INSTITUTO FEDERAL DE EDUCAÇÃO, CIÊNCIA E TECNOLOGIA DE SÃO PAULO, CÂMPUS BIRIGUI - SP
BACHARELADO EM ENGENHARIA DA COMPUTAÇÃO

ISADORA DISPOSTI BUENO DOS SANTOS

EXERCÍCIOS - TRANSFORMAÇÕES E HISTOGRAMA

BIRIGUI - SP
2023

- Utilizar as imagens Fig 3.8 e enhance-me.gif disponíveis no Moodle;
- Aplicar a transformação logarítmica, testar vários valores para o parâmetro c "s = c log (1 + r)"
- Aplicar a transformação de potência (gama), testar vários valores para o parâmetro γ e c=1 "s = cr^{γ}"
- Implemente a representação de cada plano de bits das imagens
- Implementar a equalização do histograma
- Elaborar relatório explicando a implementação de cada transformação e qual foi o efeito na imagem.

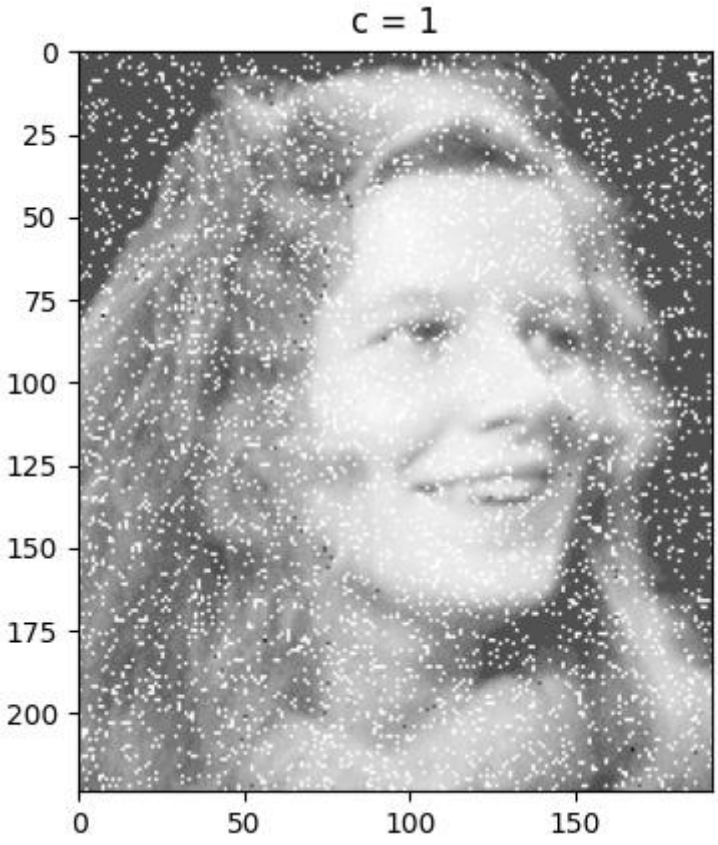
Transformação Logarítmica:

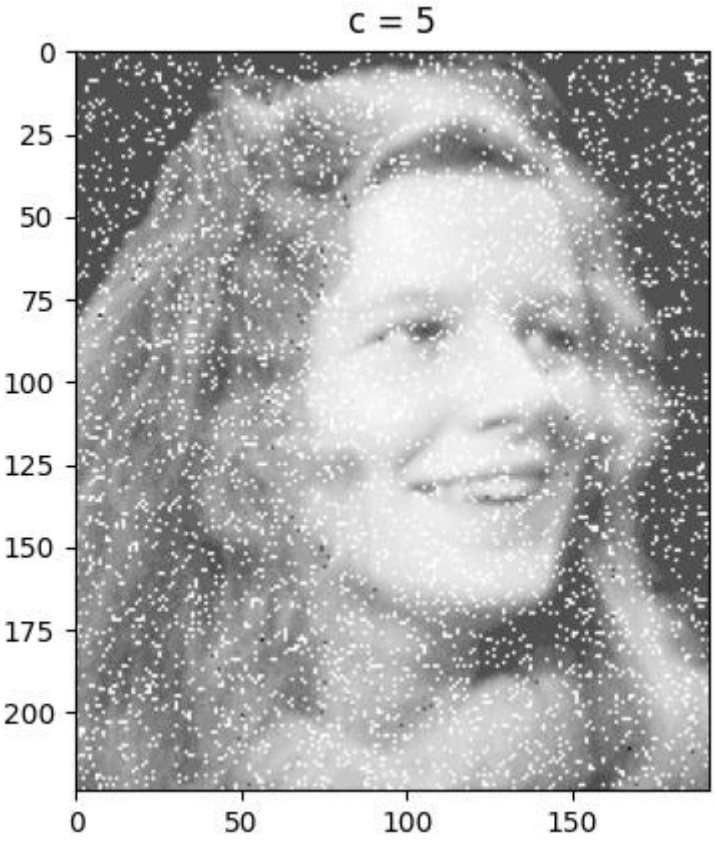
Código:

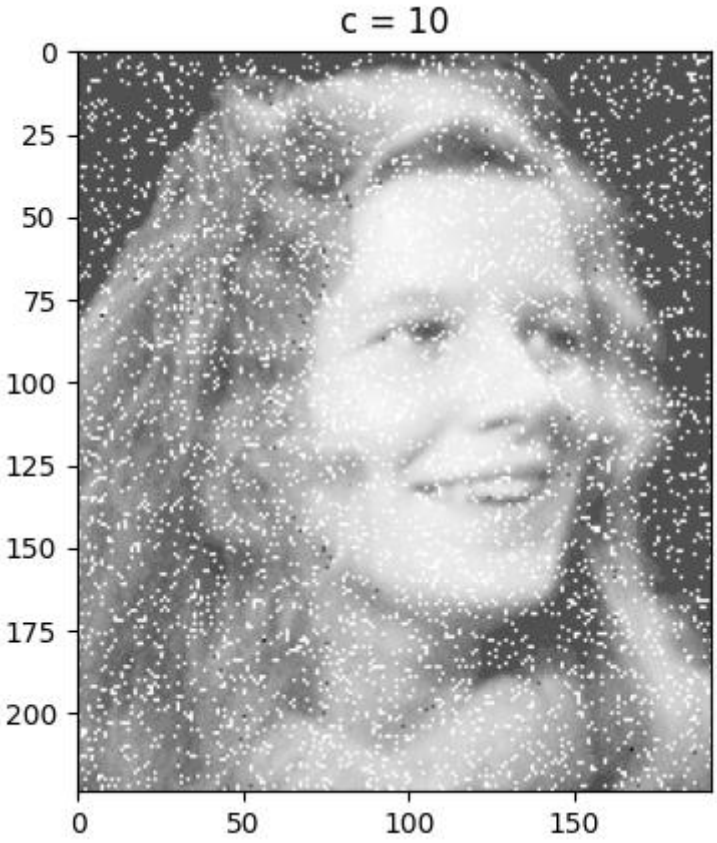
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

#Mandando a imagem para a variavel
enhance_path = 'image/enhance-me.gif'
fractured_path = 'image/fractured_spine.tif'
img = cv2.imread(enhance_path,
cv2.IMREAD_GRAYSCALE)
img2 =
cv2.imread(fractured_path,cv2.IMREAD_GRAYSCALE)

#Abrindo imagem
image_pil = Image.open(enhance_path)
image_pil2 = Image.open(fractured_path)

# Convertendo a imagem PIL para um array numpy
img = np.array(image_pil)
img2 = np.array(image_pil2)

def log_transform(c, img):
 return c * np.log(1 + img)
c_values = [1, 5, 10, 20]
```

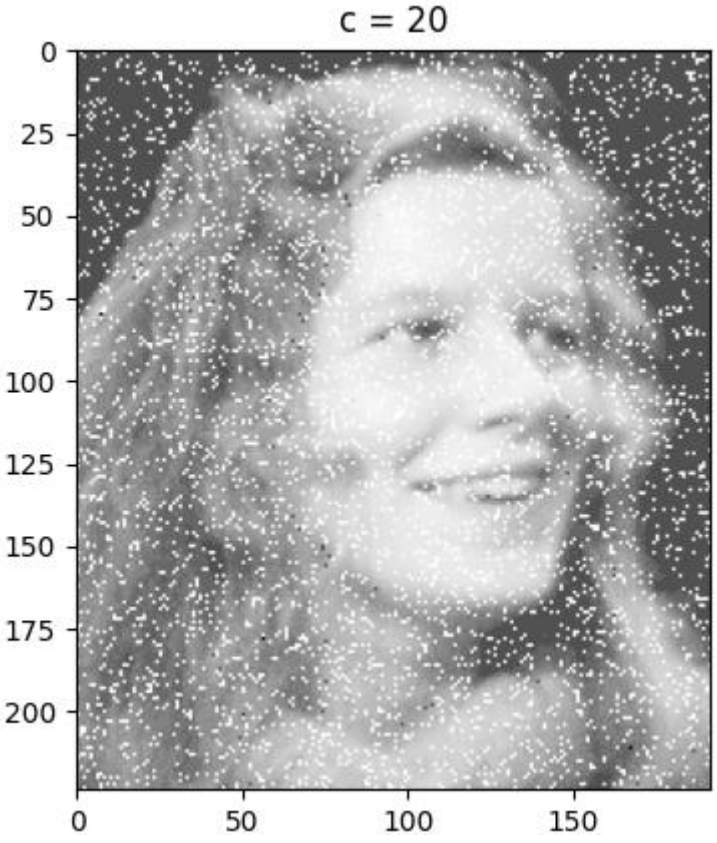
```
def log_transformF(c, img2):
 return c * np.log(1 + img2)
c_valuesF = [1, 5, 10, 20]

for c in c_values:
 transformed_img = log_transform(c, img)
 plt.imshow(transformed_img, cmap='gray')
 plt.title(f'c = {c}')
 plt.show()

for c in c_valuesF:
 transformed_imgF = log_transform(c, img2)
 plt.imshow(transformed_imgF, cmap='gray')
 plt.title(f'c = {c}')
 plt.show()
```

Resultado:




c = 5
0
25
50
75
100
125
150
175
200
0
50
100
150


c = 10
0
25
50
75
100
125
150
175
200
0
50
100
150

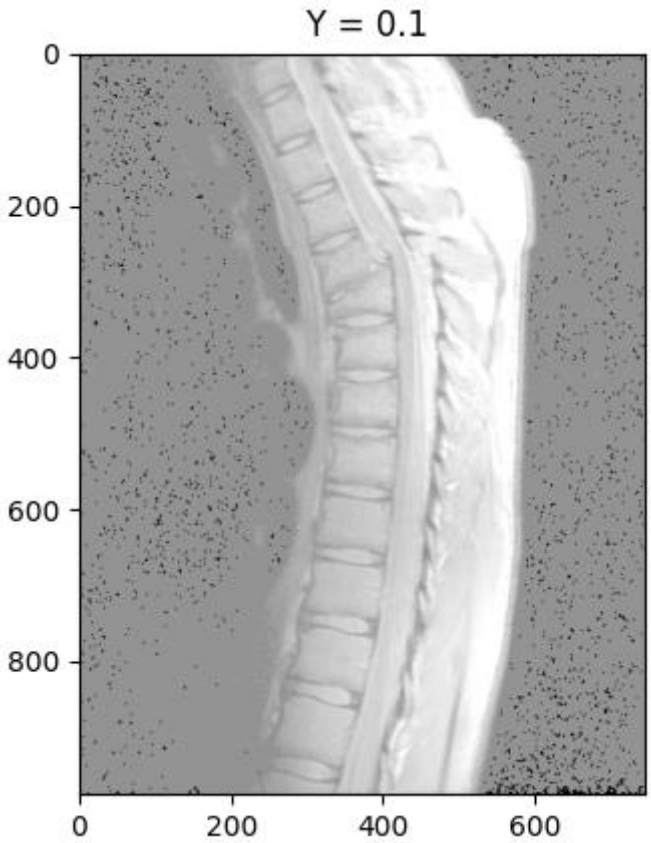

c = 20
0
25
50
75
100
125
150
175
200
0
50
100
150


c = 1
0
200
400
600
800
0
200
400
600


c = 5
0
200
400
600
800
0
200
400
600

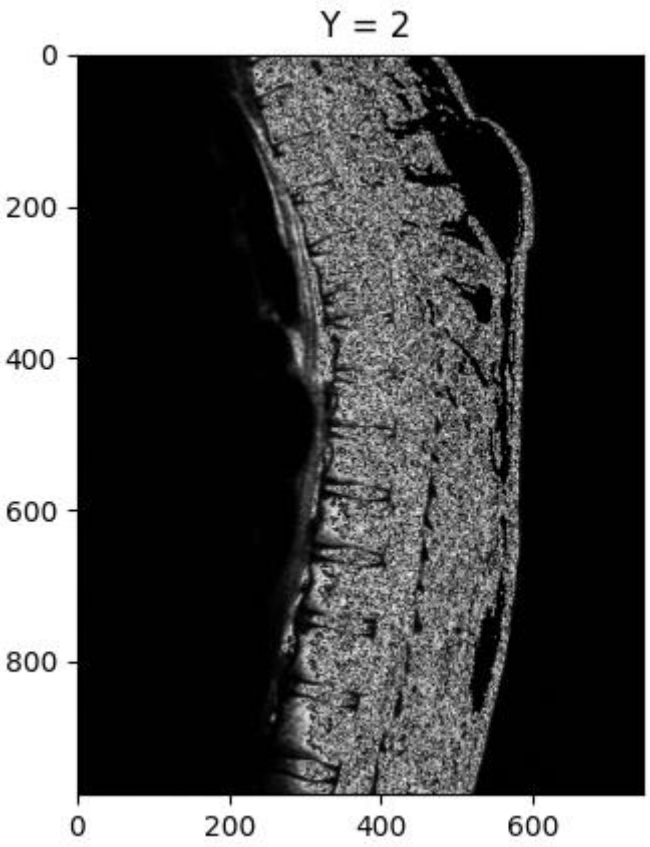

c = 10
0
200
400
600
800
0
200
400
600

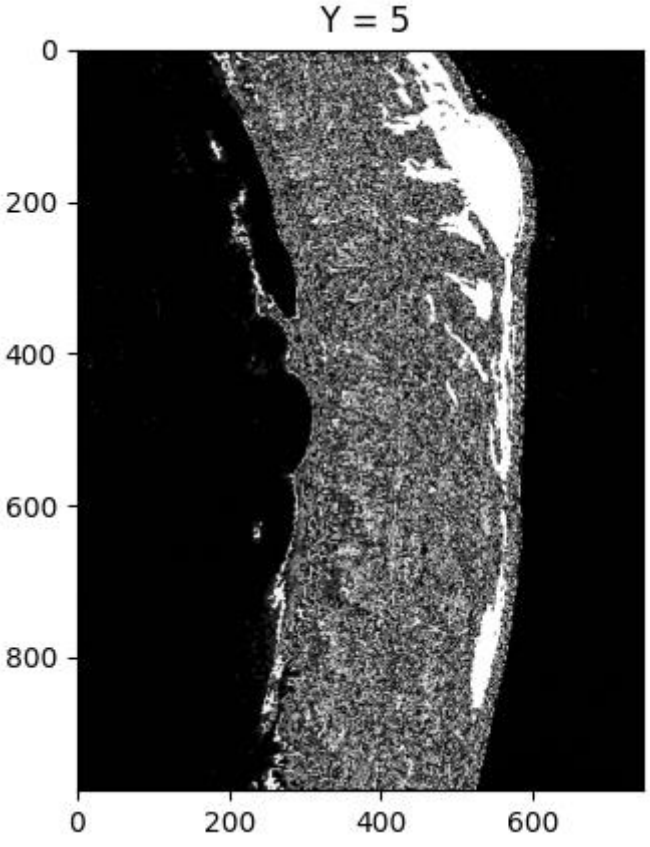
Transformação em Gama:

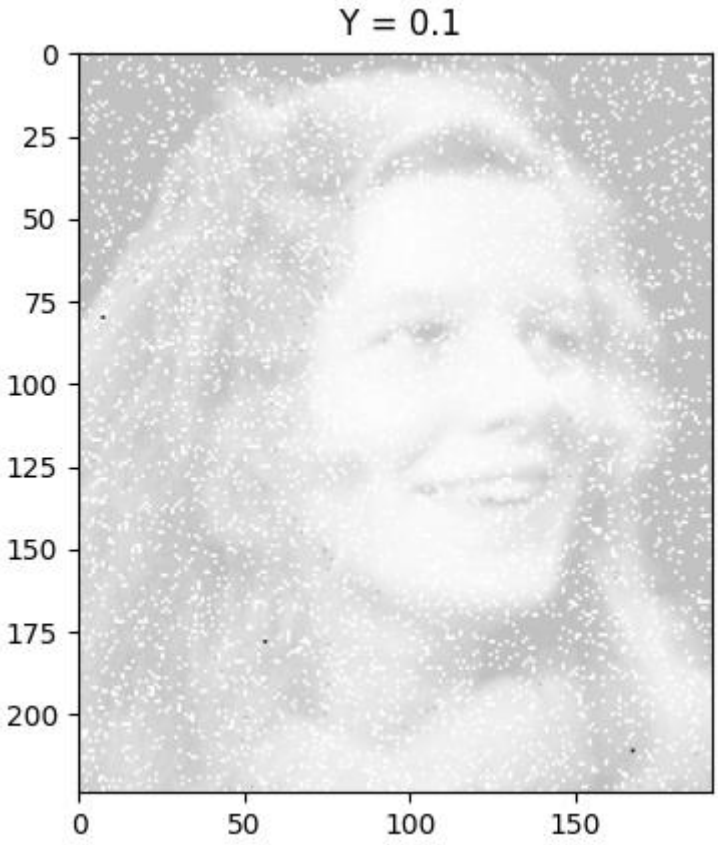


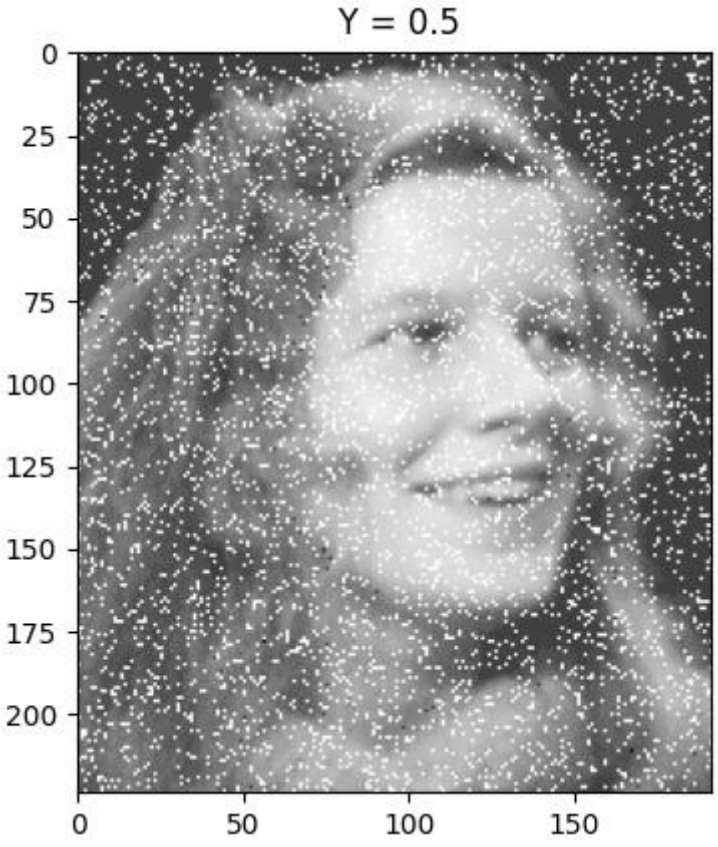

Y = 0.5
0
200
400
600
800
0
200
400
600

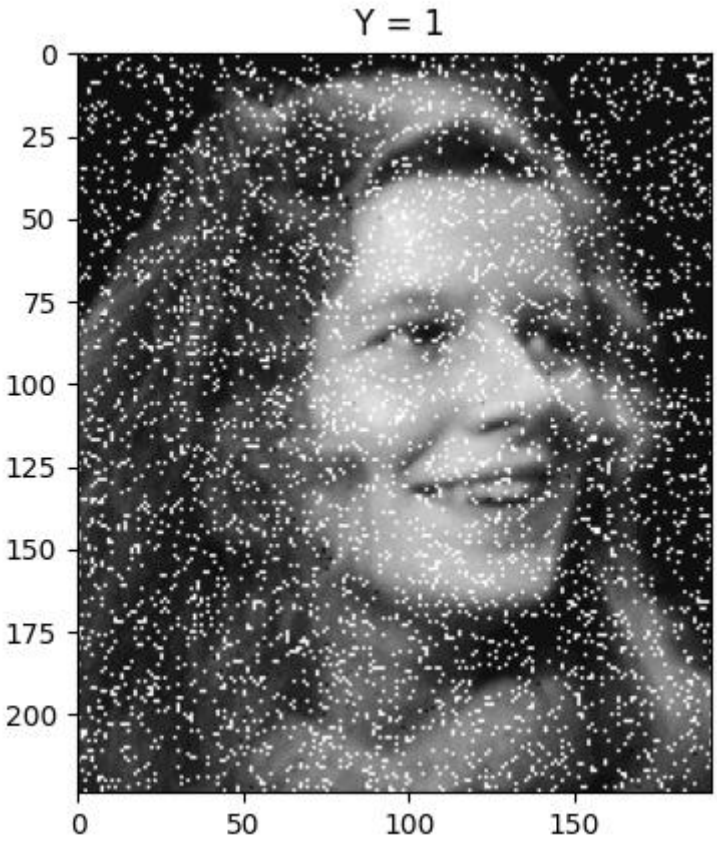

Y = 1
0
200
400
600
800
0
200
400
600

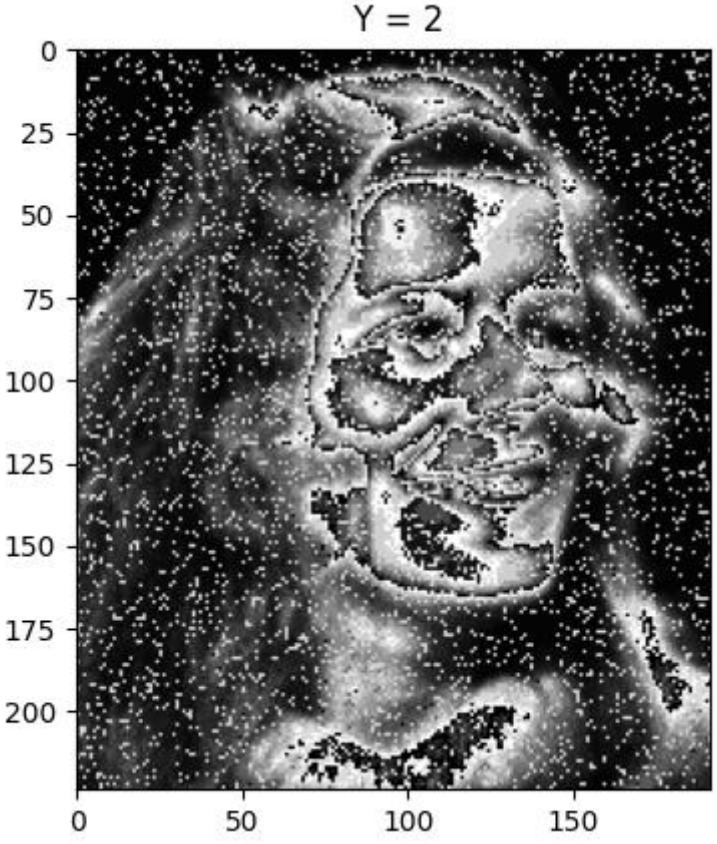

Y = 2
0
200
400
600
800
0
200
400
600

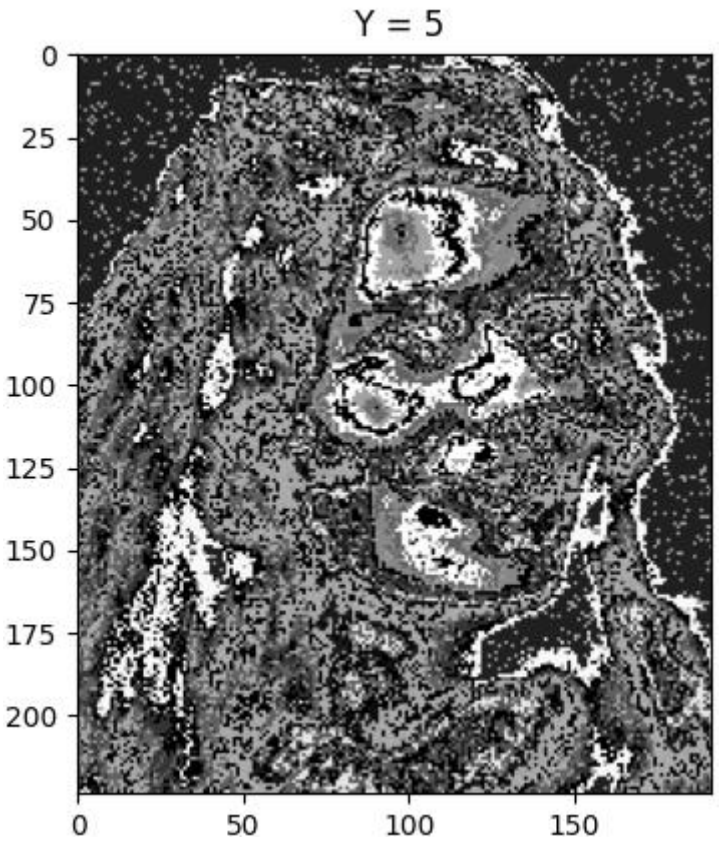

Y = 5
0
200
400
600
800
0
200
400
600


Y = 0.1
0
25
50
75
100
125
150
175
200
0
50
100
150


Y = 0.5
0
25
50
75
100
125
150
175
200
0
50
100
150


Y = 1
0
25
50
75
100
125
150
175
200
0
50
100
150


Y = 2
0
25
50
75
100
125
150
175
200
0
50
100
150

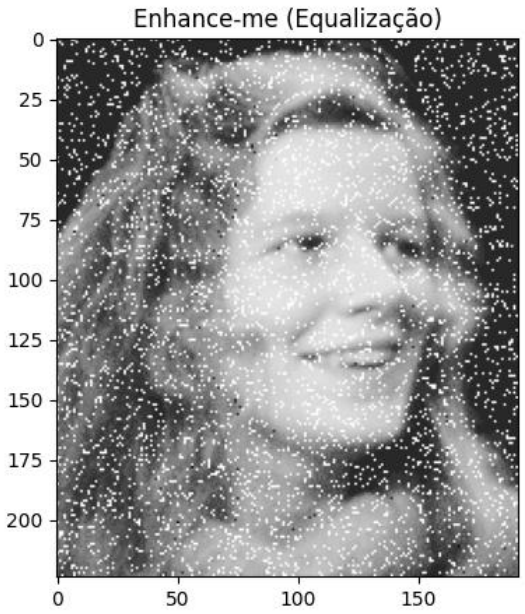


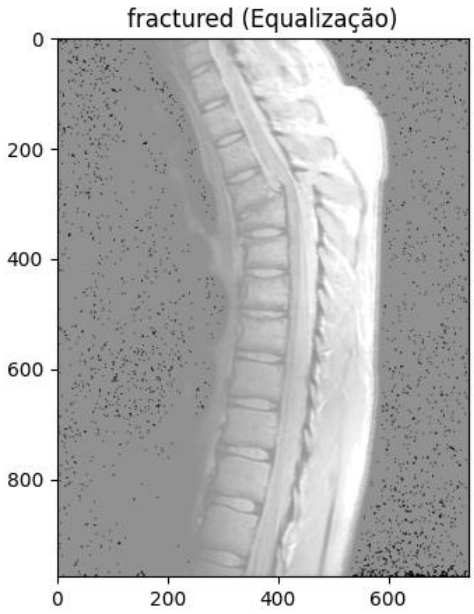
Código:

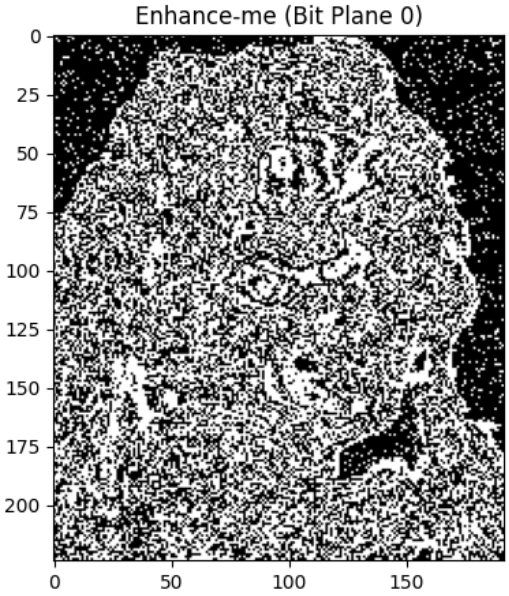
```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

enhance_path ='image/enhance-me.gif'
#enhance_path ='image/fractured_spine.tif'

img = cv2.imread(enhance_path,
cv2.IMREAD_GRAYSCALE)

# Usando PIL para abrir a imagem
image_pil = Image.open(enhance_path)

# Convertendo a imagem PIL para um array numpy
img = np.array(image_pil)
```

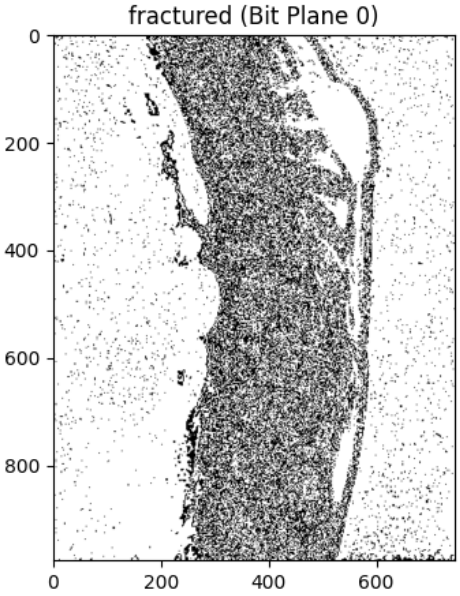
```
def power_transform(c, gamma, img):
    return c * np.power(img, gamma)
gamma_values = [0.1, 0.5, 1, 2, 5]
for gamma in gamma_values:
    transformed_img = power_transform(1, gamma,
img)
    plt.imshow(transformed_img, cmap='gray')
    plt.title(f' Y = {gamma}')
    plt.show()
```

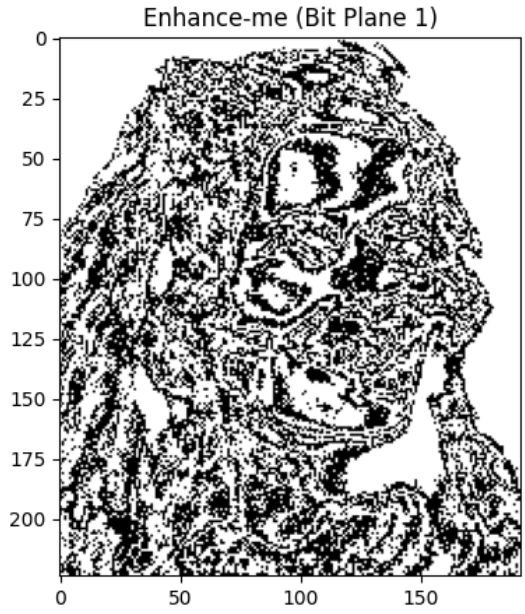
Implemente a representação de cada plano de bits das imagens:

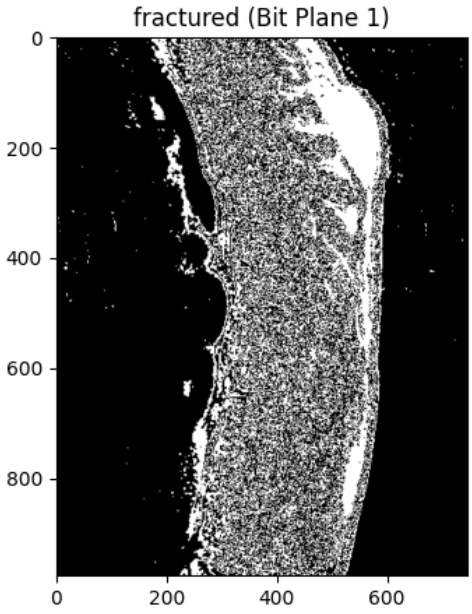


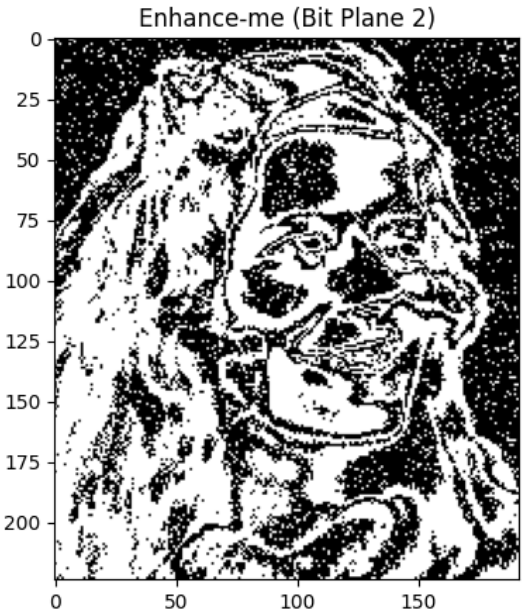


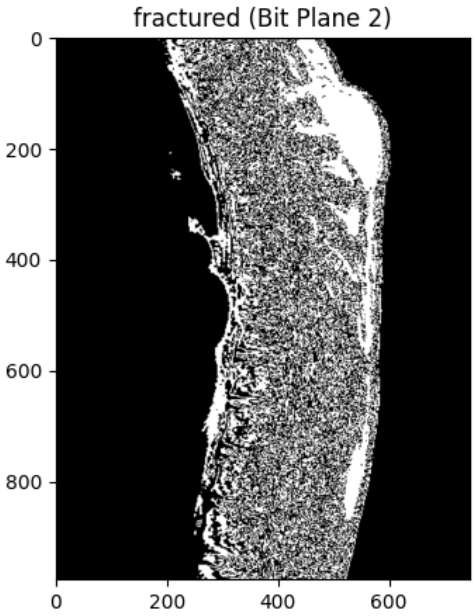

Enhance-me (Bit Plane 0)
0
25
50
75
100
125
150
175
200
0
50
100
150

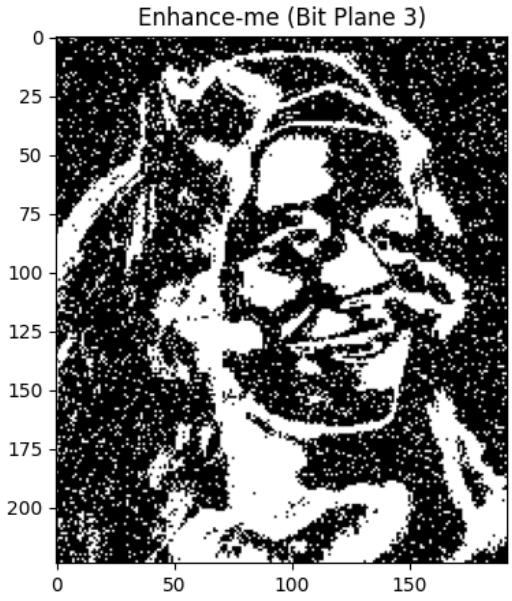

fractured (Bit Plane 0)
0
200
400
600
800
0
200
400
600

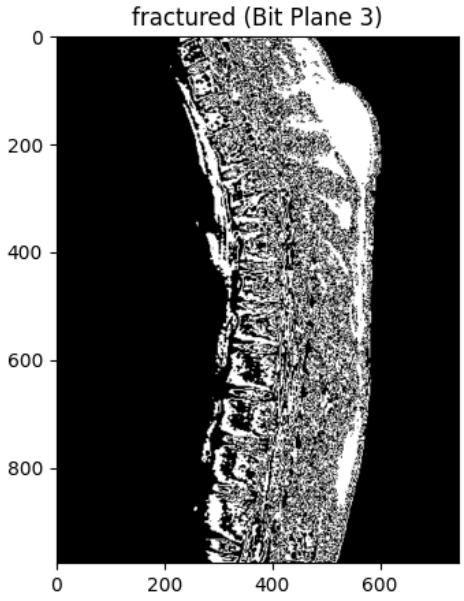

Enhance-me (Bit Plane 1)
0
25
50
75
100
125
150
175
200
0
50
100
150

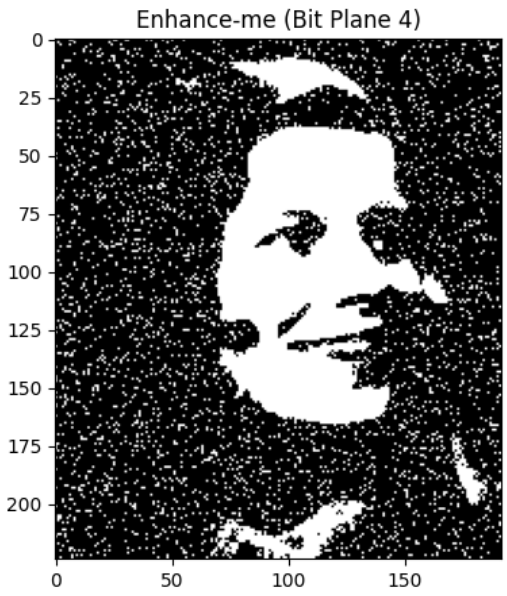

fractured (Bit Plane 1)
0
200
400
600
800
0
200
400
600

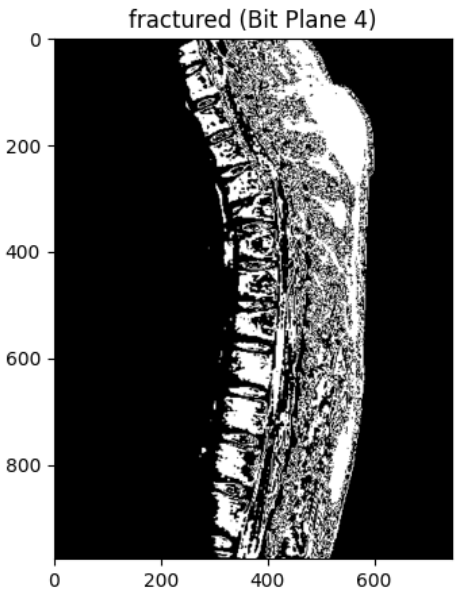

Enhance-me (Bit Plane 2)
0
25
50
75
100
125
150
175
200
0
50
100
150

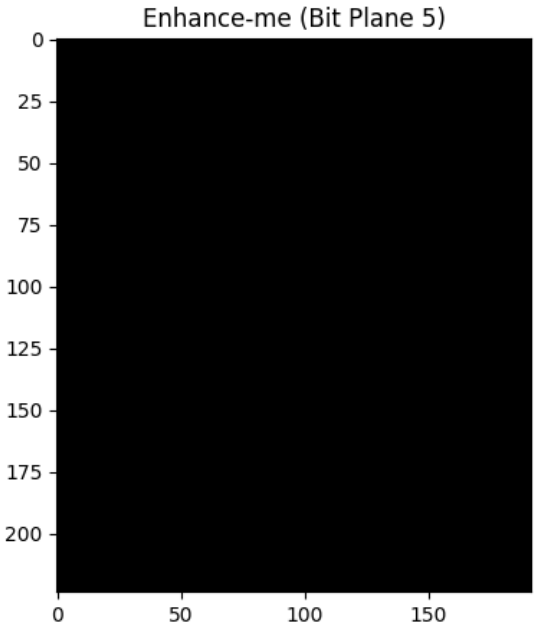

fractured (Bit Plane 2)
0
200
400
600
800
0
200
400
600

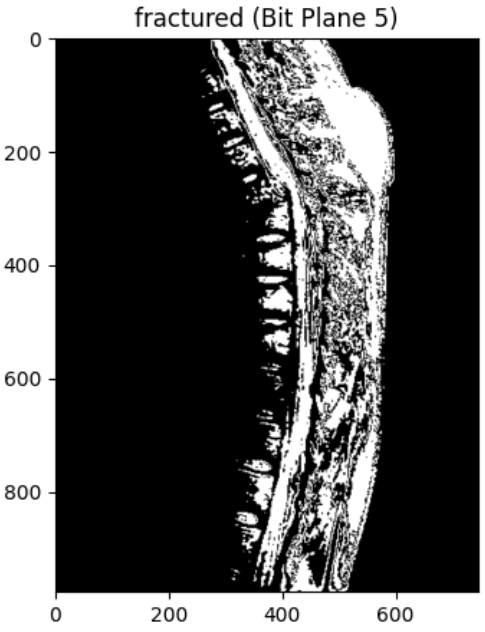

Enhance-me (Bit Plane 3)
0
25
50
75
100
125
150
175
200
0
50
100
150

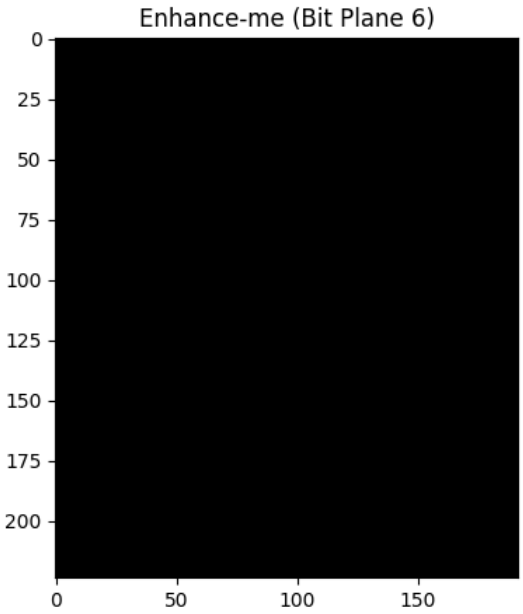

fractured (Bit Plane 3)
0
200
400
600
800
0
200
400
600

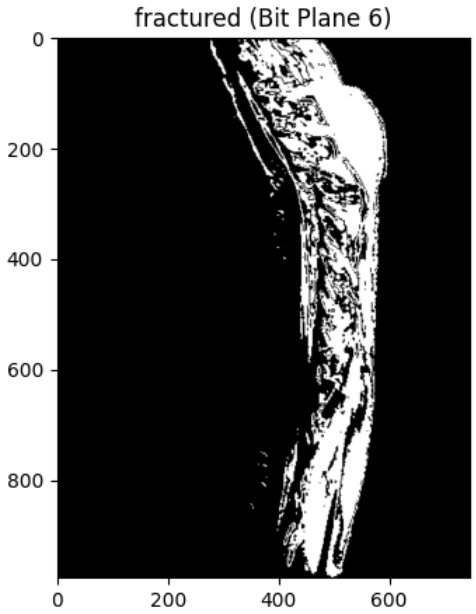

Enhance-me (Bit Plane 4)
0
25
50
75
100
125
150
175
200
0
50
100
150

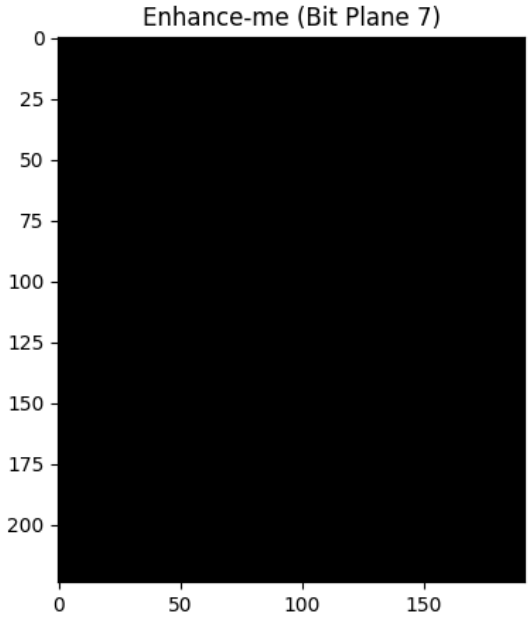

fractured (Bit Plane 4)
0
200
400
600
800
0
200
400
600

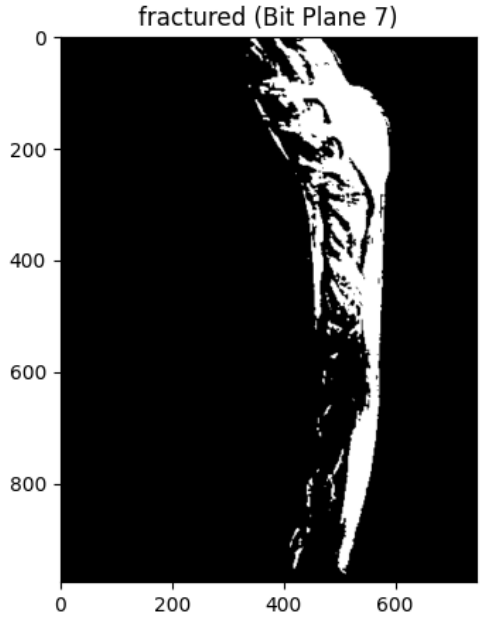

Enhance-me (Bit Plane 5)
0
25
50
75
100
125
150
175
200
0
50
100
150


fractured (Bit Plane 5)
0
200
400
600
800
0
200
400
600









Código:

```
import cv2
import numpy as np
import matplotlib.pyplot as plt
from PIL import Image

enhance_path ='image/enhance-me.gif'
fractured_path ='image/fractured_spine.tif'
enhance_img = np.array(Image.open(enhance_path))
```

```
fractured_img =
np.array(Image.open(fractured_path))

def bit_plane(img, bit):
    return (img & (1 << bit)) >> bit
16
# Representação de cada plano de bits
for i in range(8):
    bit_enhance = bit_plane(enhance_img, i)
    bit_Fig0308 = bit_plane(fractured_img, i)

# Exibição
    fig, axs = plt.subplots(1, 2, figsize=(10, 5))
    axs[0].imshow(bit_enhance, cmap='gray')
    axs[0].set_title(f'Enhance-me (Bit Plane {i})')
    axs[1].imshow(bit_Fig0308, cmap='gray')
    axs[1].set_title(f'fractured (Bit Plane {i})')
    plt.show()

# Equalização do histograma
equalized_enhance = cv2.equalizeHist(enhance_img)
equalized_fractured=
cv2.equalizeHist(fractured_img)
# Exibição
fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(equalized_enhance, cmap='gray')
axs[0].set_title('Enhance-me (Equalização)')
axs[1].imshow(equalized_fractured, cmap='gray')
axs[1].set_title('fractured (Equalização)')
plt.show()
```

4.1 Transformação Logarítmica:

A técnica da transformação logarítmica é utilizada para ampliar a escala de valores associados aos pixels mais escuros, ao mesmo tempo que reduz a

faixa de valores correspondente aos pixels mais claros. Essa estratégia é especialmente eficaz quando se lida com imagens que apresentam amplas variações de brilho. Na programação com a biblioteca numpy, a aplicação dessa transformação é realizada por meio da seguinte fórmula: "s = c × log(1 + r)", onde 'r' representa o valor original do pixel e 'c' é uma constante que controla o nível de contraste da transformação. A alteração do valor de 'c' provoca mudanças notáveis no contraste da imagem resultante, com valores maiores de 'c' intensificando o contraste e valores menores suavizando o efeito.

4.2 Transformação Potência (Gama):

A técnica de transformação potência tem como objetivo expandir a faixa de valores dos pixels escuros e comprimir a dos pixels claros, sendo particularmente útil em imagens com ampla variação de brilho. Com a ajuda da biblioteca numpy, essa transformação pode ser implementada usando a seguinte fórmula: "s = c × log(1 + r)". Nessa equação, "r" representa o valor do pixel original e "c" é uma constante que controla o nível de contraste da transformação. Ajustar o valor de "c" resulta em modificações no contraste da imagem, com valores maiores de "c" produzindo um contraste mais acentuado e valores menores suavizando esse efeito.

4.3 Representação de Cada Plano de Bits:

Uma imagem em escala de cinza pode ser representada em planos de bits separados, com cada plano correspondendo a um bit específico no valor do pixel. A obtenção da representação de cada plano de bits envolve isolar cada bit presente no valor do pixel. É importante destacar que o plano de bits mais significativo contém a maior parte da informação visual da imagem, enquanto os planos de bits menos significativos contribuem para a representação de detalhes mais sutis, mas também podem introduzir ruído na imagem com maior frequência.

4.4 Equalização do Histograma:

A equalização do histograma é uma técnica que reorganiza os valores dos pixels em uma imagem com o objetivo de criar um histograma mais uniforme. No contexto da biblioteca OpenCV, essa equalização pode ser facilmente realizada com a função cv2.equalizeHist(). Essa abordagem melhora o contraste global da imagem, tornando os detalhes mais nítidos e distribuindo de forma mais uniforme a intensidade dos pixels.

GitHub: https://github.com/isadoradisposti/Transformacoes-e-Histograma.