63o-bin195161ria-e-poda-wemerson

February 10, 2025

1 Esqueletização e Poda de Imagem Binária

1.1 Atividade Pontuada 03 - Processamento de Imagens

1.1.1 I - Introdução

Este trabalho tem como objetivo implementar o processo de esqueletização (skeletonization) binária e o algoritmo de poda (pruning) para refinamento do resultado, conforme o método proposto por Gonzalez & Woods em seu livro "Digital Image Processing". A técnica de esqueletização é aplicada a uma imagem binária de uma impressão digital, utilizando operações morfológicas implementadas manualmente como erosões e aberturas para reduzir a imagem às suas estruturas esqueléticas.

A implementação segue as abordagens descritas por Gonzalez & Woods, aplicando o conceito de erosões e dilatacões para iterativamente remover pixels das bordas da imagem, preservando as características gerais da forma. O algoritmo de poda é então utilizado para refinar o esqueleto, eliminando ramificações desnecessárias e melhorando a precisão do resultado final.

Imagem Original

[1]:
```
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()
```

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.

[2]:
```
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.]
```

1.1.2 II - Esqueletização (*Skeletonization*)

A **esqueletização** reduz a imagem binária a um esqueleto, preservando suas formas principais. Isso é feito através de **erosão sucessiva**, onde a imagem é erodida repetidamente, removendo pixels das bordas. Após cada erosão, aplica-se a **dilatação** para capturar o esqueleto, com a diferença entre a imagem original e a dilatação da erosão.

Antes da esqueletonização, são realizadas operações de **abertura** (erosão seguida de dilatação) e **fechamento** (dilatação seguida de erosão) para suavizar e remover detalhes irrelevantes. O processo é repetido até que a imagem atinja seu esqueleto final.

Erosão (A⊖B) A erosão remove pixels das bordas das formas brancas da imagem. A função realiza isso verificando, para cada pixel da imagem, se o elemento estruturante se ajusta à região ao redor dele. Se a região for completamente compatível com o elemento estruturante (no caso, se todos os pixels de valor 1 coincidirem com o valor do elemento), o pixel central é mantido.

[3]:
```
#função de erosão (A⊖B): erosão de uma imagem A com o elemento estruturante B
def erosao(imagem, elemento_estruturante):
    m, n = imagem.shape  #dimensões da imagem
    h, w = elemento_estruturante.shape  #dimensões do elemento estruturante
    resultado = np.zeros_like(imagem)  #imagem de saída
    pad_h, pad_w = h // 2, w // 2  #padding para bordas

    #iteração sobre a imagem (ignorando as bordas)
    for i in range(pad_h, m - pad_h):
        for j in range(pad_w, n - pad_w):
            region = imagem[i - pad_h:i + pad_h + 1, j - pad_w:j + pad_w + 1] #região da imagem ao redor do pixel (i, j)

            #a erosão só mantém o pixel se a região for totalmente compatível com o elemento estruturante
            if np.all(region[elemento_estruturante == 1] == 1):
                resultado[i, j] = 1
    return resultado  #retorna a imagem erodida
```

Dilatação (A⊕B) A dilatação é o oposto da erosão. Ela expande as regiões brancas na imagem. Para cada pixel da imagem, a dilatação mantém o pixel central se algum pixel da região ao redor (definida pelo elemento estruturante) for branco. Assim, as formas da imagem se expandem.

[4]:
```
#função de dilatação (A⊕B): dilatação de uma imagem A com o elemento estruturante B
def dilatacao(imagem, elemento_estruturante):
    m, n = imagem.shape  #dimensões da imagem
    h, w = elemento_estruturante.shape  #dimensões do elemento estruturante
    resultado = np.zeros_like(imagem)  #imagem de saída
    pad_h, pad_w = h // 2, w // 2  #padding para bordas

    #iteração sobre a imagem (ignorando as bordas)
    for i in range(pad_h, m - pad_h):
        for j in range(pad_w, n - pad_w):
            region = imagem[i - pad_h:i + pad_h + 1, j - pad_w:j + pad_w + 1] #região da imagem ao redor do pixel (i, j)

            #a dilatação acontece se pelo menos um pixel da região corresponder ao elemento estruturante
            if np.any(region[elemento_estruturante == 1] == 1):
                resultado[i, j] = 1
    return resultado  #retorna a imagem dilatada
```

Abertura A abertura é uma operação composta pela erosão seguida da dilatação. Ela é útil para remover pequenos ruídos ou detalhes finos da imagem. Basicamente, ela primeiro reduz a imagem (erosão) e depois expande as regiões remanescentes (dilatação).

[5]:
```
#abertura (A⊖B) ⊕B: primeiro a erosão (A⊖B) e depois a dilatação ((A⊖B) ⊕B)
def abertura(imagem, elemento_estruturante):
    return dilatacao(erosao(imagem, elemento_estruturante),
elemento_estruturante)
```

Fechamento O fechamento realiza a operação inversa da abertura: primeiro a dilatação e depois a erosão. Isso ajuda a preencher pequenos buracos ou lacunas nas regiões brancas da imagem.

[6]:
```
#fechamento (A⊕B) ⊖B: primeiro a dilatação (A⊕B) e depois a erosão ((A⊕B) ⊖B)
def fechamento(imagem, elemento_estruturante):
    return erosao(dilatacao(imagem, elemento_estruturante),
elemento_estruturante)
```

Esqueletização A esqueletização é o processo central da morfologia matemática, extraindo a estrutura óssea da imagem. Utilizando erosões sucessivas e comparando a diferença entre a imagem original e a dilatação das erosões, a função vai acumulando os "ossos" da imagem até que toda a estrutura seja extraída. A cada iteração, a imagem é erodida e dilatada, e o esqueleto é atualizado.

[7]:
```
#função de esqueletização (usando erosões sucessivas)
def esqueletizacao(imagem, elemento_estruturante, max_k=10):
    A = imagem.copy()

    #limpeza com abertura e fechamento (operação preparatória)
    A = abertura(A, elemento_estruturante)    #abertura para remoção de ruído
    A = fechamento(A, elemento_estruturante)    #fechamento para suavizar os
limites

    #criar a imagem de esqueleto (inicialmente toda preta)
    esqueleto = np.zeros_like(imagem)

    for k in range(max_k):
        #erosão sucessiva de A (A ⊖B) repetido k vezes
        A_erosao = erosao(A, elemento_estruturante)

        #dilatação da erosão (A ⊖B) ⊕B
        A_dilatacao = dilatacao(A_erosao, elemento_estruturante)

        #calcular a diferença entre a imagem original e a dilatação da erosão
para obter o esqueleto (miolo)
        esqueleto_k = A - A_dilatacao    # Sk(A) = (A ⊖B) - (A ⊖B) ⊕B
```

```
        #acumular o esqueleto (união dos esqueletos parciais)
        esqueleto = np.bitwise_or(esqueleto, esqueleto_k)

        #atualizar A para a próxima iteração (acumula a erosão sucessiva)
        A = A_erosao

        #se a imagem se tornar completamente vazia, parar
        if np.all(A == 0):
            break

    return esqueleto
```

Exibição e Execução Após aplicar o processo de esqueletização, o resultado é uma versão simplificada e esquelética da imagem original. A imagem resultante mostra apenas o esqueleto central dos objetos presentes na imagem, permitindo visualizar suas formas e estruturas principais, enquanto elimina detalhes e preenchimentos desnecessários.

[8]:
```
#elemento estruturante (kernel 3x3 em forma de cruz)
elemento_estruturante = np.array([[0, 1, 0],
                                  [1, 1, 1],
                                  [0, 1, 0]], dtype=np.uint8)

#realizar a esqueletização
imagem_esqueleto = esqueletizacao(imagem.astype(np.uint8),
 elemento_estruturante)

#exibir o resultado da esqueletização
plt.figure(figsize=(6, 6))
plt.imshow(imagem_esqueleto.astype(np.uint8), cmap="gray") #tipo uint8 para
 exibição
plt.title("Imagem Esqueleto", loc="left")
plt.axis("off")
plt.show()
```

1.1.3 III - Poda (*Pruning*)

A poda é um processo utilizado para remover ramificações espúrias do esqueleto da imagem. Após a esqueletização, pequenos segmentos podem permanecer conectados, mas não contribuem para a estrutura principal do objeto. O objetivo da poda é remover essas ramificações, preservando apenas as partes mais significativas do esqueleto.

A técnica se baseia na identificação e remoção de pixels de extremidade, ou seja, aqueles que possuem um único vizinho conectado. O algoritmo percorre a imagem iterativamente, removendo esses pixels até que não haja mais mudanças ou até atingir um número máximo de iterações.

Poda A poda recebe como entrada a imagem esqueletizada e um elemento estruturante, e realiza um número limitado de iterações para remover ramificações supérfluas. Esse processo ajuda a refinar a estrutura do esqueleto, deixando apenas os componentes principais e eliminando artefatos menores.

Funcionamento da Poda:

1. Percorre a Imagem: A função varre a imagem pixel por pixel, identificando pontos que pertencem ao esqueleto.
2. Identifica Extremidades: Um pixel do esqueleto com apenas um vizinho é considerado um ponto de extremidade e pode ser removido.
3. Iteração Controlada: O processo se repete por um número máximo de iterações (max_iter), garantindo que a poda não seja excessiva.

4. Critério de Parada: Se uma iteração não fizer mais mudanças na imagem, o processo é interrompido.
5. Resultado Final: A imagem podada mantém apenas as estruturas centrais mais significativas do esqueleto.

[9]:
```python
#função de poda (pruning)
def poda(imagem_esqueletizada, elemento_estruturante, max_iter=3):
    esqueleto_podado = imagem_esqueletizada.copy()

    for _ in range(max_iter):
        esqueleto_temp = esqueleto_podado.copy()    #criar uma cópia da imagem atual para verificar alterações

        #vizinhança 3x3
        vizinhanca = [[-1, -1], [-1, 0], [-1, 1],
                      [ 0, -1], [ 0, 1],
                      [ 1, -1], [ 1, 0], [ 1, 1]]

        #percorre cada pixel da imagem
        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] == 1:    #se o pixel é parte do esqueleto
                    #contar o número de vizinhos conectados
                    vizinhos_conectados = sum(
                        esqueleto_podado[y + dy, x + dx] == 1 for dy, dx in vizinhanca
                    )

                    #se o pixel tem apenas um vizinho conectado, pode ser uma extremidade
                    if vizinhos_conectados <= 1:
                        esqueleto_temp[y, x] = 0    #remover o pixel

        #verificar se houve alguma alteração. Se não houver, interromper a poda
        if np.array_equal(esqueleto_temp, esqueleto_podado):
            break    #se a imagem não mudou, parar a iteração

        #atualizar a imagem esqueleto podada
        esqueleto_podado = esqueleto_temp

    return esqueleto_podado
```

Execução e Exibição do Resultado A poda refina o esqueleto, removendo detalhes irrelevantes e deixando apenas a estrutura principal. O resultado é uma versão mais limpa e simplificada do

esqueleto, facilitando sua interpretação e aplicação em análise de formas.

[10]:
```
#aplicar a poda ao esqueleto obtido
esqueleto_podado = poda(imagem_esqueleto, elemento_estruturante, max_iter=5)

#exibir o resultado final da poda
plt.figure(figsize=(6, 6))
plt.imshow(esqueleto_podado, cmap='gray')
plt.title("Esqueleto Final Após Poda", loc="left")
plt.axis('off')
plt.show()
```

1.1.4 IV - Conclusão

A **esqueletização** e a **poda** são técnicas utilizadas para extrair e refinar a estrutura central de objetos em imagens binárias. A esqueletização aplica operações morfológicas, como erosão e dilatação, para obter o esqueleto da imagem. Em seguida, a poda é realizada para remover ramificações e detalhes irrelevantes, resultando em uma representação mais limpa e simplificada do objeto. Esse processo pode ser utilizado para aplicações em visão computacional, como reconhecimento de padrões e segmentação de imagens.

Wemerson Soares | 202300084020