Filtragem no Domínio Espacial em Processamento Digital de Imagens com Python e OpenCV

Objetivo da Aula

Compreender e aplicar técnicas de filtragem no domínio espacial, transformações de intensidade, processamento de histograma, e operações lógicas e aritméticas em imagens digitais. Utilizaremos a imagem padrão de Lena para ilustrar cada conceito.

1. Introdução à Filtragem no Domínio Espacial

Filtragem no domínio espacial é uma técnica em processamento de imagens onde uma máscara ou kernel é aplicada diretamente aos pixels da imagem. Isso permite modificar aspectos como nitidez e suavidade.

Princípios Básicos:

- **Kernel**: Matriz (geralmente 3x3, 5x5, etc.) aplicada para calcular novos valores de pixel.
- **Convolução**: Processo de movimentar o kernel sobre cada pixel para gerar a imagem resultante.

2. Funções de Transformação de Intensidade

Essas funções ajustam os níveis de intensidade de uma imagem, modificando brilho e contraste.

Exemplo 1: Ajuste de Brilho e Contraste

Este exemplo aumenta o brilho e o contraste da imagem de Lena.

```
import cv2
import numpy as np

# Carregar imagem de Lena em tons de cinza
img = cv2.imread('lena.png', cv2.IMREAD_GRAYSCALE)

# Ajuste de brilho e contraste
alpha = 1.5  # Fator de contraste
beta = 50    # Fator de brilho
bright_contrast_img = cv2.convertScaleAbs(img, alpha=alpha, beta=beta)

# Exibir a imagem resultante
cv2.imshow("Brilho e Contraste", bright_contrast_img)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

Explicação:

- `alpha` controla o contraste (maior que 1 aumenta o contraste).
- `beta` adiciona brilho ao valor do pixel.

3. Processamento de Histograma

O histograma representa a distribuição dos níveis de intensidade. A equalização de histograma melhora o contraste ao distribuir melhor os valores de intensidade.

Exemplo 2: Equalização de Histograma

Este exemplo aumenta o contraste de Lena ao redistribuir os níveis de intensidade da imagem.

```
# Equalização de histograma
equalized_img = cv2.equalizeHist(img)

# Exibir imagem equalizada
cv2.imshow("Equalização de Histograma", equalized_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Exemplo 3: Visualização do Histograma

Abaixo, mostramos como visualizar o histograma da imagem para observar a distribuição de intensidade antes e depois da equalização.

```
import matplotlib.pyplot as plt

# Calcular e plotar o histograma da imagem original
hist_orig = cv2.calcHist([img], [0], None, [256], [0, 256])
plt.plot(hist_orig, color='gray')
plt.title("Histograma Original de Lena")
plt.xlabel("Intensidade de Pixel")
plt.ylabel("Frequência")
plt.show()

# Calcular e plotar o histograma da imagem equalizada
hist_eq = cv2.calcHist([equalized_img], [0], None, [256], [0, 256])
plt.plot(hist_eq, color='blue')
plt.title("Histograma Equalizado de Lena")
plt.xlabel("Intensidade de Pixel")
```

```
plt.ylabel("Frequência")
plt.show()
```

4. Operações Lógicas em Imagens

Operações lógicas permitem combinar ou modificar imagens. Estes são úteis para realçar partes específicas de uma imagem ou aplicar máscaras.

Exemplo 4: Operação Lógica AND

Aqui aplicamos uma máscara circular na imagem de Lena usando a operação AND.

```
# Criar uma máscara circular
mask = np.zeros(img.shape, dtype="uint8")
cv2.circle(mask, (img.shape[1] // 2, img.shape[0] // 2), 100, 255, -1)

# Aplicar a operação lógica AND
and_img = cv2.bitwise_and(img, mask)

# Exibir o resultado
cv2.imshow("Operação AND com Máscara", and_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

5. Operações Aritméticas em Imagens

Essas operações ajustam o conteúdo da imagem. A adição e a subtração são usadas para ajustar brilho, combinar imagens, ou destacar diferenças.

Exemplo 5: Adição e Subtração de Imagens

Adicionamos e subtraímos uma imagem borrada de Lena para destacar detalhes.

```
python
# Aplicar um filtro de suavização na imagem
blurred_img = cv2.GaussianBlur(img, (15, 15), 0)

# Operação de adição
added_img = cv2.addWeighted(img, 0.5, blurred_img, 0.5, 0)

# Operação de subtração
subtracted_img = cv2.subtract(img, blurred_img)

# Exibir os resultados
cv2.imshow("Imagem Adicionada", added_img)
cv2.imshow("Imagem Subtraída", subtracted_img)
cv2.waitKey(0)
```

```
cv2.destroyAllWindows()
```

Explicação:

- Na **adição**, fundimos suavemente a imagem original com a borrada.
- Na **subtração**, destacamos os detalhes ao remover as regiões suavizadas.

6. Exemplo Prático Completo

Combina as técnicas acima em uma aplicação prática com a imagem de Lena.

```
# Carregar imagem original
img = cv2.imread('lena.png', cv2.IMREAD_GRAYSCALE)

# 1. Ajuste de Brilho e Contraste
alpha, beta = 1.5, 30
bright_contrast_img = cv2.convertScaleAbs(img, alpha=alpha, beta=beta)

# 2. Equalização de Histograma
equalized_img = cv2.equalizeHist(bright_contrast_img)

# 3. Filtro de Suavização
smooth_img = cv2.blur(equalized_img, (5, 5))

# 4. Operação Lógica (AND com máscara circular)
mask = np.zeros(img.shape, dtype="uint8")
cv2.circle(mask, (img.shape[1] // 2, img.shape[0] // 2), 100, 255, -1)
and_img = cv2.bitwise_and(smooth_img, mask)

# 5. Operação Aritmética - Subtração para Destaque
subtracted_img = cv2.subtract(img, smooth_img)

# Exibir Resultados
cv2.imshow("Original", img)
cv2.imshow("Brilho e Contraste", bright_contrast_img)
cv2.imshow("Equalização de Histograma", equalized_img)
cv2.imshow("Suavização", smooth_img)
cv2.imshow("AND com Máscara", and_img)
cv2.imshow("Subtração", subtracted_img)
cv2.waitKey(0)
cv2.destroyAllWindows()
```

Conclusão

Nesta aula, abordamos o conceito de filtragem no domínio espacial, ajustando intensidade, manipulando histogramas e aplicando operações lógicas e aritméticas na

imagem padrão de Lena. Essas técnicas são fundamentais para realçar detalhes, ajustar características visuais e combinar diferentes imagens, preparando o caminho para manipulações avançadas em processamento de imagens.

Exercícios

1. **Experimente diferentes valores de alpha e beta** na transformação de intensidade para ver o efeito no brilho e contraste.
2. **Teste diferentes tamanhos de kernel** na suavização e observe o impacto no detalhamento da imagem.
3. **Aplique diferentes operações lógicas** (como XOR) entre uma imagem e sua máscara para entender o comportamento de cada operação.

Essas operações são essenciais para manipulação básica e realce de imagens, preparando o terreno para abordagens mais sofisticadas em Processamento Digital de Imagens.