Operações Pontuais

Conceito

As operações pontuais são usadas para modificar os valores de pixel em uma imagem sem afetar seu tamanho, geometria ou estrutura local.

Cada novo valor de pixel depende apenas do valor anterior na mesma posição e é determinado por uma função, que pode ser global (a mesma em toda a imagem) ou não.

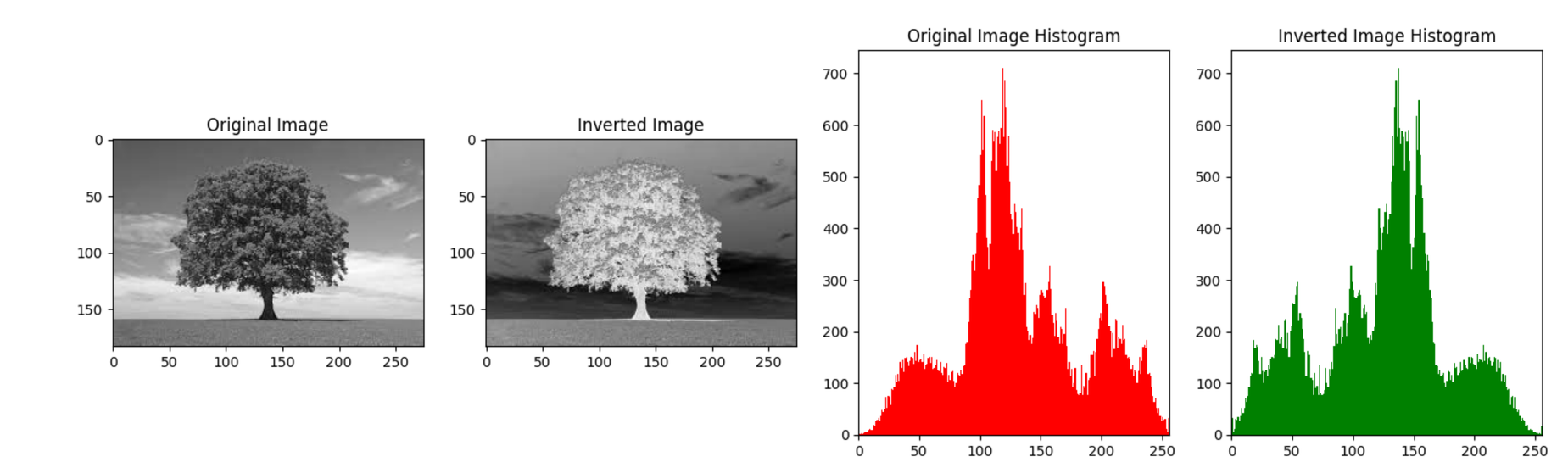
Se a operação for independente das coordenadas do pixel, ela é chamada de homogênia. Exemplos de operações pontuais homogêneas incluem mudanças no brilho, contraste e cor, limiarização global e correção gama. Essas operações são úteis em muitas aplicações de processamento de imagem.

Ao contrário das operações pontuais homogêneas, as operações pontuais não homogêneas levam em consideração a coordenada atual da imagem ao determinar o novo valor de pixel. Um exemplo comum de operação pontual não homogênea é o ajuste local de contraste ou brilho, usado para compensar a iluminação irregular durante a aquisição da imagem.

Inversão de Intensidade

A operação de inversão de intensidade de uma imagem é um processo que inverte a ordem dos valores de pixel, multiplicando cada valor por -1 e adicionando um valor constante para mapear o resultado novamente para a faixa admissível.

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

# load img in gray scale
img = cv2.imread("./images/tree.jpeg", 0)

# invertion a = (aMax - a)
inverted_img = np.max(img) - img

# img show using matplotlib
fig, (ax1, ax2, ax3, ax4) = plt.subplots(ncols=4, figsize=(20, 5))

ax1.imshow(cv2.cvtColor(img, cv2.COLOR_BGR2RGB))
ax1.set_title("Original Image")

ax2.imshow(cv2.cvtColor(inverted_img, cv2.COLOR_BGR2RGB))
ax2.set_title("Inverted Image")

ax3.hist(img.ravel(), 256, [0, 256], color='red')
ax3.set_title("Original Image Histogram")
ax3.set_xlim([0, 256])

ax4.hist(inverted_img.ravel(), 256, [0, 256], color='green')
ax4.set_title("Inverted Image Histogram")
ax4.set_xlim([0, 256])

plt.show()
```

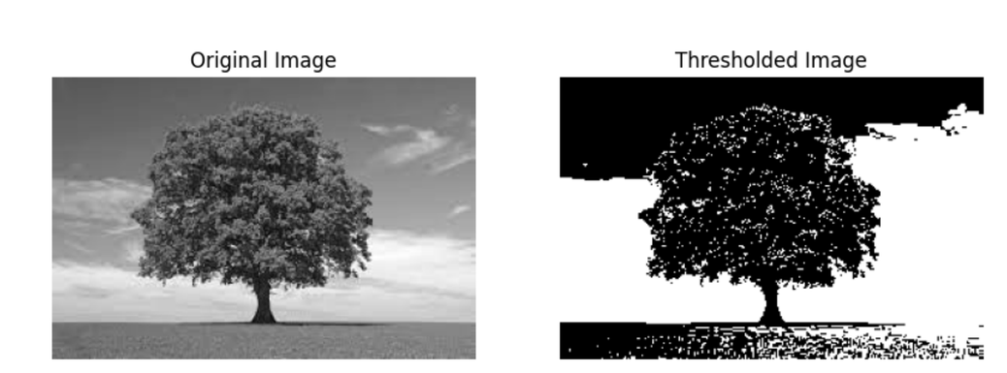


Limiarização

O limiar de imagem é uma técnica utilizada para binarizar imagens com base em um valor de limite, onde a intensidade dos pixels acima desse valor é marcada como branco e abaixo ou igual é marcada como preto.

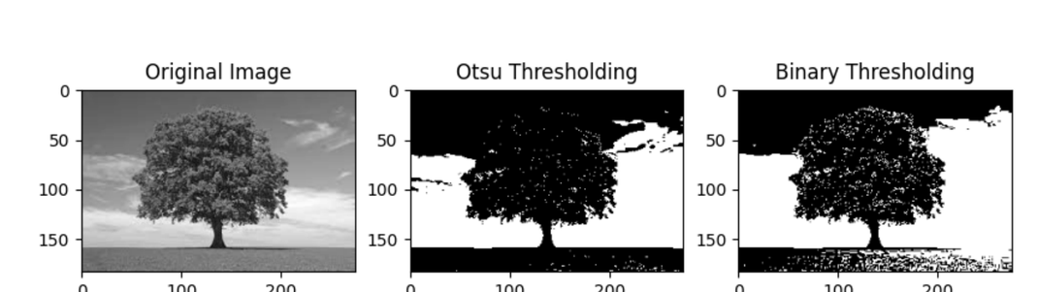
É uma etapa comum de pré-processamento em aplicações como processamento de imagens médicas e análise de imagens de satélite. Matematicamente, o limite de imagem é um tipo especial de quantização que mapeia todos os pixels para um de dois valores de intensidade fixa, dependendo do valor de limite definido.

```python
import cv2
import matplotlib.pyplot as plt

# load image in grayscale
img = cv2.imread('./images/tree.jpeg', cv2.IMREAD_GRAYSCALE)

# apply binary thresholding with threshold value of 127
_, thresh = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

# plot original and thresholded images side by side
fig, axs = plt.subplots(1, 2, figsize=(10, 5))
axs[0].imshow(img, cmap='gray')
axs[0].set_title('Original Image')
axs[0].axis('off')
axs[1].imshow(thresh, cmap='gray')
axs[1].set_title('Thresholded Image')
axs[1].axis('off')
plt.show()
```



O método de limiarização de Otsu é uma técnica de segmentação de imagem GLOBAL que tenta encontrar o melhor limite para separar os objetos da imagem de fundo. O método de Otsu funciona calculando uma medida de variância intra-classe para diferentes valores de limiar e, em seguida, escolhe o limiar que minimiza a variância intra-classe. Isso permite encontrar um limiar que maximiza a diferença entre as intensidades de pixels do objeto e do fundo, resultando em uma segmentação mais precisa.

```python
import cv2
import matplotlib.pyplot as plt

img = cv2.imread('./images/tree.jpeg', 0)

# apply otsu
otsu_thresh, otsu_img = cv2.threshold(img, 0, 255, cv2.THRESH_BINARY + cv2.THRESH_OTSU)

# apply binary
binary_thresh, binary_img = cv2.threshold(img, 127, 255, cv2.THRESH_BINARY)

# display image, otsu and binary
fig, ax = plt.subplots(1, 3, figsize=(10, 5))
ax[0].imshow(img, cmap='gray')
ax[0].set_title('Original Image')
ax[1].imshow(otsu_img, cmap='gray')
ax[1].set_title('Otsu Thresholding')
ax[2].imshow(binary_img, cmap='gray')
ax[2].set_title('Binary Thresholding')
plt.show()
```

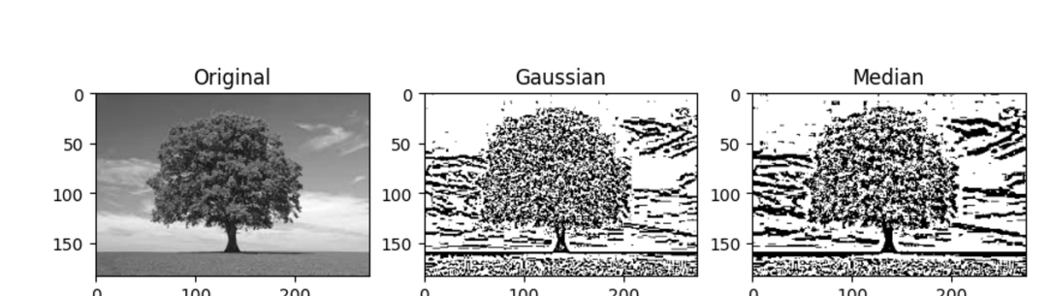


Limiarização Adaptativa

A limiarização adaptativa é uma técnica utilizada para separar objetos de interesse do fundo em imagens com variações espaciais na iluminação. Essa técnica usa um limite dinâmico que varia de acordo com as condições de iluminação em diferentes áreas da imagem.

Existem três parâmetros de entrada para a técnica de limiarização adaptativa: método adaptativo, tamanho do bloco e constante C. O método adaptativo pode ser cv2.ADAPTIVE_THRESH_MEAN_C ou cv2.ADAPTIVE_THRESH_GAUSSIAN_C, e o tamanho do bloco e a constante C afetam a precisão do resultado.

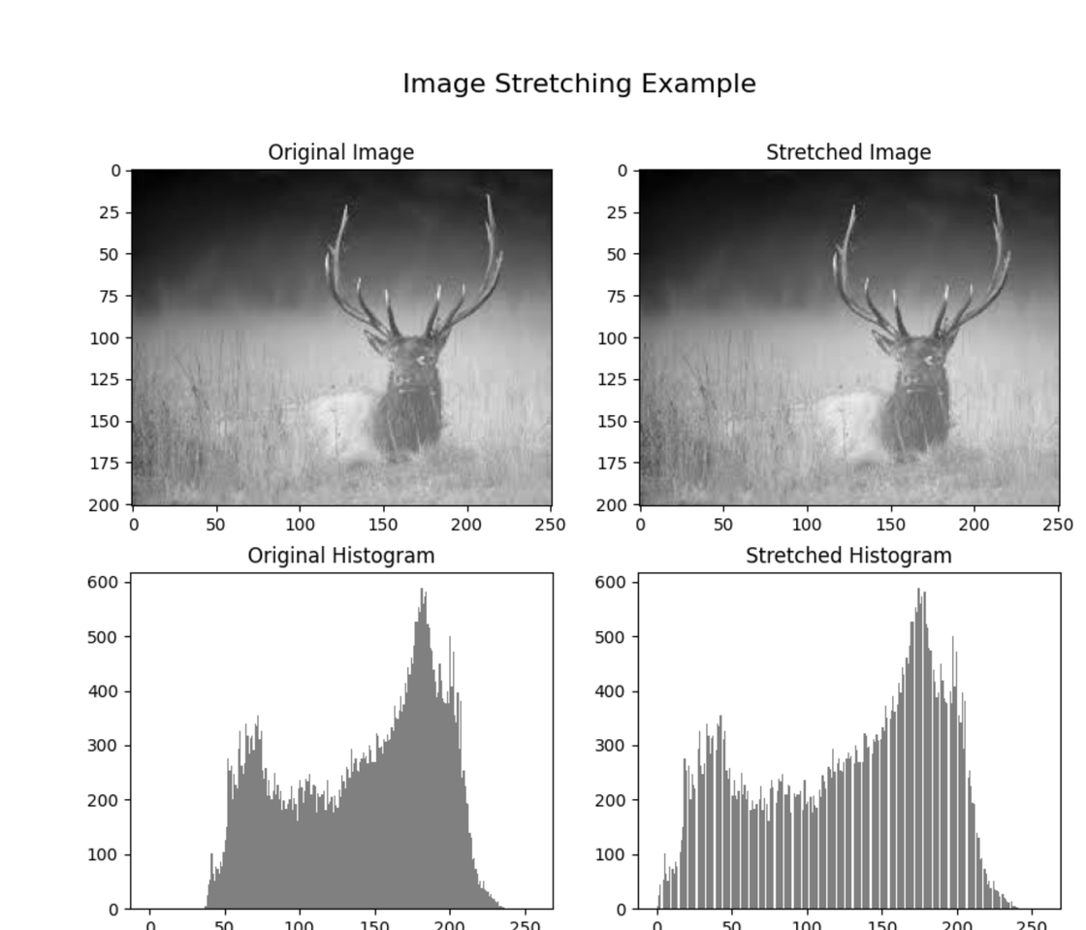
```python
import cv2
import numpy as np
import matplotlib.pyplot as plt

# load image in gray
img = cv2.imread('./images/tree.jpeg', 0)

# gausian adaptive threshold
gaussian = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_GAUSSIAN_C, cv2.THRESH_BINARY, 11, 2)
# median adaptive threshold
median = cv2.adaptiveThreshold(img, 255, cv2.ADAPTIVE_THRESH_MEAN_C, cv2.THRESH_BINARY, 11, 2)

#plot
fig, axs = plt.subplots(1, 3, figsize=(10, 10))
axs[0].imshow(img, cmap='gray')
axs[0].set_title('Original')
axs[1].imshow(gaussian, cmap='gray')
axs[1].set_title('Gaussian')
axs[2].imshow(median, cmap='gray')
axs[2].set_title('Median')
plt.show()
```



Normalização do Histograma

A normalização é um processo que muda a faixa de valores de intensidade de pixel em imagens ou sinais, com o objetivo de trazê-los para uma faixa mais familiar ou normal aos sentidos. A normalização linear de uma imagem digital em tons de cinza é realizada por meio de uma fórmula que envolve a subtração do valor mínimo da faixa original e a multiplicação pelo valor máximo desejado dividido pela diferença entre os valores mínimo e máximo originais. Isso resulta em uma nova faixa de valores para a imagem. A normalização é útil para melhorar o contraste em imagens com baixo contraste devido ao brilho e para garantir consistência na faixa dinâmica de um conjunto de dados.

```python
import cv2
import matplotlib.pyplot as plt


img = cv2.imread('./images/deer.jpeg', cv2.IMREAD_GRAYSCALE)

# normalization
img_stretched = cv2.normalize(img, None, 0, 255, cv2.NORM_MINMAX)

# plot both images and histograms
fig, axs = plt.subplots(2, 2, figsize=(10, 8))
fig.suptitle('Image Stretching Example', fontsize=16)
axs[0, 0].imshow(img, cmap='gray')
axs[0, 0].set_title('Original Image')
axs[1, 0].hist(img.ravel(), 256, [0, 256], color='gray')
axs[1, 0].set_title('Original Histogram')
axs[0, 1].imshow(img_stretched, cmap='gray')
axs[0, 1].set_title('Stretched Image')
axs[1, 1].hist(img_stretched.ravel(), 256, [0, 256], color='gray')
axs[1, 1].set_title('Stretched Histogram')
plt.show()
```



Equalização do Histograma

A equalização do histograma é uma técnica utilizada para ajustar duas imagens diferentes de forma que suas distribuições de intensidade resultantes sejam semelhantes, tornando-as mais fáceis de comparar. O objetivo é encontrar e aplicar uma operação pontual que torne o histograma da imagem modificada mais uniforme.

A operação pontual desejada para equalização do histograma é obtida a partir do histograma cumulativo da imagem original. A função resultante é monotônica crescente e transforma o histograma da imagem original em um histograma aproximadamente uniforme. A equalização linear do histograma não deve modificar uma imagem uniformemente distribuída e aplicações repetidas não devem alterar a imagem após a primeira vez.

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


img = cv2.imread('./images/tree.jpeg', cv2.IMREAD_GRAYSCALE)

# equalizing
img_eq = cv2.equalizeHist(img)

# plots
fig, axs = plt.subplots(3, 2, figsize=(10, 10))
axs[0, 0].imshow(img, cmap='gray')
axs[0, 0].set_title('Original Image')
axs[1, 0].hist(img.flatten(), 256, [0, 256], color='r')
axs[1, 0].set_xlim([0, 256])
axs[1, 0].set_title('Original Histogram')
axs[0, 1].imshow(img_eq, cmap='gray')
axs[0, 1].set_title('Equalized Image')
axs[1, 1].hist(img_eq.flatten(), 256, [0, 256], color='r')
axs[1, 1].set_xlim([0, 256])
axs[1, 1].set_title('Equalized Histogram')
axs[2, 0].plot(cv2.calcHist([img], [0], None, [256], [0, 256]).cumsum())
axs[2, 0].set_title('Original Cumulative Histogram')
axs[2, 1].plot(cv2.calcHist([img_eq], [0], None, [256], [0, 256]).cumsum())
axs[2, 1].set_title('Equalized Cumulative Histogram')
plt.show()
```