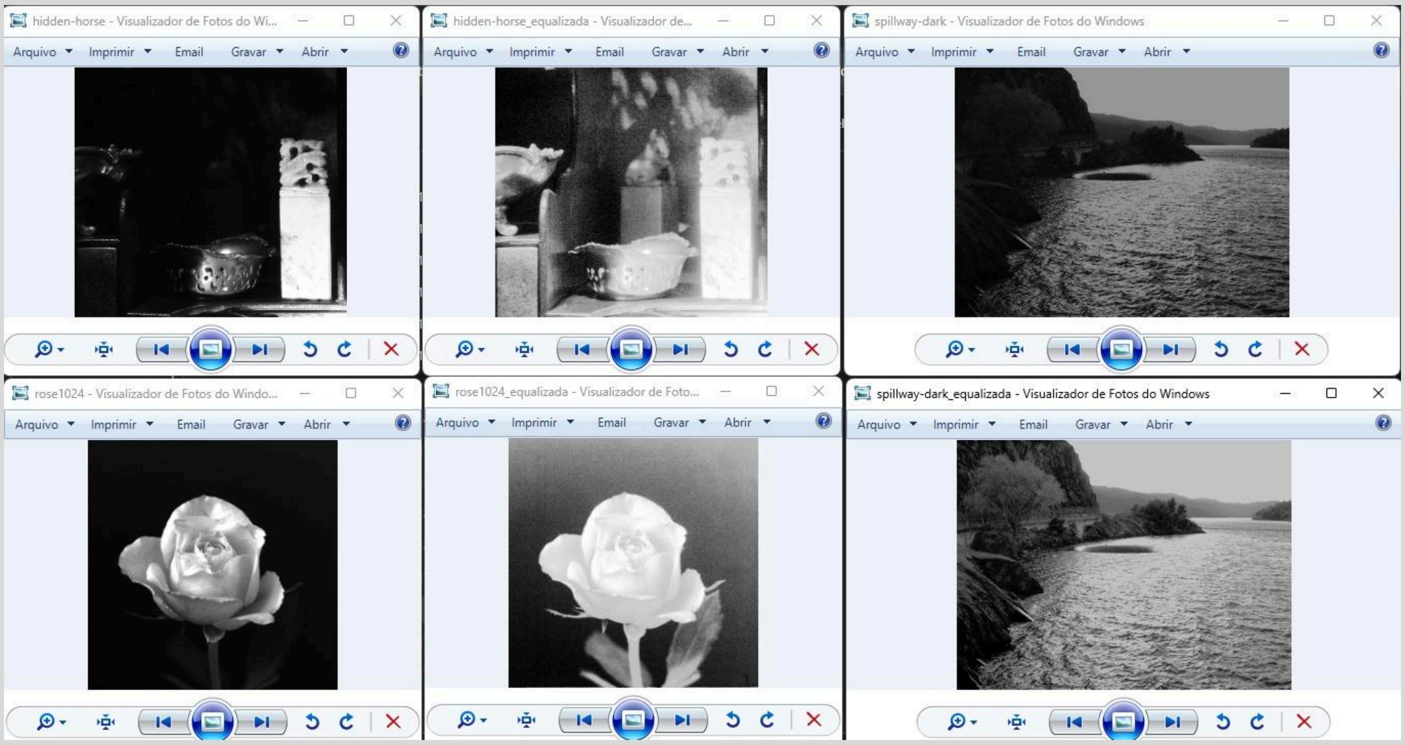
The following report aims to primarily report on the development of a project to explore the concepts learned in the course of Digital Image Processing. The project consists of elaborating an algorithm to compute the histogram of a 256-level grayscale image.

The first function called ***histEqual4e*** receives as parameter an array that represents the histogram of an image. It initializes an empty list to save the accumulated probability and a variable to save the accumulated sum of probabilities. Then, it iterates between the possible values of intensity of pixels (0 and 255) and calculates the accumulated probability. It also adds the probability of the actual pixel to the sum of probabilities and that sum is added to the list of accumulated probabilities. After that, an empty list is initialized and the algorithm iterates over the possible values of intensity of pixels (0 and 255) again to calculate the equalized values, multiplying the accumulated probability to the maximum value of intensity and that value is rounded to the next integer value (if necessary). In the end, the function returns the new equalized values.

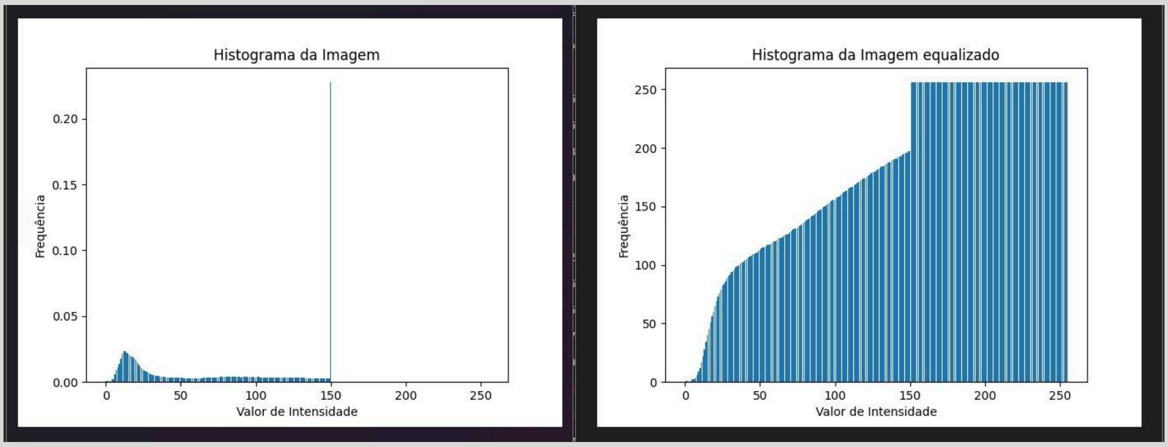
```
import numpy as np

def histEqual4e(f):

   #cálculo da probabilidade acumulada
   probability_accumulated = []
   sum_probability = 0
   for k in range(256):
       if k == 0:
           pass
       else:
           sum_probability += f[k-1]
       probability_accumulated.append(sum_probability)

   #mapeamento dos respectivos valores de cinza em novos valores equalizados.
   new_value_f = []
   for j in range(256):
       new_value = 255 * probability_accumulated[j]
       new_value_f.append(np.ceil(new_value))
   return np.array(new_value_f)
```

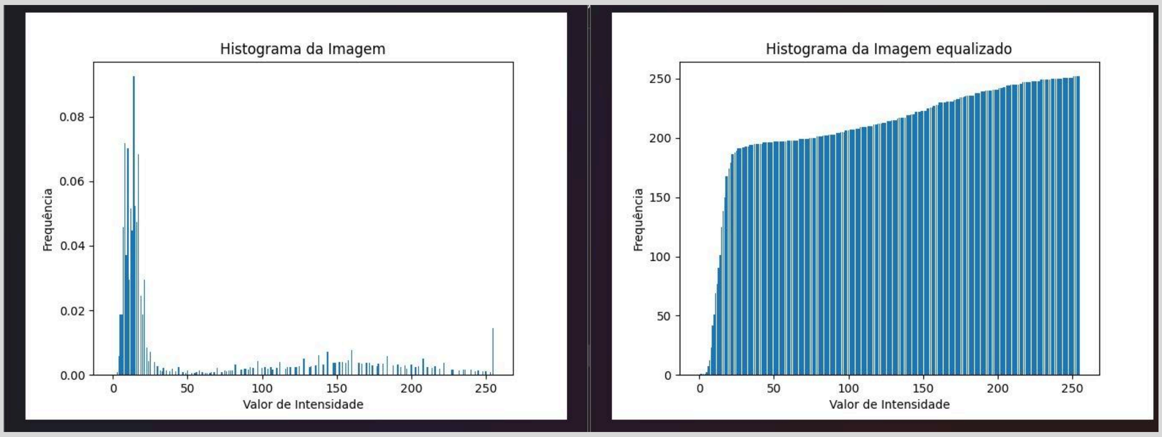
The other functioncalled ***imagehist4e*** receives as parameters the matrix that represents the image and the mode that the histogram has to be calculated. An array with 256 zeros is initialized to represent the histogram where each element is the count of pixels with determined quantity. The function calculates the total number of pixels multiplying the number of rows and columns of the image, then it iterates over the total of pixels. If the mode required is 'normalized', it calculates the relative frequency of each intensity value divided by the total number of pixels of the image. Then it adds the relative frequency to the a list of the normalized histogram. In the end it returns the normalized histogram. If the mode selected was unnormalized it just return the histogram array.

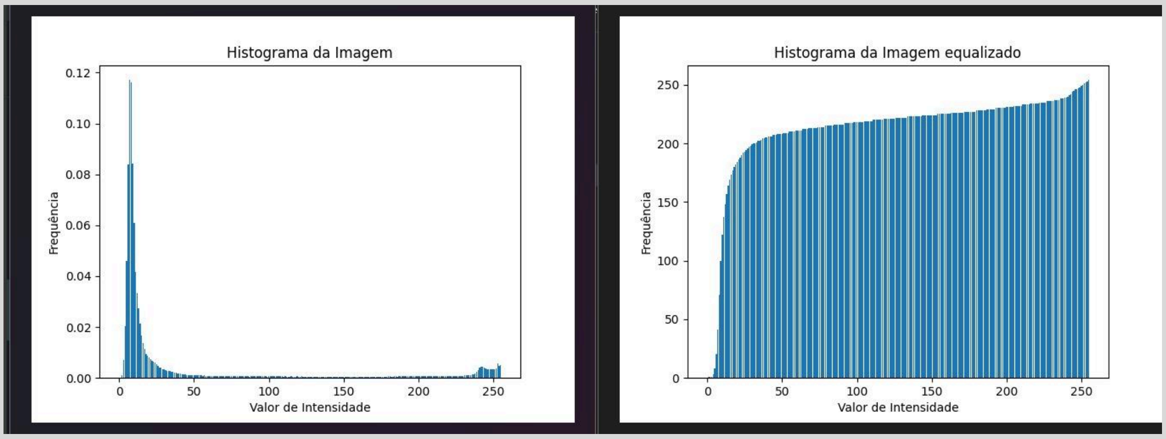
```
def imagehist4e(f, modo):
   #inicializa o histograma
   histograma = np.zeros(256)
   numero_total_pixels = f.shape[0] * f.shape[1]

   #conta quantas vezes cada valor de intensidade aparece na imagem
   for row in f:
       for column in row:
           histograma[column] = histograma[column] + 1

   # normaliza o histograma, se necessário
   # cálculo probabilístico da divisão de quantas vezes o valor apareceu pelo número
total de pixels na imagem
   histograma_normalizado = []
   if modo == 'n':
       for k in range(256):
           valor_normalizado = histograma[k]/numero_total_pixels
           histograma_normalizado.append(valor_normalizado)
       return histograma_normalizado
   if modo == 'u':
       return np.array(histograma)
```


hidden-horse - Visualizador de Fotos do Wi...
hidden-horse_equalizada - Visualizador de...
spillway-dark - Visualizador de Fotos do Windows
rose1024 - Visualizador de Fotos do Windo...
rose1024_equalizada - Visualizador de Foto...
spillway-dark_equalizada - Visualizador de Fotos do Windows
Arquivo
Imprimir
Email
Gravar
Abrir

The function was tested on 3 different images and the results comparing the normal histograms and equalized ones shows that it is possibly to generate images with more balanced values of intensity, where the equalized images have more visible details that are not initially perceptive on the original images.







References

(submit here the references cited).

Python. (2020). *The Python Standard Library — Python 3.8.1 Documentation*. Disponível em: https://docs.python.org/3/library/index.html

Guimarães, M. (Data desconhecida). UNIDADE III - Fundamentos das transformações de intensidade - Equalização de histograma. Disponível em: https://drive.google.com/file/d/1MMfVjH6Iby1qnHaw9f6b5vRQspA3GUjo/view

Data Hackers. (2018). Equalização de Histograma em Python. *Medium*. Disponível em: https://medium.com/data-hackers/equaliza%C3%A7%C3%A3o-de-histograma-em-python-378830368d60

Cagnin, A. A. (2013). Processamento de Imagens Digitais - Histogramas (Notas de Aula). Universidade Estadual do Oeste do Paraná. Disponível em: https://www.inf.unioeste.br/~adair/PID/Notas%20Aula/Processamento%20de%20Imagens%20Digitais%20-%20Histogramas.pdf

Python Imaging Library Handbook. (n.d.). open, rotate, and display an image using the default viewer. Disponível em: https://pillow.readthedocs.io/en/stable/reference/Image.html#open-rotate-and-display-an-image-using-the-default-viewer

Gonzalez, R. C., & Woods, R. E. (2018). *Digital Image Processing* (4ª ed.). Prentice Hall.