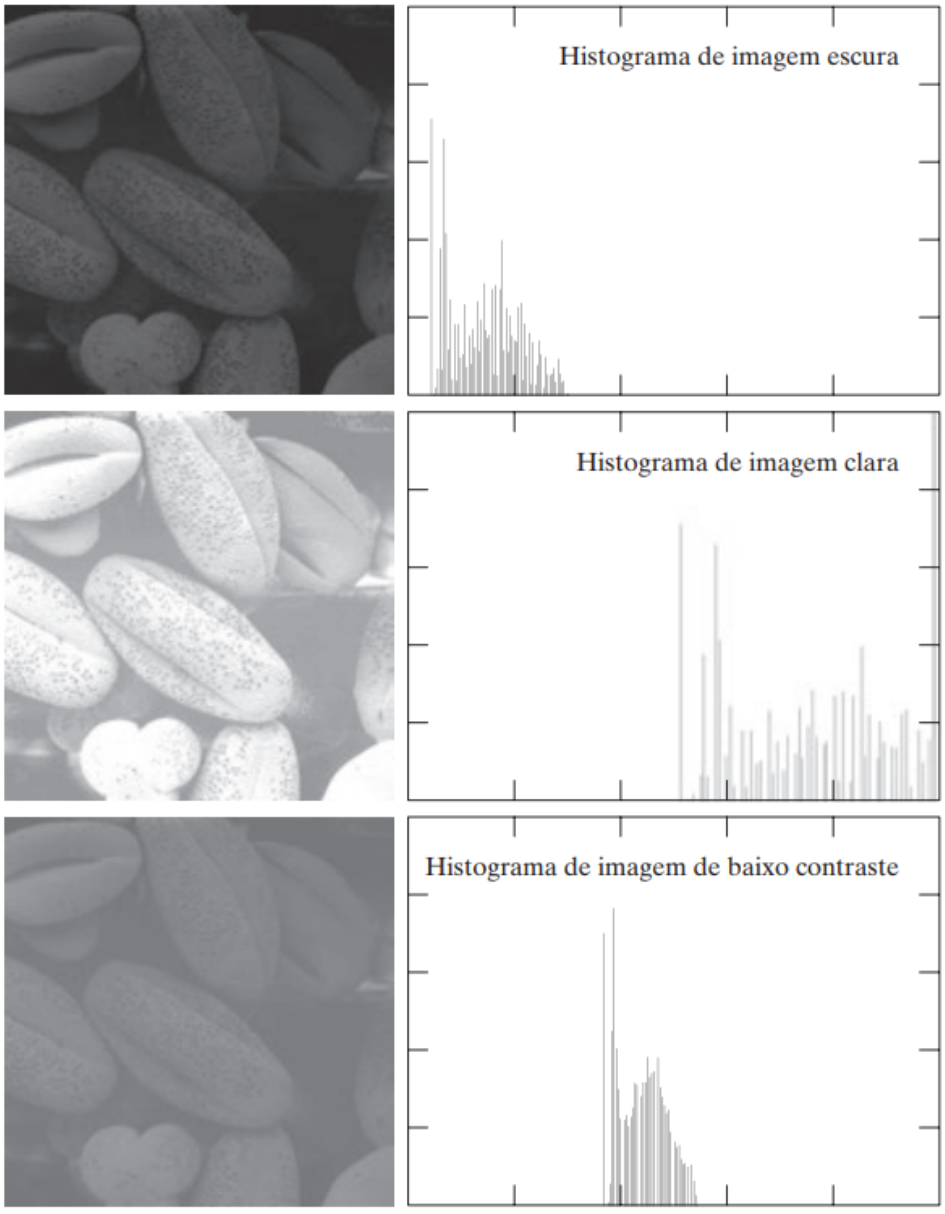
Relatório

Disciplina: Processamento digital de imagens
Discentes:
Diego Takahashi RA: 109889
Pedro H. Landins RA: 103572

Introdução

Para fazer uma normalização, primeiro é preciso entender que é o histograma de uma imagem. Assim, o histograma de uma imagem descreve a distribuição estatística dos níveis de intensidade de uma imagem em termos do número de pixels. Assim, como mostrado abaixo, o lado esquerdo do histograma representa escalas de intensidades “escuras”, o lado direito escalas de intensidades “claras”. Já uma imagem com baixo contraste tem um histograma normalmente localizado no meio das escalas de intensidades.



Histogramas são usados para várias técnicas de realce de imagem, sendo a **normalização** uma delas. Que, em poucas palavras, significa que vamos tentar "esticar" o histograma de uma imagem para que, ao contrário das imagens acima, o histograma cubra uma faixa mais ampla da escala de intensidade. Sendo assim, neste trabalho, faremos a implementação em *Python* da **normalização local**, o que significa que para cada conjunto de pixels m x n aplicaremos a Equação abaixo.

$$T[i] = \left\lfloor \frac{i - \min(f)}{\max(f) - \min(f)} \cdot k \right\rfloor$$

A implementação da função de aumento local de contraste utilizando normalização de contraste e filtros de ordem é a seguinte:

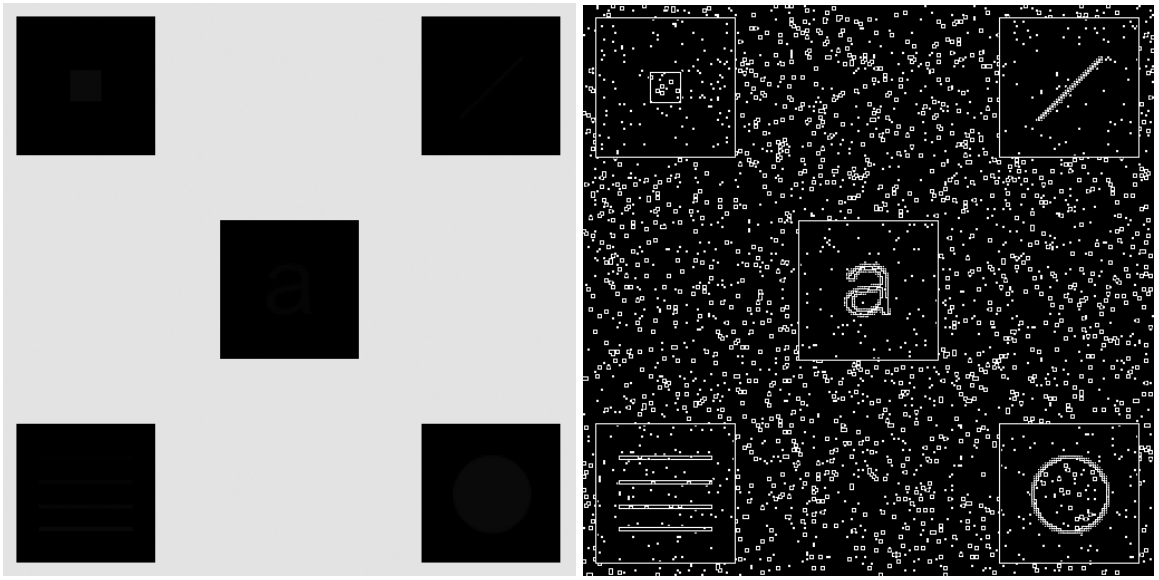
```
import numpy as np
import matplotlib.pyplot as plt
import cv2
from skimage.filters.rank import maximum, minimum
from skimage.morphology import rectangle

def normalizacao(img, m, n):
    min = minimum(img, rectangle(m, n))
    max = maximum(img, rectangle(m, n))
    return
(255*((img-min)/(max-min+np.finfo(float).eps))).astype(np.uint8)


img = cv2.imread('img3.tif', 0)
#Alterar 'm' e 'n' para escolher as dimensões da vizinhança
m = 3
n = 3
img_n = normalizacao(img, m, n)
cv2.imshow('original', img)
cv2.imshow('normalizada', img_n)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

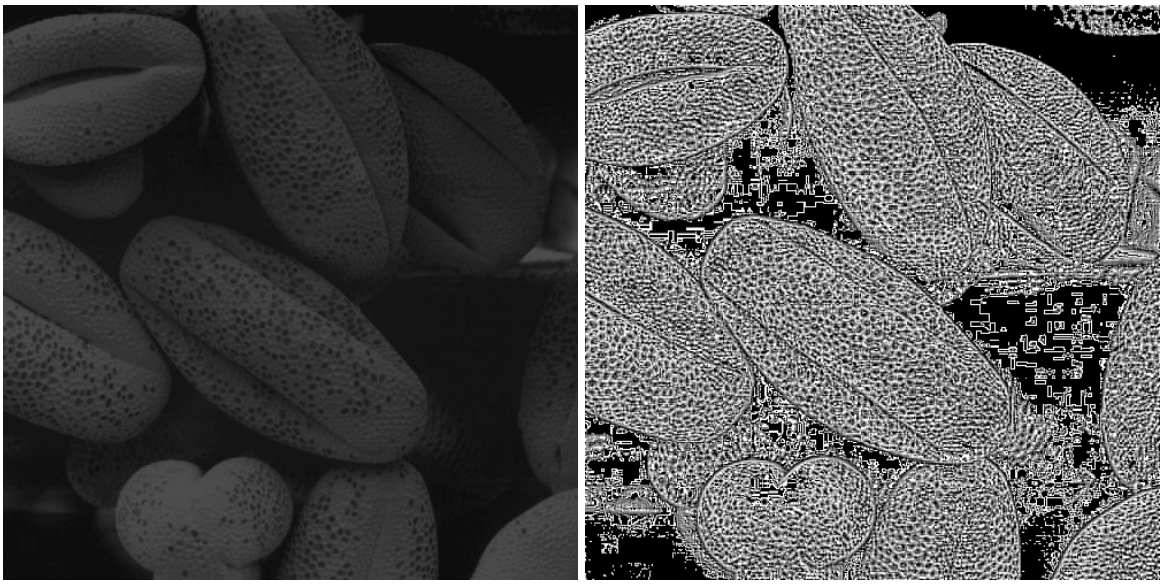
Acima, temos basicamente a implementação da equação apresentada. Em uma vizinhança de dimensões m x n tudo que temos que achar é o menor e o maior valor de pixel. No entanto, como há uma divisão, é preciso tomar precauções para "**max-min**" não resultar em 0 já que esse número está no denominador. E por isso, é necessário a soma de "**np.finfo(float).eps**", tendo em vista que essa função

retorna um número muito próximo de zero, mas não de fato zero. Além disso, para garantir que os valores são inteiros de 8 bits, necessários para representar até 255, a conversão **“astype(np.uint8)**” também é feita.

Resultados

Abaixo, estão as imagens originais à esquerda, e o resultado da normalização local utilizando uma vizinhança de dimensões 3 x 3 à direita.





As imagens abaixo apresentam as originais à esquerda e o resultado da normalização local utilizando uma vizinhança de dimensões 36 x 36 à direita.