Notas Técnicas

CBPF-NT-007/97

Dezembro 1997

Aplicação do Processamento Digital de Imagens na Detecção de Bordas em Imagens de Domínios Magnéticos: Algorítmo -2+4

Flávio Luis de Mello e Marcio Portes de Albuquerque

Flávio Luis de Mello
Fmello@cat.cbpf.br

Marcio Portes de Albuquerque
Mpa@cat.cbpf.br

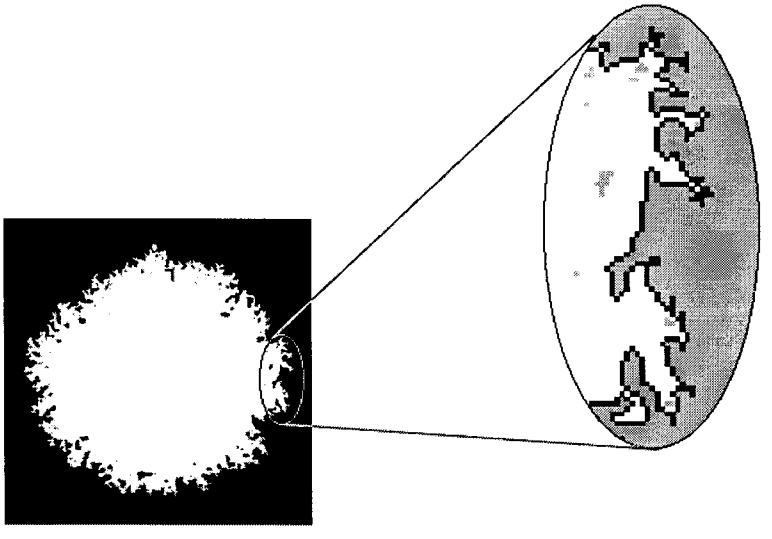
Resumo

Esta nota técnica tem a finalidade de apresentar um algoritmo de detecção de borda implementado no CBPF. A motivação deste trabalho surgiu com a necessidade de estudar a evolução das bordas de domínios magnéticos. O trabalho se propõem a fornecer informações básicas sobre o processamento de imagens para compreender o funcionamento do algoritmo.

Inicialmente são apresentados os métodos convencionais de detecção de borda. Eles são explicados de maneira sucinta e também são apresentados os resultados obtidos através da utilização dos mesmos. Estes métodos são aplicados na imagem que é o objeto de estudo e os resultados são avaliados.

Em seguida, apresenta-se um algoritmo para a detecção de borda de imagens de domínios magnéticos. É feita a documentação do funcionamento do mesmo e são realizados alguns exemplos do que é feito a cada passo do programa.

Por fim, encontra-se a listagem do programa em Pascal. Este programa constitui uma aplicação desenvolvida em um software de processamento de imagens conhecido como NIH-Image. Este software é de domínio público e trabalha na plataforma Macintosh.



Introdução

O interesse em aplicações de processamento de imagem tem crescido paralelamente com a evolução dos computadores digitais. O baixo custo de hardwares dedicados ao processamento de imagens, bem como o alto desempenho dos computadores atuais têm influenciado o crescimento no interesse por aplicações que trabalhem com dados em duas dimensões, isto é, imagens.

Existem vários métodos para se tratar esta espécie de dado. De maneira geral estes métodos atendem bem as necessidades provenientes do processamento. Entretanto não existe um procedimento que pode ser aplicado a todos os tipos de imagens. Na verdade, o que ocorre é que existem diversos procedimentos para executar um mesmo processamento. Cada um destes procedimentos irá executar a tarefa melhor, ou pior que outro análogo, dependendo do tipo de imagem que lhe é fornecida.

Entretanto, o que é esta classificação quanto ao tipo de imagem? As imagens diferenciam-se entre si de acordo com determinadas características, tal como a iluminação de quando ela foi adquirida, o foco, o grau de complexidade das estruturas que nela estão retratadas, e outras mais. A qualidade do processamento está intimamente ligada ao tipo da imagem. Em casos extremos se faz necessário implementar algoritmos dedicados a realizar um determinado processamento em uma imagem específica.

Este trabalho discute uma situação semelhante. A imagem em questão é o resultado de uma simulação da evolução de domínios magnéticos. O que se deseja estudar é a borda destes domínios para se obter alguma informação sobre o grau de rugosidade dos mesmos. Um estudo desta natureza exige que uma identificação de borda dos objetos seja feita. Naturalmente, existem algoritmos para a detecção de borda, porém nenhum deles fornece resultados satisfatórios. O que será apresentado é um algoritmo capaz de resolver o problema.

A proposta do capítulo a seguir é apresentar os aspectos mais importantes para se compreender o estudo da borda de um objeto e que tipo de informação pode-se obter através dela. A fim de se atingir tal objetivo, são apresentadas noções básicas sobre o conjunto de operações a serem realizadas. Os tópicos são descritos de forma a trazer informações importantes como: o funcionamento dos métodos utilizados para detecção de borda, as diferentes abordagens realizadas de acordo com as características de cada imagem e alguns algoritmos alternativos que fornecem resultados surpreendentes a algumas imagens problemáticas.

De fato, a atenção principal está voltada justamente a estes algoritmos alternativos. A literatura clássica traz somente as ferramentas mais utilizadas, dentre elas podemos citar o "thresholding" e os operadores de gradiente. Métodos mais específicos de análise são difíceis de serem encontrados. Assim, apresenta-se nos capítulos subseqüentes um algoritmo chamado "menos dois mais quatro" (-2+4), excelente para detecção de bordas de objetos maciços e cheios de reentranças. Será feito o estudo de caso utilizando imagens de domínios magnéticos recentemente produzidas através de simulação em computadores do CBPF.

Existem diversos conceitos importantes para realizar o tratamento de alguma imagem, porém o objetivo deste trabalho não é compreender todos eles. Todavia é necessário introduzir um conceito bastante comum em processamento de imagens chamado segmentação.

Segmentação é um processo que subdivide uma imagem em partes ou objetos. A segmentação é um dos mais importantes elementos da análise de imagens porque é, neste passo, que objetos ou outras áreas de interesse para o estudo da imagem são extraídos a fim de possibilitar um processamento subseqüente, tal como descrições ou reconhecimentos.

Os algoritmos de segmentação são baseados em duas propriedades básicas dos níveis de cinza de uma imagem: descontinuidade e similaridade. Na primeira categoria, particionamos a imagem, baseados em mudanças abruptas dos níveis de cinza. As principais áreas de interesse, incluídas nesta categoria são a detecção de pontos, de linhas e de bordas de uma imagem. As principais abordagens da segunda categoria são o "thresholding", o crescimento de regiões, a separação e o colapso de células.

As técnicas de detecção de pontos linhas e bordas são facilmente achadas na literatura [Gonzales]. Os métodos mais comuns utilizados na detecção destas propriedades baseiam-se em pequenas máscaras na forma de matrizes 3x3. Apesar da detecção de pontos e linhas ser relevante, a de bordas é significativamente a técnica mais importante. A razão disto é que pontos isolados e linhas muito finas não ocorrem com muita freqüência na maioria das aplicações práticas.

Detecção de Borda

Definimos borda como sendo a fronteira entre duas regiões com valores de níveis de cinza distintos. Assumimos que estas regiões em questão são homogêneas o suficiente para que a transição entre elas possa ser baseada unicamente na descontinuidades dos níveis de cinza.

Basicamente, a idéia da maioria das técnicas de detecção de bordas está fundada no cálculo de uma variação dos valores que um pixel pode assumir em uma determinada imagem. Isto pode ser facilmente compreendido observando a *Fig. 1*. A primeira derivada deste modelo apresentado é 0 em todas as regiões com níveis de cinza constantes e assume um valor constante na transição dos níveis de cinza. A segunda derivada, por outro lado, é 0 em todas as regiões exceto no início e no fim das transições dos níveis de cinza. Baseado nisto, é evidente que a primeira derivada detecta a presença de uma borda, enquanto a segunda derivada pode ser usada para determinar se o pixel está sobre um fundo escuro ou sobre um objeto claro. A primeira derivada de qualquer ponto de uma imagem pode ser obtida avaliando a magnitude do gradiente naquele ponto, enquanto que a segunda derivada é dada por um Laplaciano.

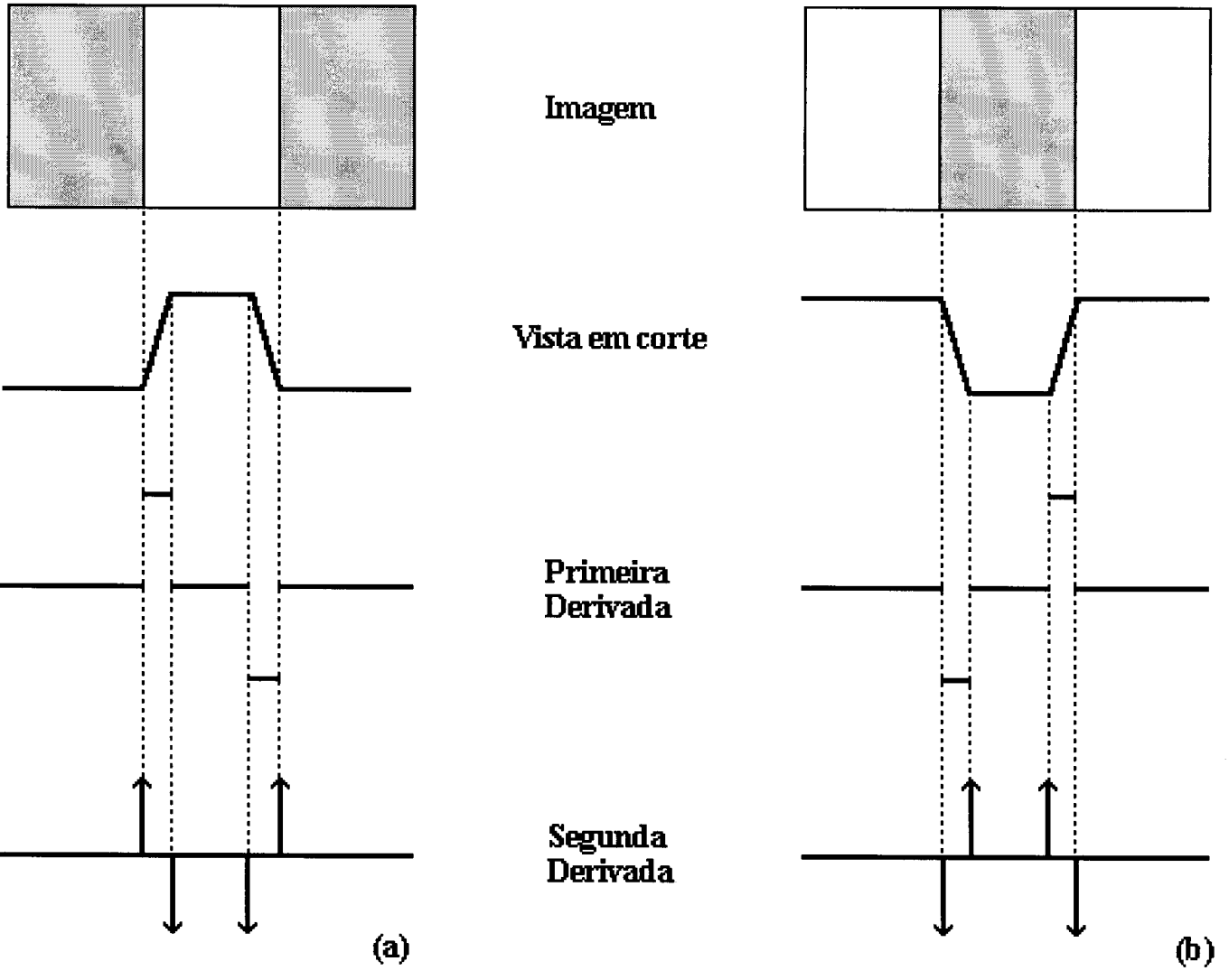


Figura 1 – Detecção de borda usando derivada. (a) Objeto claro em um fundo escuro. (b) Objeto escuro em um fundo claro.

O gradiente ∇f é um vetor definido em pontos quaisquer *a* onde as derivadas parciais $D_1f(a),...,D_nf(a)$ existem. As componentes deste vetor são as derivadas parciais de f em a,

$$\nabla f = (D_1f(a),...,D_nf(a)).$$

Quando aplicado a imagens, este conceito pode ser escrito da seguinte forma: gradiente é a taxa de variação dos valores dos pixels que compõe a imagem em uma dada direção. Isto significa que quando realizamos o gradiente de uma certa imagem, em uma direção qualquer, estamos privilegiando os valores de borda daquela direção. É comum realizarmos o cálculo nas direções X e Y. A implementação deste cálculo é feita utilizando-se máscaras, matrizes 3x3 nas quais são atribuídos valores aos seus elementos. Esses valores constituirão pesos a serem dados para cada pixel da imagem. Considere uma região de uma imagem conforme mostrado na *Fig. 2(a)*, A quadrícula x_5 representa o valor de nível de cinza na posição (x,y) enquanto que os outros x_i representam os valores de nível de cinza dos vizinhos deste pixel. Vamos escolher a máscara da *Fig. 2(b)* por exemplo. Ela é conhecida como operador de Sobel e realiza detecção na direção X. Assim teremos que:

$$G_x = (x_7+2x_8+x_9) - (x_1+2x_2+x_3)$$

x_1	x_2	x_3
x_4	x_5	x_6
x_7	x_8	x_9

(a)

-1	-2	-1
0	0	0
1	2	1

(b)

Figura 2 - (a) Região 3x3 da imagem. (b) Operador de Sobel na direção X.

É possível calcular o gradiente, baseando-se em máscaras maiores, no entanto matrizes 3x3 são mais utilizadas devido a rapidez do cálculo e por não necessitarem de um hardware muito complexo. Há diversas outras máscaras, cada qual com suas características, que podem ser utilizadas para detecção através do gradiente. A Figura a seguir relaciona algumas delas. A *Fig. 4* exibe os resultados obtidos aplicando Sobel na *Fig. 4(a)*.

1	1	1
0	0	0
-1	-1	-1

(a)

-1	0	1
-2	0	2
-1	0	1

(b)

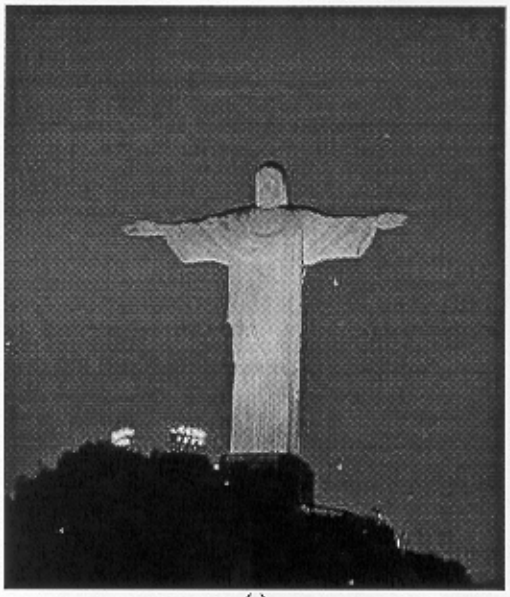
0	0	0
0	-1	1
0	0	0

(c)

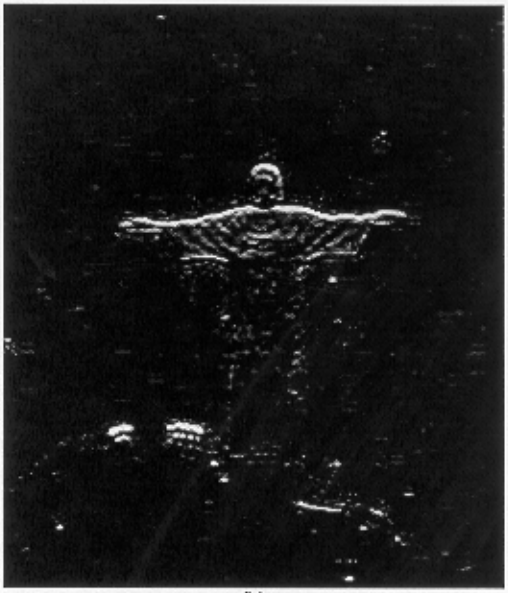
0	0	1
0	-1	0
0	0	0

(d)

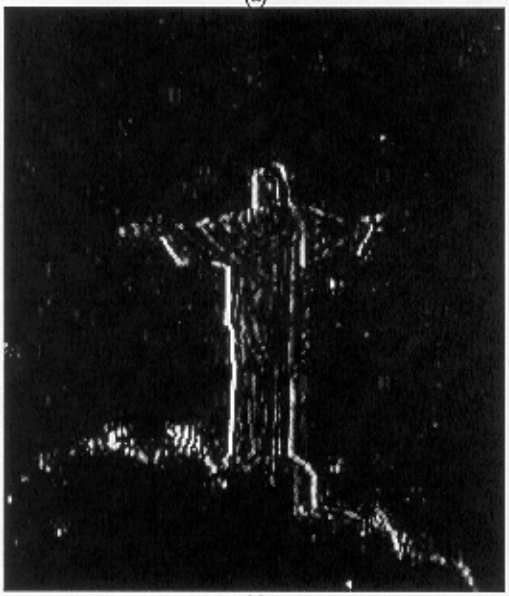
Figura 3 – (a) Máscara de Prewitt na direção X. (b) Máscara de Sobel na direção Y. (c) Máscara de Robert na direção X. (d) Máscara de Robert na direção 45° .



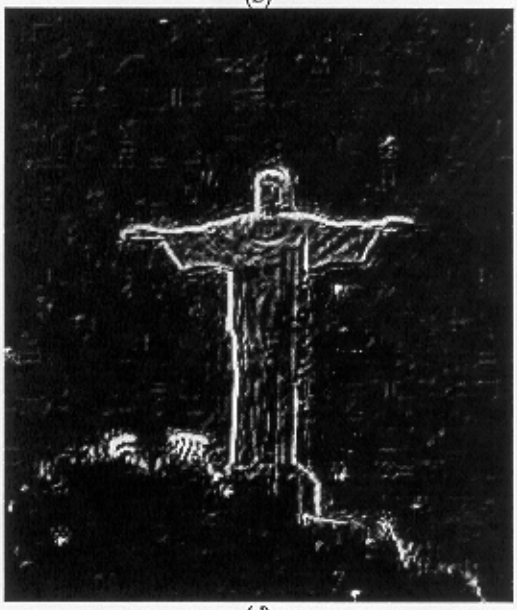
(a)



(b)



(c)



(d)

Figura 4 - (a) Imagem original. (b) Resultado da máscara de Sobel na direção X. (c) Resultado da máscara de Sobel na direção Y. (d) Resultado obtido com a soma das imagens nas duas direções.

Algoritmo -2+4

Realizar a detecção de borda, utilizando o gradiente, é uma solução bastante eficiente, quando temos bordas predominantemente em uma dada direção. Contudo, se esta borda assume um aspecto aleatório, o método já não é muito adequado. Primeiro, porque é bastante custoso realizar o gradiente em diversas direções para sobrepor as imagens ao final das operações. Segundo, a imagem obtida apresenta bastante ruído e inviabiliza qualquer análise com a mesma. Nesses tipos de imagens, seria interessante "caminhar" sobre a borda do objeto, isto é, a partir do momento que se descobre um pixel da borda, deve-se checar qual o pixel ao redor da presente posição é um pixel de borda, e em seguida movimentar-se para ele. Este procedimento pode ser ainda bastante subjetivo, e por hora, um pouco confuso, mas serve como idéia inicial de como funciona o algoritmo.

A opção em utilizar o algoritmo -2+4 veio da necessidade de analisar imagens do estudo de domínios magnéticos. Estas imagens são o resultado de simulações de relaxação magnética e formação de domínios em filmes finos.

Observando-se a Fig. 5 pode-se notar uma borda com várias irregularidades. Esta figura não parece, a princípio, muito complexa, mas se atentarmos para uma visão em detalhe desta imagem, veremos que a borda é bastante rugosa. Percebe-se que a análise utilizando gradiente é inviável. Isto porque não conseguiríamos responder a questões essenciais como: Em quantas e quais direções devemos aplicar os operadores gradiente?

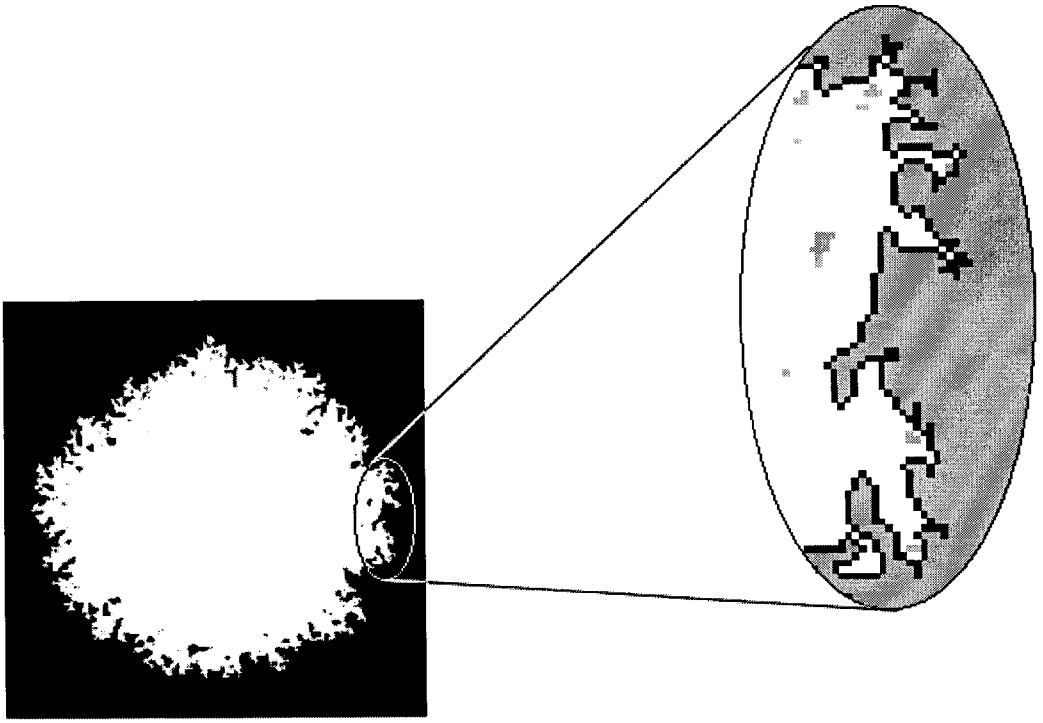


Figura 5 – Detalhes da borda rugosa de um domínio magnético.

Foi com o intuito de solucionar este tipo de problema que implementamos o algoritmo -2+4. Para compreendermos o seu funcionamento é necessário conhecer as ferramentas que serão utilizadas para abstrair a solução. Basicamente, elas se resumem a duas tabelas e uma pequena grade do tamanho de uma matriz 3x3. Cada célula desta matriz representa um pixel.

Suponha que o algoritmo esteja posicionado em um pixel da imagem que é um pixel de borda. Este pixel, de agora em diante chamado de pixel central, tem características especiais que irão ser aproveitadas para auxiliar na solução do problema. A primeira observação a ser feita é que a cor deste pixel é branca. Segundo, este pixel está em uma região de fronteira entre o fundo preto da imagem e o domínio de cor branca. Deste modo, avaliando-se a cor dos pixels pode-se afirmar que quando ocorrer uma transição de um pixel preto para um pixel branco existirá uma borda. Para compreender o procedimento de avaliação das cores do pixel observe a Fig.6 . Note a representação da matriz 3x3, o pixel central e as diversas setas indicando direções. Nas extremidades destas setas encontra-se uma marcação de ângulos tomada a partir de uma posição inicial 0°. Todos os pixels que são possíveis candidatos a serem bordas recebem uma "etiqueta" para suas posições (-3, -2, -1, 0, 1, 2, 3, 4).

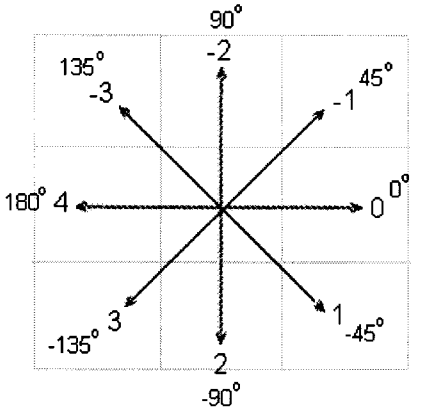


Figura 6 – Esquema da matriz 3x3. O pixel central corresponde à aquele onde as retas das direções se interceptam. As posições dos pixels são identificadas através de etiquetas, e as direções em graus são medidas a partir do pixel 0.

O algoritmo avalia a cor do pixel a partir da posição -3, girando no sentido horário até chegar a posição 4. Como na posição -3 sempre existirá um pixel preto afirma-se que quando ocorrer uma mudança de cor, um pixel de borda terá sido encontrado. O algoritmo movimenta-se para esta nova posição na imagem e atribui uma cor que foi previamente estabelecida como a cor característica da borda. Este procedimento é repetido até que um ciclo seja fechado, ou seja, até que toda a borda tenha sido percorrida. Observe o exemplo ilustrativo da Fig.7.

Conforme esperado, a cor do pixel na posição -3 não é branca, logo o algoritmo avalia a posição seguinte. Na posição -2 a cor também não é branca e o algoritmo movimenta-se para a posição -1. Nesta posição existe um pixel branco que nestas condições é um pixel de borda. Em seguida, o algoritmo movimenta o centro da matriz para as coordenadas (X,Y) deste pixel, pinta-o e retoma o procedimento de avaliação de cores.

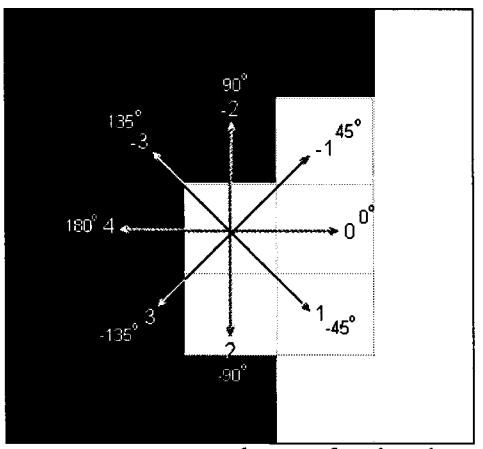


Figura 7- A área preta corresponde ao fundo da imagem enquanto que a branca corresponde ao domínio ("objeto"). O pixel central está localizado em cima de um pixel de borda.

Um ponto chave deste algoritmo é que pelo menos um pixel preto estará sendo lido inicialmente, e que sendo assim, ao ser lido um pixel branco encontraremos um pixel de borda. De modo simples, isto significa que a posição -3 sempre estará ocupada por um pixel preto. Entretanto a maneira que o método está colocado ainda não assegura isto. Imagine que o domínio magnético não estivesse a direita da figura conforme o que está exposto na Fig.7, mas sim na esquerda. Repare que um domínio na esquerda da imagem possui na sua posição -3 um pixel branco e que na posição -1 existirá um pixel preto e nestas condições o algoritmo não funciona. Ora, se o algoritmo é valido para o lado esquerdo do domínio e não o é para o lado direito, como pode ser possível detectar toda borda?

Para solucionar esta questão é necessário apresentar mais uma operação que deve ser realizada com a matriz 3x3 de maneira que as condições iniciais do algoritmo sejam sempre aceitas. Para compreender melhor o que irá ocorrer é essencial manter uma visão global do que esta sendo feito até o presente momento:

1. O pixel central esta posicionado em cima de um pixel de borda;
2. O algoritmo realiza uma procura no sentido horário para descobrir quem é o primeiro pixel branco;
3. Ao descobrir um pixel branco, o pixel central se move para esta nova posição;
4. O novo pixel descoberto é pintado com uma cor predefinida e retoma-se o passo 1.

Note que a matriz 3x3 está sofrendo uma translação a posição do pixel detectado. A intenção é que além desta translação, ocorra uma rotação em torno do centro da matriz. Esta rotação irá ocorrer no intervalo entre os passos 2 e 3. Observe a seguir, na Fig.8 todas as possíveis configurações que o sistema de eixos montado sobre a matriz 3x3 pode assumir.

Basicamente, o mecanismo de rotação envolve a direção para a qual aponta o vetor 0 e a posição do pixel de borda. O algoritmo de rotação inicia-se após este pixel ter sido descoberto, e a translação começa quando o mesmo tiver terminado. Entretanto é necessário compreender como ocorre a rotação do sistema de eixos. Após o pixel ter sido descoberto, o algoritmo descobre o quanto o sistema de eixos deve girar para que a direção 0 aponte para o novo pixel. Neste momento, cada elemento da matriz 3x3 é atualizado e a matriz assume uma das formas apresentadas na Fig8, e só então, a matriz sofre a translação.

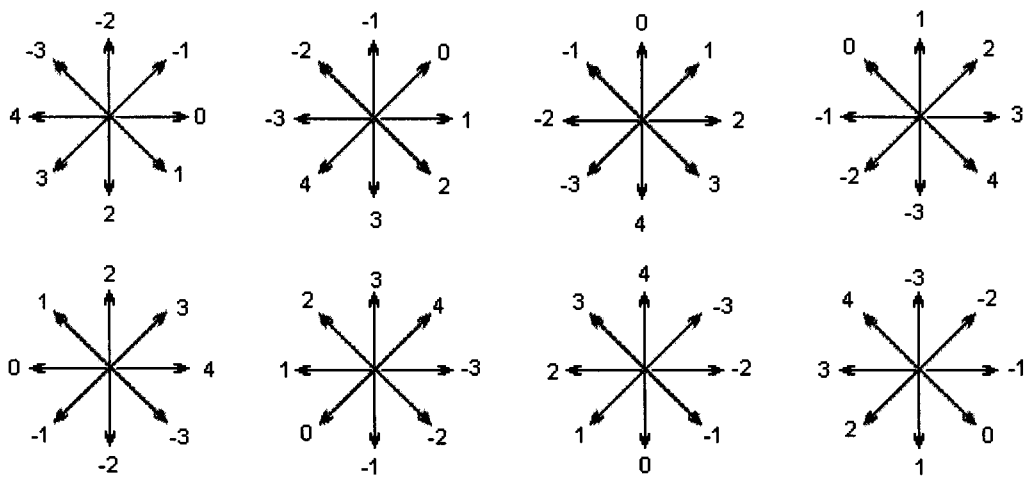


Figura 8 – Possíveis configurações do sistema de eixos da matriz 3x3.

Deste modo, o algoritmo completo de detecção fica organizado da seguinte forma:

1. O pixel central esta posicionado em cima de um pixel de borda;
2. O algoritmo realiza uma procura no sentido horário para descobrir quem é o primeiro pixel branco;
3. Ao descobrir um pixel branco, o sistema de eixos sofre uma rotação de modo que a direção 0 passe a apontar para este novo pixel, e a matriz 3x3 é atualizada;
4. O pixel central desloca-se para esta nova posição, isto é, a matriz 3x3 é transladada para a posição do pixel de borda recentemente encontrado;
5. O novo pixel descoberto é pintado com uma cor predefinida e retoma-se o passo 1.

Implementação

O programa possui três funções que são essenciais para o algoritmo. A primeira é um procedimento chamado *InicializaMatriz* que é responsável por preencher com os valores descritos na Tabela A cada elemento da matriz. Estes valores de ângulo correspondem a rotação que o sistema de eixos da matriz 3x3 deverá sofrer. Os motivos e a maneira pela qual é feita esta rotação será discutida mais adiante. A utilidade da Tabela A será compreendida quando for discutido o funcionamento do procedimento Menos2Mais4.

j \ i	1	2	3	4	5	6	7	8
1	180°	135°	90°	45°	0°	-45°	-90°	-135°
2	135°	90°	45°	0°	-45°	-90°	-135°	180°
3	90°	45°	0°	-45°	-90°	-135°	180°	135°
4	45°	0°	-45°	-90°	-135°	180°	135°	90°
5	0°	-45°	-90°	-135°	180°	135°	90°	45°
6	-45°	-90°	-135°	180°	135°	90°	45°	0°
7	-90°	-135°	180°	135°	90°	45°	0°	-45°
8	-135°	180°	135°	90°	45°	0°	-45°	-90°

Tabela A - Tabela implementada pelo vetor MatrizRotação

Em seguida, tem-se a função *NextPixel* que é o coração do funcionamento do algoritmo. Ele recebe como parâmetro um dos valores de *j* apresentados na Tabela A. No corpo desta função encontra-se implementado a Tabela B a qual possui a assinatura dos elementos da matriz 3x3 para cada uma das possíveis configurações do sistema de eixos apresentado na seção anterior. Com o valor *j* da linha na tabela, a função pesquisa dentro daquela configuração do sistema de eixos, qual a primeira direção que é encontrado um pixel de cor branca. Isto é feito avaliando o valor do pixel na posição da primeira coluna, da segunda coluna, até alcançar a última coluna. Quando este pixel é encontrado, retorna-se o valor da *direção* e a posição (x,y) do pixel central é atualizada.

j \ direção	-2(90°)	-1(45°)	0(0°)	1(-45°)	2(-90°)	3(-135°)	4(180°)
1	x , y+1	x-1 , y+1	x-1 , y	x-1 , y-1	x , y-1	x+1 , y-1	x+1 , y
2	x- , y+1	x-1 , y	x-1 , y-1	x , y-1	x+1 , y-1	x+1 , y	x+1 , y+1
3	x-1 , y	x-1 , y-1	x , y-1	x+1 , y-1	x+1 , y	x+1 , y+1	x , y+1
4	x-1 , y-1	x , y-1	x+1 , y-1	x+1 , y	x+1 , y+1	x , y+1	x-1 , y+1
5	x , y-1	x+1 , y-1	x+1 , y	x+1 , y+1	x , y+1	x-1 , y+1	x-1 , y
6	x+1 , y-1	x+1 , y	x+1 , y+1	x , y+1	x-1 , y+1	x-1 , y	x-1 , y-1
7	x+1 , y	x+1 , y+1	x , y+1	x-1 , y+1	x-1 , y	x-1 , y-1	x , y-1
8	x+1 , y+1	x , y+1	x-1 , y+1	x-1 , y	x-1 , y-1	x , y-1	x+1 , y-1

Tabela B – Tabela contendo as possíveis configurações que o sistema de eixos pode assumir.

Por fim tem-se o procedimento *Menos2Mais4* que utiliza as funções descritas para compor o algoritmo de detecção de borda. Inicialmente, posiciona-se o pixel central na lateral esquerda a uma altura média da imagem. Em seguida, o pixel central percorre a imagem da esquerda para a direita verificando a cor dos pixels. A cor sempre será preta, a não ser que se encontre um pixel do domínio, uma borda, que branco. Quando este pixel é encontrado, o deslocamento do pixel central pára e pinta-se de vermelho este primeiro pixel de borda. Partir de então o movimento do pixel central, e consequentemente da matriz 3x3, passa a ser regido pela função *NextPixel*. Após a nova direção ter sido retornada, procura-se na *MatrizRotação* (Tabela A) qual a nova assinatura da matriz 3x3, isto é, com qual linha *j* deve ser feita a nova procura de pixels brancos.

Dentro da coluna da Tabela A determinada pela direção, procura-se qual a linha que possui o valor 0°. Esta linha corresponde ao valor de *j* que será utilizado para a próxima interação. Este procedimento se repete sucessivamente, até que encontra-se o pixel vermelho, o primeiro pixel encontrado, que constitui a condição de parada total do algoritmo.

Listagem

```
unit User5;
{Importante:                                                                    }
{           255  -> preto                                                       }
{             0 -> branco                                                       }
interface

   uses
      QuickDraw, Palettes, PrintTraps, globals, Utilities, Graphics, Functions, Analysis, Edit, Lut, File1, Camera, Sound, User2, User, User4;

      procedure WriteAeraFiles;
      procedure InicializaMatriz;
      function NextPixel (linha: integer): integer;
      procedure Menos2Mais4;
      procedure MyInitializeInfoPtr;
      procedure Menos2Mais4doLuiz;
      procedure Calculos (horizontal: integer; vertical: integer);
      procedure BinarizeTheImage;
      procedure MySetOptionsForAP;


implementation
   const
      BRANCO = 0;
      VERMELHO = 1;
      AZUL = 3;
       XINICIAL = 0;
      YINICIAL = 499;


   var
      MatrizRotacao: array[1..8, 1..8] of integer;
      i, j, x, y, BinaryValue, x_cg, y_cg: integer;
       x_center, y_center, perimeter: real;
      saida: text;
      FileName: str255;
      workfile: Text;
      rescode: OSErr;
      per: integer;

{ ... WriteAeraFiles ..... }
{Este procedimento uma interface com o usuario e permite especificar qual sera o arquivo de saida de dados }
procedure WriteAeraFiles;
   var
      str, str1: str255;
      k, ignore: integer;
      area: Real;
   begin
      FileName := '024cs.dat';
      WriteWhatFile(FileName, 'Enter the contour file name ?');
       if reply.good then begin
          rescode := Setvol(nil, reply.vRefnum);
          rewrite(workfile, Reply.fname);
          writeln(workfile, 'WorkImage:', info^.title);
          writeln(workfile, ' x       y      pho     theta');
          UpdatePicWindow;
      end;

      SaveInfo := info; { Save info Pointer of the work image in SaveInfo pointer }
      if (Duplicate('Image Work Buffer 2', true) = false) then begin
         PutMessage('Can not create a work buffer n°2, can not start... . Try change allocated memory in the finder. ');
         Exit(WriteAeraFiles);
       end;
       BinaryValue := 127;
       BinarizeTheImage;
       InvertPic;
       MySetOptionsForAP;
       AnalyzeParticles;
        x_center := xcenter^[1];
        y_center := ycenter^[1];
        info^.changes := false; { to Close the image withou asking to save. }

        ignore := CloseAWindow(PicWindow[nPics]); { Close Work Buffer }
        Info := SaveInfo;
        UpdatePicWindow;
```

```
        x := MyGetPixel(100, 100);

        Menos2Mais4doLuiz;
        ShowMessage('Fim.');

    end;

{procedimento para binarizar a imagem}
 procedure BinarizeTheImage;
    var
    i: integer;
    MyTable: LookUpTable;
    begin
       for i := 0 to BinaryValue do
          MyTable[i] := WhiteIndex;
       for i := BinaryValue to 255 do
          MyTable[i] := BlackIndex;
       ApplyTable(MyTable);
    end;


procedure MySetOptionsForAP;
    begin
        MinParticleSize := 1;
        MaxParticleSize := 300000;
        OutlineParticles := true;
        LabelParticles := false;
        IgnoreParticlesTouchingEdge := true;
        IncludeHoles := false;
    end;

procedure MyInitializeInfoPtr;
    begin
       Info^.BytesPerRow := 1024; {Init. Info Pointer with Image Parameters. }
       Info^.ImageSize := 1048576;
       Info^.nlines := 1024;
       Info^.PixelsPerLine := 1024;
       Info^.ColorTable := Grays;
       Info^.LUTMode := GrayScale;
    end;


{ ... Inicializa_Matriz ..... }
procedure InicializaMatriz;
    begin
       MatrizRotacao[1, 1] := 180;
       MatrizRotacao[1, 2] := 135;
       MatrizRotacao[1, 3] := 90;
       MatrizRotacao[1, 4] := 45;
       MatrizRotacao[1, 5] := 0;
       MatrizRotacao[1, 6] := -45;
       MatrizRotacao[1, 7] := -90;
       MatrizRotacao[1, 8] := -135;

       MatrizRotacao[2, 1] := 135;
       MatrizRotacao[2, 2] := 90;
       MatrizRotacao[2, 3] := 45;
       MatrizRotacao[2, 4] := 0;
       MatrizRotacao[2, 5] := -45;
       MatrizRotacao[2, 6] := -90;
       MatrizRotacao[2, 7] := -135;
       MatrizRotacao[2, 8] := 180;

       MatrizRotacao[3, 1] := 90;
       MatrizRotacao[3, 2] := 45;
       MatrizRotacao[3, 3] := 0;
       MatrizRotacao[3, 4] := -45;
       MatrizRotacao[3, 5] := -90;
       MatrizRotacao[3, 6] := -135;
       MatrizRotacao[3, 7] := 180;
       MatrizRotacao[3, 8] := 135;

       MatrizRotacao[4, 1] := 45;
       MatrizRotacao[4, 2] := 0;
       MatrizRotacao[4, 3] := -45;
       MatrizRotacao[4, 4] := -90;
       MatrizRotacao[4, 5] := -135;
       MatrizRotacao[4, 6] := 180;
```

```
    MatrizRotacao[4, 7] := 135;
    MatrizRotacao[4, 8] := 90;

    MatrizRotacao[5, 1] := 0;
    MatrizRotacao[5, 2] := -45;
    MatrizRotacao[5, 3] := -90;
    MatrizRotacao[5, 4] := -135;
    MatrizRotacao[5, 5] := 180;
    MatrizRotacao[5, 6] := 135;
    MatrizRotacao[5, 7] := 90;
    MatrizRotacao[5, 8] := 45;
    MatrizRotacao[6, 1] := -45;
    MatrizRotacao[6, 2] := -90;
    MatrizRotacao[6, 3] := -135;
    MatrizRotacao[6, 4] := 180;
    MatrizRotacao[6, 5] := 135;
    MatrizRotacao[6, 6] := 90;
    MatrizRotacao[6, 7] := 45;
    MatrizRotacao[6, 8] := 0;

    MatrizRotacao[7, 1] := -90;
    MatrizRotacao[7, 2] := -135;
    MatrizRotacao[7, 3] := 180;
    MatrizRotacao[7, 4] := 135;
    MatrizRotacao[7, 5] := 90;
    MatrizRotacao[7, 6] := 45;
    MatrizRotacao[7, 7] := 0;
    MatrizRotacao[7, 8] := -45;

    MatrizRotacao[8, 1] := -135;
    MatrizRotacao[8, 2] := 180;
    MatrizRotacao[8, 3] := 135;
    MatrizRotacao[8, 4] := 90;
    MatrizRotacao[8, 5] := 45;
    MatrizRotacao[8, 6] := 0;
    MatrizRotacao[8, 7] := -45;
    MatrizRotacao[8, 8] := -90;
end;


{ ... NextPixel ..... }
function NextPixel (linha: integer): integer;
  var
    v, h, vteste, hteste, ValorAux, direcao: integer;
  begin
    h := x;
    v := y;
    direcao := -99;

    case linha of
      1: begin
        ValorAux := MyGetPixel(h, v + 1);   { .....Direcao -2 }
        if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
          direcao := 90;
          x := h;
          y := v + 1;
        end
        else begin
          ValorAux := MyGetPixel(h - 1, v + 1);   { .....Direcao -1 }
          if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
            direcao := 45;
            x := h - 1;
            y := v + 1;
          end
          else begin
            ValorAux := MyGetPixel(h - 1, v);   { .....Direcao 0 }
            if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
              direcao := 0;
              x := h - 1;
              y := v;
            end
            else begin
              ValorAux := MyGetPixel(h - 1, v - 1);   { .....Direcao 1 }
              if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                direcao := -45;
                x := h - 1;
                y := v - 1;
              end
              else begin
```

```
            ValorAux := MyGetPixel(h, v - 1);   {..... Direcao 2 }
            if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
               direcao := -90;
               x := h;
               y := v - 1;
            end
            else begin
               ValorAux := MyGEtPixel(h + 1, v - 1);   {..... Direcao 3 }
               if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                  direcao := -135;
                  x := h + 1;

                  y := v - 1;
               end
               else begin
                     ValorAux := MyGetPixel(h + 1, v);   {..... Direcao 4 }
                     if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                        direcao := 180;
                        x := h + 1;
                        y := v;
                     end;
               end;
            end;
         end;
      end;
   end;
end;
2: begin
   ValorAux := MyGetPixel(h - 1, v + 1);   { .....Direcao -2 }
   if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
      direcao := 90;
      x := h - 1;
      y := v + 1;
   end
   else begin
      ValorAux := MyGetPixel(h - 1, v);   { .....Direcao -1 }
      if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
         direcao := 45;
         x := h - 1;
         y := v;
      end
      else begin
         ValorAux := MyGetPixel(h - 1, v - 1);   { .....Direcao 0 }
         if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
            direcao := 0;
            x := h - 1;
            y := v - 1;
         end
         else begin
            ValorAux := MyGetPixel(h, v - 1);   { .....Direcao 1 }
            if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
               direcao := -45;
               x := h;
               y := v - 1;
            end
            else begin
               ValorAux := MyGetPixel(h + 1, v - 1);   {..... Direcao 2 }
               if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                  direcao := -90;
                  x := h + 1;
                  y := v - 1;
               end
               else begin
                  ValorAux := MyGEtPixel(h + 1, v);   {..... Direcao 3 }
                  if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                     direcao := -135;
                     x := h + 1;
                     y := v;
                  end
                  else begin
                     ValorAux := MyGetPixel(h + 1, v + 1);   {... Direcao 4 }
                     if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                        direcao := 180;
                        x := h + 1;
                        y := v + 1;
                     end;
                  end;
               end;
```

```
            end;
          end;
        end;
      end;
    end;
3: begin
     ValorAux := MyGetPixel(h - 1, v);   { .....Direcao -2 }
     if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
       direcao := 90;
       x := h - 1;
       y := v;
     end
     else begin
       ValorAux := MyGetPixel(h - 1, v - 1);  { .....Direcao -1 }
       if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
         direcao := 45;
         x := h - 1;
         y := v - 1;
       end
       else begin
         ValorAux := MyGetPixel(h, v - 1);  { .....Direcao 0 }
         if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
           direcao := 0;
           x := h;
           y := v - 1;
         end
         else begin
           ValorAux := MyGetPixel(h + 1, v - 1);  { .....Direcao 1 }
           if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
             direcao := -45;
             x := h + 1;
             y := v - 1;
           end
           else begin
             ValorAux := MyGetPixel(h + 1, v);  {..... Direcao 2 }
             if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
               direcao := -90;
               x := h + 1;
               y := v;
             end
             else begin
               ValorAux := MyGEtPixel(h + 1, v + 1);  {..... Direcao 3 }
               if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                 direcao := -135;
                 x := h + 1;
                 y := v + 1;
               end
               else begin
                 ValorAux := MyGetPixel(h, v + 1);  {..... Direcao 4 }
                 if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                   direcao := 180;
                   x := h;
                   y := v + 1;
                 end;
               end;
             end;
           end;
         end;
       end;
     end;
   end;
4: begin
   ValorAux := MyGetPixel(h - 1, v - 1);   { .....Direcao -2 }
   if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
     direcao := 90;
     x := h - 1;
     y := v - 1;
   end
   else begin
     ValorAux := MyGetPixel(h, v - 1);  { .....Direcao -1 }
     if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
       direcao := 45;
       x := h;
       y := v - 1;
     end
     else begin
       ValorAux := MyGetPixel(h + 1, v - 1);  { .....Direcao 0 }
       if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
         direcao := 0;
```

```
            x := h + 1;
            y := v - 1;
          end
          else begin
            ValorAux := MyGetPixel(h + 1, v);  { .....Direcao 1 }
            if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
              direcao := -45;
              x := h + 1;
              y := v;
            end
            else begin
              ValorAux := MyGetPixel(h + 1, v + 1);  {..... Direcao 2 }
              if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                direcao := -90;
                x := h + 1;
                y := v + 1;
              end
              else begin
                ValorAux := MyGEtPixel(h, v + 1);  {..... Direcao 3 }
                if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                  direcao := -135;
                  x := h;
                  y := v + 1;
                end
                else begin
                  ValorAux := MyGetPixel(h - 1, v + 1);  {.... Direcao 4 }
                  if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                    direcao := 180;
                    x := h - 1;
                    y := v + 1;
                  end;
                end;
              end;
            end;
          end;
        end;
      end;
    end;
  end;
5: begin{posicao inicial}
    ValorAux := MyGetPixel(h, v - 1);  { .....Direcao -2 }
    if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
      direcao := 90;
      x := h;
      y := v - 1;
    end
    else begin
      ValorAux := MyGetPixel(h + 1, v - 1);  { .....Direcao -1 }
      if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
        direcao := 45;
        x := h + 1;
        y := v - 1;
      end
      else begin
        ValorAux := MyGetPixel(h + 1, v);  { .....Direcao 0 }
        if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
          direcao := 0;
          x := h + 1;
          y := v;
        end
        else begin
          ValorAux := MyGetPixel(h + 1, v + 1);  { .....Direcao 1 }
          if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
            direcao := -45;
            x := h + 1;
            y := v + 1;
          end
          else begin
            ValorAux := MyGetPixel(h, v + 1);  {..... Direcao 2 }
            if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
              direcao := -90;
              x := h;
              y := v + 1;
            end
            else begin
              ValorAux := MyGEtPixel(h - 1, v + 1);  {..... Direcao 3 }
              if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                direcao := -135;
                x := h - 1;
                y := v + 1;
```

```
                    end
                    else begin
                        ValorAux := MyGetPixel(h - 1, v);   {..... Direcao 4 }
                        if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                            direcao := 180;
                            x := h - 1;
                            y := v;
                        end;
                    end;
                end;
            end;
        end;
    end;
end;
6: begin
    ValorAux := MyGetPixel(h + 1, v - 1);   { .....Direcao -2 }
    if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
        direcao := 90;
        x := h + 1;
        y := v - 1;
    end
    else begin
        ValorAux := MyGetPixel(h + 1, v);   { .....Direcao -1 }
        if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
            direcao := 45;
            x := h + 1;
            y := v;
        end
        else begin
        ValorAux := MyGetPixel(h + 1, v + 1);   { .....Direcao 0 }
        if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
            direcao := 0;
            x := h + 1;
            y := v + 1;
        end
        else begin
            ValorAux := MyGetPixel(h, v + 1);   { .....Direcao 1 }
            if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                direcao := -45;
                x := h;
                y := v + 1;
            end
            else begin
                ValorAux := MyGetPixel(h - 1, v + 1);   {..... Direcao 2 }
                if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                    direcao := -90;
                    x := h - 1;
                    y := v + 1;
                end
                else begin
                    ValorAux := MyGEtPixel(h - 1, v);   {..... Direcao 3 }
                    if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                        direcao := -135;
                        x := h - 1;
                        y := v;
                    end
                    else begin
                        ValorAux := MyGetPixel(h - 1, v - 1);   {..... Direcao 4 }
                        if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                            direcao := 180;
                            x := h - 1;
                            y := v - 1;
                        end;
                    end;
                end;
            end;
        end;
    end;
end;
': begin
    ValorAux := MyGetPixel(h + 1, v);   { .....Direcao -2 }
    if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
        direcao := 90;
        x := h + 1;
        y := v;
    end
    else begin
```

```
      ValorAux := MyGetPixel(h + 1, v + 1);  { .....Direcao -1 }
      if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
         direcao := 45;
         x := h + 1;
         y := v + 1;
      end
      else begin
         ValorAux := MyGetPixel(h, v + 1);  { .....Direcao 0 }
         if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
            direcao := 0;
            x := h;
            y := v + 1;
         end
         else begin
            ValorAux := MyGetPixel(h - 1, v + 1);  { .....Direcao 1 }
            if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
               direcao := -45;
               x := h - 1;
               y := v + 1;
            end
            else begin
               ValorAux := MyGetPixel(h - 1, v);  {..... Direcao 2 }
               if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                  direcao := -90;
                  x := h - 1;
                  y := v;
               end
               else begin
                  ValorAux := MyGEtPixel(h - 1, v - 1);  {..... Direcao 3 }
                  if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                     direcao := -135;
                     x := h - 1;
                     y := v - 1;
                  end
                  else begin
                     ValorAux := MyGetPixel(h, v - 1);  {..... Direcao 4 }
                     if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                        direcao := 180;
                        x := h;
                        y := v - 1;
                     end;
                  end;
               end;
            end;
         end;
      end;
   end;
end;
8: begin
      ValorAux := MyGetPixel(h + 1, v + 1);  { .....Direcao -2 }
      if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
         direcao := 90;
         x := h + 1;
         y := v + 1;
      end
      else begin
         ValorAux := MyGetPixel(h, v + 1);  { .....Direcao -1 }
         if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
            direcao := 45;
            x := h;
            y := v + 1;
         end
         else begin
            ValorAux := MyGetPixel(h - 1, v + 1);  { .....Direcao 0 }
            if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
               direcao := 0;
               x := h - 1;
               y := v + 1;
            end
            else begin
               ValorAux := MyGetPixel(h - 1, v);  { .....Direcao 1 }
               if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                  direcao := -45;
                  x := h - 1;
                  y := v;
               end
               else begin
                  ValorAux := MyGetPixel(h - 1, v - 1);  {..... Direcao 2 }
                  if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
```

```
                        direcao := -90;
                        x := h - 1;
                        y := v - 1;
                     end
                     else begin
                        ValorAux := MyGEtPixel(h, v - 1);   {..... Direcao 3 }
                        if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                           direcao := -135;
                           x := h;
                           y := v - 1;
                        end
                        else begin
                           ValorAux := MyGetPixel(h + 1, v - 1);   {..... Direcao 4 }
                           if ((ValorAux = BRANCO) or (ValorAux = AZUL) or (ValorAux = VERMELHO)) then begin
                              direcao := 180;
                              x := h + 1;
                              y := v - 1;
                           end;
                        end;
                     end;
                  end;
               end;
            end;
         end;
      end;
      NextPixel := direcao;
   end;

{... Menos2Mais4 .....}
procedure Menos2Mais4;
   var
      valor, direcao: integer;
   begin
      x := XINICIAL;
      y := YINICIAL;
      InicializaMatriz;
      j := 5;{posicao inicial na matriz de rotacao}
      i := 1;{posicao inicial na matriz de rotacao}

      {   Busca a borda da celula}
      valor := MyGetPixel(x, y);
      while (valor <> 0) do begin
         x := x + 1;
         valor := MyGetPixel(x, y);
      end;

      {  Achou a celula}
      {  Pixel Inicial -> marco ele com uma cor diferente (VERMELHO) porque ele sera a condicao de parada}
      PutPixel(x, y, VERMELHO);
      direcao := NextPixel(j);

      j := 1;
      while (MatrizRotacao[i, j] <> direcao) do
         j := j + 1;
      i := 1;
      while (MatrizRotacao[i, j] <> 0) do
         i := i + 1;
      valor := MyGetPixel(x, y);
      while (valor <> VERMELHO) do begin
         PutPixel(x, y, AZUL);       {Valor para os pixels da borda}

         {*************************** }
         {**    Calculos com os pixels     **}
         {**    devem ser inseridos aqui   **}
         {*************************** }

         direcao := NextPixel(j);
         j := 1;
         while (MatrizRotacao[i, j] <> direcao) do
            j := j + 1;
         i := 1;
         while (MatrizRotacao[i, j] <> 0) do
            i := i + 1;
         valor := MyGetPixel(x, y);
         end;
      end;

end. { User5 }
```

Bibliografia

GONZALEZ, Rafael C. & WINTZ, Paul. "Digital Image Processing". 2ed. Addison-Wesley Publishing Company, Reading, Massachusetts, 1987.

"A Tutorial on Image Processing". Noesis Vision Inc. ,Ville St Laurent, Quebec, 1991.

GOMES, Jonas & VELHO, Luiz. "Computação Gráfica: Imagem". IMPA/SBM, Rio de Janeiro, RJ, 1994.

SCHALKOFF, Robert J. "Digital Image Processing and Computer Vision". Johm Wiley & Sons, New York, 1989.