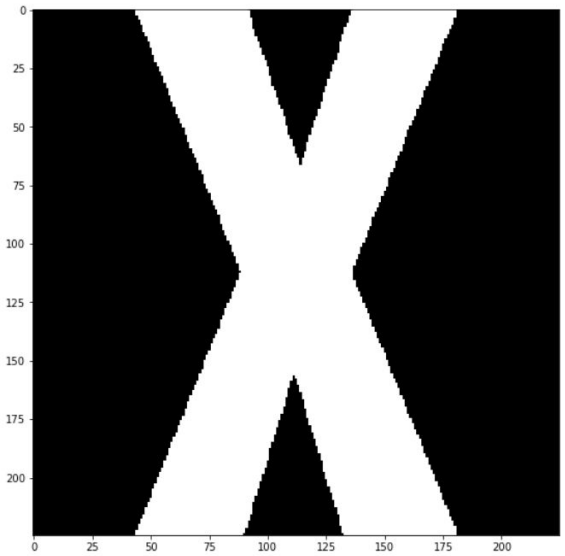
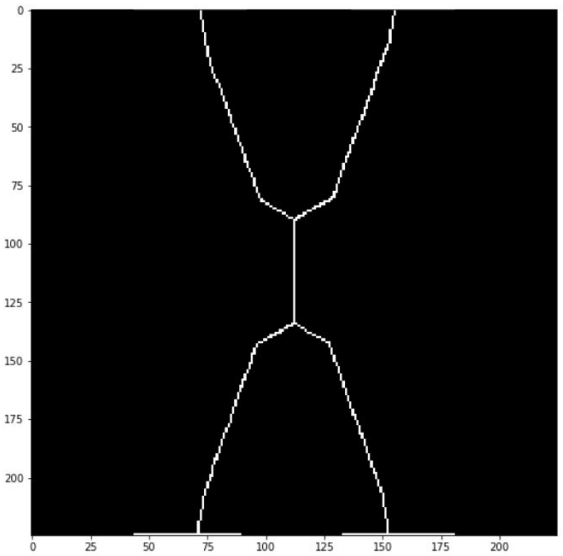
Trabalho 3 - Tópicos em Informática 11
Tema: Esqueletonização
Membros:
Guilherme Nishi Kanashiro, RA: 628298
Leonardo Utida Alcântara, RA: 628182
Rodolfo Krambeck Asbahr, RA: 628042
Tiago Bachiega de Almeida, RA: 628247

Introdução

Na área de processamentos de imagens, uma das subáreas que é muito estudada é o reconhecimento de padrões. Uma das questões no reconhecimento de padrões é a grande quantidade de informações que precisam ser processadas em uma imagem para reconhecer um objeto. Isso motivou pesquisadores no desenvolvimento de algoritmos de esqueletonização.

Os algoritmos de esqueletonização são procedimentos computacionais de processamento de imagens para transformar uma imagem binária num conjunto de pontos presentes no interior do objeto de forma que a topologia do objeto seja representada. Esse conjunto de pontos é o esqueleto da imagem. O principal objetivo da esqueletonização de uma imagem é diminuir a quantidade de informações de uma imagem, reduzindo também as quantidades de informações que precisam ser processadas, sem perder as características topológicas dos objetos presentes na imagem. Na figuras abaixo, mostra-se à esquerda uma imagem binária que contém um objeto e à direita o esqueleto da imagem.





Aplicações

Os esqueletos possuem diversas aplicações na área de processamento de imagem, como reconhecimento de caracteres, a descrição de formas, identificação da posição de pessoas, análise de impressões digitais, entre outras. Nas imagens abaixo, mostra-se na esquerda a imagem de caracteres do alfabeto romano e caracteres usados nas línguas orientais, conhecidos como kanjis, e na direita o esqueleto de imagens que contém caracteres, o que diminui a quantidade de informações que precisam ser processadas para reconhecer os caracteres.

Algoritmo

Ao longo dos anos foram desenvolvidos diversos algoritmos para se obter o esqueleto de uma imagem. O algoritmo que foi implementado neste trabalho foi o algoritmo Zhang-Suen de esqueletonização, onde são utilizadas operações morfológicas. O algoritmo faz erosões sucessivas no objeto.

Para cada pixel p1, os pixels em sua vizinhança foram nomeados da seguinte forma:

p_9	p_2	p_3
p_8	p_1	p_4
p_7	p_6	p_5

Na figura abaixo, mostra-se a parte principal do algoritmo implementado. Em cada iteração os pixels são modificados apenas depois que todos os pixels tenham sido checados para remoção. Para isso, foi criada uma lista de pixels que serão removidos, assim os pixels que precisam ser removidos são incluídos nessa lista.

```
def esqueletonizacao(img):
    mudou = True
    esqueleto_img = img

    while mudou:
        mudou = False
        #Passo 1, pega todos os pixels menos os das bordas
        for row in range(1, esqueleto_img.shape[0]-1):
            for col in range(1, esqueleto_img.shape[1]-1):

                posicao = [row,col]
                #a)
                if esqueleto_img[row][col]:
                    #b)
                    if (vizinhancaPixelsNaoNulos(esqueleto_img, posicao) >=2
                        and vizinhancaPixelsNaoNulos(esqueleto_img, posicao) <= 6):
                        #c)
                        if transicaoVizinhanca(esqueleto_img, posicao) == 1:
                            #d)
                            if p2p4p6(esqueleto_img, posicao) == 0:
                                #e)
                                if p4p6p8(esqueleto_img, posicao) == 0:
                                    pixelsASeremRemovidos.append(posicao)
                                    mudou = True

        #Remova todos os pixels da lista
        esqueleto_img = removePixels(esqueleto_img)
```

A ideia do algoritmo é verificar todos os pixels da imagem usando dois passos de verificações. No primeiro, um pixel é removido, ou seja adicionado à lista de remoção, se obedecer às seguintes condições:

a) O pixel tem valor 1(branco, na imagem binária);

b) O número de vizinhos não nulos for maior ou igual a 2 e menor ou igual a 6;

c) Se o número de transições de 0 para 1 for exatamente 1;

d) Se houver pelo menos um pixel nulo entre P2, P4 e P6;

e) Se houver pelo menos um pixel nulo entre P4, P6 e P8.

Note que as condições estão identificadas da mesma forma no código.

Após isso, todos os pixels na lista de remoção são removidos, ou seja, seu valor é alterado para zero(preto, na imagem binária).

```
        #Passo2, pega todos os pixels menos os das bordas
        for row in range(1, esqueleto_img.shape[0]-1):
            for col in range(1, esqueleto_img.shape[1]-1):
                posicao = [row,col]
                #a)
                if esqueleto_img[row][col]:
                    #b)
                    if (vizinhancaPixelsNaoNulos(esqueleto_img, posicao) >=2
                        and vizinhancaPixelsNaoNulos(esqueleto_img, posicao) <= 6):
                        #c)
                        if transicaoVizinhanca(esqueleto_img, posicao) == 1:
                            #d)
                            if p2p4p8(esqueleto_img, posicao) == 0:
                                #e)
                                if p2p6p8(esqueleto_img, posicao) == 0:
                                    pixelsASeremRemovidos.append(posicao)
                                    mudou = True

        #Remova todos os pixels da lista
        esqueleto_img = removePixels(esqueleto_img)

    return esqueleto_img
```

No segundo passo, os pixels são adicionados na lista de remoção se o pixel obedecer às seguintes condições:

a) O pixel tiver valor 1;

b) O número de vizinhos não nulos for maior ou igual a 2 e menor ou igual a 6;

c)Se o número de transições de 0 para 1 for exatamente 1;

d)Se houver pelo menos um pixel nulo entre P2, P4 e P8;

e)Se houver pelo menos um pixel nulo entre P2, P6 e P8.

As condições no passo dois também estão identificadas da mesma forma no código.

Feito isso, passar todos os pixels pelo passo 2, remova-os(altere o valor para zero). Esse processo é repetido até que nenhum pixel seja removido.