Uso da Transformada de Hough na Detecção de Círculos em Imagens Digitais

Glaucius Décio Duarte[1] [2]

Resumo - ***A Transformada de Hough (TH) pode ser usada para descobrir padrões paramétricos, como linhas e círculos, embutidas em imagens ruidosas. A Transformada de Hough em Múltiplas Resoluções (THMR) é muito eficiente na redução do processamento computacional e memória necessária para armazenamento dos dados computados. Este artigo mostra a eficiência da Transformada de Hough em Múltiplas Resoluções como um método para a detecção de Círculos (THC) em imagens digitais.***

Palavras Chave - ***Processamento de Imagens Digitais, Transformada de Hough, Detecção de círculos em imagens digitais.***

Introdução

Padrões em imagens normalmente são encontrados com descontinuidades e com inclusão de ruídos. A Transformada de Hough (TH) é conhecida como uma técnica eficiente para descobrir padrões descontínuos inseridos em imagens ruidosas [1] [3] [5].

Para a detecção de círculos em imagens, utiliza-se um processo de votação onde os votos são atribuídos aos pontos de passagem dos possíveis círculos existentes na imagem [2] [5] [7] [9]. Os votos são acumulados em um vetor de acumulação de votos, sendo que a detecção de um possível círculo é obtida quando um valor máximo (cume) é obtido no acumulador de votos (Fig. 1).

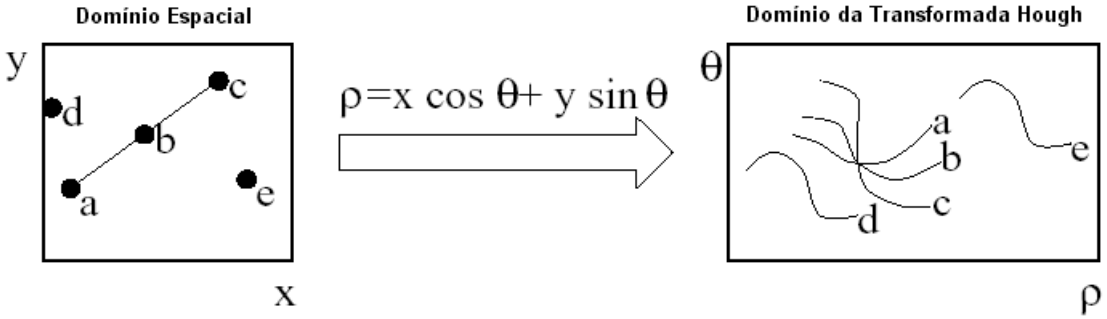

Fig. 1 - Mapeamento de uma linha existente no domínio espacial para o domínio da Transformada de Hough

A transformação apresenta a desvantagem de ser computacionalmente limitada e exigir uma grande quantidade de memória. O processamento computacional aumenta com o acréscimo no tamanho do vetor de acumulação e com a exatidão com que os parâmetros são

[1] Professor Adjunto da ESIN/UCPel, Rua Félix da Cunha 412, Pelotas-RS-Brasil, 0xx53-284-8227, glaucius@atlas.ucpel.tche.br

[2] Professor de Ensino Técnico e Tecnológico do CEFET-RS, Praça 20 de Setembro 455, Pelotas-RS-Brasil, 0xx53-284-5000, glaucius@cefetrs.tche.br

determinados. A resolução do vetor de acumulação determina a exatidão com que os parâmetros podem ser determinados.

Um trabalho interessante a ser desenvolvido consiste na obtenção de um procedimento que permita aliar um processamento computacional otimizado a uma eficiência de armazenamento da TH. Informações de gradiente também podem ser utilizadas para reduzir o tempo computacional em um sexto, quando comparado ao método que não usa as informações de gradiente. Implementações da transformada em sistemas de multiprocessamento também são propostas com o objetivo de reduzir o tempo de processamento, usando vários processadores [1] [9].

As estratégias de detecção simplificadas são algoritmos computacionalmente eficientes, sendo apropriadas para implementação em sistemas de processador único. A complexidade computacional reduzida obtida pela Transformada de Hough em Múltiplas Resoluções (THMR) resulta do uso de um algoritmo simples para detecção de cumes, além de imagens em múltiplas resoluções e vetores de acumulação nas iterações sucessivas do algoritmo. A efetividade da THMR pode ser demonstrada aplicando-se este método para imagens que contenham linhas na imagem.

Uma característica da Transformada de Hough é o fato de não permitir a determinação do comprimento e dos pontos de extremidade das linhas existentes em uma imagem. Neste caso, torna-se necessário utilizar outros métodos para atingir estes objetivos.

Entre os algoritmos que utilizam a TH, baseados em uma quantização dinâmica, o THMR ainda é o algoritmo mais eficiente entre os que estão disponíveis na literatura. Os objetivos deste trabalho incluem demonstrar a eficiência da Transformada de Hough na detecção de Círculos (THC) em imagens digitais, além de determinar a exatidão com que os parâmetros dos círculos podem ser descobertos, por meio da análise visual dos resultados obtidos pelo método para duas imagens diferentes.

Um passo necessário no processo de aplicação da TH é a detecção de bordas e a limiarização. Sugere-se o uso do método de *Canny* [4] para a realização desta tarefa. A detecção de bordas e a escolha de um limiar ótimo pode ser realizada por meio de métodos existentes na literatura. Visto que a detecção de bordas em imagens não está incluída no escopo deste trabalho, assume-se que as bordas da imagem já foram detectadas por um método adequado, antes de iniciar a aplicação da TH (Fig. 2).

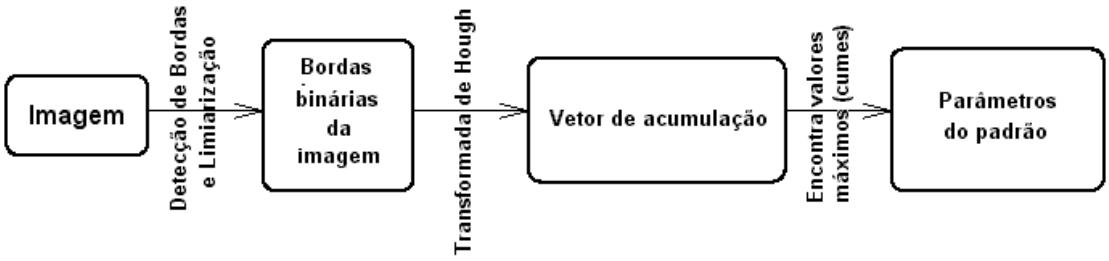

Fig. 2 - Estágios do processo de detecção de padrões em imagens pela TH

Descrição do Método para Detecção de Linhas

Inicialmente a imagem original é lida e se estiver no domínio RGB (24 bits), deve ser convertida para tons de cinza (8 bits). A seguir, utiliza-se um método eficiente para a detecção das bordas da imagem, gerando-se uma imagem binária cujos pixels ativos indicam as bordas detectadas na imagem. Sugere-se a utilização do método de *Canny* [4], por ser reconhecidamente eficiente nesta tarefa. Maiores detalhes sobre o método de *Canny* não serão discutidos, pois este detalhamente não pertence ao escopo deste trabalho, mas podem ser encontrados facilmente na literatura de processamento de imagens.

Uma vez que as bordas da imagem tenham sido detectadas, converte-se os pixels para o domínio da TH. Para isto, utiliza-se um vetor de acumulação de votos, para as maiores ocorrências de intersecção das senóides obtidas pela equação indicada na Fig. 1. Esta equação é utilizada para a obtenção dos valores de θ e ρ em função das coordenadas x,y dos pixels detectados pelo método de *Canny*.

O ângulo θ é aplicado num intervalo de 0 a 179 graus. Para implementação em MATLAB, utilizou-se a função `radon` do software MATLAB, como exemplificado a seguir. Note-se que imb é a imagem binária obtida, `R` (valores de θ) e `xp` (número de ocorrências das intersecções das senóides – cumes) são os valores que serão utilizados para a plotagem das senóides [6] [8].

```
theta = (0:179)';
[R,xp] = radon(imb, theta);
```

A seguir, os cumes dos valores armazenados no vetor de acumulação de intersecções das senóides são determinados em função do limiar fornecido, ordenando-se então a saída e escolhendo-se as linhas com máximos na TH. O passo seguinte inclui a conversão dos índices lineares dos cumes para coordenadas x, y.

Obtém-se então os valores de θ e ρ para os valores das coordenadas obtidas, determinando-se os parâmetros das linhas em função de `cos(t)`, `sen(t)` e `–r`. Finalmente, converte-se a origem do sistema de coordenadas do centro da imagem para o vértice superior esquerdo, exibindo-se as linhas detectadas sobre a imagem original.

Implementação da TH para linhas em MATLAB

A seguir, apresenta-se o código da implementação em MATLAB para a detecção de linhas pelo uso da TH.

```
% ----------------------------------------
% Uso da Transformada de Hough na Deteccao de
% Linhas em Imagens Digitais
% ----------------------------------------

function achalinhas(limiar)

im = imread('imagem1.png');
im = rgb2gray(im);
im = double(im)/255;
```

```
figure(1)
imb = edge(im);
imshow(imb);
title ('Imagem original')
theta = (0:179)';
[R,xp] = radon(imb, theta);
figure(2)
imagesc(theta,xp,R), colorbar;
xlabel ('theta (graus)'), ylabel ('rho (pixels do centro)')
title('Dominio da TH');
plt=1;
cumes = limiar;
i = find(R>cumes);
[foo,ind] = sort(-R(i));
k = i(ind(1:size(i)));
[y,x] = ind2sub(size(R),k);
t = -theta(x)*pi/180;
r = xp(y);
linhas = [cos(t) sin(t) -r];
cx = size(im,2)/2-1;
cy = size(im,1)/2-1;
linhas(:,3) = linhas(:,3) - linhas(:,1)*cx - linhas(:,2)*cy;
figure(3+plt)
texto1 = 'Imagem com as linhas detectadas para limiar = ';
texto2 = num2str(cumes);
texto = strcat(texto1,texto2);
imshow(imb);
title(texto);
desenha_linhas(linhas);
plt=plt+1;
```

Resultados obtidos na detecção de linhas

A Fig. 3 mostra a imagem original utilizada para os testes, assim como o domínio da TH, onde podem ser observadas as diversas senóides obtidas a partir das coordenadas x,y dos pixels de borda detectados.

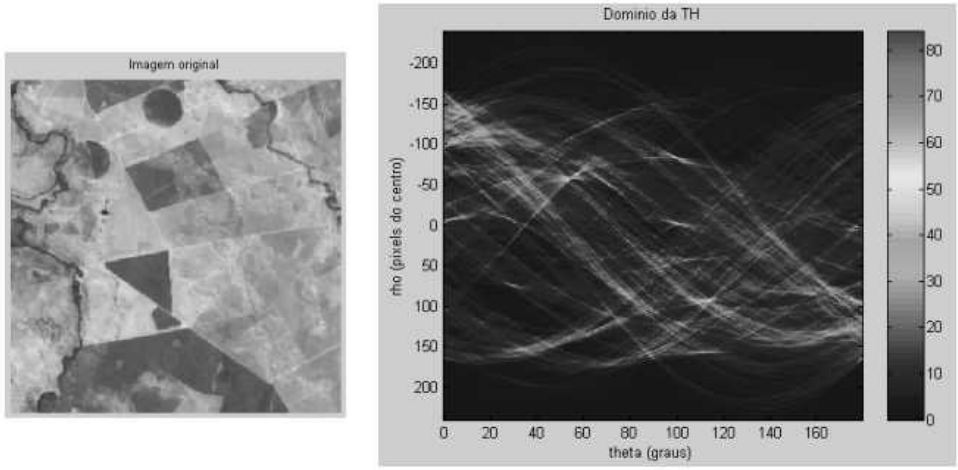

Fig. 3 – Imagem original e as senóides obtidas no domínio da TH

A Fig. 4 apresenta as linhas detectadas para 2 limiares diferentes. Note-se que o valor do limiar indica o número de votos armazenados no vetor de acumulação de votos. Assim, são

mostradas as linhas correspondentes aos valores máximos (cumes) maiores que o valor do limiar. Por esta razão, para `limiar = 40`, aparecem mais linhas do que para o valor de `limiar = 60`.

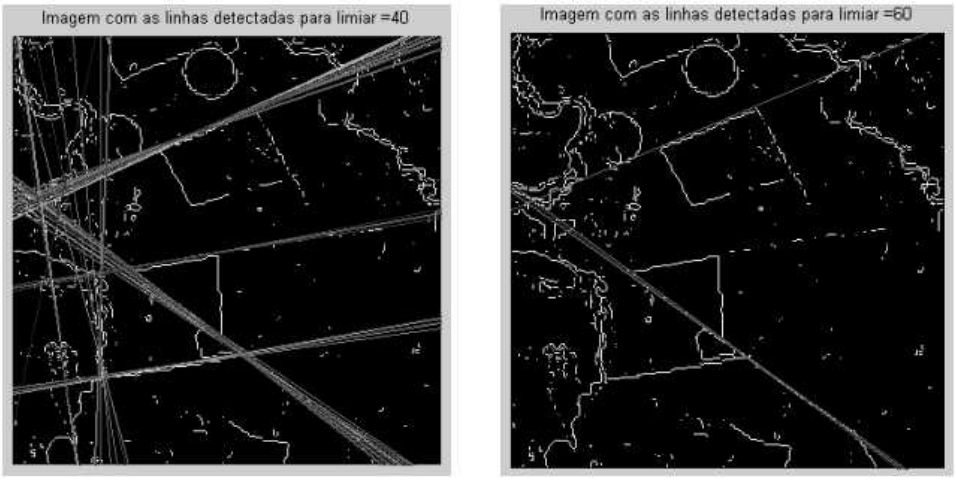

Fig. 4 – Detecção de linhas para 2 limiares diferentes (40 e 60)

Descrição do Método para Detecção de Círculos

Para a detecção de círculos em imagens, será utilizado um processo de eleição de círculos onde os votos são atribuídos aos pontos de passagem dos possíveis círculos existentes na imagem. Os votos são acumulados em uma matriz de acumulação de votos, sendo que a detecção de um possível círculo é obtida quando um valor máximo (cume) é obtido no acumulador de votos [6] [8].

A pesquisa nos pixels da imagem binária que contém as bordas da imagem original é feita utilizando-se a definição matemática (1), onde `a` e `b` são as coordenadas do centro do círculo e `r` é o raio do círculo.

$$(x - a)^2 + (y - b)^2 = r^2 \qquad (1)$$

Note-se que o algoritmo de pesquisa considera, por uma questão de simplificação computacional (para tornar o algoritmo mais rápido), os pixels encontrados em relação às direções horizontal e vertical de -45^o a 45^o, como pode ser visto na Fig. 5. Os demais pixels, nos outros quadrantes, são obtidos por reflexão.

Assim, o algoritmo inicia com a leitura da imagem original, convertendo-se esta imagem para tons de cinza, usando-se logo a seguir, o método de Canny para a obtenção da imagem binária que contém os pixels das bordas da imagem. Note-se que o usuário deve fornecer, além da imagem, os parâmetros `raio` (raio dos círculos a serem encontrados na imagem) e `dist` (resolução da matriz de acumulação de votos).

Feito isto, determinam-se as coordenadas x, y dos centros dos candidatos a círculos, aplicando-se a determinação da Transformada de Hough para detecção de Círculos (THC), em

uma dada resolução definida pelo parâmetro `dist` definido pelo usuário. Note-se que este parâmetro define uma resolução inicial para a determinação da THC. Se este valor for muito grande, a precisão na detecção dos círculos diminui, caso contrário, para valores próximos de 1, a precisão é maior, mas em contrapartida, o processamento computacional é mais demorado.

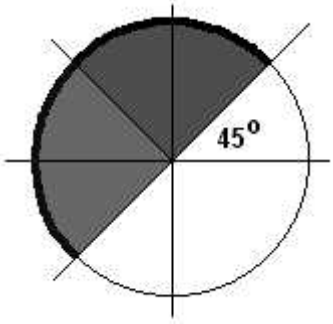

Fig. 5 – Pesquisa de pixels pertencentes ao círculo por quadrantes

A seguir, exibe-se a matriz de acumulação de votos para análise. Observe-se que neste trabalho, isto foi realizado, mostrando-se tanto a representação bidimensional da matriz como a sua representação tridimensional, que facilita a identificação visual dos cumes, mostrando claramente a posição dos centros dos círculos encontrados. Para finalizar, a imagem original é exibida mostrando-se os contornos dos círculos encontrados na cor branca, sobrepostos a esta imagem, para facilitar a identificação visual.

A matriz de acumulação de votos `acumuladorTH` é obtida a partir da seguinte descrição. Inicialmente, determina-se o número de barras `xnb`, `ynb` e `rnb` no domínio da THC, obtendo-se logo a seguir os valores iniciais de x e y assim como as diferenças iniciais para os candidatos a círculos. Determinam-se então, as linhas e colunas que contém as coordenadas dos pixels das bordas e inicializa-se a matriz de acumulação de votos no `acumuladorTH` com os valores iniciais armazenados em `xnb` e `ynb`.

Inclui-se então, os votos no acumulador em função de uma pesquisa realizada na imagem binária que contém as bordas da imagem original em função do raio fornecido (isto é feito para os pixels das bordas encontrados em linhas e colunas). Finalmente, redimensiona-se a imagem do domínio da THC para o tamanho original usando-se uma interpolação bicúbica e retornando-se a THC da imagem obtida para o programa principal.

Implementação da THC em MATLAB

A seguir, apresenta-se o código da implementação em MATLAB para a detecção de linhas pelo uso da THC.

```
% ========================================
% FUNCAO PARA A LOCALIZACAO DOS CIRCULOS
% EXISTENTES EM UMA IMAGEM
% ========================================

function achacirc(rbv, dist);

im = imread('imagem5.png');
```

```
figure(1);
subplot(1,3,1);
imshow(im);
title('Imagem Original');
imc = rgb2gray(im);
subplot(1,3,2);
imshow(imc);
title('Tons de Cinza');
limiar =  [0.1 0.45];
imb = edge(im(:,:,1), 'canny', limiar);
subplot(1,3,3);
imshow(imb);
title('Bordas');
xbv = 1:dist:size(imb, 2);
ybv = 1:dist:size(imb, 1);
himg = thc(imb, xbv, ybv, rbv);
figure(2);
contourf(himg(:,:,1)); colorbar;
title('Projecao da THC');
figure(3);
surfc(himg(:,:,1));
title('Dominio 3D da THC');
circimg = mostra_circulos(im, himg, rbv);
figure(4);
imshow(circimg);
title('Imagem com circulos encontrados');
```

```
% ========================================
% FUNCAO PARA DETERMINACAO DA TRASNFORMADA DE % HOUGH PARA CIRCULOS
% ========================================

function [himg] = thc(imb, xbv, ybv, rbv)

xnb = size(xbv, 2);
ynb = size(ybv, 2);
rnb = size(rbv, 2);
xa = xbv(1);
ya = ybv(1);
xd = xbv(2) - xa;
yd = ybv(2) - ya;
[linhas, colunas] = find(imb);
acumuladorTH(1:ynb, 1:xnb, 1:rnb) = 0;
for bcont = 1:size(linhas, 1)
  xp = colunas(bcont);
  yp = linhas(bcont);
  for rcount = 1:rnb
    raio = rbv(rcount);
    rsqr = raio * raio;
    xinic = pesquisa(xp-(raio/sqrt(2)), xa,
            xnb, xd, 1);
    xfim = pesquisa(xp+(raio/sqrt(2)), xa,
           xnb, xd, 1);
    yinic = pesquisa(yp-(raio/sqrt(2)), ya,
            ynb, yd, 1);
    yfim = pesquisa(yp+(raio/sqrt(2)), ya,
           ynb, yd, 1);
   for xcont = xinic:xfim
      xc = xbv(xcont);
```

```
      xsqr = (xp - xc)*(xp - xc);
      if (rsqr >= xsqr)
        ysqrt = sqrt(rsqr - xsqr);
        yc = yp + ysqrt;
        ycount = pesquisa(yc, ya, ynb, yd);
        if (ycount ~= 0)
          acc_count = acumuladorTH(ycount,
                      xcont, rcount);
          acumuladorTH(ycount, xcont, rcount)
                               = acc_count + 1;
        end
        yc = yp - ysqrt;
        ycount = pesquisa(yc, ya, ynb, yd);
        if (ycount ~= 0)
          acc_count = acumuladorTH(ycount,
                      xcont, rcount);
          acumuladorTH(ycount, xcont, rcount)
                               = acc_count + 1;
        end
      end
    end
    for ycont = yinic:yfim
      yc = ybv(ycont);
      ysqr = (yp - yc)*(yp - yc);
      if (rsqr >= ysqr)
        xsqrt = sqrt(rsqr - ysqr);
        xc = xp + xsqrt;
        xcont = pesquisa(xc, xa, xnb, xd);
        if (xcont ~= 0)
          acc_count = acumuladorTH(ycont,
                      xcont, rcount);
          acumuladorTH(ycont, xcont, rcount)
                              = acc_count + 1;
        end
        xc = xp - xsqrt;
        xcont = pesquisa(xc, xa, xnb, xd);
        if (xcont ~= 0)
          acc_count = acumuladorTH(ycont,
                      xcont, rcount);
          acumuladorTH(ycont, xcont, rcount)
                              = acc_count + 1;
        end

      end
    end
  end
end
for rcount = 1:rnb
  himg(:, :, rcount) = imresize(acumuladorTH
              (:, :, rcount), yd, 'bicubic');
  himg(:, :, rcount) = himg(:, :,
        rcount)/max(max(himg(:, :, rcount)));
end
```

```
% ========================================
% FUNCAO PARA A PESQUISA DE INDICES NAS
% BARRAS
% ========================================

function [cont] = pesquisa(valor, a, n, d,
                  varargin)
cont = ((valor - a)/d) + 1;
cont = round(cont);
if ((cont < 1) | (cont > n))
  if (nargin == 4)
    cont = 0;
  elseif (cont < 1)
    cont = 1;
  elseif (cont > n)
    cont = n;
  end
end
```

```
% ========================================
% FUNCAO QUE DESENHA OS CIRCULOS ENCONTRADOS % NA IMAGEM ORIGINAL
% ========================================

function [circimg] = mostra_circulos(img,
                     himg, rbv)

imsize = size(img);
circ(1:imsize(1), 1:imsize(2)) = 0;
for rcont = 1:size(rbv,2)
  bwh = im2bw(himg(:,:,rcont), 0.67);
  bwh = imdilate(bwh, ones(3, 3));
  bwh = imerode(bwh, ones(3, 3));
  bwh = bwlabel(bwh);
  stat = imfeature(bwh, 'Centroid');
  for c = 1:size(stat,1)
    centro = stat(c).Centroid;
    circ = circ + desenha_circulo(centro, rbv(rcont), imsize);
  end
end
circimg = im2double(img);
circimg(:,:,1) = circimg(:,:,1) + circ;
circimg(:,:,2) = circimg(:,:,2) + circ;
circimg(:,:,3) = circimg(:,:,3) + circ;
circimg = imadjust(circimg, [0 1], [0 1]);
```

```
% ==================================
% FUNCAO PARA O DESENHO DE UM CIRCULO
% ==================================

function [circle] = desenha_circulos
                  (centro, raio, tam_img)
dimx = tam_img(2);
dimy = tam_img(1);
circle(1:dimy, 1:dimx) = 0;
```

```
for theta=1:360
  pt = centro + raio * [cos(theta*pi/180)
       sin(theta*pi/180)];
  xp = pesquisa(pt(1), 1, dimx, 1);
  yp = pesquisa(pt(2), 1, dimy, 1);
  if ((xp ~= 0) & (yp ~= 0))
    circle(yp, xp) = 1.0;
  end
end
```

Resultados obtidos na detecção de círculos - Exemplo 1

A Fig. 6 mostra a imagem original, a imagem em tons de cinza e as bordas binárias do exemplo 1. Considere-se para este exemplo raio = 11 e dist = 2.

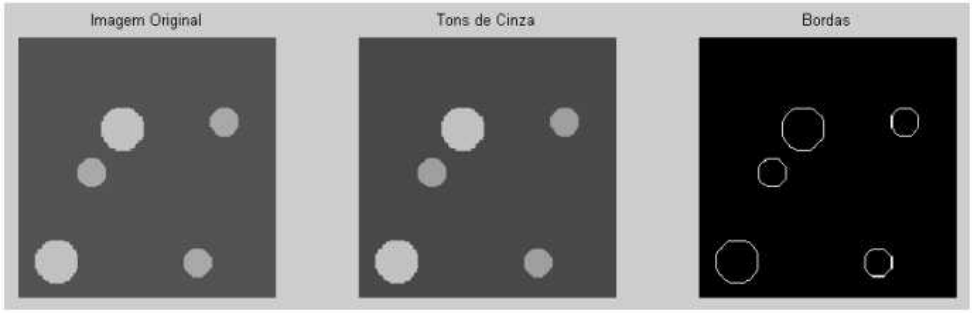

Fig. 6 - Imagem original, imagem em tons de cinza e bordas binárias

A Fig. 7 mostra a imagem 3D da matriz de acumulação de votos para o exemplo 1.

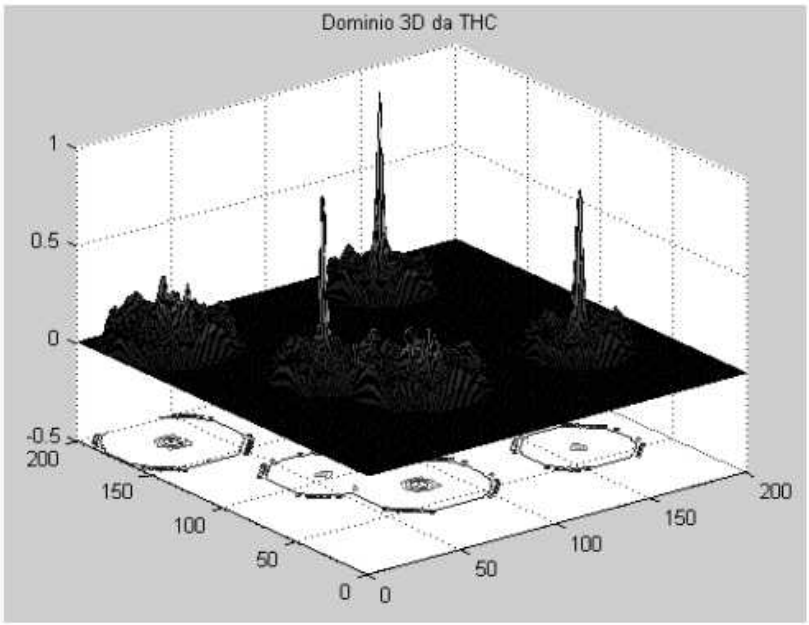

Fig. 7 - Representação 3D da matriz de acumulação de votos
da imagem do exemplo 1 para raio = 11 e dist = 2

A Fig. 8 mostra a Representação 2D da matriz de acumulação de votos e o resultado final com os círculos detectados na imagem original para o exemplo 1.

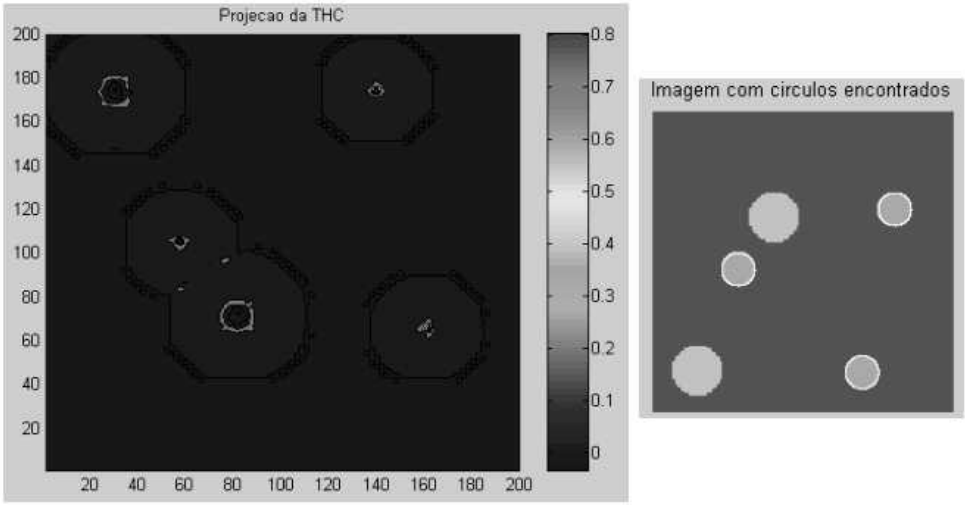

Fig. 8 - Representação 2D da matriz de acumulação de votos e o resultado final com os círculos detectados na imagem original

Resultados obtidos na detecção de círculos - Exemplo 2

A Fig. 9 mostra a imagem original, a imagem em tons de cinza e as bordas binárias do exemplo 2 [6]. Considere-se para este exemplo raio = 19 e dist = 2.

Fig. 9 - Imagem original, imagem em tons de cinza e bordas binárias

A Fig. 10 mostra a imagem 3D da matriz de acumulação de votos para o exemplo 2.

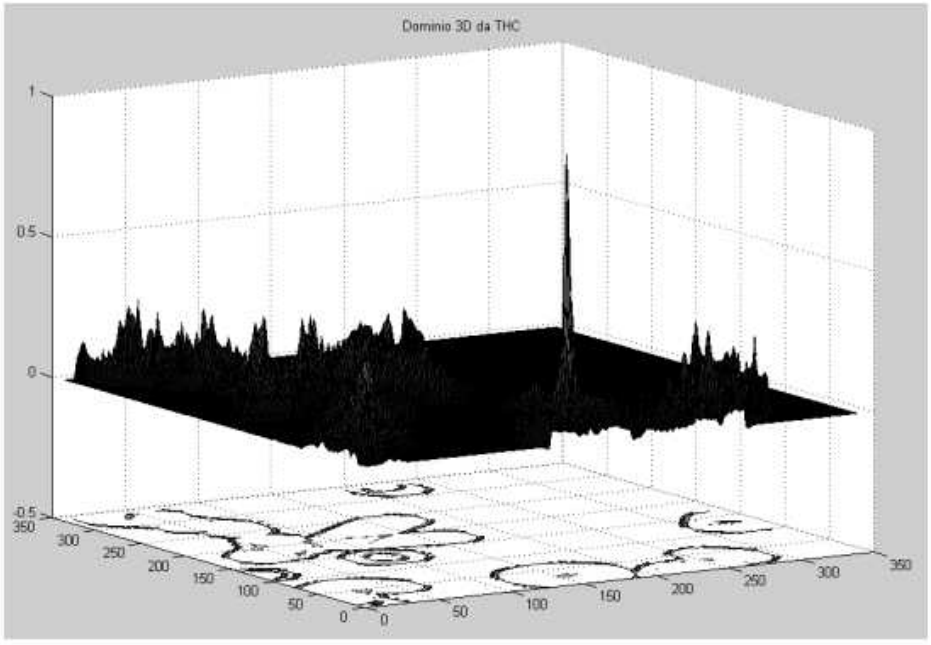

Fig. 10 - Representação 3D da matriz de acumulação de votos
da imagem do exemplo 2 para raio = 19 e dist = 2

A Fig. 11 mostra a Representação 2D da matriz de acumulação de votos e o resultado final com os círculos detectados na imagem original para o exemplo 2.

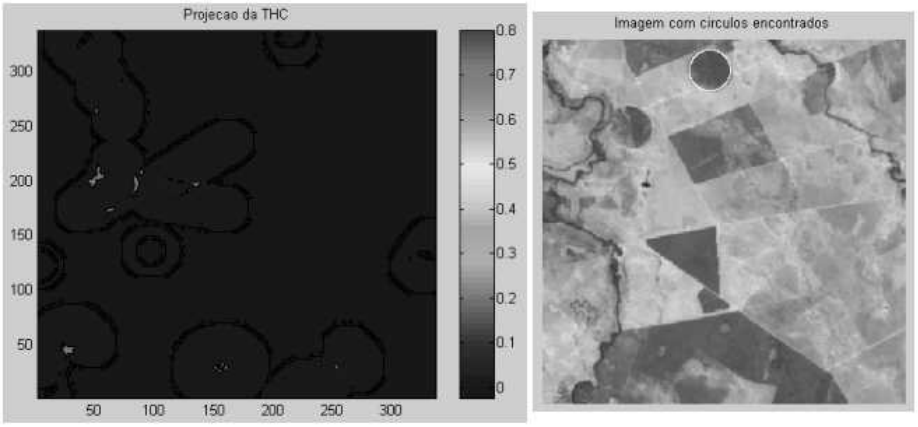

Fig. 11 - Representação 2D da matriz de acumulação de votos e o resultado
final com os círculos detectados na imagem original

Para efeito de comparação dos resultados, observe-se na Fig. 12 a mesma matriz de acumulação da Fig. 11 nas resoluções da THCMR obtidas para dist = 10 e dist = 50. Nestes casos, o círculo não foi detectado corretamente. A Fig. 13 mostra os resultados obtidos nestas resoluções.

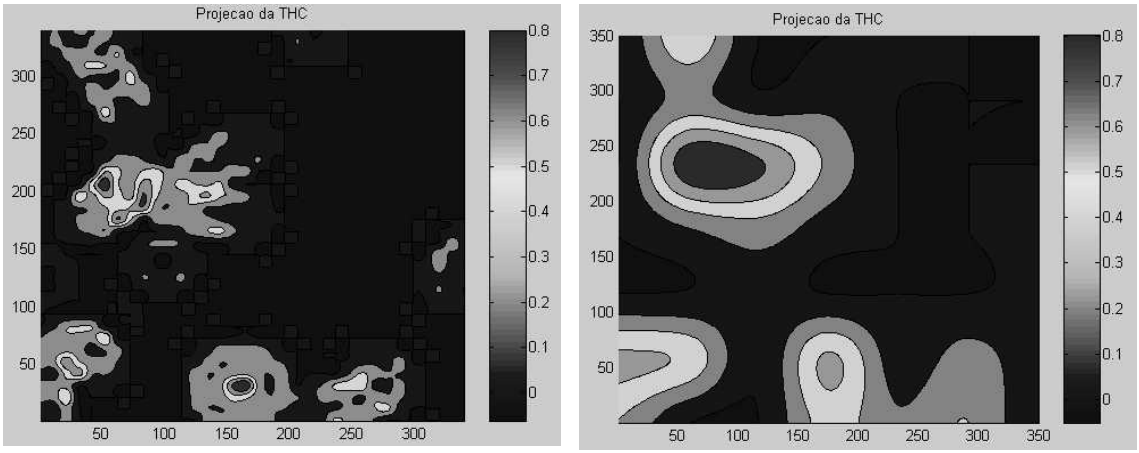

Fig. 12 - Matriz de acumulação para dist = 10 e dist = 50

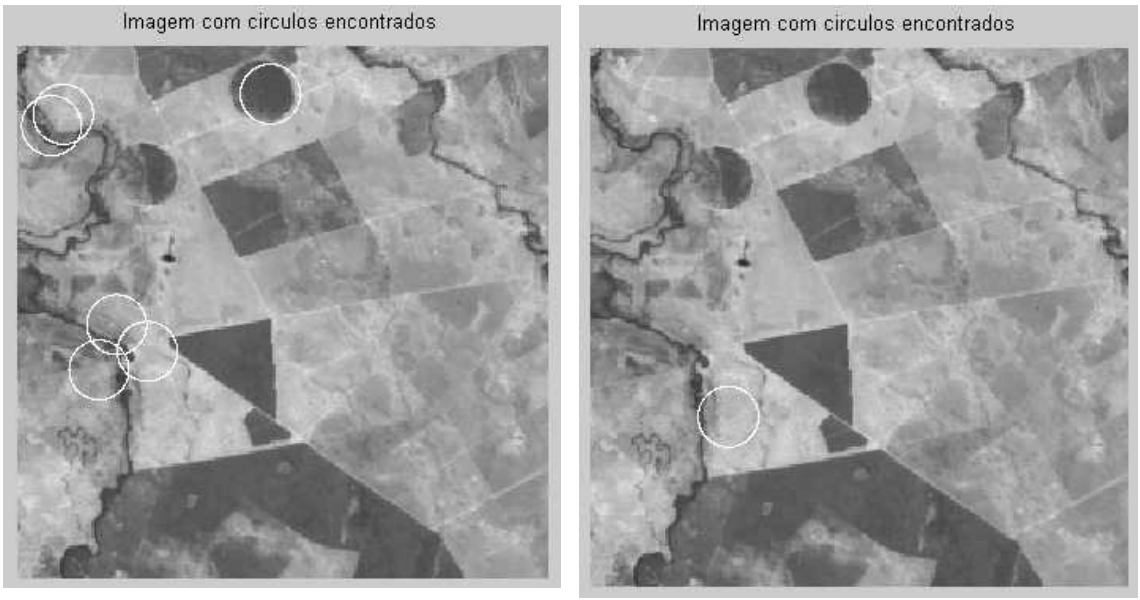

Fig. 13 - Resultados obtidos para as THC mostradas na Fig. 12

Conclusões e Futuros Trabalhos

A partir dos testes realizados com algumas imagens que apresentavam círculos, conclui-se que o algoritmo é eficiente na detecção de círculos.

O método pode ser expandido para a determinação de outras formas geométricas, desde que a sua equação seja conhecida (bastariam algumas alterações na função THC).

Pretende-se realizar algumas alterações na função THC para permitir a localização de outras formas geométricas, incluindo o emprego de um algoritmo de geração de polígonos para a detecção de triângulos, pentágonos, hexágonos, etc.

Referências

[1] **ANDERSEN**, J.; Seibel, E. Real-time Hazard detection via machine vision for wearable low vision aids. IEEE Fifth International Symposium on Wearable Computers (ISWC'01), October 08 - 09, Zurich, Switzerland, 2001.

[2] **ATIQUZZAMAN**, M. Coarse-to-Fine Search Technique to Detect Circles in Images. International Journal of Advanced Manufacturing Technology, Vol. 15, Issue 12, pp. 96-102, 1999.

[3] **ATIQUZZAMAN**, M. Multiresolution Hough transform - na efficient method of detecting pattern in images. IEEE Transactions on Pattern Analysis and Machine Intelligence, vol. 14, no. 11, pp. 1090-1095, November 1992.

[4] **CANNY**, J. A computational approach to edge detection. IEEE Transactions on PAMI-8, no. 6, pp. 679-698, 1986.

[5] **DUDA**, R. O.; Hart, P. E. Use of the Hough transformation to detect lines and curves in pictures. Communications of the ACM., vol. 15, pp. 11-15, 1972.

[6] **HADAD**, R. M.; Araújo A. A.; Martins Júnior, P. P. Identificação de Formas Circulares em Imagens de Satélites. I Workshop em Tratamento de Imagens, Belo Horizonte, Junho, 2000.

[7] **HIERKEGAARD**, P. A method for detection of circular arcs based on the Hough transform. Machine Vision and Applications, vol. 5, pp. 249-263, 1992.

[8] **KWATRA,** V. Detecting coins using Hough Transform. Disponível na web em http://www.cc.gatech.edu/~kwatra/computer_vision/coins/coins.html.

[9] **YUEN**, H. K.; Princen, J.; Illingworth, J.; Kittler, J. Comparative study of Hough transform methods for circle finding. Image and Vision Computing, vol. 8, no. 1, pp. 71-77, February 1990.