 Instituto Federal de Educação, Ciência e Tecnologia de São Paulo	Engenharia de Computação
Câmpus Birigui	Processamento digital de imagens

Transformações de Intensidade

Discente(s): Leonardo Sanchez Garcia

Docente: Prof. Dr. Murilo Varges da Silva

Birigui
2023

Introdução

Este relatório tem como objetivo detalhar e analisar as transformações de imagem aplicadas à imagem 'fractured_spine.tif'. Foram utilizadas três técnicas de processamento de imagem: Transformação Logarítmica, Transformação de Potência (Gama) e Equalização de Histograma. Cada transformação tem características distintas e afeta a imagem de maneira específica.
GitHub: https://github.com/LeonardoSanchez1/TransformacoesDeIntensidade.git

Imagem Original

1. Transformação Logarítmica:

A Transformação Logarítmica é uma técnica que se utiliza de uma função logarítmica para expandir a faixa dinâmica de uma imagem, particularmente quando a imagem possui áreas escuras com baixo contraste.

Fórmula: A transformação é expressa pela fórmula s = c * log(1 + r), onde s é o valor de saída, c é um coeficiente de ajuste e r é o valor de entrada (intensidade do pixel).

Implementação: Foi realizada a implementação da transformação logarítmica usando Python e a biblioteca OpenCV. Diferentes valores para o coeficiente c foram testados para observar os efeitos na imagem.

```
import cv2
import numpy as np
```

```
import matplotlib.pyplot as plt

# Carregar a imagem em escala de cinza
imagem = cv2.imread('fractured_spine.tif', 0)

# Criar uma figura para exibir as imagens
plt.figure(figsize=(12, 6))

# Aplicar a transformação logarítmica com diferentes valores de c
valores_c = [10, 50, 100]  # Você pode ajustar esses valores
conforme necessário

for i, c in enumerate(valores_c):
    # Aplicar a transformação logarítmica
    imagem_transformada = c * np.log1p(imagem)
    imagem_transformada = np.uint8(imagem_transformada)

    # Plotar a imagem transformada em uma grade
    plt.subplot(1, len(valores_c), i + 1)
    plt.imshow(imagem_transformada, cmap='gray')
    plt.title(f'Transformação Logarítmica (c={c})')
    plt.axis('off')

plt.tight_layout()
plt.show()
```

Efeito na imagem: A transformação logarítmica é eficaz em realçar detalhes em áreas escuras da imagem. Aumenta os valores dos pixels mais escuros, melhorando o contraste global e tornando a imagem mais visível.

Transformação Logarítmica (c=50)

Transformação Logarítmica (c=100)

2. Transformação de Potência (Gama):

A Transformação de Potência, também conhecida como transformação gama, é usada para ajustar o contraste em uma imagem. Ela é especialmente útil para realçar ou suavizar áreas de intensidade de pixel.

Fórmula: A transformação de potência é expressa pela fórmula s = c * r^y, onde s é o valor de saída, c é um coeficiente de ajuste, r é o valor de entrada (intensidade do pixel) e y é o parâmetro gama.

Implementação: A transformação de potência foi implementada utilizando Python e a biblioteca OpenCV. Diferentes valores para o parâmetro y foram testados, mantendo c fixo em 1.

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

# Carregar a imagem em escala de cinza
imagem = cv2.imread('fractured_spine.tif', 0)

# Criar uma figura para exibir as imagens
plt.figure(figsize=(12, 6))

# Aplicar a transformação de potência com diferentes valores de y
valores_y = [0.5, 1.0, 1.5]  # Você pode ajustar esses valores
conforme necessário
c = 1

for i, y in enumerate(valores_y):
    # Aplicar a transformação de potência
    imagem_transformada = c * np.power(imagem, y)
    imagem_transformada = np.uint8(imagem_transformada)

    # Plotar a imagem transformada em uma grade
    plt.subplot(1, len(valores_y), i + 1)
    plt.imshow(imagem_transformada, cmap='gray')
    plt.title(f'Transformação de Potência (y={y}, c={c})')
    plt.axis('off')

plt.tight_layout()
plt.show()
```

Efeito na imagem: A transformação de potência altera o contraste e a luminosidade da imagem. Valores de y menores que 1 aumentam o contraste, enquanto valores maiores que 1 reduzem o contraste. Permite um controle flexível sobre a intensidade da transformação.

Transformação de Potência (y=0.5, c=1)

Transformação de Potência (y=1.0, c=1)

3. Representação de Planos de Bits

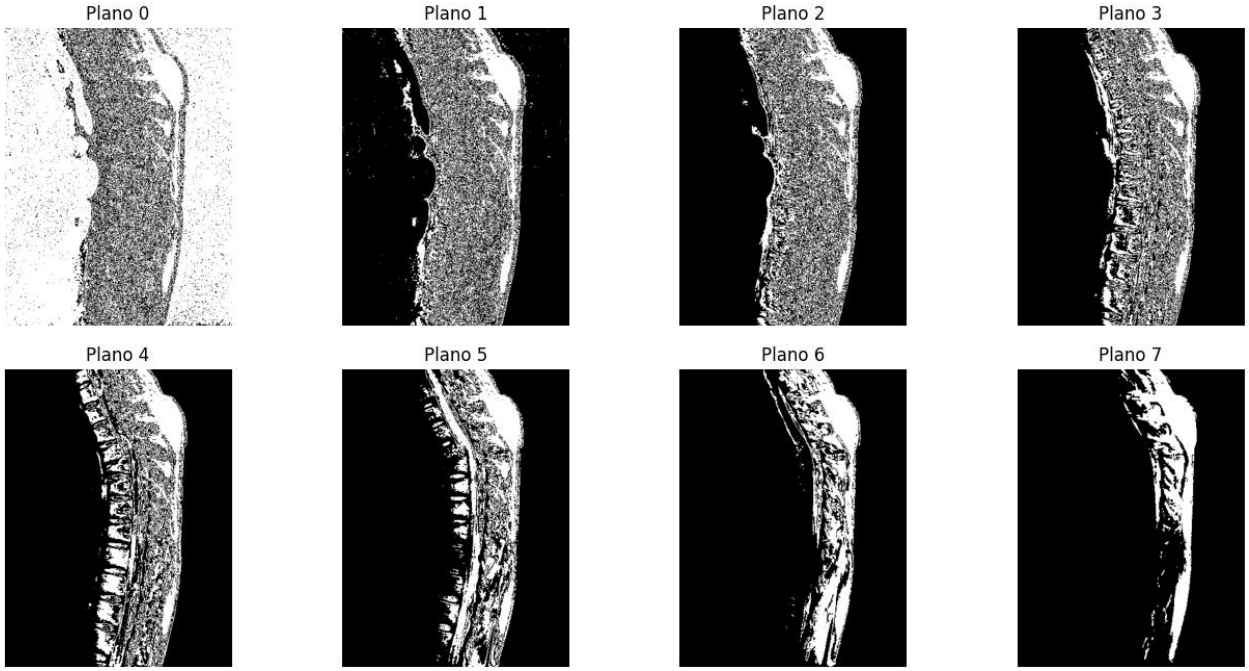
Além das transformações, representamos cada plano de bits da imagem original 'fractured_spine.tif'. Isso envolve a extração de cada bit individual que compõe a imagem e sua exibição.

Implementação: Utilizamos Python e a biblioteca OpenCV para extrair cada plano de bits. Cada plano foi criado como uma imagem binária, onde os pixels brancos representam 1 e os pixels pretos representam 0.

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

# Carregar a imagem
imagem = cv2.imread('fractured_spine.tif', 0)  # 0 para carregar a
imagem em escala de cinza

# Obter as dimensões da imagem
altura, largura = imagem.shape

# Criar uma lista para armazenar os planos de bits
planos_de_bits = []

# Extrair cada plano de bits
for i in range(8):  # Vamos considerar 8 bits, mas você pode ajustar
conforme necessário
    plano = (imagem >> i) & 1  # Extrair o bit de cada pixel
```

```
    planos_de_bits.append(plano)

# Configurar a grade para exibir as imagens
linhas = 2
colunas = 4

# Criar uma figura para exibir as imagens
fig, axs = plt.subplots(linhas, colunas, figsize=(12, 6))

# Exibir os planos de bits na grade
for i, plano in enumerate(planos_de_bits):
    row = i // colunas
    col = i % colunas
    axs[row, col].imshow(plano, cmap='gray')
    axs[row, col].set_title(f'Plano {i}')
    axs[row, col].axis('off')

plt.tight_layout()
plt.show()
```

Efeito na imagem: A representação de planos de bits permite observar a contribuição de cada bit para a imagem original. Isso pode ser útil para análise e processamento de imagem em níveis de bits separados.



4. Equalização de Histograma

A Equalização de Histograma é uma técnica que redistribui os valores de intensidade dos pixels para melhorar o contraste em uma imagem, tornando-a mais equilibrada e nítida.

Implementação: A equalização de histograma foi implementada utilizando Python e a biblioteca OpenCV.

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

# Carregar a imagem em escala de cinza
imagem = cv2.imread('fractured_spine.tif', 0)

# Aplicar a equalização do histograma
imagem_equalizada = cv2.equalizeHist(imagem)

# Exibir a imagem original e a imagem equalizada
plt.figure(figsize=(10, 5))
plt.subplot(1, 2, 1)
plt.imshow(imagem, cmap='gray')
plt.title('Imagem Original')
plt.axis('off')

plt.subplot(1, 2, 2)
plt.imshow(imagem_equalizada, cmap='gray')
plt.title('Imagem Equalizada')
plt.axis('off')

plt.tight_layout()
plt.show()
```

Efeito na imagem: A equalização de histograma melhora significativamente o contraste da imagem. Ela estica o histograma para ocupar toda a faixa de valores disponíveis, resultando em uma imagem com mais detalhes visíveis, especialmente em áreas de baixo contraste.

Conclusão

As transformações de imagem e a representação de planos de bits oferecem técnicas poderosas para o processamento e análise de imagens. A transformação logarítmica e a transformação de potência permitem ajustar o contraste e realçar detalhes, enquanto a equalização de histograma melhora o contraste global. A representação de planos de bits fornece insights sobre a estrutura da imagem em níveis de bits individuais. A escolha das técnicas depende dos objetivos específicos de processamento de imagem e do efeito desejado na imagem. Em conjunto, essas técnicas podem melhorar significativamente a qualidade visual e a utilidade da imagem processada.