8º CAPÍTULO

Segmentação de imagem

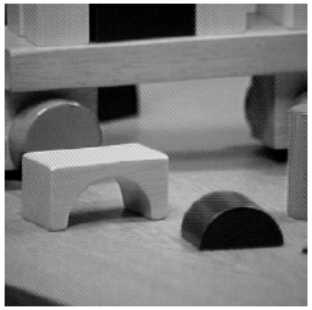
Prof. Arnaldo Abrantes

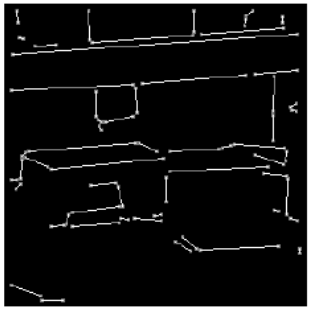
- Objectivo: decompôr uma imagem (ou sequência de imagens) num conjunto de regiões (1d, 2d ou 3d), podendo obter-se
  - Áreas com um dado significado semântico, ou
  - Conjuntos de *pixels* de fronteira, ou
  - Estruturas de *pixels* com forma determinada (linhas, círculos, polígonos)

- Porquê realizar a operação de segmentação?
  - Isolar zonas de interesse, para posterior processamento.
  - Mudança de representação da imagem, para um nível hierárquico superior


entrada


saída


entrada


saída

O que é uma região?

- Conjuntos de *pixels* conexos com ‘propriedades’ homogéneas, relativamente à(s) característica(s) (*features*) usadas na operação de segmentação

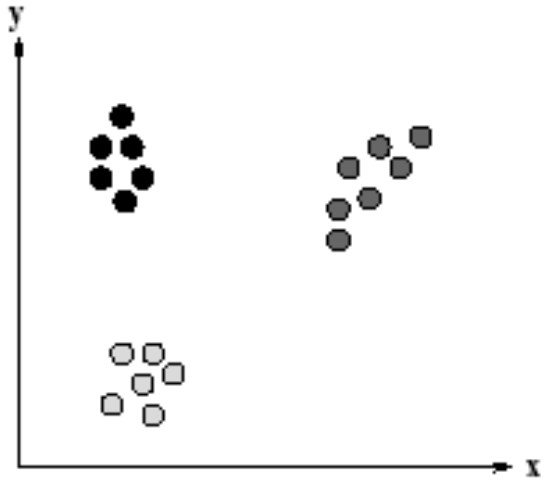
- Regiões adjacentes deverão ser significativamente diferentes, relativamente a essas mesmas *features*

- As fronteiras das regiões deverão ser suaves

- O interior duma região deverá ser simples, não devendo conter um grande número de buracos

Métodos de *clustering*

- *Clustering*
  - Processo de decompôr (classificar) um conjunto de padrões (vectores de *features*) em diferentes subconjuntos, designados por *clusters*
- Exemplos de *features*
  - Intensidade
  - Cor
  - Textura
  - Forma
  - Movimento
- Alguns métodos
  - Algoritmos clássicos: K-médias, Isodata
  - Baseados em histogramas: Otsu, Ohlander
  - Baseados em partições de grafos: Shi



K-médias / Isodata

K-médias

Objectivo: $D = \sum_{k=1}^{K} \sum_{x_i \in C_k} \|x_i - m_k\|^2$

K = 6

Form K-means clusters from a set of n-dimensional vectors.

1. Set ic (iteration count) to 1.
2. Choose randomly a set of K means $m_1(1), m_2(1), \ldots, m_K(1)$.
3. For each vector x_i compute $D(x_i, m_k(ic))$ for each $k = 1, \ldots, K$ and assign x_i to the cluster C_j with the nearest mean.
4. Increment ic by 1 and update the means to get a new set $m_1(ic), m_2(ic), \ldots, m_K(ic)$.
5. Repeat steps 3 and 4 until $C_k(ic) = C_k(ic+1)$ for all k.

Isodata

K = 5

Form isodata clusters from a set of n-dimensional vectors.

1. assign x_i to the cluster l that minimizes

$$D_\Sigma = [x_i - m_l]' \Sigma_l^{-1} [x_i - m_l].$$

2. Merge clusters i and j if

$$| m_i - m_j | < \tau_v$$

where τ_v is a variance threshold.

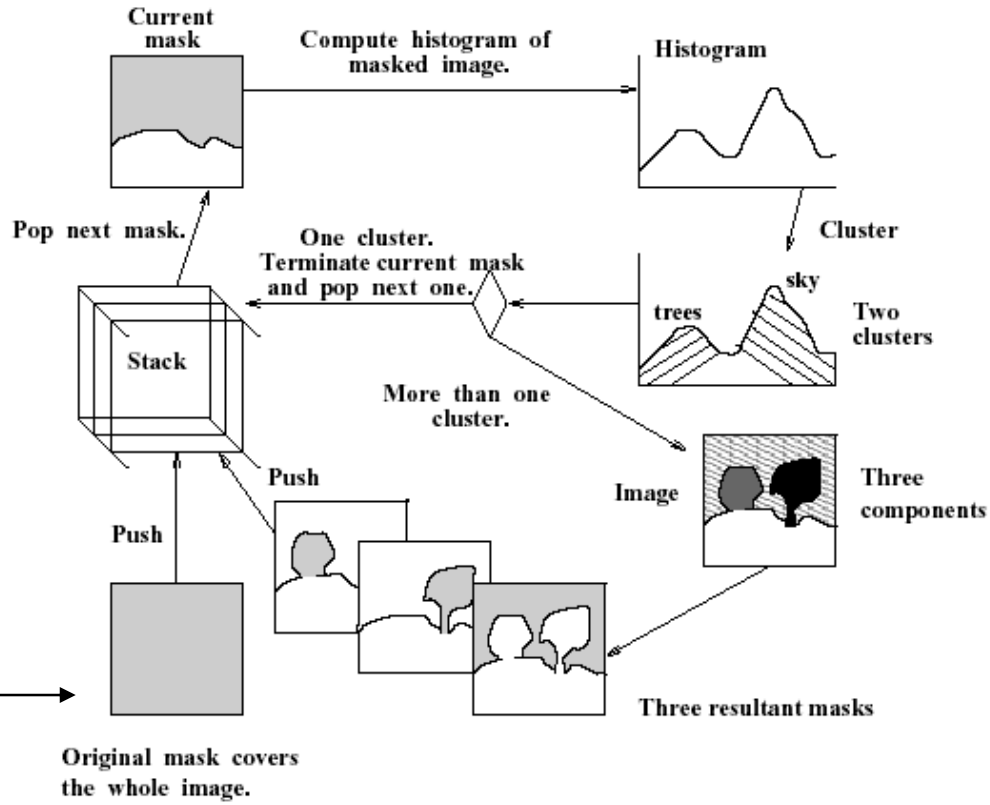
3. Split cluster k if the maximum eigenvalue of σ_k is larger than τ_v.
4. Stop when

$$| m_i(t) - m_i(t+1) | < \epsilon$$

for every cluster i or when the maximum number of iterations has been reached.

Métodos baseados em histogramas

- Hipótese
  - Os subconjuntos de *pixels* que originam os diferentes modos do histograma, correspondem a diferentes objectos na cena (caso binário – método de Otsu)
- Vantagem
  - Métodos deste tipo necessitam de apenas uma passagem pelos dados
- Exemplo
  - Algoritmo recursivo de Ohlander



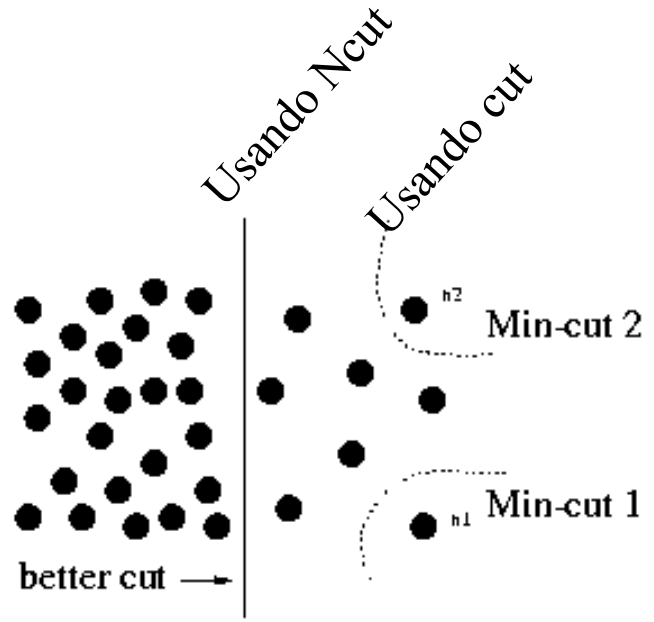
Partição de grafos – algoritmo de Shi

- Seja o grafo $G = (V, E)$ definido pelo conjunto de nós V e ligações E
  - Cada nó representa um ponto no espaço de *features*
  - Cada ligação tem um peso, $w(i,j)$, que representa o grau de semelhança entre os nós i e j
- Problema da segmentação
  - Encontrar partições do grafo, tais que as medidas de semelhança intra-partições sejam elevadas, e as inter-partições baixas
- Caso binário

$$\text{cut}(A,B) = \sum_{u \in A, v \in B} w(u,v)$$

$$\text{asso}(A,V) = \sum_{u \in A, t \in V} w(u,t)$$

  - Caso M-ário: recursão do algoritmo binário



(pesos inversamente proporcionais à distância entre nós)

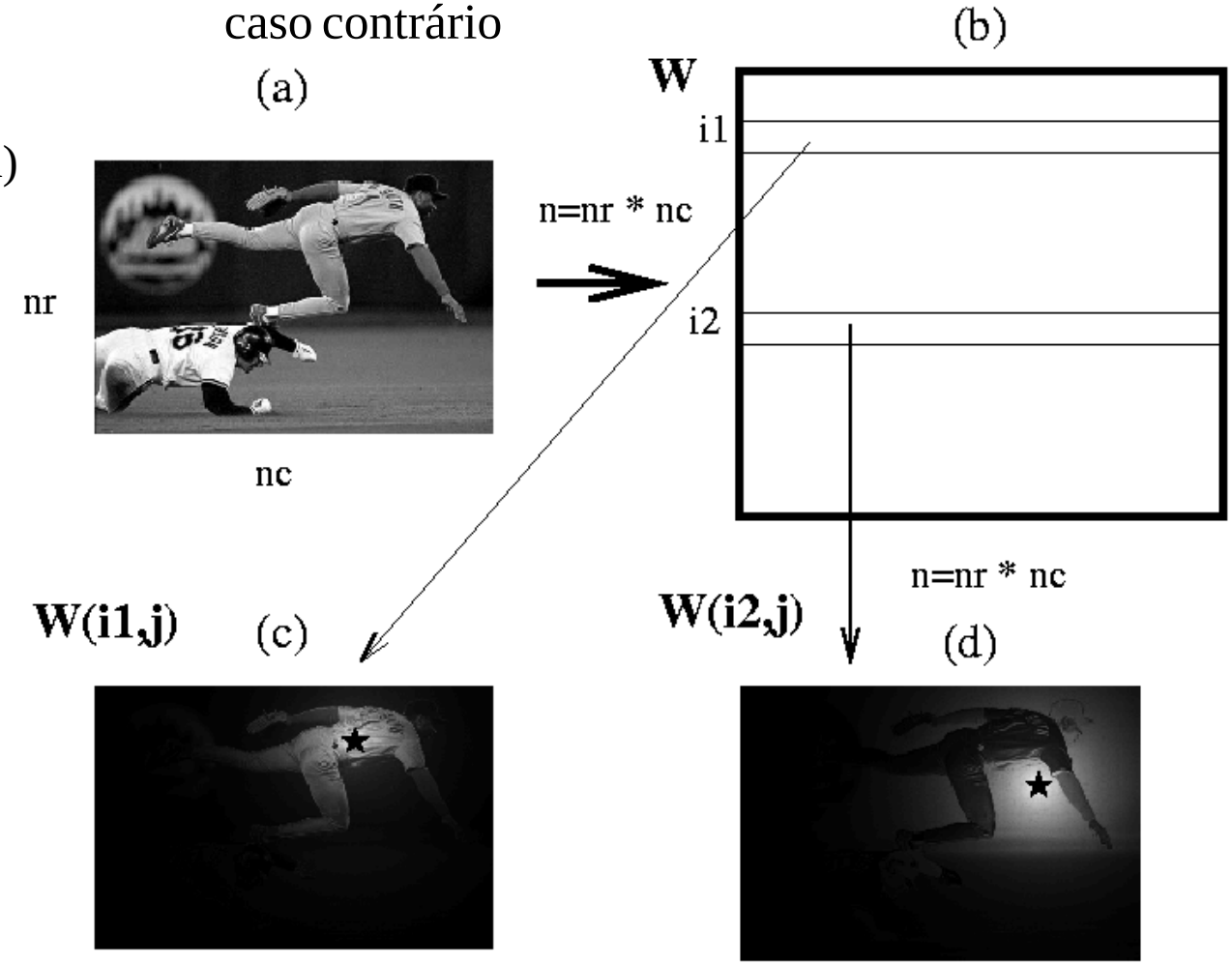
$$\text{Ncut}(A,B) = \frac{\text{cut}(A,B)}{\text{asso}(A,V)} + \frac{\text{cut}(A,B)}{\text{asso}(B,V)}$$

Minimizar Ncut

Exemplo – Escolha da função de pesos

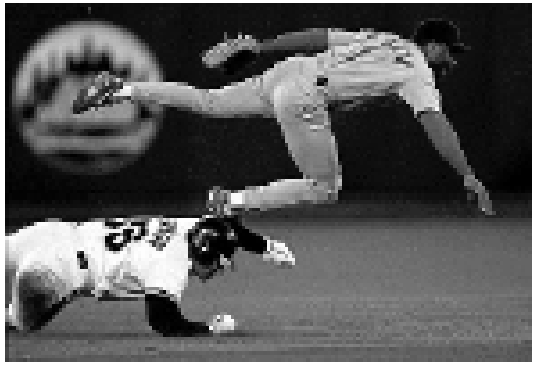
$$w(i,j) = e^{-\frac{\|F(i)-F(j)\|_2}{\sigma_I}} * \begin{cases} e^{-\frac{\|X(i)-X(j)\|_2}{\sigma_X}} & \text{se } \|X(i)-X(j)\|_2 < r \\ 0 & \text{caso contrário} \end{cases}$$

- $F(i)$, $F(j)$ são vectores de *features*, (por exemplo intensidade luminosa)
- $X(i)$, $X(j)$ são as correspondentes coordenadas dos *pixels* i, j

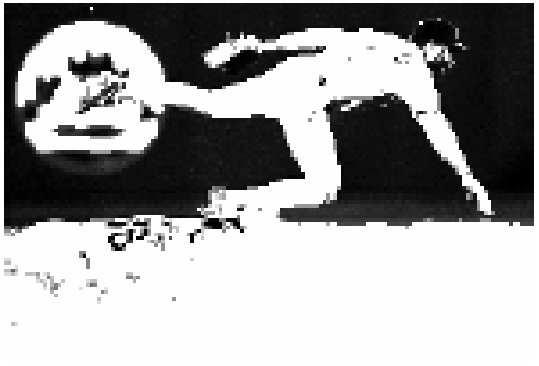


Resultados de segmentação – trabalho de Shi e Malik

original



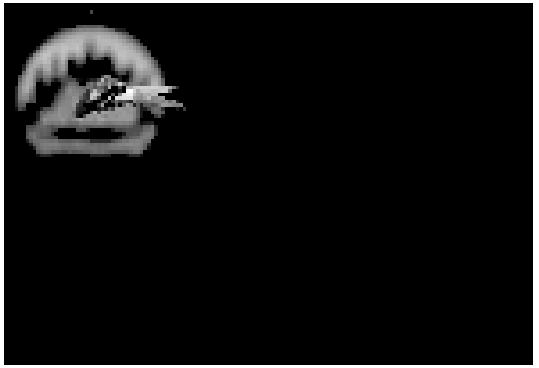
fundo



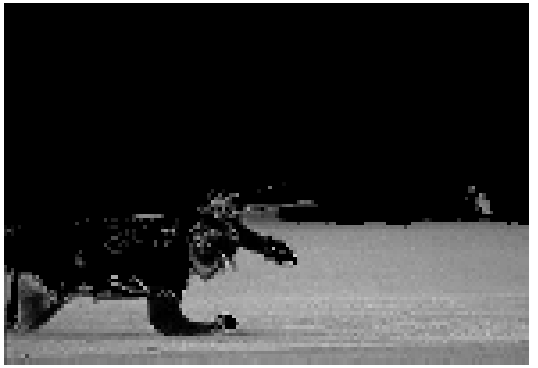
player1

player2

símbolo



solo



- Problema de optimização formulado como um problema de cálculo de vectores e valores próprios

- Ideias
  - Começar com uma região de um só pixel (canto superior esquerdo) e ir agrupando *pixels* a essa região enquanto as medidas de semelhação forem suficientemente elevadas
  - Iniciar uma nova região quando falha
- Critério estatístico

$$T = \left[\frac{(N-1)N}{N+1} (y - \bar{X})^2 \Big/ S^2 \right]^{\frac{1}{2}} \quad \begin{array}{ll} T < \tau & \text{semelhante} \\ T \geq \tau & \text{não semelhante} \end{array}$$

- Actualizar, recursivamente, a média e a medida de dispersão

$$\bar{X}_{new} \leftarrow (N\bar{X}_{old} + y)/(N+1)$$

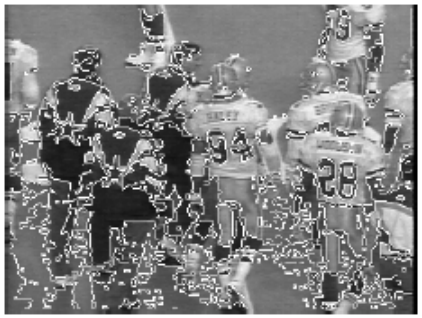
$$\bar{X} = \frac{1}{N} \sum_{(r,c) \in R} I(r,c)$$

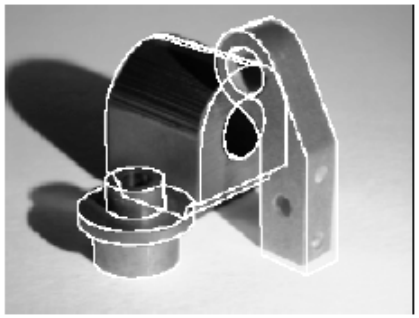
$$S^2_{new} \leftarrow S^2_{old} + (y - \bar{X}_{new})^2 + N(\bar{X}_{new} - \bar{X}_{old})^2$$

$$S^2 = \sum_{(r,c) \in R} (I(r,c) - \bar{X})^2$$

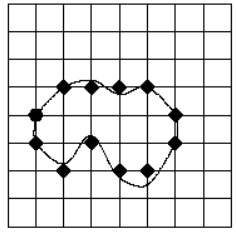
Representação de regiões

- *Overlays*

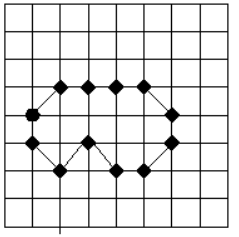




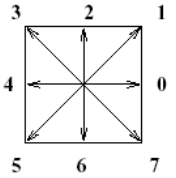
- *Chain codes*



original curve



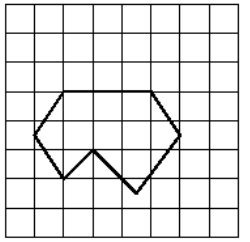
chain code links



encoding scheme

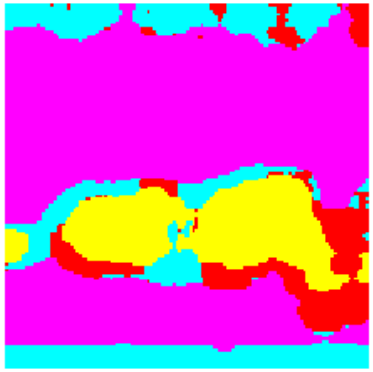
100076543532

chain code representation



polygonal approximation

- Imagem de *labels*



- *Quadtrees*

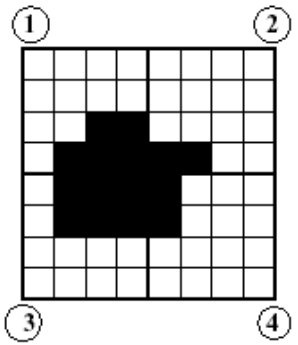
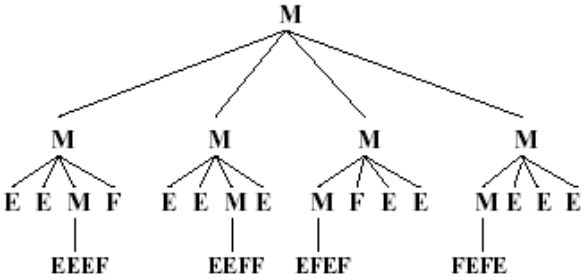


image region



quad tree representation

Extracção de fronteira

```
procedure border(S);
{
  for R:= 1 to NLINES
    {
      for C:= 1 to NPIXELS
        {
        LABEL:= S[R,C];
        if new_region(LABEL) then add(CURRENT,LABEL);
        NEIGHB:= neighbors(R,C,LABEL);
        T:= pixeltype(R,C,NEIGHB);
        if T == 'border'
        then for each pixel N in NEIGHB
          {
          CHAINSET:= chainlist(LABEL);
          NEWCHAIN:= true;
          for each chain X in CHAINSET while NEWCHAIN
            if N==rear(X)
            then { add(X,[R,C]); NEWCHAIN:= false }
          if NEWCHAIN
          then make_new_chain(CHAINSET,[R,C],LABEL);
          }
        }
      for each region REG in CURRENT
        if complete(REG)
        then { connect_chains(REG); output(REG); free(REG) }
    }
}
```

	1	2	3	4	5	6	7
1	0	0	0	0	0	0	0
2	0	0	0	0	2	2	0
3	0	1	1	1	2	2	0
4	0	1	1	1	2	2	0
5	0	1	1	1	2	2	0
6	0	0	0	0	2	2	0
7	0	0	0	0	0	0	0

(a) A labeled image with two regions.

Region	Length	List
1	8	(3,2)(3,3)(3,4)(4,4)(5,4)(5,3)(5,2)(4,2)
2	10	(2,5)(2,6)(3,6)(4,6)(5,6)(6,6)(6,5)(5,5) (4,5)(3,5)

(b). The output of the border procedure for the labeled image.

Detecção e seguimento de contornos – algoritmo de Canny

- Ideias chave:
  - Supressão de não-máximos
  - Seguimento com histerese


```
Compute thin connected edge segments in the input image.
I[x,y] : input intensity image
E[x,y] : output binary image
IS[x,y] : smoothed intensity image
Mag[x,y] : gradient magnitude
Dir[x,y] : gradient direction
T_low is low intensity threshold
T_high is high intensity threshold

procedure Canny( I[], E[] ):
{
    IS[] = image I[] smoothed by convolution with Gaussian G_σ(x,y)
    use Roberts operator to compute Mag[x,y] and Dir[x,y] from IS[];
    Suppress_Nonmaximal( Mag[], Dir[], T_low, T_high );
    Edge_Detect( Mag[], T_low, T_high, E[] );
}
procedure Suppress_Nonmaximal( Mag[], Dir[] ):
{
    define +Del[] = (1,0), (1,1), (0,1), (-1,1);
    define -Del[] = (-1,0), (-1,-1), (0,-1), (1,-1);
    for x := 0 to MaxX-1;
    for y := 0 to MaxY-1;
    {
        direction := ( Dir[x,y] + π/8 ) modulo π/4;
        if ( Mag[x,y] ≤ Mag[x,y] + Del[direction] ) then Mag[x,y] := 0;
        if ( Mag[x,y] ≤ Mag[x,y] + -Del[direction] ) then Mag[x,y] := 0;
    }
}
procedure Edge_Detect( Mag[], T_low, T_high, E[] ):
{
    for x := 0 to MaxX - 1;
    for y := 0 to MaxY - 1;
    {
        if (Mag[x,y] ≥ T_high) then Follow_Edge( x,y,Mag[], T_low, T_high, E[] );
    };
}
procedure Follow_Edge( x,y, Mag[], T_low, T_high, E[] ):
{
    E[x,y] := 1;
    while Mag[x,y] > T_low for some 8-neighbor [u,v] of [x,y]
    {
        E[u,v] := 1;
        [x,y] := [u,v];
    };
}
```

Algoritmo de Canny - Resultados

original

σ = 1

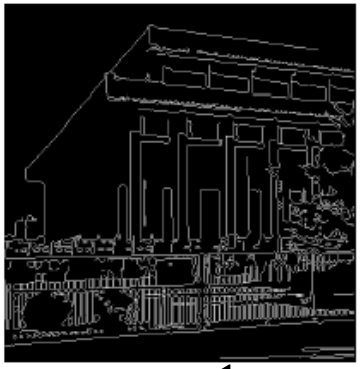
σ = 4

original

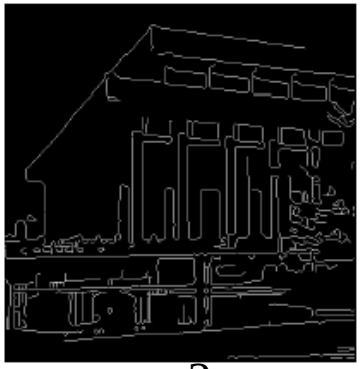
σ = 1

Roberts

original

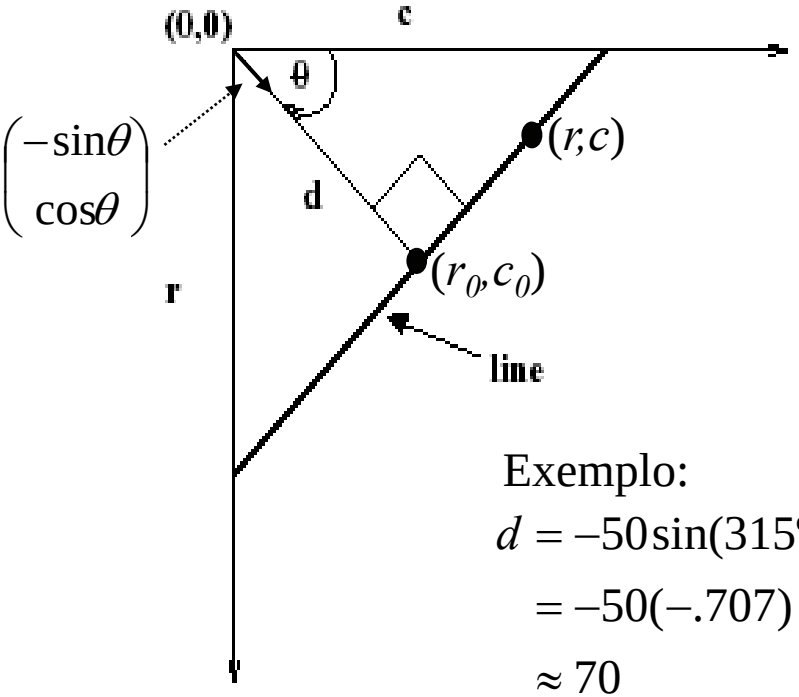


σ = 1



σ = 2

Transformada de Hough – Extracção de segmentos de recta



Equação da recta:

$$d = -r\sin\theta + c\cos\theta$$

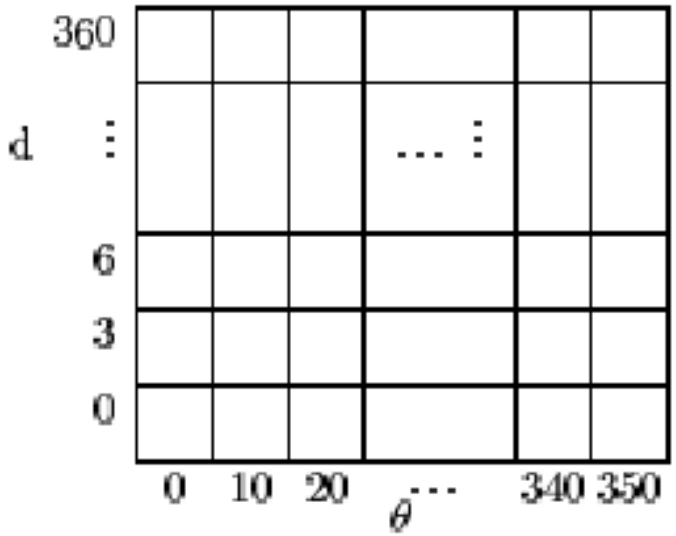
$$\left\langle \begin{pmatrix} r - r_0 \\ c - c_0 \end{pmatrix} , \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix} \right\rangle = 0$$

$$\left\langle \begin{pmatrix} r_0 \\ c_0 \end{pmatrix} , \begin{pmatrix} -\sin\theta \\ \cos\theta \end{pmatrix} \right\rangle = d$$

Exemplo:

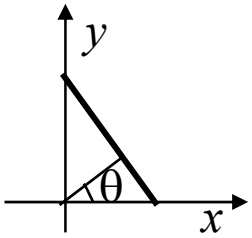
$$\begin{aligned} d &= -50\sin(315^\circ) + 50\cos(315^\circ) \\ &= -50(-.707) + 50(.707) \\ &\approx 70 \end{aligned}$$

Acumulador $\boldsymbol{A}(d_q, \theta_q)$



H: imagem 256x256

Sistema de coordenadas xy:



$$d = x\cos\theta + y\sin\theta$$

$$x \to c$$

$$y \to -r$$

Algoritmo – transformada Hough

Accumulate the straight line segments in gray-tone image S to accumulator A.
S[R, C] is the input gray-tone image.
NLINES is the number of rows in the image.
NPIXELS is the number of pixels per row.
A[DQ, THETAQ] is the accumulator array.
DQ is the quantized distance from a line to the origin.
THETAQ is the quantized angle of the normal to the line.

```
procedure accumulate_lines(S,A);
{
A := 0;
PTLIST := NIL;
for R := 1 to NLINES
   for C := 1 to NPIXELS
      {
      DR := row_gradient(S,R,C);
      DC := col_gradient(S,R,C);
      GMAG := gradient(DR,DC);
      if GMAG > gradient_threshold
         {
         THETA := atan2(DR,DC);
         THETAQ := quantize_angle(THETA);
         D := abs(C*cos(THETAQ) - R*sin(THETAQ));
         DQ := quantize_distance(D);
         A[DQ,THETAQ] := A[DQ,THETAQ]+GMAG;
         PTLIST(DQ,THETAQ) := append(PTLIST(DQ,THETAQ),[R,C])
         }
      }
}
```

Exemplo

Pré-processamento

	1	2	3	4	5
1	0	0	0	100	100
2	0	0	0	100	100
3	0	0	0	100	100
4	100	100	100	100	100
5	100	100	100	100	100

gray level image

	1	2	3	4	5
1	0	0	300	300	0
2	0	0	300	300	0
3	0	0	200	200	0
4	0	0	0	100	0
5	0	0	0	0	0

column gradient

	1	2	3	4	5
1	0	0	0	0	0
2	0	0	0	0	0
3	300	300	200	100	0
4	300	300	200	100	0
5	0	0	0	0	0

row gradient

	1	2	3	4	5
1	-	-	0	0	-
2	-	-	0	0	-
3	90	90	45	26.6	-
4	90	90	90	45	-
5	-	-	-	-	-

THETA

	1	2	3	4	5
1	-	-	0	0	-
2	-	-	0	0	-
3	90	90	40	20	-
4	90	90	90	40	-
5	-	-	-	-	-

THETAQ

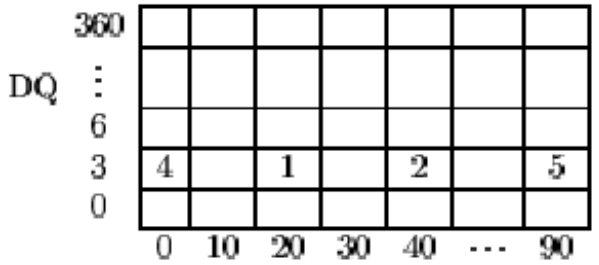
	1	2	3	4	5
1			3	4	
2			3	4	
3	3	3	4.2	4.8	
4	4	4	4	5.6	
5					

D

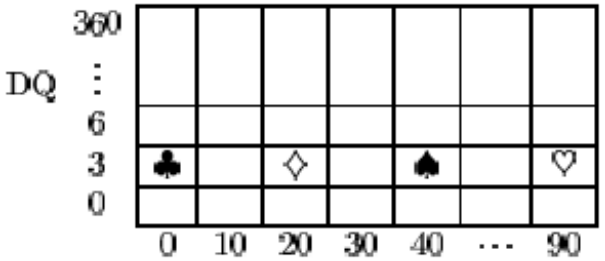
	1	2	3	4	5
1			3	3	
2			3	3	
3	3	3	3	3	
4	3	3	3	3	
5					

DQ

Acumulador



accumulator A



PTLIST

♣ (1,3)(1,4)(2,3)(2,4)
♢ (3,4)
♠ (3,3)(4,4)
♡ (3,1)(3,2)(4,1)(4,2)(4,3)

Transformada de Hough Generalizada

- A TH pode ser extendida para qualquer curva com expressão analítica da forma $f(\boldsymbol{x},\boldsymbol{a}) = 0$,
  - $\boldsymbol{x}$ é um vector 2D com as coordenadas de um ponto da curva
  - $\boldsymbol{a}$ é um vector de parâmetros

- Algoritmo genérico:
  - Inicializar acumulador $\boldsymbol{A}(\boldsymbol{a})$ a zero
  - Por cada pixel $\boldsymbol{x}$ na imagem, determinar valores de $\boldsymbol{a}$, tais que $f(\boldsymbol{x},\boldsymbol{a}) = 0$; incrementar acumulador: $\boldsymbol{A(a)} = \boldsymbol{A}(\boldsymbol{a}) + 1$
  - Máximos locais de $\boldsymbol{A}$ correspondem a curvas f na imagem