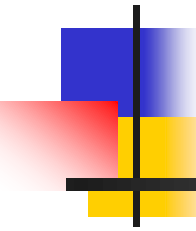


Detecção de Formas

Carlos Alexandre Mello

Pós-Graduação em Ciência da Computação

Nível Intermediário de Visão

- Baixo Nível
  - Processos de segmentação, detecção de bordas
- Nível Intermediário
  - Relacionada com a obtenção de informação abstrata sobre imagens
  - Nesse estágio, não estamos preocupados em converter uma imagem em outra como nos processamentos anteriores

Detecção de Formas

- Segmentos de Retas
- Círculos
- Elipses
- Cantos

Detecção de Segmentos de Retas

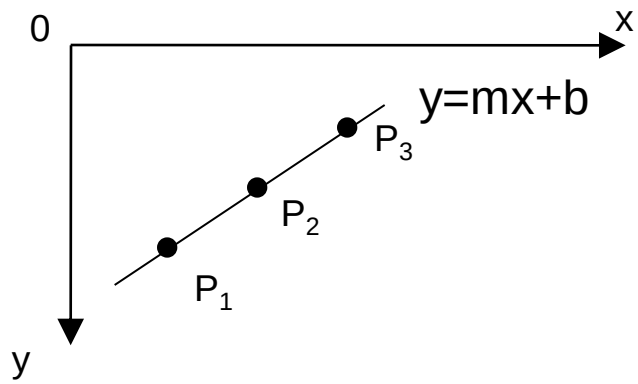
- A Transformada de Hough
  - A equação cartesiana da reta é dada por
    - y = mx + b
    - onde m = declividade e b = ordenada
  - Essa equação pode ser re-organizada como:
    - b = y – mx
    - No caso, m e b passam a ser os parâmetros com x e y constantes
  - O Plano mb é conhecido como espaço de parâmetros

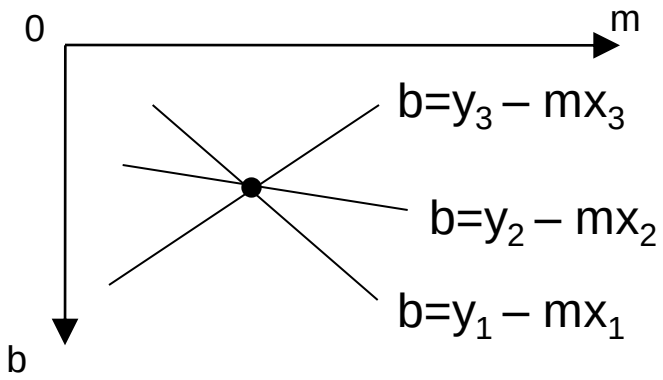
Detecção de Segmentos de Retas

- A Transformada de Hough – Ideia básica
  - Assim, todos os segmentos de retas que passam pelo ponto $P_1(x_1, y_1)$ são representados no plano mb pela equação $b = y_1 – mx_1$
  - Da mesma forma, a equação $b = y_2 – mx_2$ representa o conjunto de segmentos de reta que passam por $P_2(x_2, y_2)$
  - O ponto (m, b) é comum a esses dois conjuntos de segmentos que passam por P_1 e P_2

Detecção de Segmentos de Retas

- A Transformada de Hough – Ideia básica
  - De fato, todos os pontos que são colineares no plano da imagem se interceptam em um mesmo ponto no espaço de parâmetros





Detecção de Segmentos de Retas

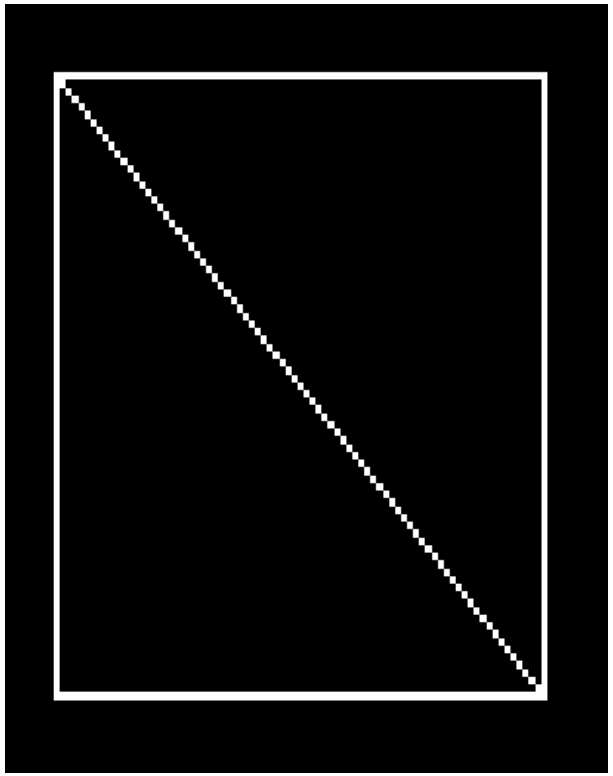
- A Transformada de Hough – Ideia básica
  - Pontos nos quais muitas retas se interceptam no espaço de parâmetros correspondem a muitos pixels colineares, os quais potencialmente formam um segmento de reta na imagem
  - Isso significa que as coordenadas no espaço (m, b) do ponto de interseção fornecem parâmetros da reta no espaço (x, y) que contém esses pontos

Detecção de Segmentos de Retas

- A Transformada de Hough – Ideia básica

Cada linha reta na imagem pode ser descrita por uma equação

Cada ponto branco, se considerado em separado, poderia estar em um número infinito de linhas retas



Detecção de Segmentos de Retas

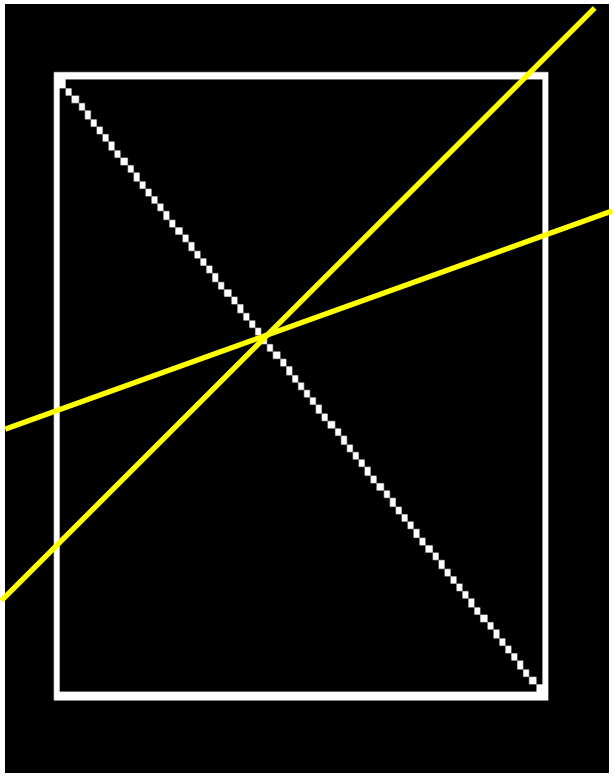
- A Transformada de Hough – Ideia básica

Cada linha reta na imagem pode ser descrita por uma equação

Cada ponto branco, se considerado em separado, poderia estar em um número infinito de linhas retas

Na transformada de Hough, cada ponto acrescenta um voto para cada linha que ele poderia estar

As linhas com a maior votação ganham



Detecção de Segmentos de Retas

- A Transformada de Hough
  - Esse conceito forma a base da transformada de Hough para detecção de retas
  - Pixels são convertidos em retas no espaço (m, b) e então os pontos de interseção de várias retas são localizados e agrupados em segmentos de retas

Detecção de Segmentos de Retas

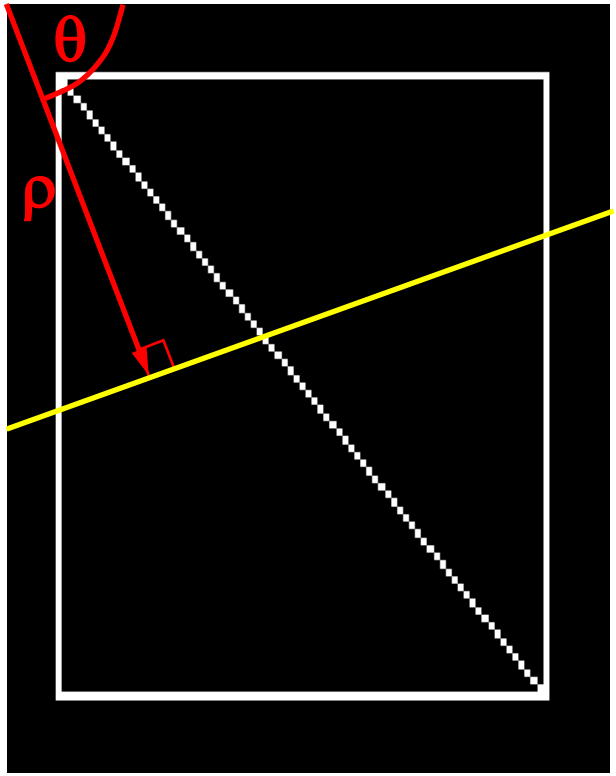
- A Transformada de Hough
  - A equação cartesiana da reta apresenta problemas quando as retas forem aproximadamente verticais (declividade tende a infinito)
  - Uma maneira de resolver isso é usar a equação da reta na forma polar
    - $\rho = x.\cos\theta + y.\text{sen}\theta$
    - em que ρ é a distância perpendicular da origem (0,0) à reta e θ é o ângulo formado entre a reta perpendicular e o eixo x

Detecção de Segmentos de Retas

- A Transformada de Hough

Qualquer linha pode ser representada por dois números

A linha amarela será representada por (ρ, θ)



i.e., representamos uma linha usando:

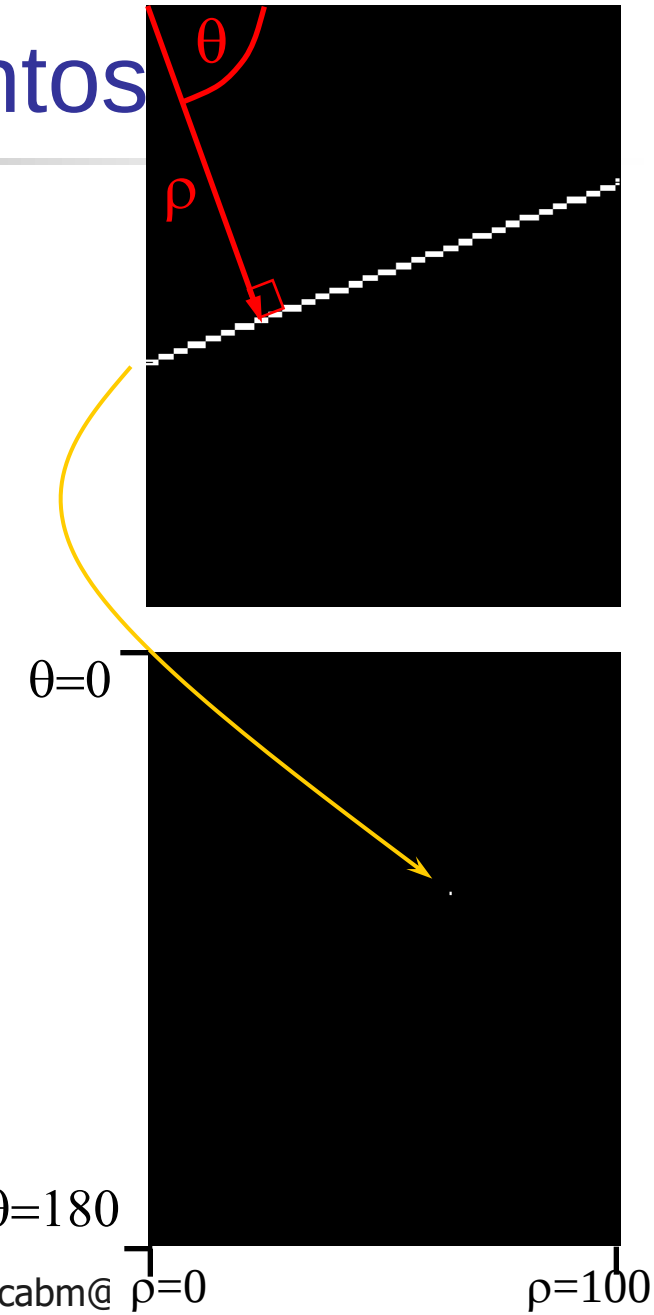
- Uma linha partindo da origem
  - de comprimento ρ
  - formando ângulo θ com a horizontal

Detecção de Segmentos

- A Transformada de Hough

  Podemos usar (ρ, θ) para representar qualquer linha no espaço da imagem

  Esse é o espaço de Hough



Detecção de Segmentos de Retas

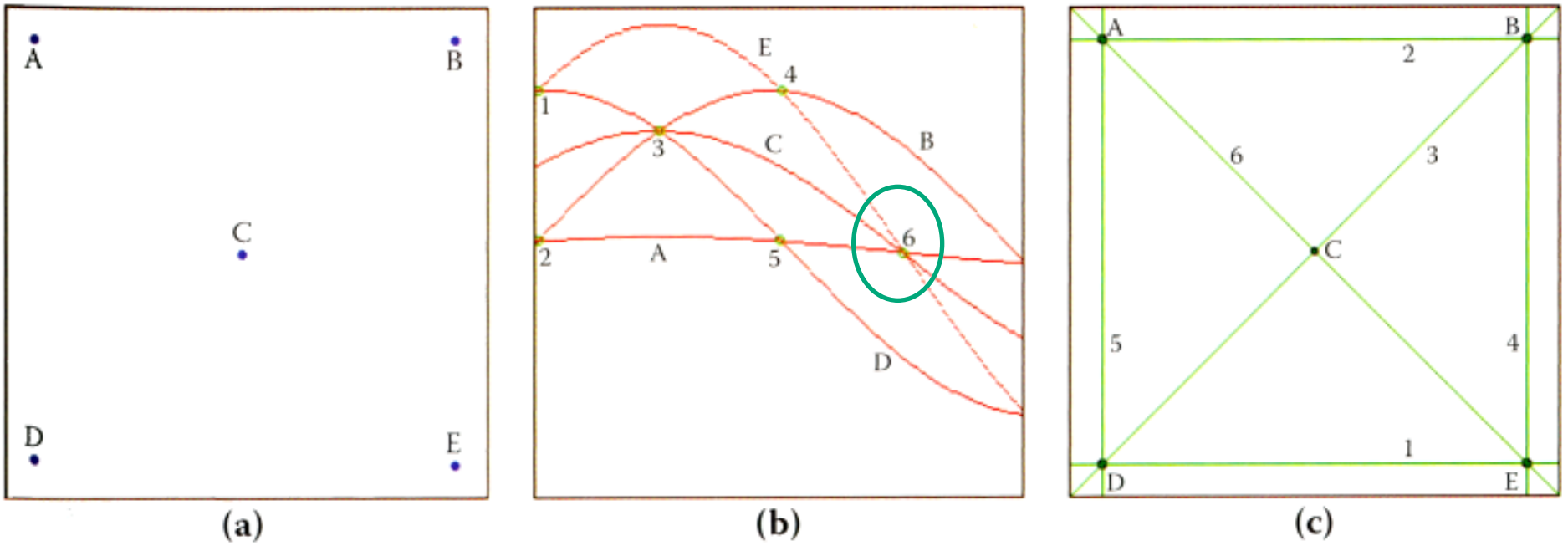
- A Transformada de Hough
  - A conversão da equação cartesiana da reta para a forma polar pode ser feita por meio das relações:
    - $\text{sen}\ \theta = -1/(\sqrt{m^2 + 1})$
    - $\cos\ \theta = m/(\sqrt{m^2 + 1})$
    - $\rho = -b/(\sqrt{m^2 + 1})$
  - Assim, ao invés de utilizar coordenadas no espaço (m, b), as coordenadas no espaço (ρ, θ) são mais utilizadas

Detecção de Segmentos de Retas

- A Transformada de Hough
  - No espaço (ρ, θ) ou *espaço de Hough* pontos colineares no espaço (x, y) correspondem agora a curvas senoidais que se interceptam no espaço (ρ, θ)
  - Para implementar a transformada de Hough, o espaço (ρ, θ) deve ser discretizado
  - Com θ medido em relação ao eixo x, seu valor varia de 0 a 180º
  - ρ varia de 0 a $\sqrt{m^2 + n^2}$ para uma imagem mxn
    - Maior distância da imagem (comprimento da diagonal)

Detecção de Segmentos de Retas

- A Transformada de Hough – Ideia básica



Cada ponto em (a) gera senóides em no espaço de Hough (b); cada ponto de encontro entre linhas legendado com um número nesse espaço gera linhas na imagem no espaço real (c). Em (b) ainda, circulamos a interseção de A, C, E, correspondendo a uma reta.

Detecção de Segmentos de Retas

- A Transformada de Hough
  - Após essa discretização do espaço (ρ, θ), cada célula do espaço é considerada como um acumulador, inicializadas com valor zero
  - Para cada ponto (x, y) na imagem, k pontos colineares de uma reta x.cos θ + y.sen θ = ρ levam a k curvas senoidais no plano $\rho\theta$ que se interceptam em (ρ, θ) no espaço de parâmetros
  - Incrementando-se θ e resolvendo para o valor de ρ correspondente há k posições no acumulador associadas à célula determinada pelo ponto de interseção (ρ, θ)

Detecção de Segmentos de Retas

- A Transformada de Hough
  - Ao final desse procedimento, um valor k em uma célula corresponde a k pontos no plano $\rho\theta$ que satisfazem a equação da reta
  - A precisão da colinearidade desses pontos é determinada pelo número de subdivisões no plano $\rho\theta$
  - Os valores mais altos no espaço de Hough correspondem aos parâmetros que caracterizam as retas da imagem

Detecção de Segmentos de Retas

- A Transformada de Hough – Algoritmo

1. Inicializa H[ro, θ]=0
2. Para cada ponto [x,y] na imagem

   for θ = 0 to 180

   - H[ro, θ] += 1

3. Encontre os valores de (ro, θ) onde H[ro, θ] é máximo
4. A linha detectada na imagem é dada por ro=x.cosθ + y.senθ

Detecção de Segmentos de Retas

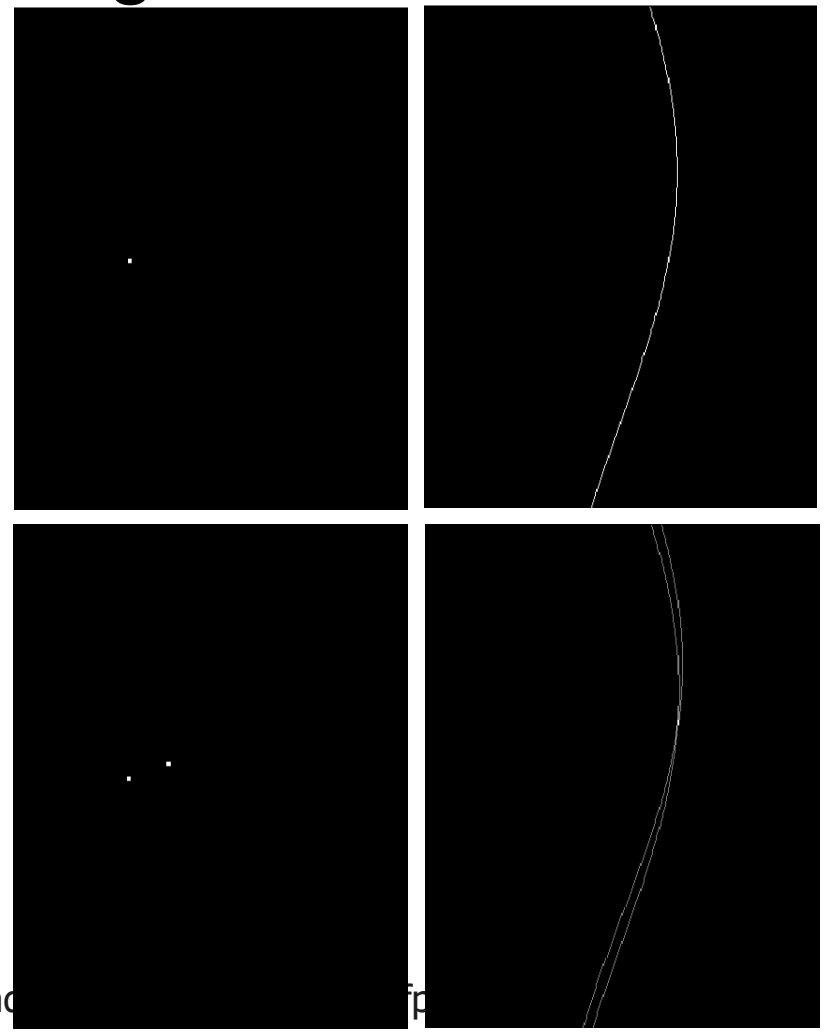
- A Transformada de Hough

Um ponto na imagem corresponde a uma curva senoidal no espaço de Hough

Dois pontos correspondem a duas curvas

A interseção dessas curvas tem dois “votos” na matriz de acumuladores

Essa interseção representa a linha reta na imagem que passa por ambos os pontos

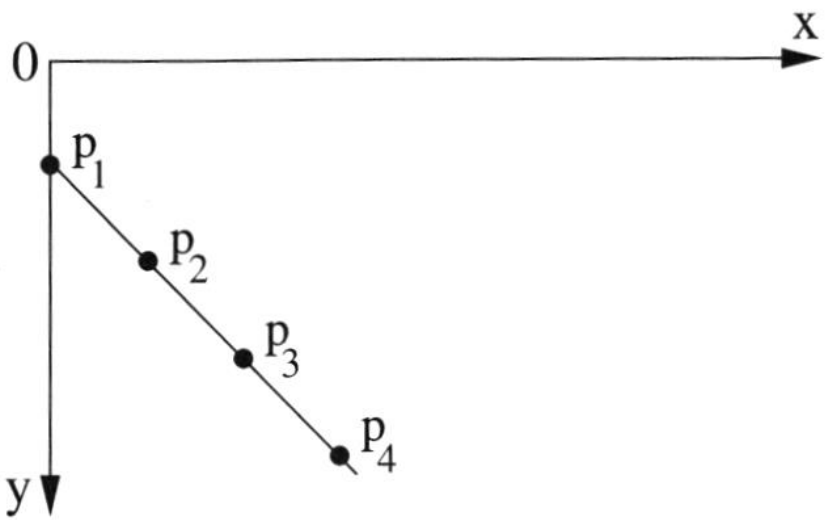


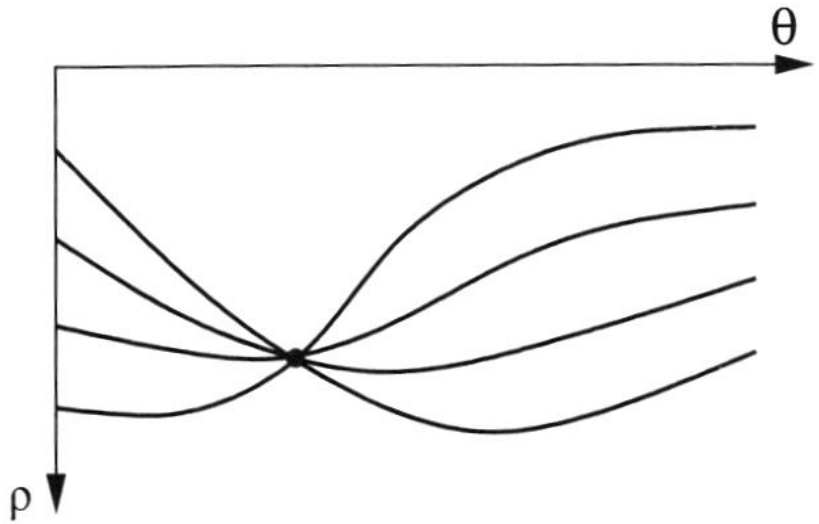
Detecção de Segmentos de Retas

- A Transformada de Hough
  - Exemplo: Considere os pontos colineares $P_1(0,1)$, $P_2(1,2)$, $P_3(2,3)$ e $P_4(3,4)$
  - Esses pontos são mapeados para o espaço de parâmetros como 4 curvas senoidais
  - Como os pontos são colineares, eles se tocam no espaço de parâmetros
    - No caso, em $\rho = -\sqrt{2}/2$, $\theta = 3\pi/4$
  - Logo: $-\sqrt{2}/2 = x.\cos(3\pi/4) + y.\text{sen}(3\pi/4)$
  - $\Rightarrow y = x + 1$

Detecção de Segmentos de Retas

- A Transformada de Hough



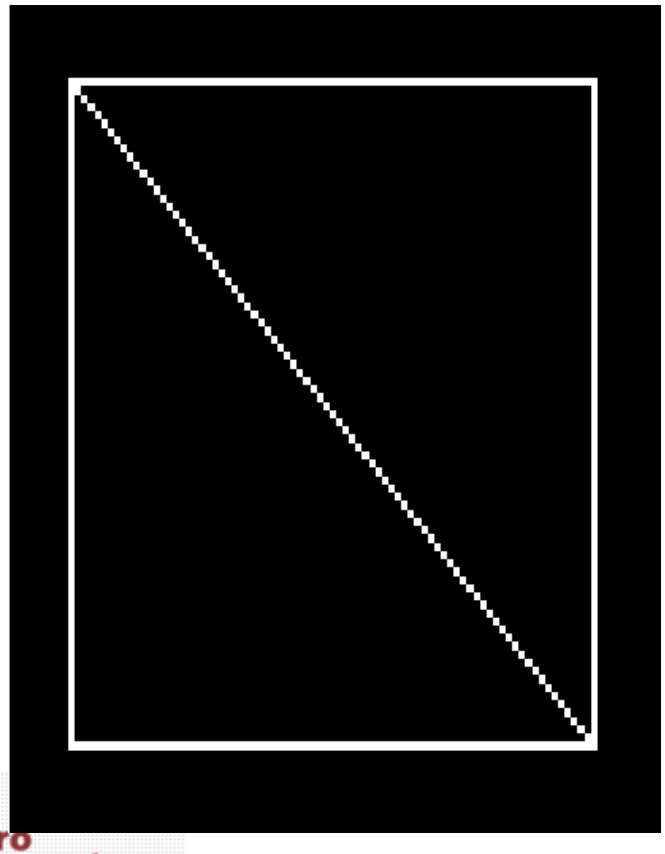


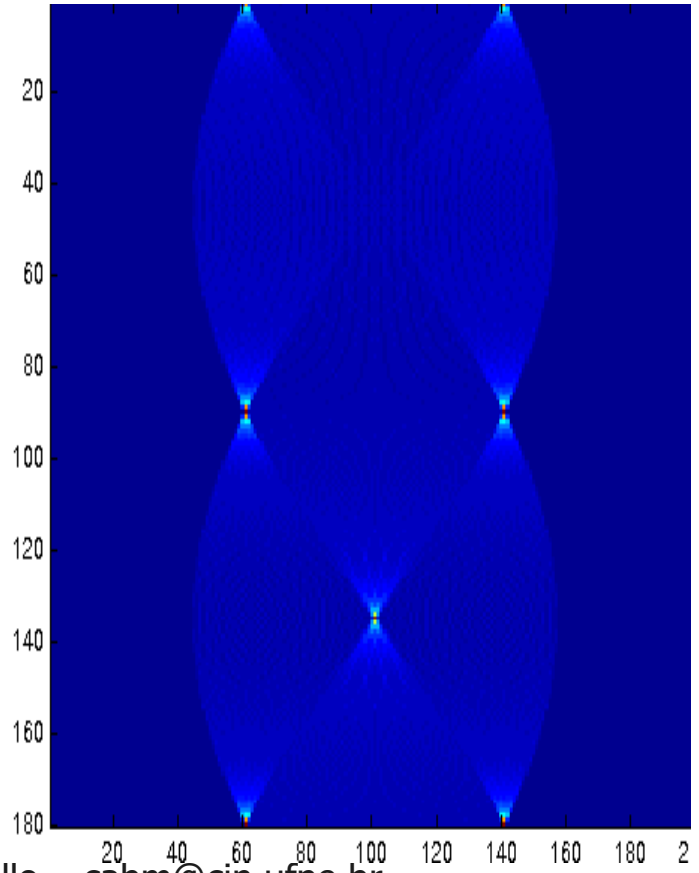
Detecção de Segmentos de Retas

- A Transformada de Hough
  - Vantagens:
    - Tolerante a falhas nas bordas
    - Pouco afetada por ruído
    - Pouco afetada por oclusões na imagem
  - Desvantagem: Alto custo computacional

Detecção de Segmentos de Retas

- A Transformada de Hough
  - Exemplo:





Detecção de Segmentos de Retas

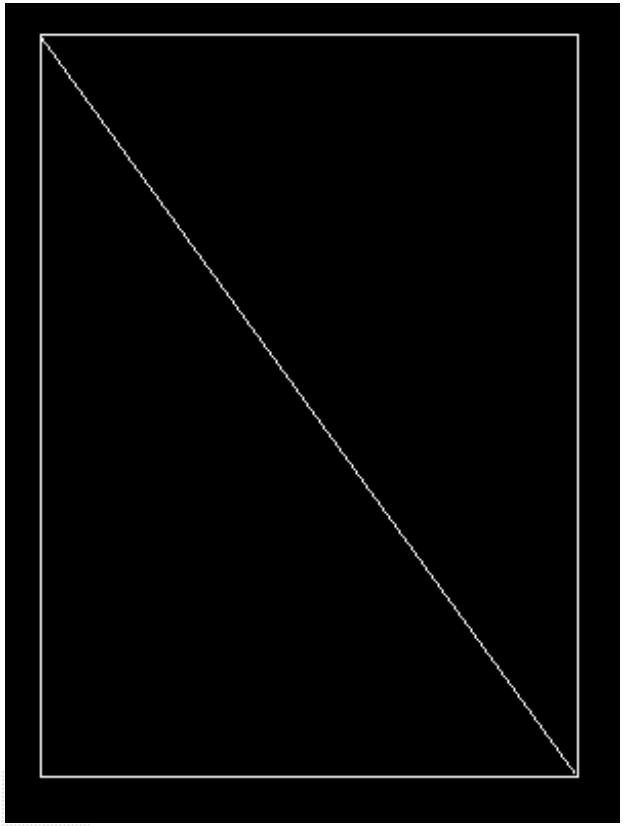
- A Transformada de Hough
  - Localizando linhas no Matlab

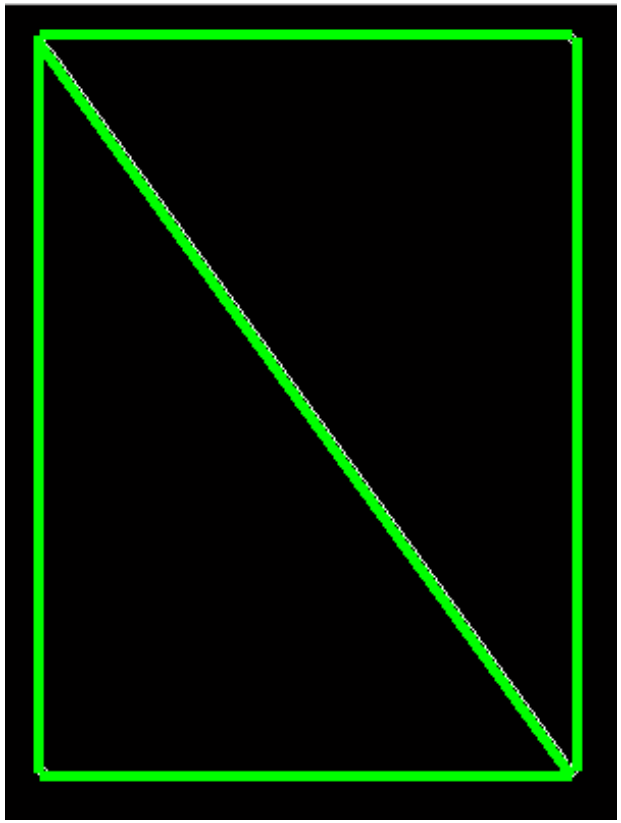
```
function ht_line(nome);
bw = imread(nome);
bw = edge (bw, 'canny');
[H,theta,rho] = hough(bw);
peaks  = houghpeaks(H,5);
lines = houghlines(bw,theta,rho,peaks);
imshow(bw)
hold on
for k = 1:numel(lines)
    x1 = lines(k).point1(1);
    y1 = lines(k).point1(2);
    x2 = lines(k).point2(1);
    y2 = lines(k).point2(2);
    plot([x1 x2],[y1 y2],'Color','g','LineWidth', 4)
end
hold off
```

Detecção de Segmentos de Retas

- A Transformada de Hough
  - Localizando linhas no Matlab

Imagem original





Linhas detectadas

Detecção de Segmentos de Retas

- A Transformada de Hough
  - A transformada de Hough pode ser usada também para detecção de inclinação
    - *Skew detection*
  - A imagem do espaço de Hough pode ter um pico em um ângulo que corresponde ao ângulo de inclinação

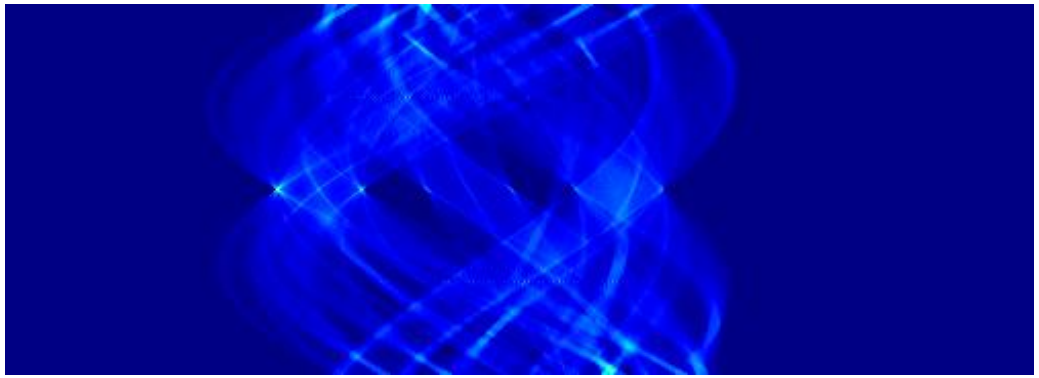
Detecção de Segmentos de Retas

- A Transformada de Hough

```
function z = hough(nome)
im = imread(nome);                          [lin, col] = size(im);
center_x = round(col/2);                    center_y = round(lin/2);
conversao = pi/180;                         rmax = round(sqrt(lin^2 + col^2)/2);
z = zeros(181, 2*rmax + 2);                 tmax = 0; tmval = 0;
for i=1:lin
    for j=1:col
        x = double(im(i, j));
        if (x == 1)
            for omega = 0:180
                r = round((i - center_y)*sin(omega*conversao) + (j - center_x)*cos(omega*conversao));
                z(omega + 1, rmax + r) = z(omega + 1, rmax + r) + 1;
            end
        end
    end
end
for i = 1:181
    for j = 1:2*rmax + 2
        if (z(i, j) > tmval)
            tmval = z(i, j);
            tmax = i;
        end
    end
end
theta = 90 – tmax;
z = uint8(z); imshow(z); map = colormap(jet);
```

Detecção de Segmentos de Retas

- A Transformada de Hough
  - Exemplo:



Inclinação detectada = -1º

Detecção de Segmentos de Retas

- A Transformada de Hough
  - Exemplo:

Inclinação imposta = 10º (direita)
Inclinação detectada = -11º

Detecção de Segmentos de Retas

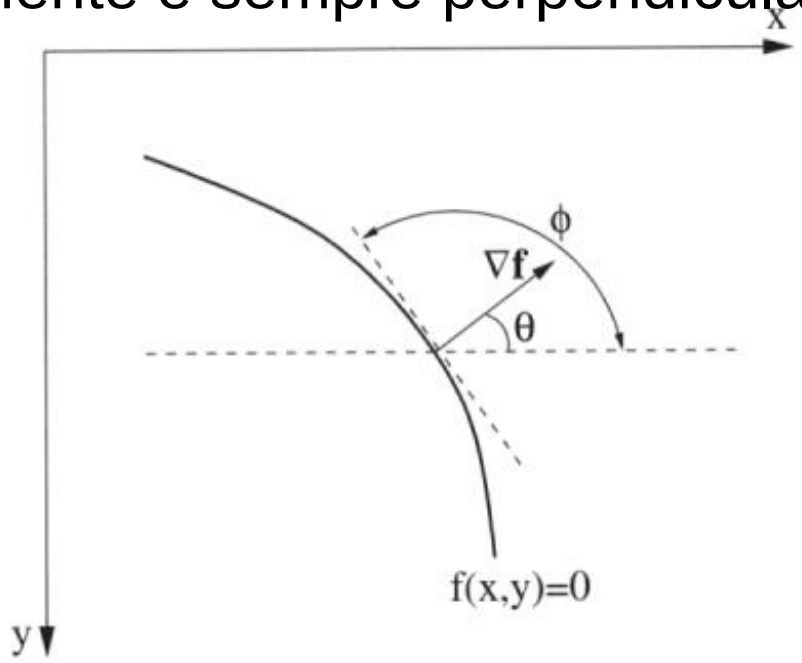
- A Transformada de Hough
  - Exemplo:

Inclinação imposta = 10º (esquerda)
Inclinação detectada = 9º

Detecção de Segmentos de Retas

- Variação
  - Uso de gradiente
    - Forma de diminuir o custo da transformada de Hough
    - A direção θ do gradiente é sempre perpendicular à direção tangente Φ

O valor do parâmetro θ coincide então com a direção do gradiente no ponto



Detecção de Segmentos de Retas

- Variação
  - Uso de gradiente
    - Dada a magnitude e a direção do gradiente
    - Com a direção conhecida, basta aplicar a relação
      - $\rho = x.\cos\theta + y.\text{sen}\theta$
    - para cada ponto (x,y) da magnitude com valor diferente de zero
    - Tornando o passo 3 do algoritmo anterior mais simples

Detecção de Circunferência

- A Transformada de Hough pode ser usada para detecção de circunferências em uma imagem
- A equação cartesiana de uma circunferência é dada por:
  - $(x - a)^2 + (y - b)^2 = r^2$
  - em que (a, b) são as coordenadas do centro da circunferência e r é seu raio
- Assim, a estrutura do acumulador é tridimensional

Detecção de Circunferência

- Para cada pixel (x, y), a célula (a, b, r) é incrementada se o ponto (a, b) estiver à distância r do ponto (x, y)
- Se um centro específico (a, b) de uma circunferência de raio r é frequentemente encontrado no espaço de parâmetros é provável que uma circunferência de raio r e centro (a, b) exista na imagem
- Picos no espaço de parâmetros correspondem a centros de circunferências

Detecção de Circunferência

- As equações paramétricas da circunferência em coordenadas polares são:
  - $x = a + r.\cos\theta$
  - $y = b + r.\sen\theta$
- ou
  - $a = x - r.\cos\theta$
  - $b = y - r.\sen\theta$

Detecção de Circunferência

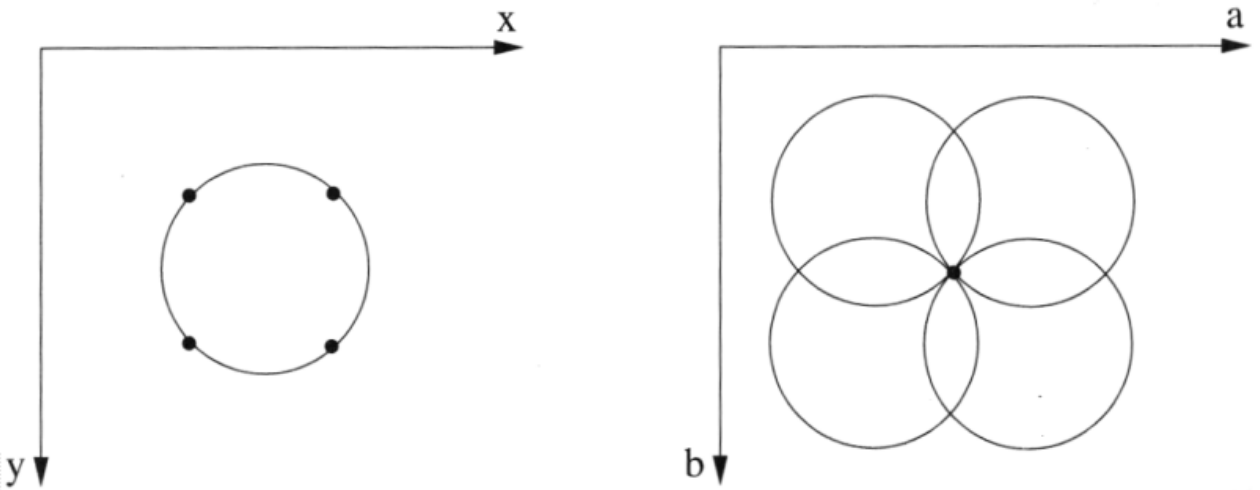
- Se o raio da circunferência for previamente conhecido, então é necessário incrementar o acumulador para o ponto (a, b) dado pelas equações anteriores
- Nesse caso, circunferências com valor de raio r poderão ser detectadas a cada aplicação do algoritmo
- Isso acontece apenas se o raio é conhecido

Detecção de Circunferência

- O custo computacional pode ser reduzido novamente com o uso de gradientes
- Dada a direção θ do gradiente em um ponto (x, y) de borda, pode-se calcular o seno e o cosseno de θ
- Juntando as equações anteriores podemos ter:
  - $b = a.tg\ \theta - x.tg\ \theta + y$
- eliminando o raio

Detecção de Circunferência

- A partir da direção θ do gradiente em cada ponto (x, y) de borda, as células acumuladoras no espaço de parâmetros (a,b) são incrementadas de acordo com a equação anterior



Detecção de Circunferência

- Algoritmo
  - 1. Criar o espaço de parâmetros (a, b)
  - 2. Inicializar as células de acumulação M(a, b) com zero
  - 3. Calcular a magnitude e a direção do gradiente
  - Para cada ponto (x, y) da magnitude diferente de zero, incrementar todos os pontos na matriz (a, b) de acordo com $b = a.tg\theta - x.tg\theta + y$
    - Para todos os valores possíveis de a

Detecção de Circunferência

- Exemplo:
  - Do Canny Edge Detector:

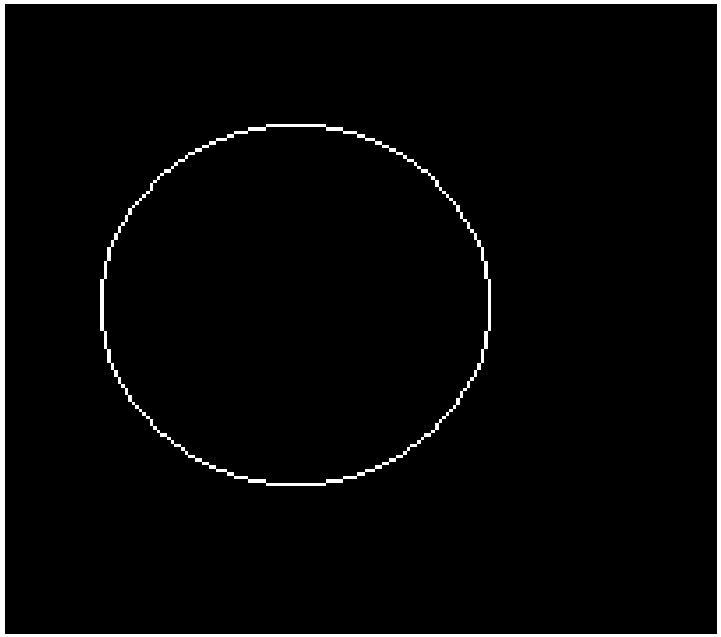
```
G = fspecial ('Gaussian', [5 5], 1.4);
im2 = filter2(G, im);
hx = [-1 0 1; -2 0 2; -1 0 1];
hy = [1 2 1; 0 0 0; -1 -2 -1];
imx = filter2 (hx, im2);
imy = filter2 (hy, im2);
gra_mag = sqrt(imx.^2 + imy.^2);
gra_dir = atan2(imy, imx);
```

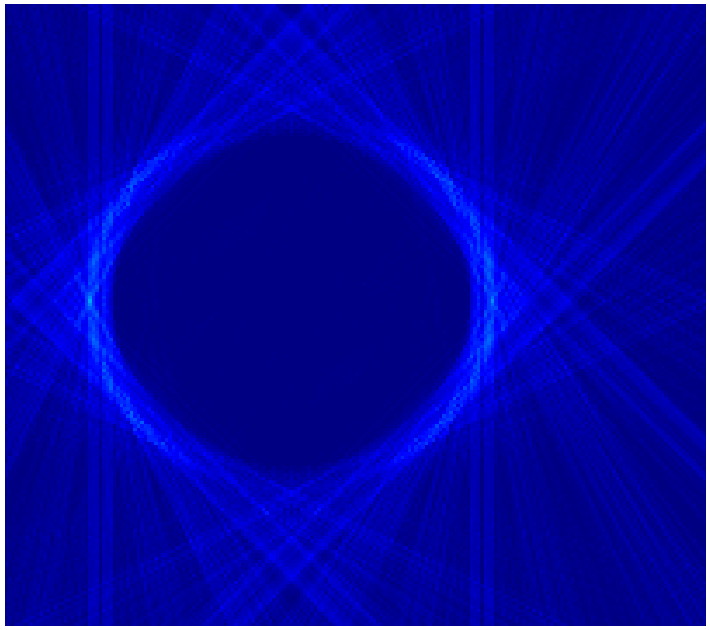
Detecção de Circunferência

- Exemplo:

```
acc = zeros(lin, col);
for i=1:lin
    for j=1:col
        elemento = gra_mag(i, j);
        if (elemento ~= 0)
            for a=1:lin
                coluna = round(a*tan(gra_dir(i, j)) - i*tan(gra_dir(i, j)) + j);
                if ((coluna > 0) & (coluna <= col))
                    acc(a, coluna) = acc(a, coluna) + 1;
                end
            end
        end
    end
end
```

Detecção de Circunferência

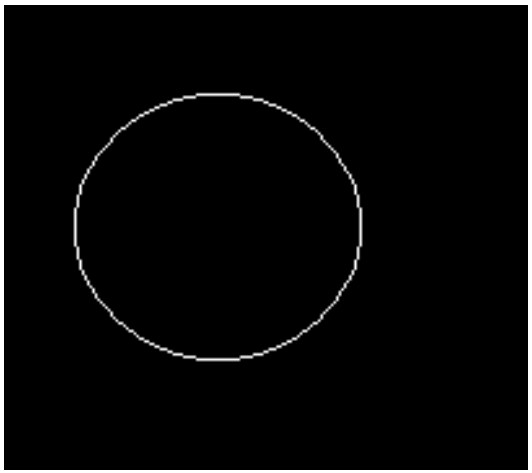


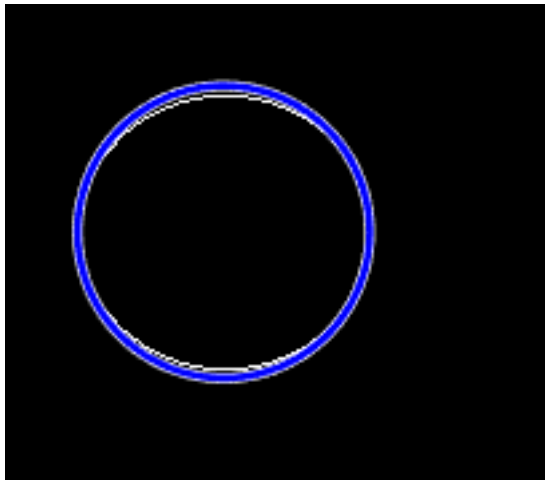


Detecção de Circunferência

Função imfindcircles (MatLab)

```
>> [centers, radii] = imfindcircles(im2,[50 100],'ObjectPolarity','bright',
...'Sensitivity',0.9, 'EdgeThreshold', 0.3);
```

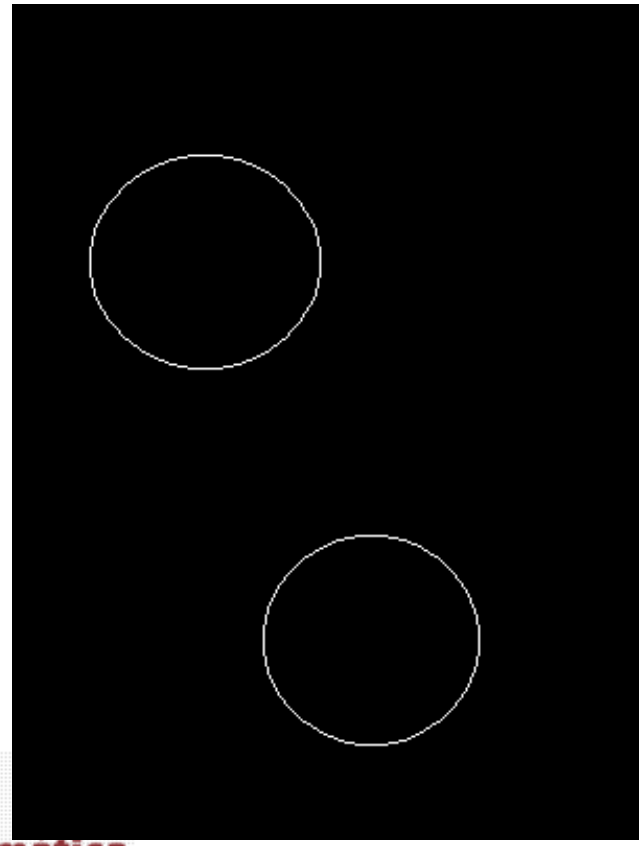


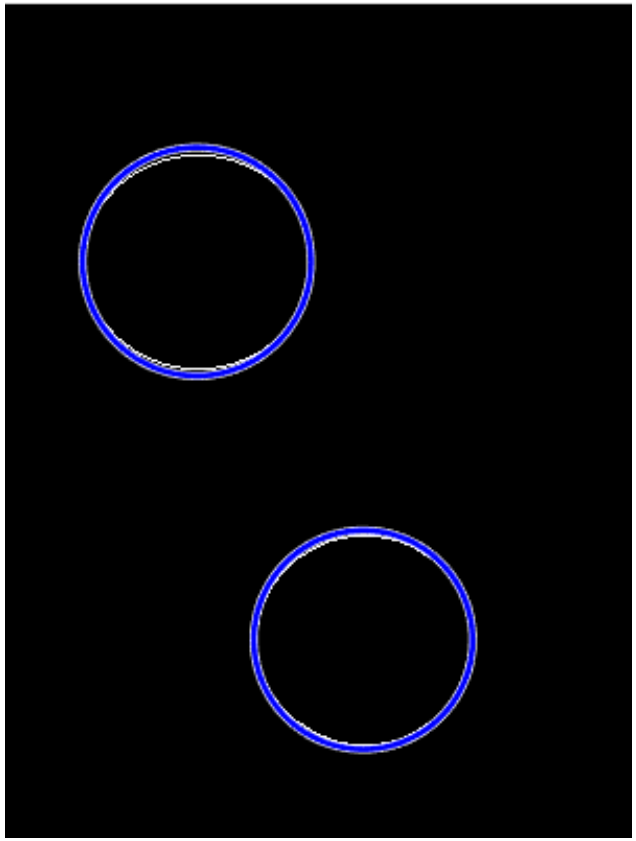


Detecção de Circunferência

Função imfindcircles (MatLab)

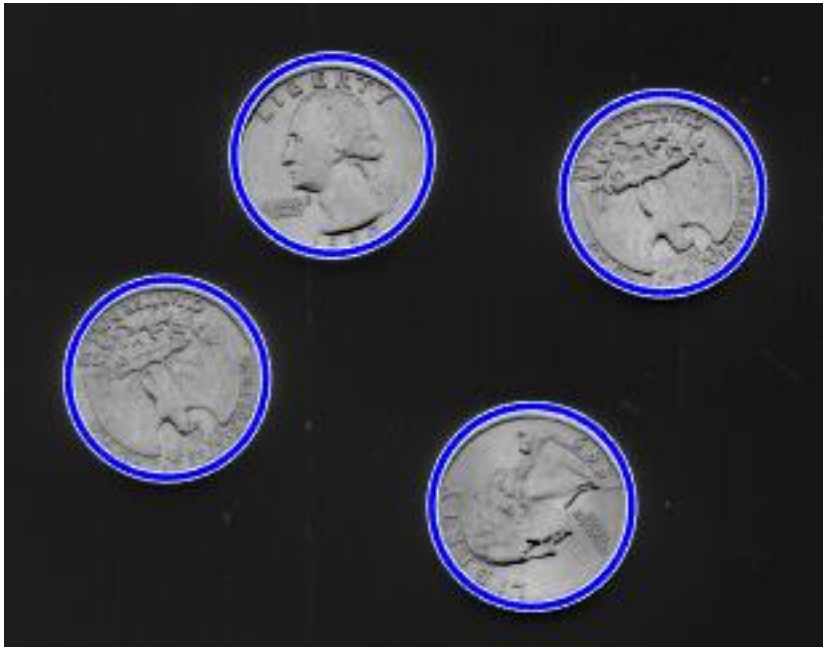
```
>> [centers, radii] = imfindcircles(im2,[50 100],'ObjectPolarity','bright',
...'Sensitivity',0.9, 'EdgeThreshold', 0.3);
```





Detecção de Circunferência

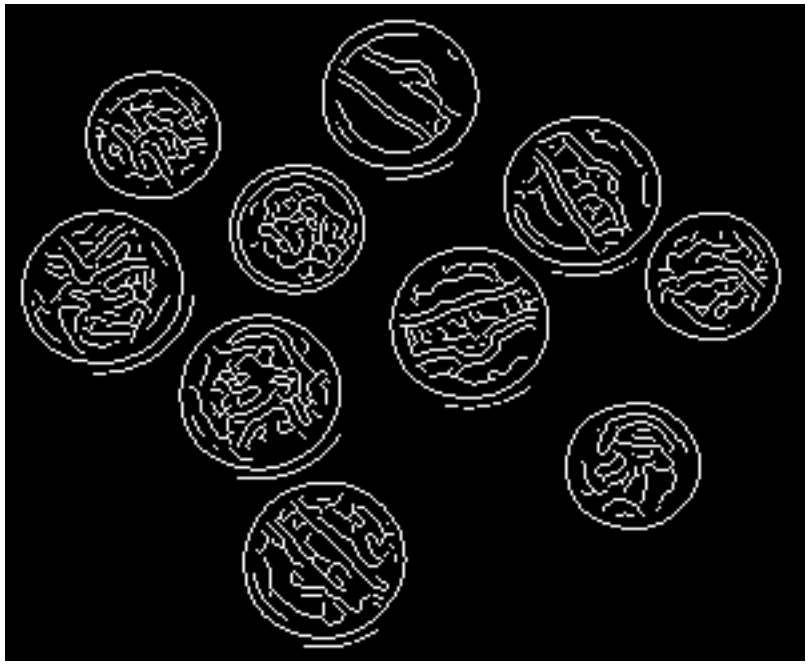
Função imfindcircles (MatLab)

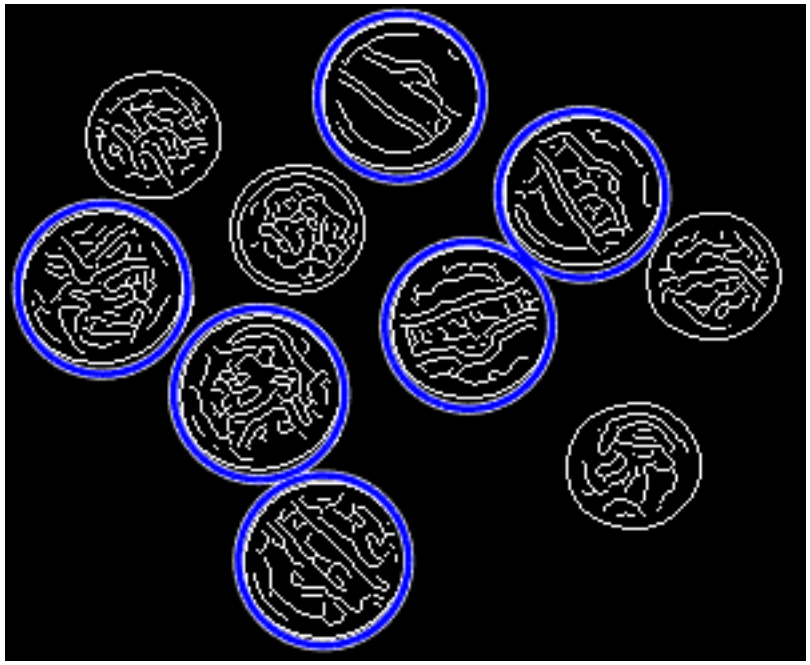


Detecção de Circunferência

Função imfindcircles (MatLab)

```
>> [centers, radii] = imfindcircles(im2,[30 60],'ObjectPolarity','bright',
...'Sensitivity',0.9, 'EdgeThreshold', 0.3);
```



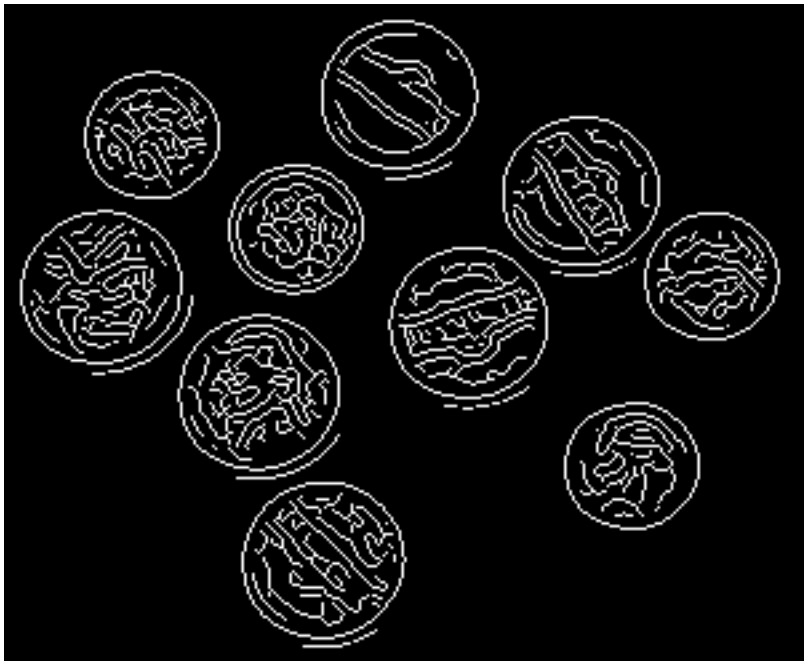


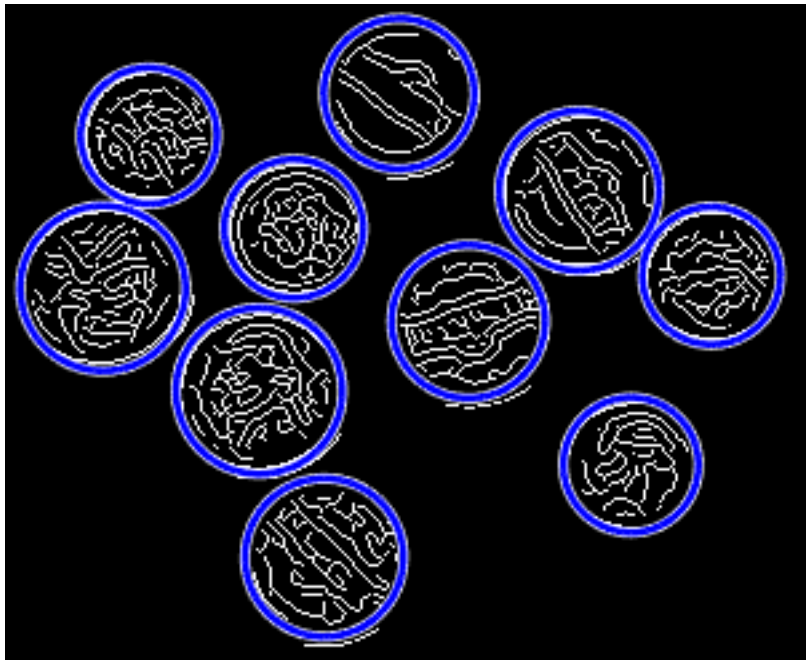
Nem todas as circunferências foram detectadas por causa do intervalo no raio.

Detecção de Circunferência

Função imfindcircles (MatLab)

>> [centers, radii] = imfindcircles(im2,[20 60],'ObjectPolarity','bright',
...'Sensitivity',0.9, 'EdgeThreshold', 0.3);



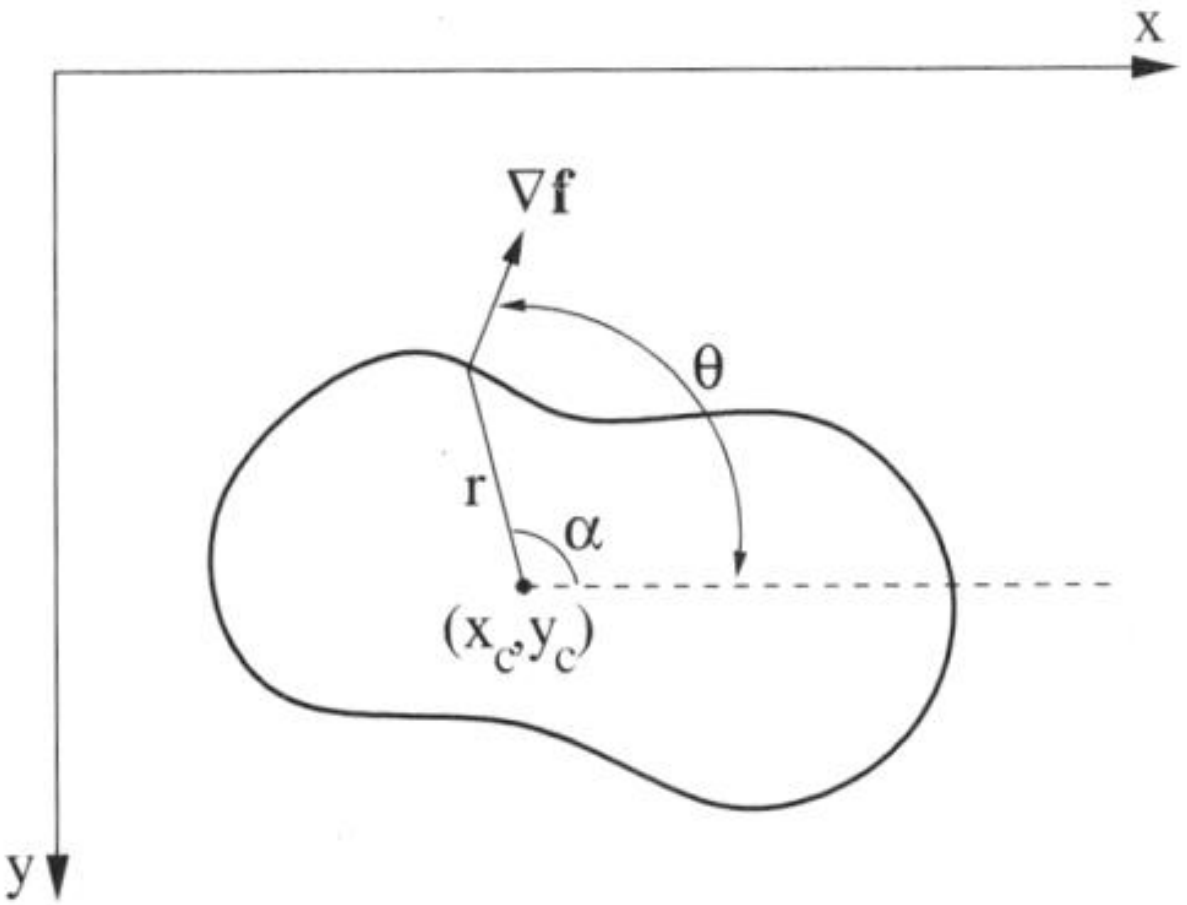


Todas foram detectadas agora.

Generalização da Transformada de Hough

- A transf de Hough pode ser generalizada para encontrar curvas com formas arbitrárias
  - Ou seja, sem representação paramétrica
- Na generalização proposta por Ballard, um ponto de referência (x_c, y_c) é escolhido no interior do objeto, por exemplo, seu centróide
- Um segmento de reta arbitrário pode ser formado unindo o ponto de referência a um ponto da borda

Generalização da Transformada de Hough



Generalização da Transformada de Hough

- A direção do gradiente pode ser calculada na interseção do segmento de reta com a borda do objeto
- Uma tabela de referência (a Tabela-R) é construída para armazenar dois parâmetros r e α como uma função da direção da borda no ponto de interseção
- Para cada ponto (x, y) da borda, os parâmetros r e α são calculados:
  - $r = \sqrt{(x - x_c)^2 + (y - y_c)^2}$
  - $\alpha = \arctan[(y - y_c)/(x - x_c)]$

Generalização da Transformada de Hough

- A tabela resultante pode ser ordenada de acordo com as direções da borda nos pontos de interseção
- Deve-se notar que diferentes pontos P_i e P_j da borda do objeto podem ter a mesma direção da borda, ou seja $\theta_{Pi} = \theta_{Pj}$
- Isso significa que pode existir mais de um par (r, α) para cada θ tal que possa determinar as coordenadas de um ponto de referência

Generalização da Transformada de Hough

Tabela-R

θ_1	$(r_1^1, \alpha_1^1), (r_2^1, \alpha_2^1), \ldots$
θ_2	$(r_1^2, \alpha_1^2), (r_2^2, \alpha_2^2), \ldots$
$\vdots$	$\vdots$
θ_k	$(r_1^k, \alpha_1^k), (r_2^k, \alpha_2^k), \ldots$

Generalização da Transformada de Hough

- A detecção da forma e sua localização na imagem pode ser realizada buscando-se, para cada ponto (x, y) da borda, a entrada na tabela cujo ângulo θ_i é mais próximo à direção do gradiente no ponto (x, y)
- Para cada par (r, α) da entrada indexada por θ_i, deve-se calcular as coordenadas dos pontos de referência candidatos:
  - $x_c = x + r.\cos \alpha$ e $y_c = y + \text{sen } \alpha$

Generalização da Transformada de Hough

- Considerando M a matriz de acumulação que armazena as células do espaço de parâmetros, a célula $M(x_c, y_c)$ deve ser incrementada de 1
- Após repetir esse procedimento para cada ponto (x, y) da borda, os valores das células da matriz M que excedam um limiar T representam as possíveis localizações da forma do objeto na imagem

Generalização da Transformada de Hough

- Para tornar a detecção da forma de um objeto invariante quanto à rotação e à escala dois parâmetros adicionais devem ser introduzidos
  - O fator de escala S
  - Ângulo de rotação Φ
- O espaço de parâmetros agora possui dimensão 4
  - M (x_c, y_c, S, Φ)

Generalização da Transformada de Hough

- A detecção da forma do objeto é similar à descrita anteriormente
- Para cada ponto da borda (x, y), busca-se a entrada da tabela cujo ângulo θ_i esteja próximo à direção do gradiente no ponto (x,y)
- Para cada par (r, α) associado a essa entrada deve-se calcular
  - $x_c = x + r.S.\cos(\alpha + \Phi)$
  - $y_c = y + r.S.\text{sen}(\alpha + \Phi)$

Generalização da Transformada de Hough

- Cada célula $M(x_c, y_c, S, \Phi)$ é incrementada de 1
- Ao final do processo, todas as células da matriz M que excedem um limiar T representam o fator de escala S, o ângulo de rotação Φ da forma, bem como a localização do ponto de referência (x_c, y_c) na imagem

Generalização da Transformada de Hough

Imagem

Template

```
[score,  y, x ] = ...
Generalized_hough_transform...
(image, template);
```

y = 79, x = 117

https://www.mathworks.com/matlabcentral/fileexchange/44166-generalized-hough-transform

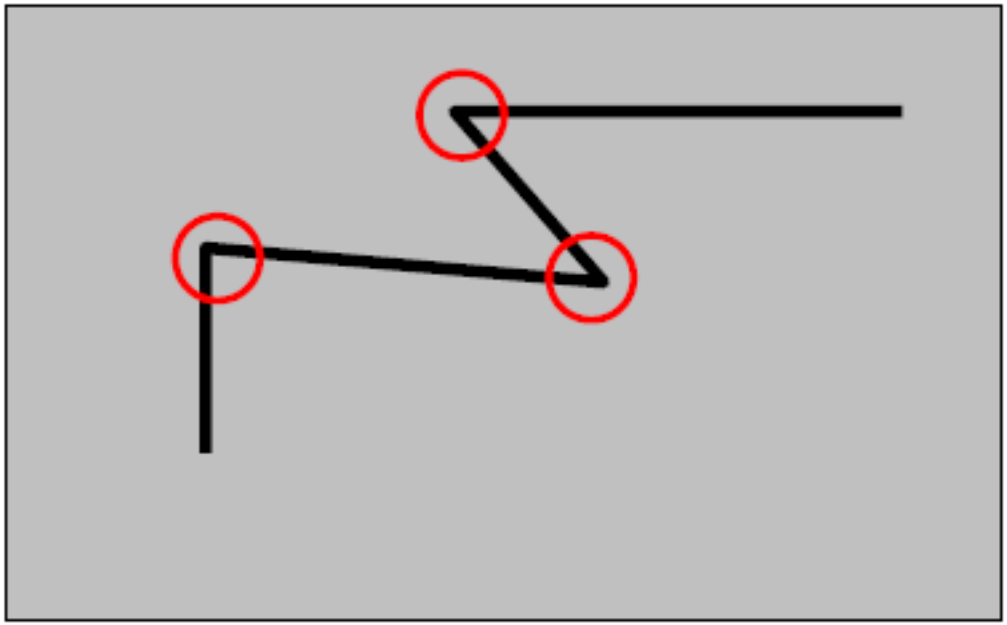
Generalização da Transformada de Hough

Detecção de elipses

- Alto custo computacional
- Espaço de parâmetros com 5 variáveis
- Mais fácil usar a generalização da transformada de Hough

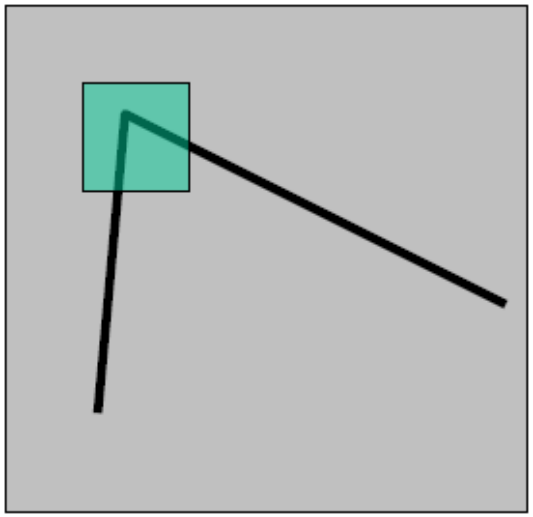
Detecção de cantos

- Harris corner detector
  - C.Harris, M.Stephens, “A Combined Corner and Edge Detector”, 1988



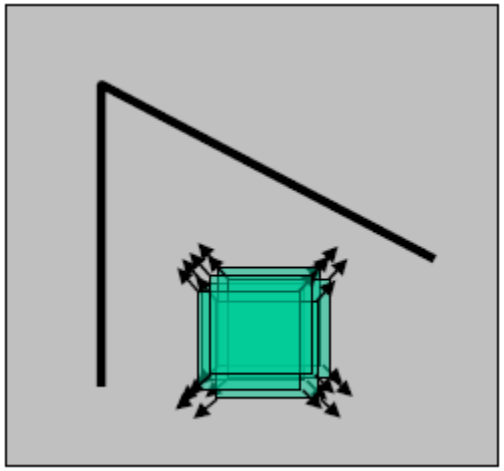
Detecção de cantos

- Harris corner detector
  - Idéia básica
    - Um canto (ou quina) não pode ser identificado por um único pixel
    - Precisamos olhar o comportamento do gradiente sob uma janela móvel

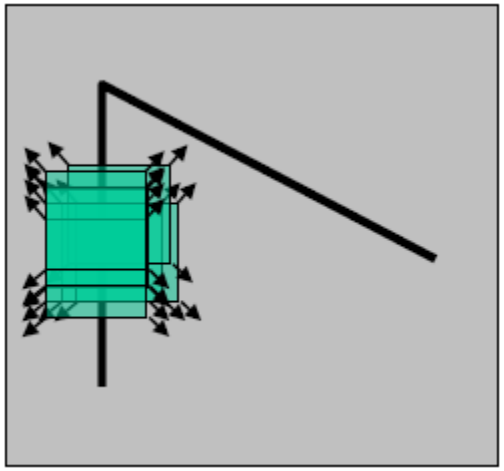


Detecção de cantos

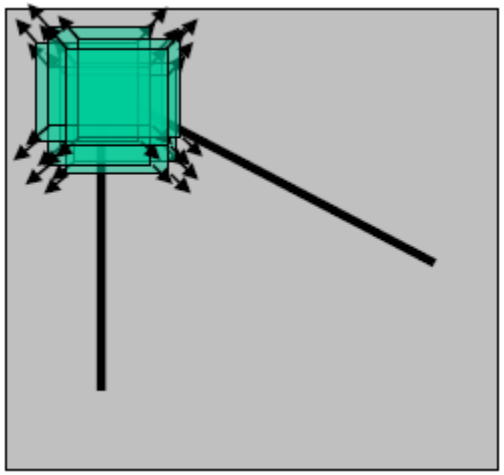
- Harris corner detector
  - Idéia básica



Região uniforme: sem mudanças em qualquer direção



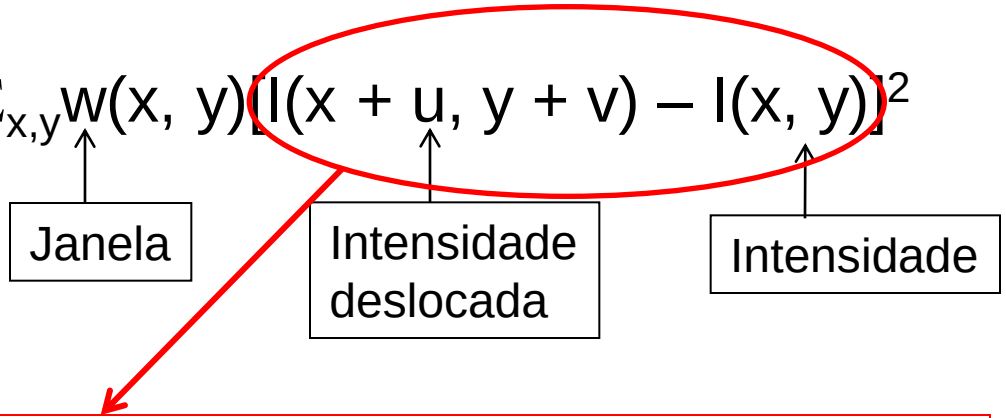
Borda: sem mudanças ao longo da direção da borda



Canto: mudanças significativas em todas as direções

Detecção de cantos

- Harris corner detector
  - Mudança de intensidade para o deslocamento (u, v)
    - $E(u, v) = \Sigma_{x,y} w(x, y)[I(x + u, y + v) - I(x, y)]^2$

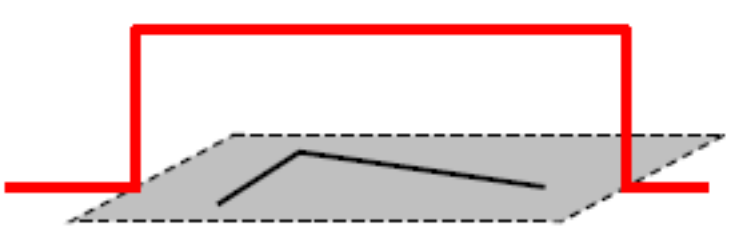


Para caminhos quase constantes, esse valor será próximo de zero.
Para caminhos bastante distintos, esse valor será alto.
Assim, queremos caminhos com E(u, v) alto.

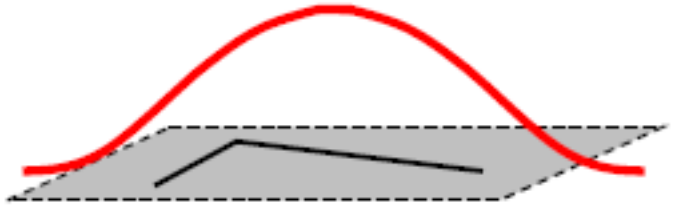
Detecção de cantos

- Harris corner detector

Janela:



Degrau Unitário ou



Gaussiana

Detecção de cantos

- Harris corner detector
  - Por expansão em série de Taylor:

$$f(x+u,y+v) = f(x,y) + uf_x(x,y) + vf_y(x,y) +$$

Primeiras derivadas parciais

$$\frac{1}{2!}\left[u^2 f_{xx}(x,y) + uvf_{xy}x,y + v^2 f_{yy}(x,y)\right] +$$

Segundas derivadas parciais

$$\frac{1}{3!}\left[u^3 f_{xxx}(x,y) + u^2 v f_{xxy}(x,y) + uv^2 f_{xyy}(x,y) + v^3 f_{yyy}(x,y)\right]$$

Terceiras derivadas parciais

$+ \cdots$ (Termos de mais alta ordem)

Aproximação de primeira ordem

$$f(x+u,y+v) \approx f(x,y) + uf_x(x,y) + vf_y(x,y)$$

Detecção de cantos

- Harris corner detector
  - Por expansão em série de Taylor:

$$\sum [I(x+u,y+v)-I(x,y)]^2$$

$$\approx \sum [I(x,y)+uI_x+vI_y-I(x,y)]^2$$ Aproximação de primeira ordem

$$=\sum u^2I_x^2+2uvI_xI_y+v^2I_y^2$$

$$=\sum \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$ Re-escrita na forma de matriz

$$=\begin{bmatrix} u & v \end{bmatrix} \left(\sum \begin{bmatrix} I_x^2 & I_xI_y \\ I_xI_y & I_y^2 \end{bmatrix} \right) \begin{bmatrix} u \\ v \end{bmatrix}$$

Detecção de cantos

- Harris corner detector
  - Para pequenos deslocamentos (u, v) temos uma aproximação:
    - $E(u, v) \approx [u \quad v]\, M \begin{bmatrix} u \\ v \end{bmatrix}$
    - onde M é uma matriz 2x2 calculada das derivadas da imagem:

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

Matriz Hessiana

Detecção de cantos

- Harris corner detector
  - Mudanças de intensidade na janela: análise de autovalores

$$M = \sum_{x,y} w(x,y) \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$$

λ_1 e λ_2 são autovalores de M

Cada célula da matriz Hessiana contém a soma dos elementos da sub-matriz. Ou seja, aqui, temos a soma dos elementos da sub-matriz IxIy.

Detecção de cantos

- Harris corner detector
  - Exemplos

Flat

Edge

Corner

Detecção de cantos

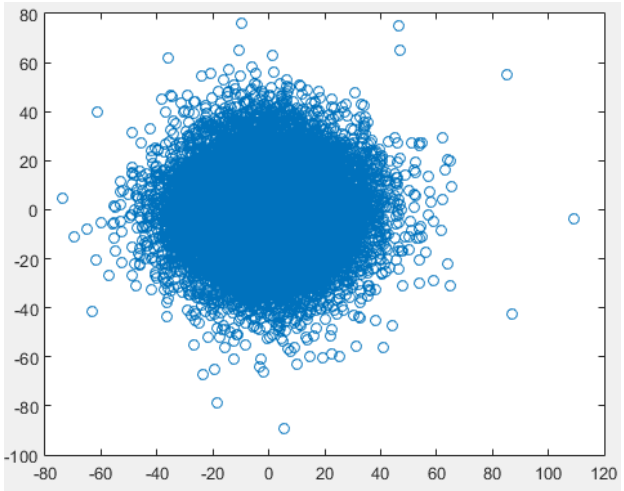
- Harris corner detector
  - Exemplos:

```
Código:
img = Imagem de entrada em tons de cinza
[gx, gy] = gradient(double(img));
[gxx, gxy] = gradient(gx);
[gxy, gyy] = gradient(gy);
gx2 = reshape (gx, [1 150*150]);
gy2 = reshape (gy, [1 150*150]);
plot (gx2, gy2, 'o');
```

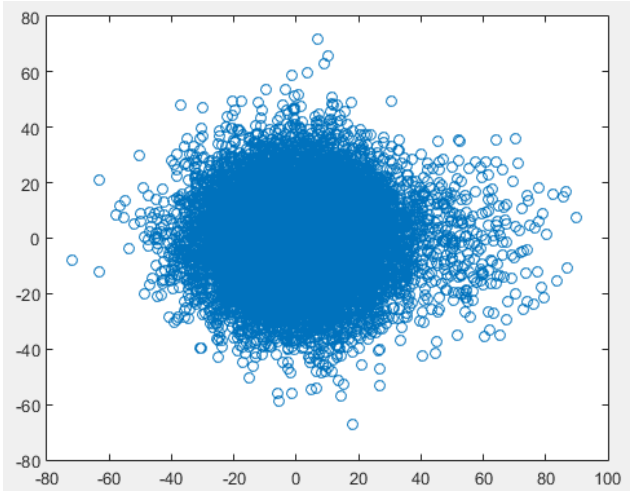
As imagens são 150x150 pixels

Detecção de cantos

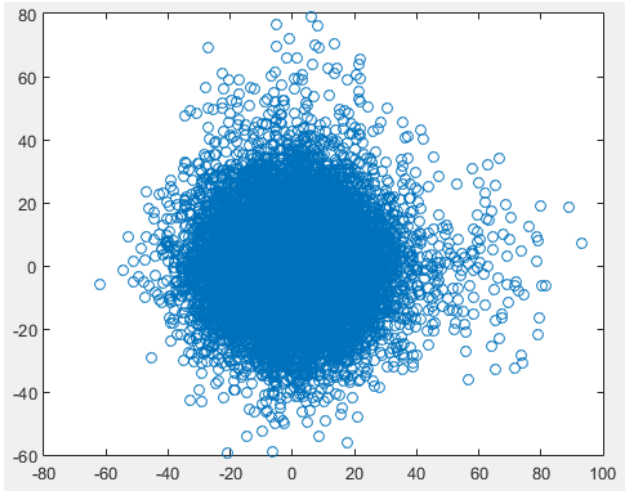
- Harris corner detector
  - Exemplos: Ix *versus* Iy



Flat
(Concentrado)



Edge
(Espalhado em uma direção)

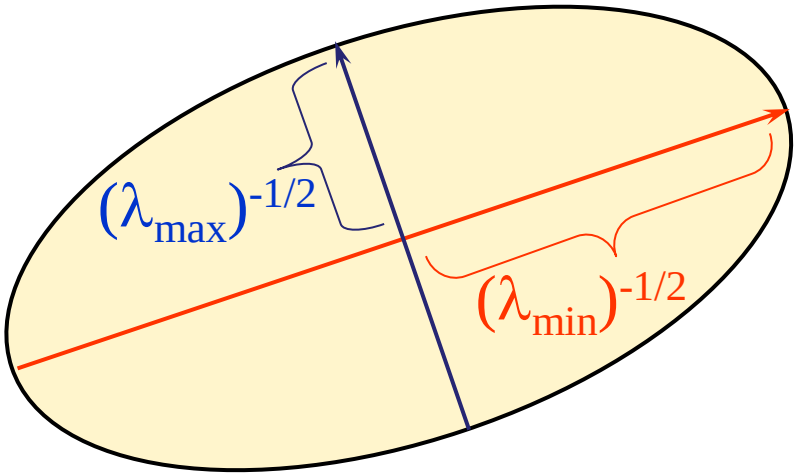


Corner
(Espalhado em mais de uma direção)

Detecção de cantos

- Harris corner detector

Elipse $E(u,v)$



Uma elipse pode ser descrita na forma canônica como:

$$\frac{x^2}{a^2} + \frac{y^2}{b^2} = 1$$

onde a e b são os raios equatoriais.

De forma geral, uma elipse de orientação arbitrária pode ser definida pela equação:

$$x^T A x = 1$$

onde A é uma matriz simétrica positiva* e x um vetor. Nesse caso, os autovalores de A definem as principais direções da elipse e o inverso da raiz quadrada dos autovalores são os correspondentes raios equatoriais.

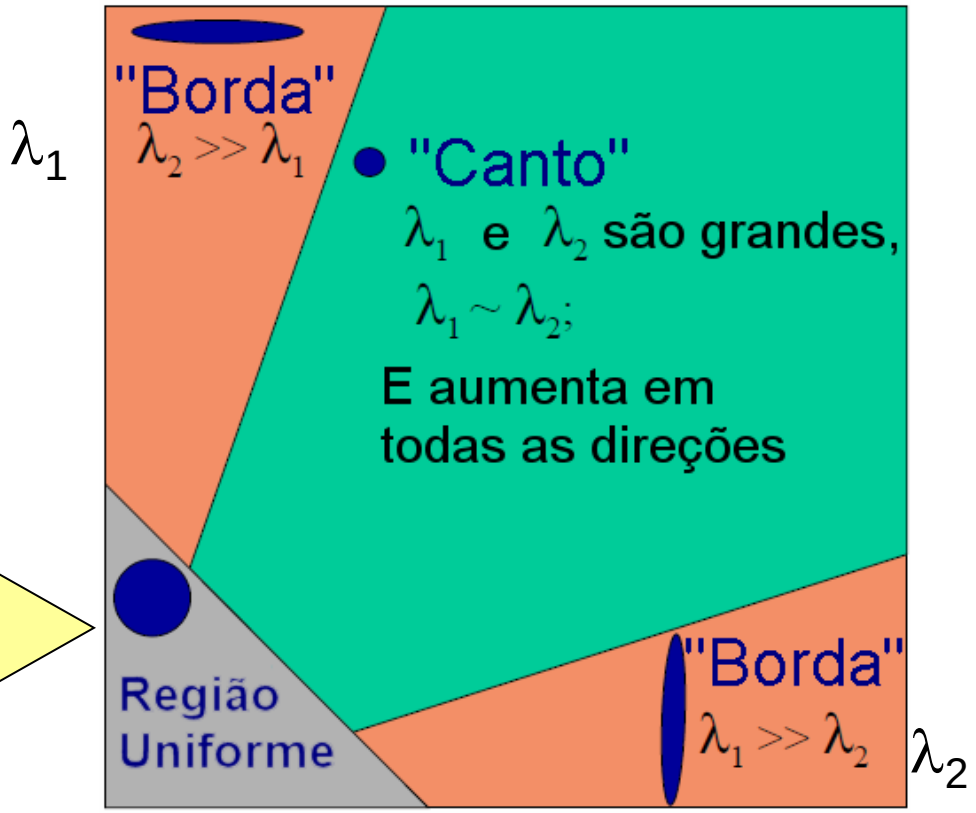
*Uma matriz M tal que $z^T M z > 0$, para todo z não nulo.

$$\begin{bmatrix} z_0 & z_1 \end{bmatrix} \begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} \begin{bmatrix} z_0 \\ z_1 \end{bmatrix} = z_0^2 + z_1^2$$

$M_1 = \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix}$ não é porque $\begin{bmatrix} 1 & -1 \end{bmatrix} \begin{bmatrix} 0 & 1 \\ 1 & 0 \end{bmatrix} \begin{bmatrix} 1 \\ -1 \end{bmatrix} = -2 < 0$

Detecção de cantos

- Harris corner detector
  - Classificação dos pontos da imagem usando os autovalores



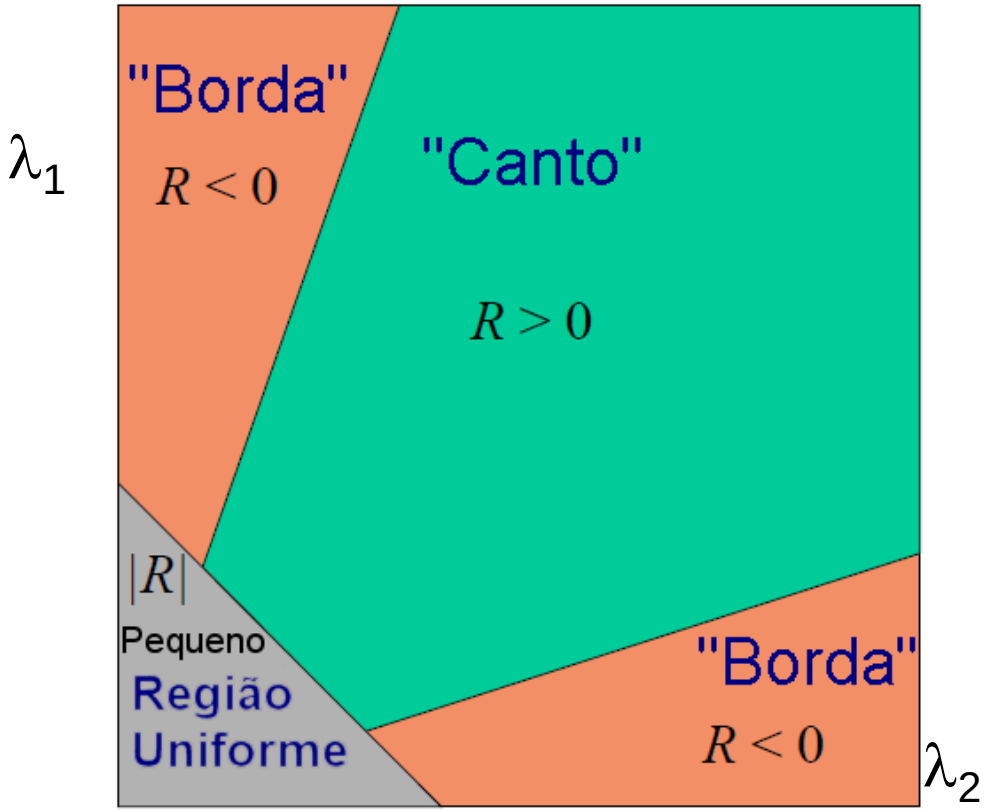
λ1 e λ2 são pequenos;
E é praticamente
constante em todas
as direções

Detecção de cantos

- Harris corner detector
  - Medida da resposta em relação ao canto:
    - $R = \det M - k.(\text{trace } M)^2$
    - $\det M = \lambda_1.\lambda_2$
    - $\text{trace } M = \lambda_1 + \lambda_2$
    - k é uma constante empírica

Detecção de cantos

- Harris corner detector
  - Classificação dos pontos da imagem usando os autovalores



Detecção de cantos

- Harris corner detector
  - Exemplos:

Código:

```
img = Imagem de entrada em tons de cinza
[gx, gy] = gradient(double(img));
gxx = gx.^2; gxx = sum(sum(gxx));
gyy = gy.^2; gyy = sum(sum(gyy));
gxy = gx.*gy; gxy = sum(sum(gxy));
hess_img = [gxx gxy; gxy gyy];
e = eig(hess_img)
R = det(hess_img) - 0.01*(trace(hess_img))^2
```

Detecção de cantos

- Harris corner detector
  - Exemplos:

Flat

$\lambda 1 = \lambda 2 = 0$
$R = 0$

Edge

$\lambda 1 = 0$
$\lambda 2 = 1.116.300$
$R = -1{,}24.10^{10}$

Corner

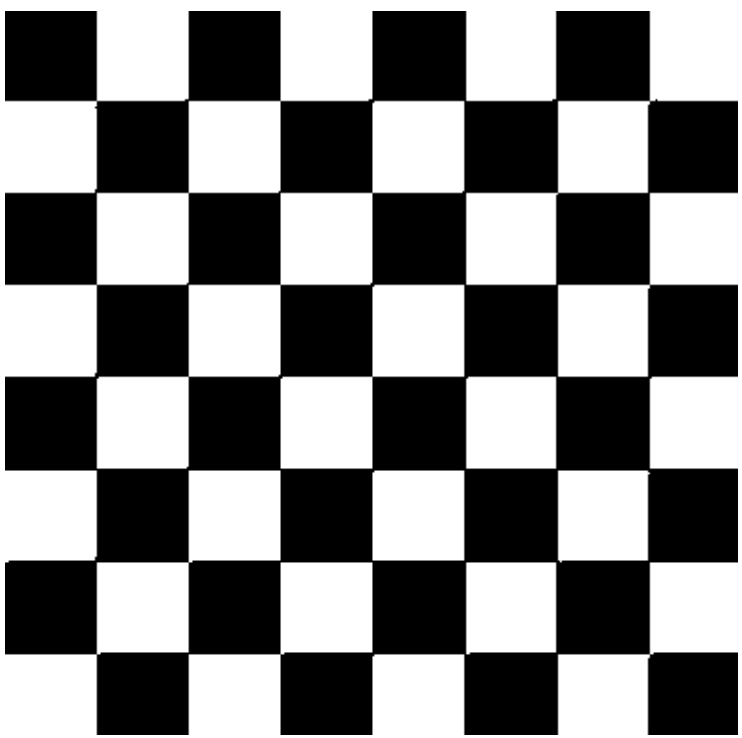
$\lambda 1 = 557.920$
$\lambda 2 = 617.920$
$R = 3{,}31.10^{11}$

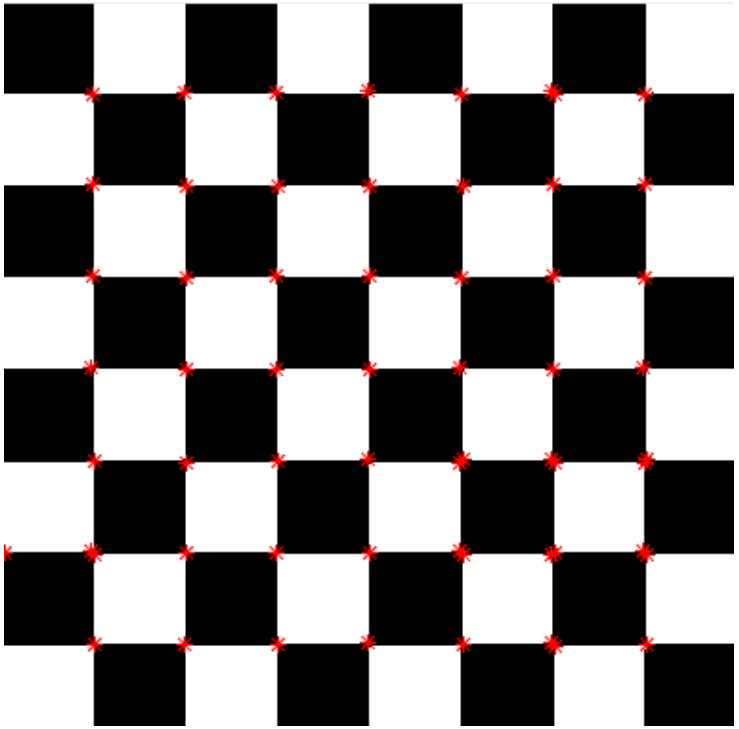
Detecção de cantos

- Harris corner detector
  - Algoritmo:
    - Encontre pontos com maiores respostas da função R (R > threshold)
    - Tome os pontos de máximo local de R

Detecção de cantos

- Harris corner detector





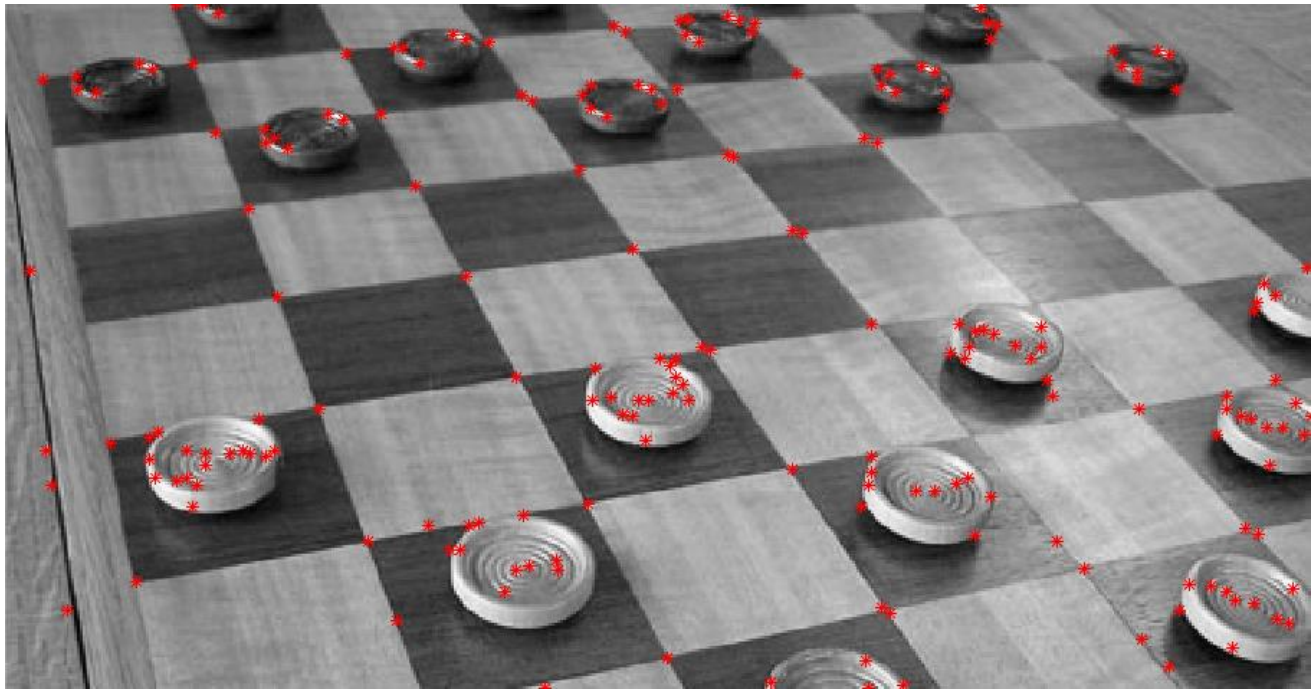
Na prática, não é tão perfeito....

Detecção de cantos

- Harris corner detector

Detecção de cantos

- Harris corner detector



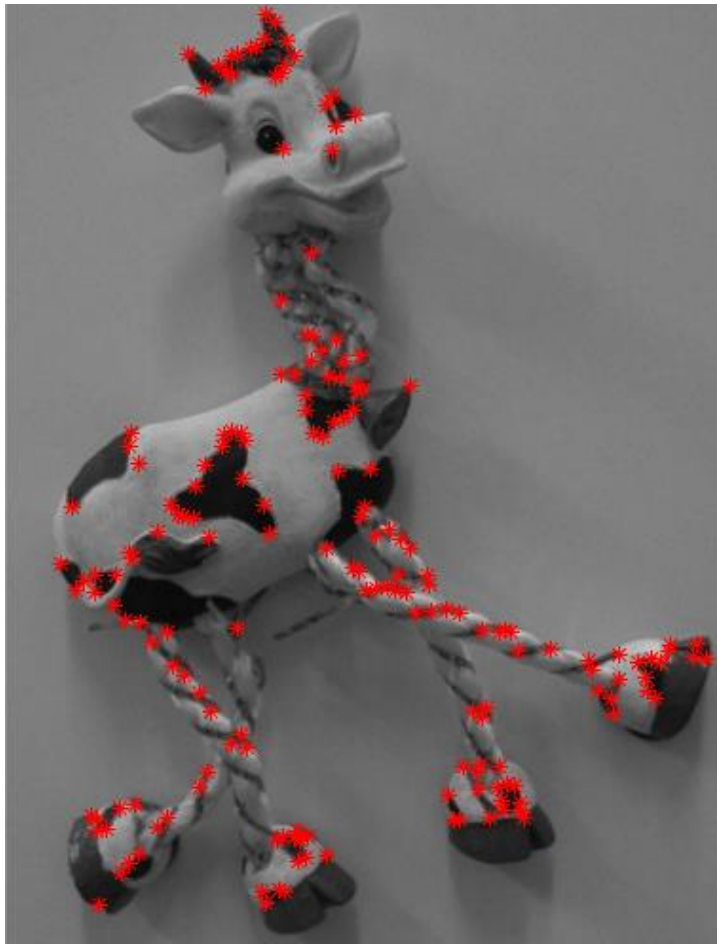
Observe falsos negativos e positivos.

Detecção de cantos

- Harris corner detector

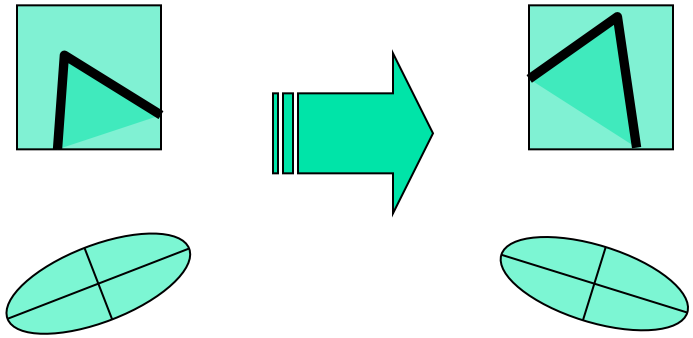
Detecção de cantos

- Harris corner detector



Detecção de cantos

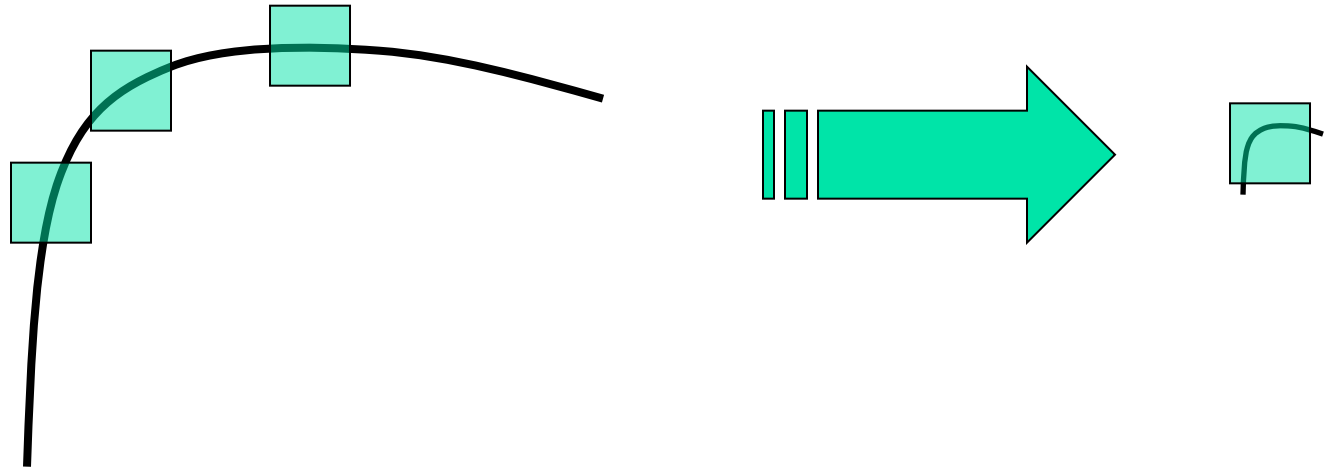
- Harris corner detector
  - Invariante à rotação



A elipse rotaciona mas sua forma (i.e. autovalores) permanecem as mesmas

Detecção de cantos

- Harris corner detector
  - Mas não é invariante à escala!!!



Todos os pontos são classificados como bordas

Aqui, é visto como canto!