Esqueletização e Poda de Imagem Binária

Atividade Pontuada 03 - Processamento de Imagens

Introdução

Este trabalho tem como objetivo implementar a esqueletização (skeletonization) binária e o algoritmo de poda (pruning) para refinamento do resultado, seguindo o método proposto por Gonzalez & Woods.

Neste notebook, implemento o algoritmo de esqueletização para uma imagem binária de uma impressão digital, utilizando o conceito de erosões e aberturas. A abordagem é baseada no livro "Digital Image Processing" de Gonzalez e Woods.

Imagem Original

```
import matplotlib.pyplot as plt #para exibir a imagem

from matplotlib.image import imread

#diretório da imagem
imagem = imread(r"C:\Users\Wemerson\Downloads\digital.png")

#exibir imagem
plt.imshow(imagem, cmap="gray")
plt.title("Impressão Digital Binária",loc="left")
plt.axis("off")
plt.show()
```

Impressão Digital Binária

Preparação da Imagem

Normalmente, antes da esqueletização, a imagem precisa ser binarizada para garantir que existam apenas dois valores de intensidade (preto=0 e branco=255). No entanto, a imagem anexada já possui apenas duas intensidades distintas, o que dispensa a necessidade dessa etapa.

```
import numpy as np  #biblioteca NumPy para manipulação de arrays/matrizes

#verificar os valores únicos na imagem
valores_unicos = np.unique(imagem)
print("Valores únicos na imagem:", valores_unicos)
```

```
Valores únicos na imagem: [0. 1.]
```

I - Esqueletização (Skeletonization)

A primeira etapa do processo é a **esqueletização** da imagem binária para reduzi-la até uma versão esquelética que preserve a estrutura e as características mais importantes da forma. Isso é feito por meio de operações de **erosão** e **abertura**.

A **erosão** é aplicada repetidamente à imagem binária, removendo gradualmente os pixels das bordas. Em seguida, a **abertura**, que é uma erosão seguida de dilatação, é realizada para remover pequenos detalhes da imagem e preservar as formas mais consistentes.

Esse processo continua iterativamente até que a imagem seja reduzida ao esqueleto desejado. O esqueleto final é obtido pela acumulação das bordas removidas em cada iteração de erosão.

Erosão

A erosão é uma operação morfológica que "reduz" as áreas brancas de uma imagem binária.

A função `erosao()` percorre a imagem pixel por pixel e aplica um elemento estruturante (kernel). Para cada posição, verifica se a região do kernel sobre a imagem atende ao critério (todos os pixels dentro do kernel devem ser brancos para que o pixel correspondente seja mantido). Se a condição for atendida, o pixel correspondente na imagem resultante é mantido

```
#função de erosão
def erosao(imagem, elemento_estruturante):
    #tamanho do kernel
    kernel_height, kernel_width = elemento_estruturante.shape
    pad_height = kernel_height // 2
    pad_width = kernel_width // 2

    #criar a imagem resultante (inicialmente toda preta)
    imagem_erodida = np.zeros_like(imagem, dtype=np.uint8)

    #percorrer a imagem
    for i in range(pad_height, imagem.shape[0] - pad_height):
        for j in range(pad_width, imagem.shape[1] - pad_width):
            #aplicar o elemento estruturante sobre a região da imagem
            region = imagem[i - pad_height:i + pad_height + 1, j -
pad_width:j + pad_width + 1]
            if np.all(region[elemento_estruturante == 1] == 255):
                imagem_erodida[i, j] = 255

    return imagem_erodida
```

Dilatação

A dilatação é a operação oposta à erosão e "expande" as áreas brancas de uma imagem binária. Semelhante à erosão, a função `dilatacao()` percorre a imagem e verifica se qualquer pixel dentro da região definida pelo elemento estruturante é branco. Se sim, o pixel correspondente na imagem resultante se torna branco.

```
#função de dilatação
def dilatacao(imagem, elemento_estruturante):
    #tamanho do kernel
    kernel_height, kernel_width = elemento_estruturante.shape
    pad_height = kernel_height // 2
    pad_width = kernel_width // 2

    #criar a imagem resultante (inicialmente toda preta)
    imagem_dilatada = np.zeros_like(imagem, dtype=np.uint8)
```

```
    #percorrer a imagem
    for i in range(pad_height, imagem.shape[0] - pad_height):
        for j in range(pad_width, imagem.shape[1] - pad_width):
            #aplicar o elemento estruturante sobre a região da imagem
            region = imagem[i - pad_height:i + pad_height + 1, j -
pad_width:j + pad_width + 1]
            if np.any(region[elemento_estruturante == 1] == 255):
                imagem_dilatada[i, j] = 255

    return imagem_dilatada
```

Esqueletização

A função `esqueletizacao()` aplica o algoritmo de esqueletização em uma imagem binária, utilizando erosões sucessivas com um elemento estruturante em forma de cruz (3x3). O algoritmo erode a imagem progressivamente, removendo as bordas dos objetos até que reste apenas a linha central de cada objeto (esqueleto). A cada iteração, é feita uma erosão dupla (erosão e abertura), e as bordas removidas são acumuladas no esqueleto. O processo continua até que a imagem não tenha mais pixels brancos significativos, ou seja, até que o esqueleto esteja completo.

```
#função de esqueletonização utilizando as funções manuais de erosão e
dilatação
def esqueletizacao(imagem_binaria):
    #definir o elemento estruturante (kernel) 3x3 (usando cruz 3x3)
    elemento_estruturante = np.array([[0,1,0],
                                      [1,1,1],
                                      [0,1,0]], dtype=np.uint8)

    #criar a imagem de esqueleto (inicialmente toda preta)
    esqueleto = np.zeros_like(imagem_binaria, dtype=np.uint8)

    img = imagem_binaria.copy()

    #continuar até que a imagem esteja completamente vazia
    while np.any(img):  #enquanto houver pelo menos um pixel branco
        erodida = erosao(img, elemento_estruturante)  # erosão manual
        dilatada = dilatacao(erodida, elemento_estruturante)
#dilatação manual

        #subtrair a dilatação da erosão para obter a borda removida
        borda = img - dilatada
        esqueleto = np.bitwise_or(esqueleto, borda)  #acumular bordas
no esqueleto

        img = erodida  #atualizar a imagem para continuar erodindo

    return esqueleto
```

Aplicação da Esqueletização e Exibição do Resultado

Neste bloco, a função de esqueletização é aplicada à imagem binária e o resultado é exibido. A imagem resultante representa o esqueleto da impressão digital, com as características mais relevantes preservadas.

```
#carregar a imagem binária
imagem = imread(r"C:\Users\Wemerson\Downloads\digital.png")

#esqueletizar a imagem binária
esqueleto = esqueletizacao(imagem_binaria)

#exibir o resultado final da esqueletização
plt.figure(figsize=(6, 6))
plt.imshow(esqueleto, cmap="gray")
plt.title("Impressão Digital Esqueletizada",loc="left")
plt.axis("off")
plt.show()
```

II - Poda (Pruning)

Após realizar a esqueletização da imagem binária da impressão digital, aplica-se a **poda** para remover segmentos expúrios ou extremidades, deixando apenas as linhas principais do esqueleto.

A técnica utilizada percorre cada pixel do esqueleto, analisando seus vizinhos em uma vizinhança 3x3. Se o pixel tiver menos de dois vizinhos conectados, ele é removido, pois provavelmente é um segmento indesejado (extremidade ou espúrio).

Etapa de Poda do Esqueleto

A poda ajuda a refinar a estrutura do esqueleto, deixando apenas os componentes principais e eliminando artefatos menores.

A função de poda foi implementada conforme os passos descritos a seguir:

1. **Contagem de vizinhos conectados**: Para cada pixel ativo, contamos quantos vizinhos ao redor estão conectados ao esqueleto.
2. **Remoção de pixels**: Se o número de vizinhos conectados for menor ou igual a 1, o pixel é removido.

O resultado da poda é a imagem com o esqueleto mais limpo e representativo.

```
def poda(imagem_esqueletizada):
    esqueleto_podado = imagem_esqueletizada.copy()  #copiar a imagem
para evitar alterar a original

    #definir a vizinhança 3x3 para analisar os vizinhos ao redor do
pixel
    #1 = pixel está ativo (parte do esqueleto), e 0 = não está.
    vizinhanca = [
        [-1, -1], [-1, 0], [-1, 1],  #vizinhos superiores (diagonais e
acima)
        [ 0, -1], [ 0, 1],           #vizinhos laterais (esquerda e
direita)
        [ 1, -1], [ 1, 0], [ 1, 1]   #vizinhos inferiores (diagonais e
abaixo)
    ]

    #percorrer cada pixel da imagem (exceto bordas)
    for y in range(1, esqueleto_podado.shape[0] - 1):  #ignorar bordas
verticais
        for x in range(1, esqueleto_podado.shape[1] - 1):  #ignorar
bordas horizontais
            if esqueleto_podado[y, x] == 255:  #se o pixel é parte do
esqueleto
                #contar o número de vizinhos conectados
                vizinhos_conectados = sum(
                    esqueleto_podado[y + dy, x + dx] == 255 for dy, dx
in vizinhanca
                )

                #se o pixel tem apenas um vizinho conectado ou menos,
pode ser uma extremidade ou segmento expúrio
                if vizinhos_conectados <= 1:
```

```
                    esqueleto_podado[y, x] = 0  #remover pixel do
esqueleto

    return esqueleto_podado

#aplicar a poda no esqueleto obtido
esqueleto_podado = poda(esqueleto)

#exibir o esqueleto após a poda
plt.figure(figsize=(6,6))
plt.imshow(esqueleto_podado, cmap="gray")
plt.title("Impressão Digital Podada", loc="left")
plt.axis("off")
plt.show()
```

Conclusão

O processo de **esqueletização binária** e **poda** foi implementado com sucesso na imagem da impressão digital. A esqueletização foi realizada utilizando o método de erosão e abertura, enquanto a poda foi aplicada para remover segmentos expúrios e extremidades, deixando apenas as partes essenciais do esqueleto.

Com a poda, foi possível limpar o esqueleto, eliminando artefatos menores e melhorando a precisão da representação da impressão digital.

Este processo pode ser útil para diversas aplicações de reconhecimento de padrões e análise de formas, especialmente em áreas como biometria e reconhecimento de impressões digitais.

A tarefa foi concluída com e o resultado final é uma versão refinada e limpa do esqueleto da impressão digital.

Autor: [Wemerson Soares]