UNIVERSIDADE FEEVALE

Lucas Batista, Josué E. E. Feldeckircher,
Murilo Tappar, Ricardo Alvarenga, Tais Baierle e Thauan Godoy

AFINAMENTO DE ZHANG-SUEN

Novo Hamburgo - RS
2020

1. INTRODUÇÃO

Em processamento de imagens a técnica da esqueletização é utilizada para obter o esqueleto de uma objeto através do afinamento. O afinamento nada mais é do que a redução de uma forma para uma versão simplificada da mesma, mas que ainda mantém suas características originais. Os esqueletos possuem várias aplicações na área de processamento de imagens, tais como, agrupamento, segmentação, vetorização, descrição de formas, reconhecimento de caracteres, inspeção, etc.

O afinamento consiste em verificar sucessivamente as bordas dos objetos (contornos), onde cada ponto da borda é atribuído ao fundo ou ao esqueleto. As várias bordas (contornos) dos objetos são processados geralmente sem perda de conexidade, ou seja, o esqueleto formado geralmente se mantém conexo. Para que a esqueletização seja dita efetiva no objeto, deve seguir três regras: 1) Remover pontos extremos; 2) Quebrar a conectividade[1]; e 3) Causar a erosão excessiva da região.

O algoritmo de Zhang Suen consiste na aplicação sucessiva de duas etapas. Na primeira etapa um pixel é eliminado, ou marcado para ser eliminado em outro momento se as seguintes condições se cumprirem:

1. O número de conectividade é 1;
2. Existem ao menos dois pixels vizinhos pretos, e menos do que sete, sendo que, vizinhos pretos de um ponto são todos os pontos pretos presentes nas 8 direções possíveis a partir do ponto corrente (pixels de P2 a P9);

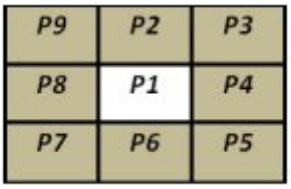

Figura 1: Quantidade de vizinhos pretos.

3. Ao menos um dos pixels P2, P4 ou P6 são brancos;

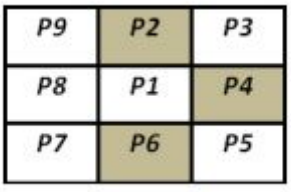

Figura 2: Vizinhos brancos.

4. Ao menos um dos pixels P4, P6 ou P8 são brancos:

[1] Conectividade é um dos conceitos mais importantes no processo de afinamento de uma imagem. Ao deletar os pixels, a conectividade da imagem deve sempre ser preservada, ou seja, deve haver ao menos um pixel que ligue duas regiões distintas quaisquer.

P9	P2	P3
P8	P1	P4
P7	P6	P5

Figura 3: Vizinhos brancos.

No final dessa etapa os pixels marcados são eliminados. Na segunda etapa, as condições 1 e 2 permanecem as mesmas, mas a 4 e a 3 assumem os seguinte critérios:

3. Ao menos um dos pixels P2, P4 ou P8 são brancos:

P9	P2	P3
P8	P1	P4
P7	P6	P5

Figura 4 : Vizinhos brancos.

4. Ao menos um dos pixels P2, P6 ou P8 são brancos:

P9	P2	P3
P8	P1	P4
P7	P6	P5

Figura 5 : Vizinhos brancos.

Após isso, novamente os pixels são marcados e eliminados. A primeira etapa deve ser aplicada a todos os pixels da borda, se as condições forem satisfeitas, o pixel é marcado e eliminado no final da primeira etapa. O mesmo deve ser feito com a segunda etapa. O algoritmo é aplicado até que não existam mais pixels a serem removidos.

Após o término da execução do afinamento, a imagem esperada é algo como mostrado no exemplo abaixo:

Figura 6 : Imagem Afinada.

5. METODOLOGIA

Segue abaixo a aplicação do afinamento de bordas em português estruturado:

```
Var
img: vetor[0..24,0..10] de inteiro
imgResultado: vetor[0..24,0..10] de inteiro
i, x, y, mudou, tmp, estado, proximo: inteiro

Início

//CRIA VETOR VAZIO
PARA y<-0 ATE 9 FACA
   PARA x<-0 ATÉ 24 FACA
   img[x,y]<-0
   FIMPARA
FIMPARA

//CRIA QUADRADO
PARA y<-1 ATÉ 8 FACA
   PARA x<-5 ATÉ 19 FACA
   img[x,y]<-1
   FIMPARA
FIMPARA

//CRIA DESENHO
PARA y<-4 ATE 5 FACA
   PARA x<-10 ATÉ 14 FACA
   img[x,y]<-0
   FIMPARA
FIMPARA

//ALGORITMO ZHANG-SUEN
estado<-1
PARA y<-1 ATÉ 8 FACA
   PARA x<-1 ATÉ 23 FACA
        ENQUANTO proximo<=0 FACA
           SE img[x,y]=0 ENTÃO  //verifica se o p1 é 0, se sim passa para próx
              imgResultado[x,y]<-0
              proximo<-1
           FIMSE
           tmp<-0
           tmp<-img[x,y-1]+ img[x+1,y-1] //soma posições 2 e 3
           tmp<-tmp + img[x+1,y  ] + img[x+1,y+1] + img[x  ,y+1] //soma posições 4, 5 e 6
           tmp<-tmp + img[x-1,y+1] + img[x-1,y  ] + img[x-1,y-1] //soma posições 7, 8 e 9
           SE tmp = 2 ENTAO  //verifica ligações, se for menor que 2 ou maior que 7, vira 1
              imgResultado[x,y]<-1
              proximo<-1
           SENAO
                imgResultado[x,y]<-0
```

```
            FIMSE
            tmp<-0
            //verifica vizinhança
            SE (img[x  ,y  ]=0) E (img[x  ,y-1]=1) ENTÃO  //p1 e p2
                 tmp<- tmp + 1
            FIMSE
            SE (img[x  ,y-1]=0) E (img[x+1,y-1]=1) ENTÃO  //p2 e p3
                 tmp<- tmp + 1
            FIMSE
            SE (img[x+1,y-1]=0) E (img[x+1,y  ]=1) ENTÃO   //p3 e p4
                 tmp<- tmp + 1
            FIMSE
            SE (img[x+1,y  ]=0) E (img[x+1,y+1]=1) ENTAO   //p4 e p5
                 tmp<- tmp + 1
            FIMSE
            SE (img[x+1,y+1]=0) E (img[x  ,y+1]=1) ENTÃO   //p5 e p6
                 tmp<- tmp + 1
            FIMSE
            SE (img[x  ,y+1]=0) E (img[x-1,y+1]=1) ENTÃO   //p6 e p7
                 tmp<- tmp + 1
            FIMSE
            SE (img[x-1,y+1]=0) E (img[x-1,y  ]=1) ENTÃO   //p7 e p8
                 tmp<- tmp + 1
            FIMSE
            SE (img[x-1,y  ]=0) E (img[x-1,y-1]=1) ENTÃO   //p8 e p9
                 tmp<- tmp + 1
            FIMSE
            SE (img[x-1,y-1]=0) E (img[x  ,y-1]=1) ENTÃO   //p9 e p2
                 tmp<- tmp + 1
            FIMSE

            SE tmp > 1 ENTAO
                 imgResultado[x,y]<-0
                 proximo<-1
            FIMSE
            //verifica se há continuidade

          SE (img[x  , y-1] + img[x+1, y] + img[x, y+1]=3) OU (img[x+1, y] + img[x  , y+1] +
img[x-1, y  ]=3) ENTAO
             imgResultado[x,y]<-1
             proximo<-1
          FIMSE

          SE (img[x  , y-1] + img[x+1, y  ] + img[x-1, y]=3) OU (img[x  , y-1] + img[x  ,
y+1] + img[x-1, y]=3) ENTAO
             imgResultado[x,y]<-1
             proximo<-1
          FIMSE
        FIMENQUANTO
        proximo<-0
   FIMPARA
```

```
FIMPARA

//PRINT VETOR BINÁRIO
PARA y<-0 ATE 9 FACA
   PARA x<-0 ATÉ 24 FACA
        ESCREVA(img[x,y])
   FIMPARA
   ESCREVAL
FIMPARA

ESCREVAL

//PRINT VETOR BINÁRIO C CARACTERES ALTERADOS
PARA y<-0 ATE 9 FACA
   PARA x<-0 ATE 24 FACA
        SE img[x,y]=0 ENTAO
        ESCREVA (".")
        SENAO
        ESCREVA ("#")
        FIMSE
   FIMPARA
   ESCREVAL
FIMPARA
ESCREVAL

//PRINT VETOR RESULTADO C CARACTERES ALTERADOS
PARA y<-0 ATE 9 FACA
   PARA x<-0 ATÉ 24 FACA
        SE imgResultado[x,y]=0 ENTAO
        ESCREVA (".")
        SENAO
        ESCREVA ("#")
        FIMSE
   FIMPARA
   ESCREVAL
FIMPARA

Fimalgoritmo
```

6. DESENVOLVIMENTO DO MÉTODO

A implementação do projeto foi feito em Python 3 com orientação a objetos. Primeiramente foi definida uma interface gráfica para o projeto utilizando os recursos da biblioteca Tinker, para fazer a interpretação de imagens foi utilizada a biblioteca PIL (Pillow).

Num primeiro momento, dentro da classe Afinamento, é feito toda a configuração da tela. Após isso, temos o método que implementa o seletor de arquivos, que permite o usuário buscar um arquivo do disco, após selecionar o

arquivo desejado, o caminho em disco desse arquivo é armazenado para ser exibido em tela e também, é feita a importação para aplicação.

Depois dos métodos de interface gráfica e da funcionalidade de escolher e exibir arquivos, são implementados os métodos de manipulação da imagem propriamente dita. Primeiramente com o método que faz a binarização[2] da imagem pegando a altura e o comprimento da imagem, e depois convertendo ela para binário em que pixels maiores que determinado valor médio assumem o valor um (1) e menores o valor zero (0). Após isso, existe um método que determina os oito vizinhos de um pixel p1 e outro método que estabelece os vizinhos pretos. Segue na sequência o método que estabelece o número de conectividade, onde cada transição de branco para preto é quantificada, pois segundo os dois autores do algoritmo (ZHANG; SUEN, 1984) número de conectividade de um pixel é definido como sendo o número de transições de branco para preto, percorrendo os pixels vizinhos no sentido de P1 a P8 ao pixel em questão;

Por fim, é feito o método que aplica de fato o afinamento de bordas na imagem, onde em cada iteração do while é feita a verificação das condições de exclusão do pixel dentro de um if, caso o pixel atenda todos os requisitos, é colocado um pixel branco no seu lugar. Como descrito na teoria do algoritmo, são feitas duas iterações com suas regras definidas de exclusão do pixel. O laço de repetição executa até que nenhum pixel atenda às condições de exclusão. Feito isso o método devolve uma imagem editada.

O núcleo do código fonte que de fato implementa com afinamento de bordas Zhang Suen pode ser visto no seguinte trecho de código:

```
def zhangSuen(self):
    width = self.img.width
    height = self.img.height
    new = self.binaria(self.img, 123)
    img = new.load()

    iteracao1 = iteracao2 = [(-1, -1)]
    while iteracao1 or iteracao2:
        iteracao1 = []
        for x in range(1, width - 1):
            for y in range(1, height - 1):
                if img[x, y] == 0:
                    vizinhos = P2, P3, P4, P5, P6, P7, P8, P9 = self.vizinhos(x, y, img)
                    if (self.conectividade(vizinhos) == 1
                        and 2 <= self.qtdeVizinhosPretos(vizinhos) < 7
                        and (P2 == PIXEL_BRANCO or P4 == PIXEL_BRANCO or P6 == PIXEL_BRANCO)
                        and (P4 == PIXEL_BRANCO or P6 == PIXEL_BRANCO or P8 == PIXEL_BRANCO)
                    ):
```

[2] Binarização: Binarizar uma imagem significa transformar uma imagem em tons de cinza em uma imagem binária. Verifica-se a intensidade dos pixels da imagem para saber se ele receberá o valor de branco ou preto (0 ou 1), essa decisão é baseada num valor pré-definido chamado de threshold ou limiar. Então o valor do pixel é comparado ao valor do threshold, se for menor, recebe o valor branco e se for maior o valor preto.

```
                    iteracao1.append((x, y))
        for x, y in iteracao1:
            img[x, y] = 1

        iteracao2 = []
        for x in range(1, width - 1):
            for y in range(1, height - 1):
                if img[x, y] == 0:
                    vizinhos = P2, P3, P4, P5, P6, P7, P8, P9 = self.vizinhos(x, y, img)
                    if(self.conectividade(vizinhos) == 1
                        and 2 <= self.qtdeVizinhosPretos(vizinhos) <= 6
                        and (P2 == PIXEL_BRANCO or P4 == PIXEL_BRANCO or P8 == PIXEL_BRANCO)
                        and (P2 == PIXEL_BRANCO or P6 == PIXEL_BRANCO or P8 == PIXEL_BRANCO)
                    ):
                        iteracao2.append((x, y))
        for x, y in iteracao2:
            img[x, y] = 1
    self.novaImg = new.convert('RGB')
    self.photoNew = ImageTk.PhotoImage(self.novaImg)
    self.imgEditada['image'] = self.photoNew
```

Neste método a imagem é recebida pela referência da classe Afinamento, feito isso, é feito um laço while, que contém as verificações das condições da primeira iteração, onde é feito um levantamento dos vizinhos do pixel analisado, após isso no if é feito a verificação do primeiro conjunto de condições do afinamento de zhang suen. Atendendo a todos os requisitos, esse pixel ganha a coloração branca, ou seja, eliminado visualmente. Na segunda iteração, também é feito o levantamento dos vizinhos, e no if o segundo conjunto de verificações do afinamento é verificado. Atendendo a todos os requisitos, esse pixel ganha a coloração branca, ou seja, eliminado visualmente.

7. CONCLUSÃO

A aplicação do algoritmo de Zhang Suen pode ser um pouco mais demorada em função das iterações pixel a pixel. Outra situação percebida nos testes feitos foi que algumas imagens afinadas com Zhang Suen apresentam distorções em relação a imagem final, resultado avaliado na imagem abaixo:

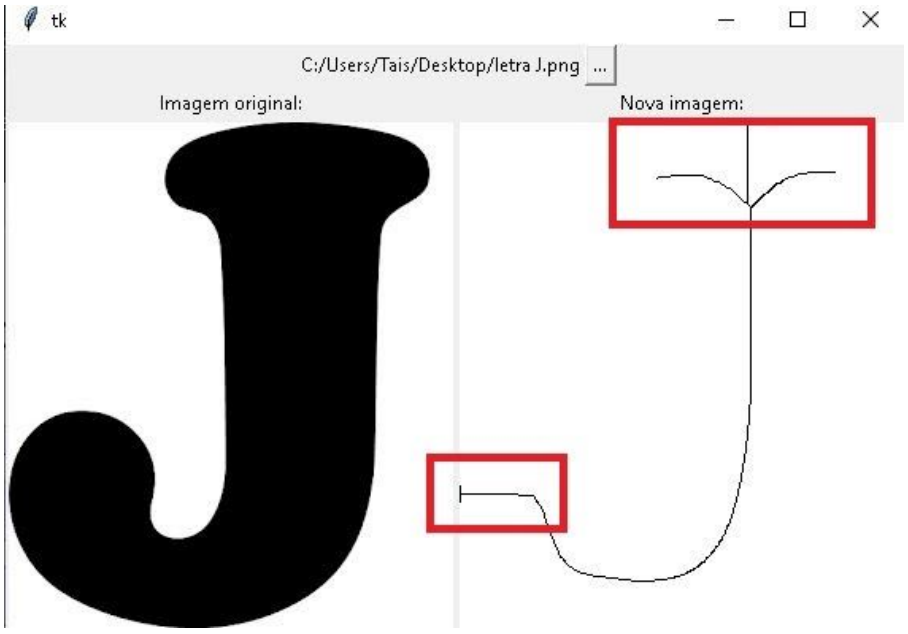

Figura 7: Distorções no afinamento

Outra questão observada na prática é que em algumas figuras que apresentam curvas, pode ocorrer um efeito escada, chamado de “staircase”, ou seja um efeito escada, que segue no exemplo abaixo:

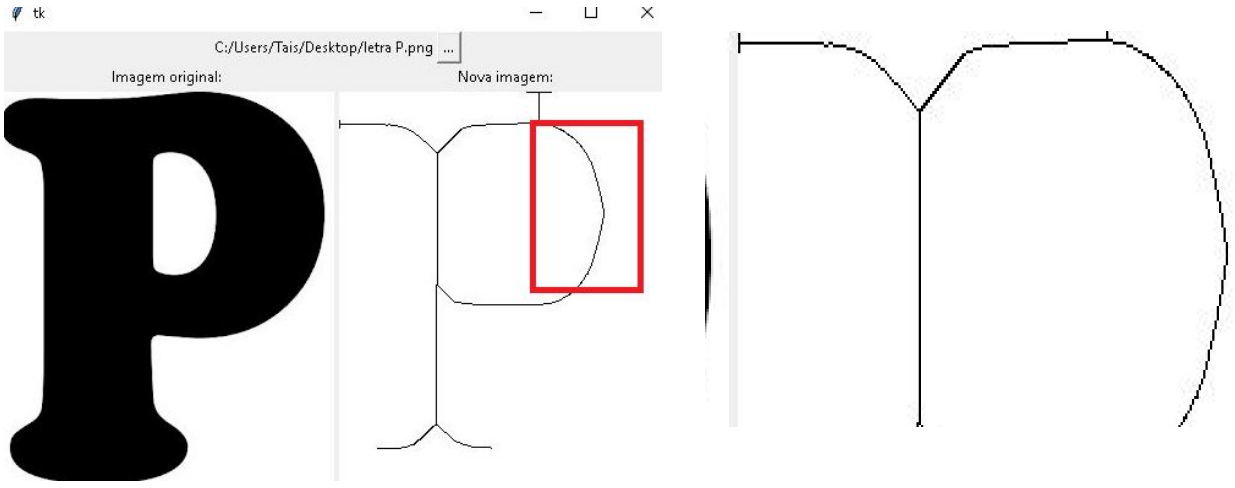

Figura 8: Efeito “Staircase” no afinamento

Apresenta em algumas imagens perdas da morfologia, como no caso do teste feito na imagem abaixo, onde o esqueleto não reflete a imagem original.

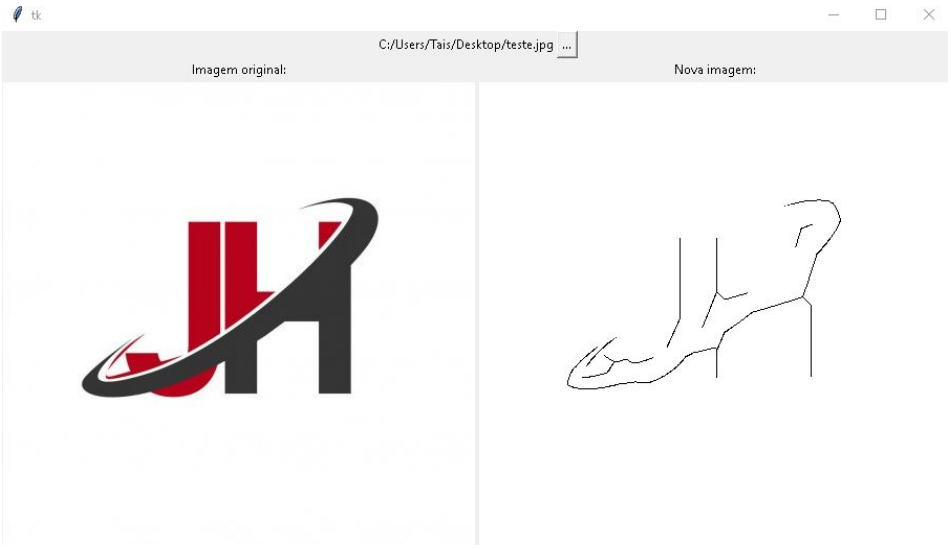

Figura 9: Perda da morfologia.

Por outro lado, imagens foram submetidas obtendo um resultado muito satisfatório, sem perda da conectividade e mantendo as características da imagem original.

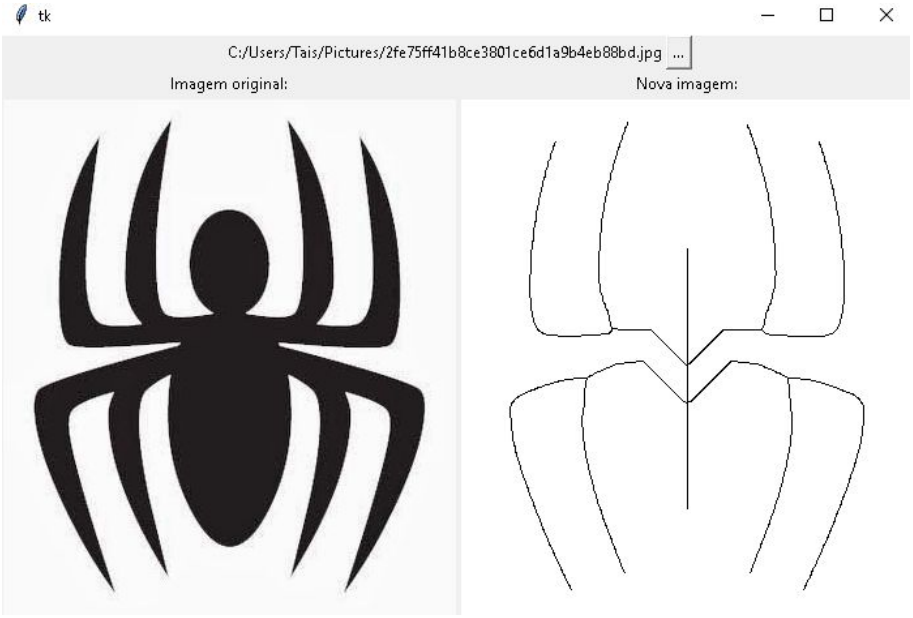

Figura 10: Afinamento com resultado satisfatório

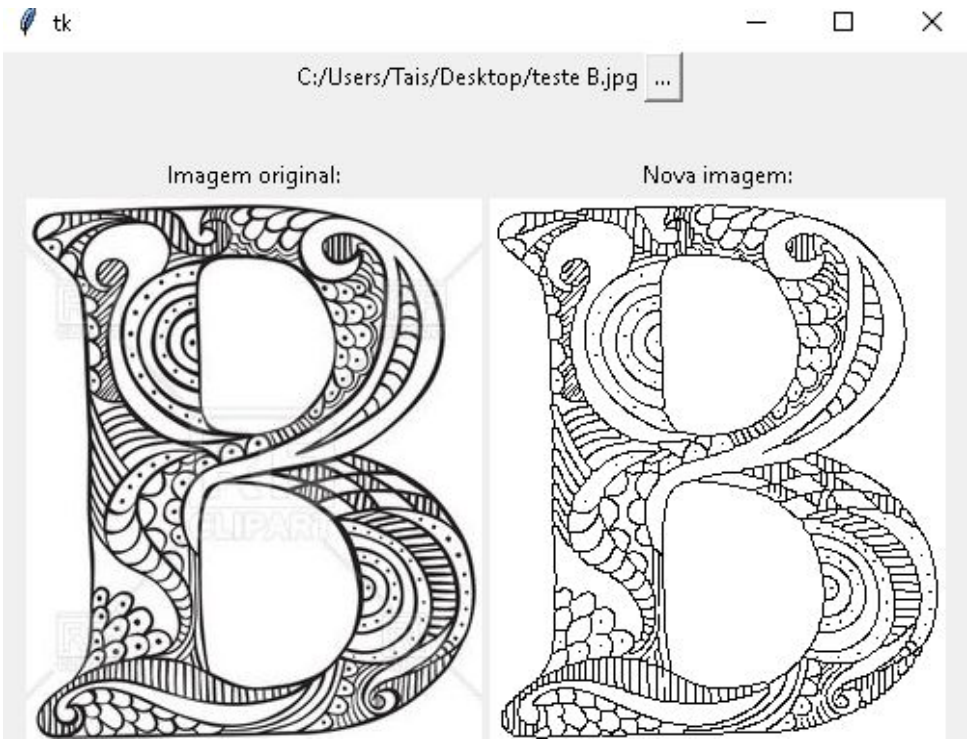

Figura 11: Afinamento com resultado satisfatório

Após os testes realizados, foi verificado que o afinamento de bordas de Zhang Suen possui limitações, e que dependendo da imagem submetida não apresenta um esqueleto de qualidade, portanto como todos os algoritmos de afinamento existentes, deve se observar a imagem que vai ser submetida e escolher o método mais adequado.

8. REFERÊNCIAS BIBLIOGRÁFICAS

Galvanin, E. A. S., Vale, G. M., Dal Poz, A. P e Telles, S. S. S. **DETECÇÃO E AFINAMENTO DE BORDAS UTILIZANDO SUAVIZAÇÃO ANISOTRÓPICA E ESQUELETIZAÇÃO,** IV Colóquio Brasileiro de Ciências Geodésicas - IV CBCG, Curitiba, 16 a 20 de maio de 2005.

R. O. Plotze, e O. M. Bruno, **Estudo e comparação de algoritmos de esqueletonização para imagens binárias**, IV Congresso Brasileiro de Computação – CBComp 2004.

CORRÊA, Fernando Porto, FESTA, Leidmar Magnus, **Avaliação de técnicas para afinamento de imagens digitais**, Curitiba, 2005, Universidade Federal do Paraná;

GUILHERME, Luis Renato Woiski, **Uma abordagem de afinamento por aprendizagem através de exemplos**, Curitiba 2007, a Pontifíicia Universidade Católica do Paraná;

PRATES, Jorge Marques, **Algoritmo de Thinning e suas aplicações**, Presidente Prudente 2011, Universidade Estadual Paulista Júlio de Mesquita Filho.