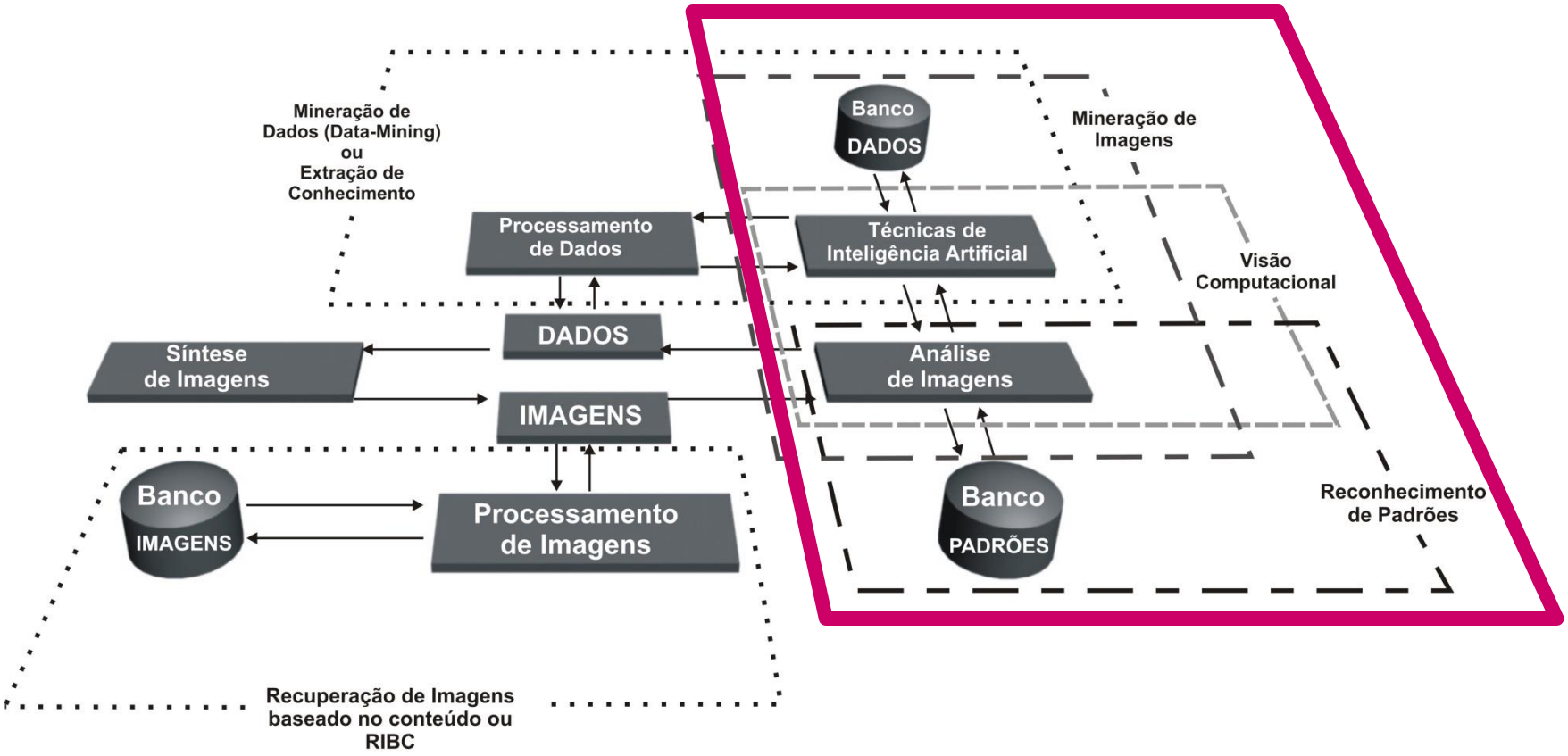
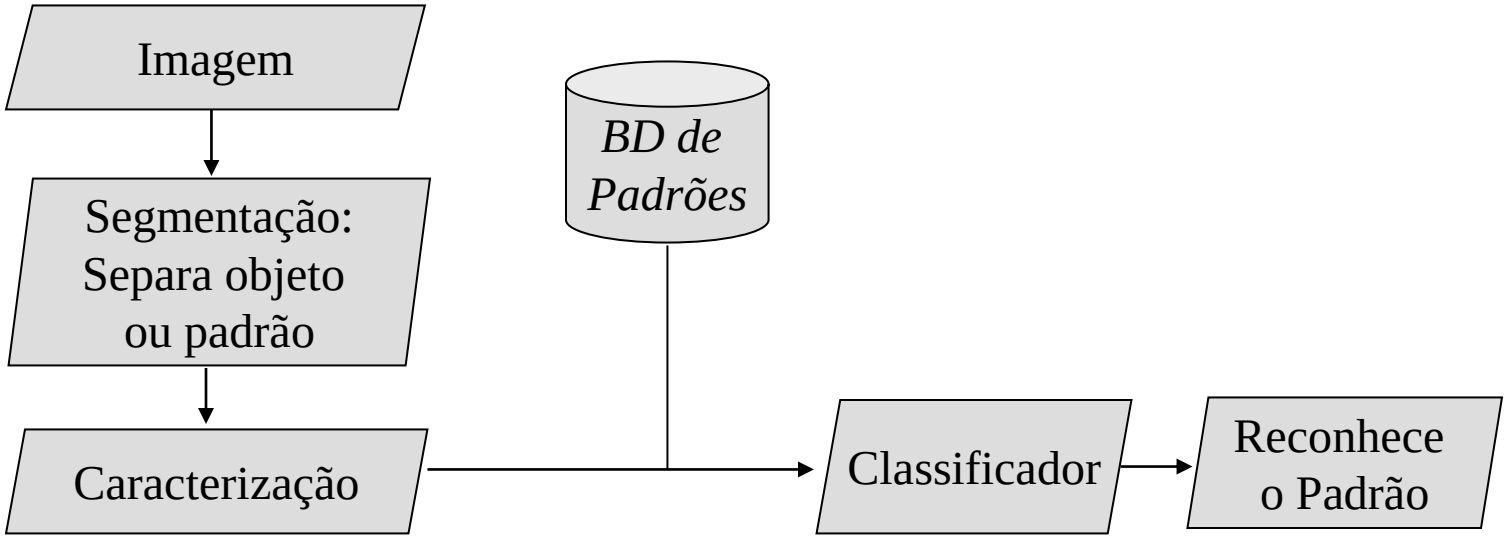
Extração de Características

Rivera



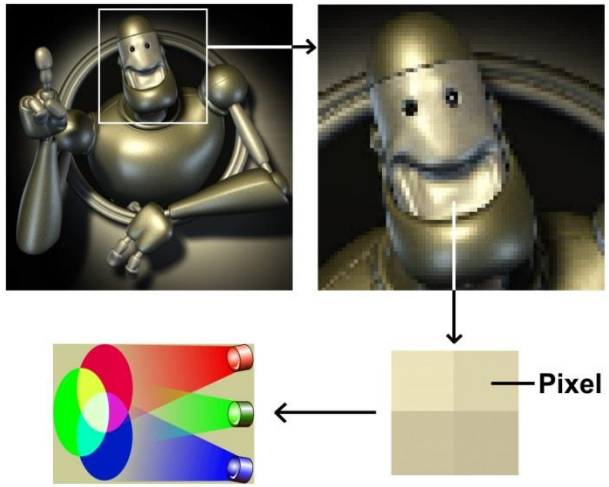
- Sistemas de análise de Imagens
  - Reconhecimento de elementos e objetos
    - Parâmetros quantificáveis
      - Cor, posição, orientação, dimensões, textura, etc.



Etapas de um sistema de reconhecimento de padrões.

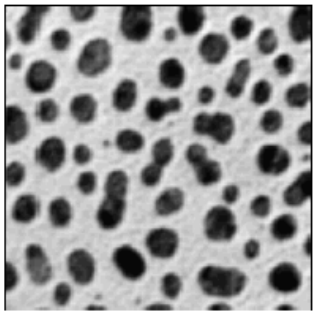
Segmentação

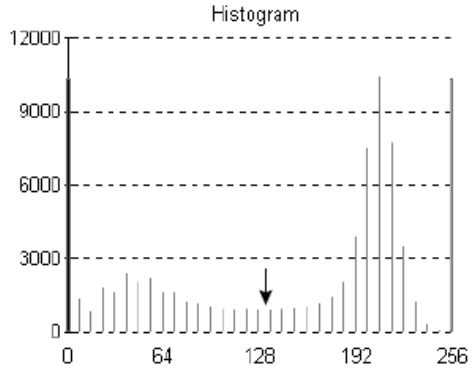
Divisão da imagem em regiões que possuem o mesmo conteúdo no contexto de uma aplicação.

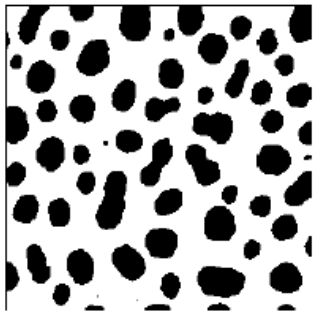


A segmentação baseada em:

- Descontinuidades
  - Mudanças bruscas de tons
- Similaridades
  - aspectos comuns com limiar
- Limites ou bordas
- Áreas ou regiões







Segmentação Baseada em Regiões

Partição da imagem baseada no conteúdos de grupos de pixels.

Premissas:

(1) Homogeneidade da região (com tolerância)

(2) Regiões delimitadas por fronteiras contínuas

(3) Pontos que correspondem a uma única região

(4) O conjunto de todas as regiões deve formar a imagem

Técnicas:

- Segmentação por crescimento de regiões
- Segmentação por divisão e fusão de regiões
- Segmentação por clusterização
- Segmentação por janelas (windows)

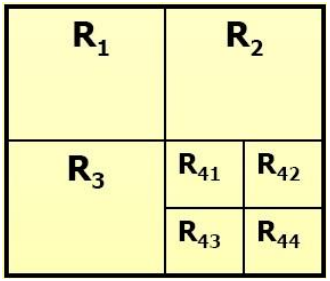
Segmentação por crescimento de regiões

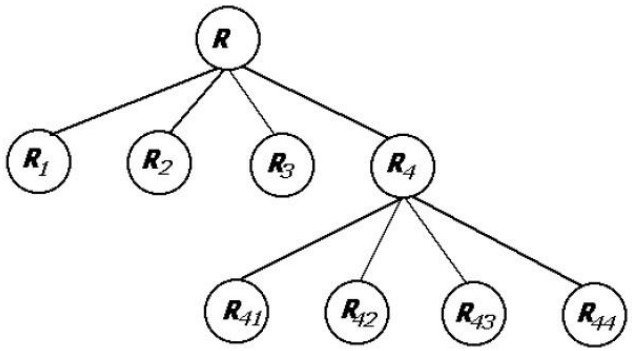
- Iniciar a partir de um *pixel* ou um conjunto de *pixels* (denominado de "***semente***").
- Para cada semente avalia-se o predicado dos *pixels* vizinhos
  - Ex. cor RGB com menos de 5% da variação de 5 pixels vizinhos
- A agregação das regiões é feita quando o critério de similaridade ou de decisão do predicado for verdadeiro.
- Critério de parada bem definido

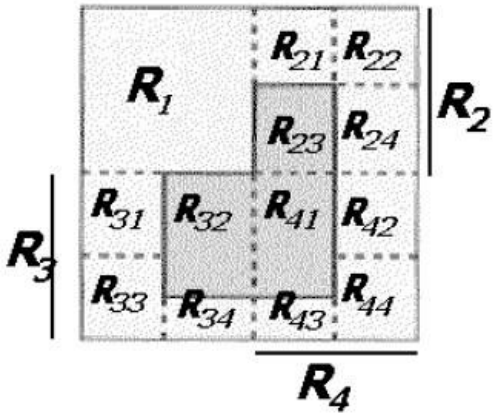
50	51	50	102
51	49	50	102
240	240	102	102
241	240	103	103

Segmentação por divisão e fusão de regiões

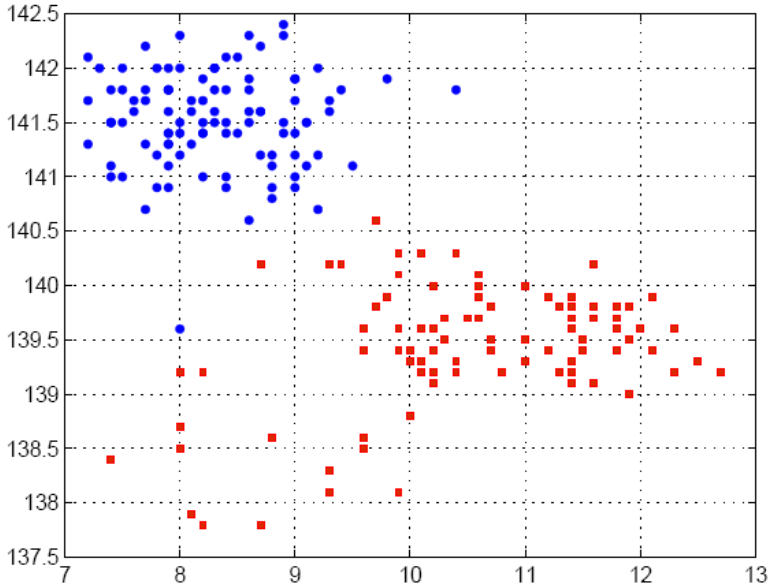
- Subdivide uma imagem em quadtree
  - Verificar se os *pixels* atendem a algum critério de homogeneidade.
- Os blocos que atenderem ao critério não serão mais divididos.
- O bloco que não atender será subdividido em blocos menores.
- Realiza a junção dos blocos vizinhos homogêneos.





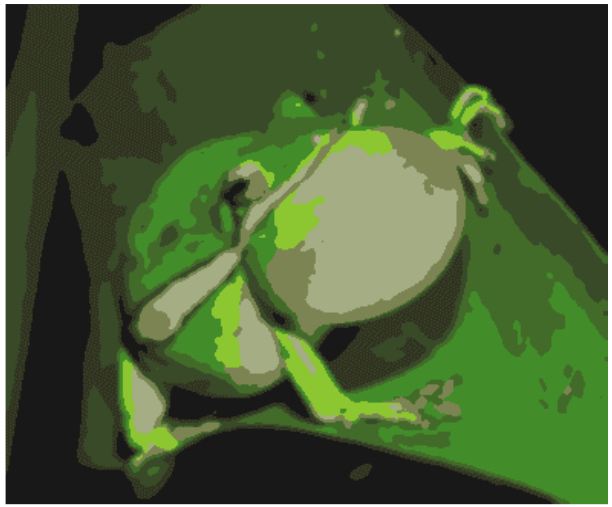


Segmentação por “clusterização”

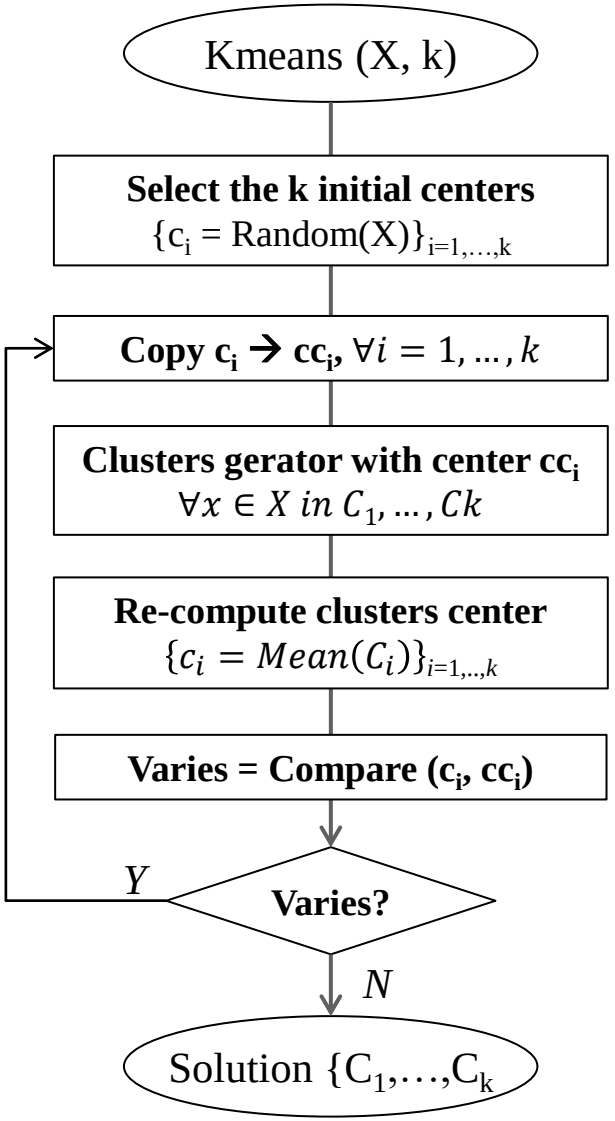


Algoritmo K-Means

- Algoritmo de classificação não-supersionada.
- O critério a ser minimizado é definido em função da distância dos elementos em relação aos centros dos agrupamentos.
- Usualmente, a métrica é a distância Euclidiana.
- Quanto menor for este valor, mais homogêneos serão os objetos dentro de cada grupo e melhor será a partição.



Algoritmo K-Means



Outras técnicas

- Filtragem no domínio espacial
  - Segmentação na própria imagem
    - Sem transformações
    - Uso de medidas calculadas na imagem
- Filtragem no domínio da freqüência
  - No espaço de transformada de Fourier
- Transformação para um espaço de medida específico
  - No espaço Euclidiano
    - Transformação linear para outro espaço
      - Ex. transformada de Hough, wavelets
- Baseadas em Morfologia Matemática
  - Transformada watershed (divisor de águas)
- Contornos ativos – ou modelos deformáveis
  - Snakes – extração de bordas de objetos da cena
    - Contorno ajustado a curvas (splines)
    - Inicialmente uma configuración inicial evolui até se ajustar ao objeto de interese

- 5/10
- Luis
- Yuri
- Maria

Propriedades do *Pixel*

Retangular ou quadrada. Três aspectos a considerar:

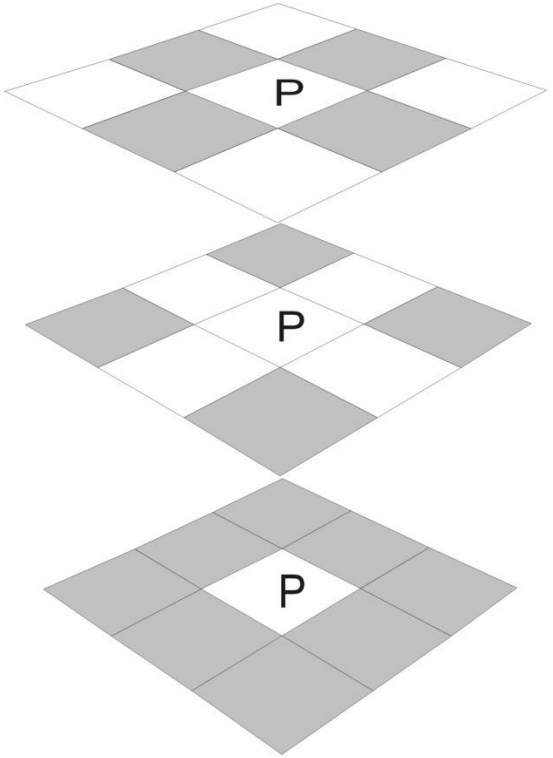
- Vizinhança em *Pixel* (Vizinhança-4 e Vizinhança-8)
- Medidas de Distância
- Conectividade

    * Propriedade de um pixel está conectado a outro)

Vizinhança em *Pixel* (Vizinhança-4 e Vizinhança-8)

Quais são os vizinhos de um determinado pixel?

- *Importante para segmentação e continuidade do objeto*



(a) *vizinhança-4* de p - $N_4(p)$

(b) *vizinhança-D* de p – $N_D(p)$

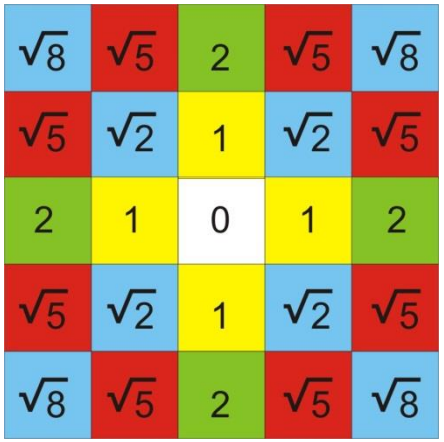
(c) *vizinhança-8* de p – $N_8(p)$

Medidas de Distância

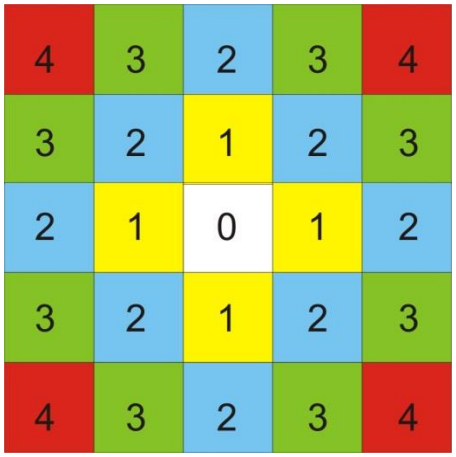
Distância *city-block*, *Manhatan* ou quarteirão para e distância Euclidiana para :

$$D(X_i, X_j) = \left[\sum_{l=1}^{n} \left| x_{il} - x_{jl} \right|^r \right]^{\frac{1}{r}}$$

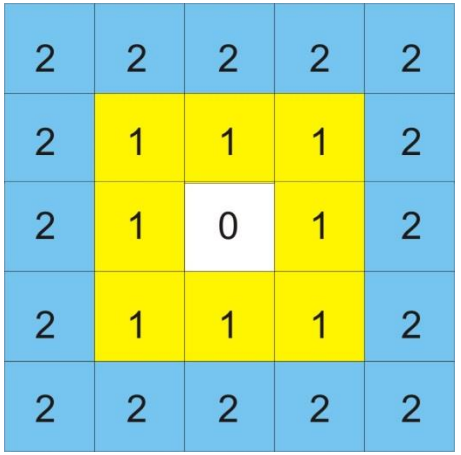
X_i, X_j *vetor de elementos*



Distância Euclidiana
$\leq \sqrt{8}$ do *pixel* central



Distância Manhattan
$D_1 = |x_{ix} - x_{jx}| + |x_{iy} - x_{jy}|$



$D(p,q) = Máx\ |x_{ix} - x_{jx}|, |x_{iy} - x_{jy}| \} \leq 2$

Conectividade

Dois *pixels* estão conectados se:

- são adjacentes ($N_4(p)$ ou $N_8(p)$); e,
- atributos (níveis de cinza, texturas ou cores) similares.

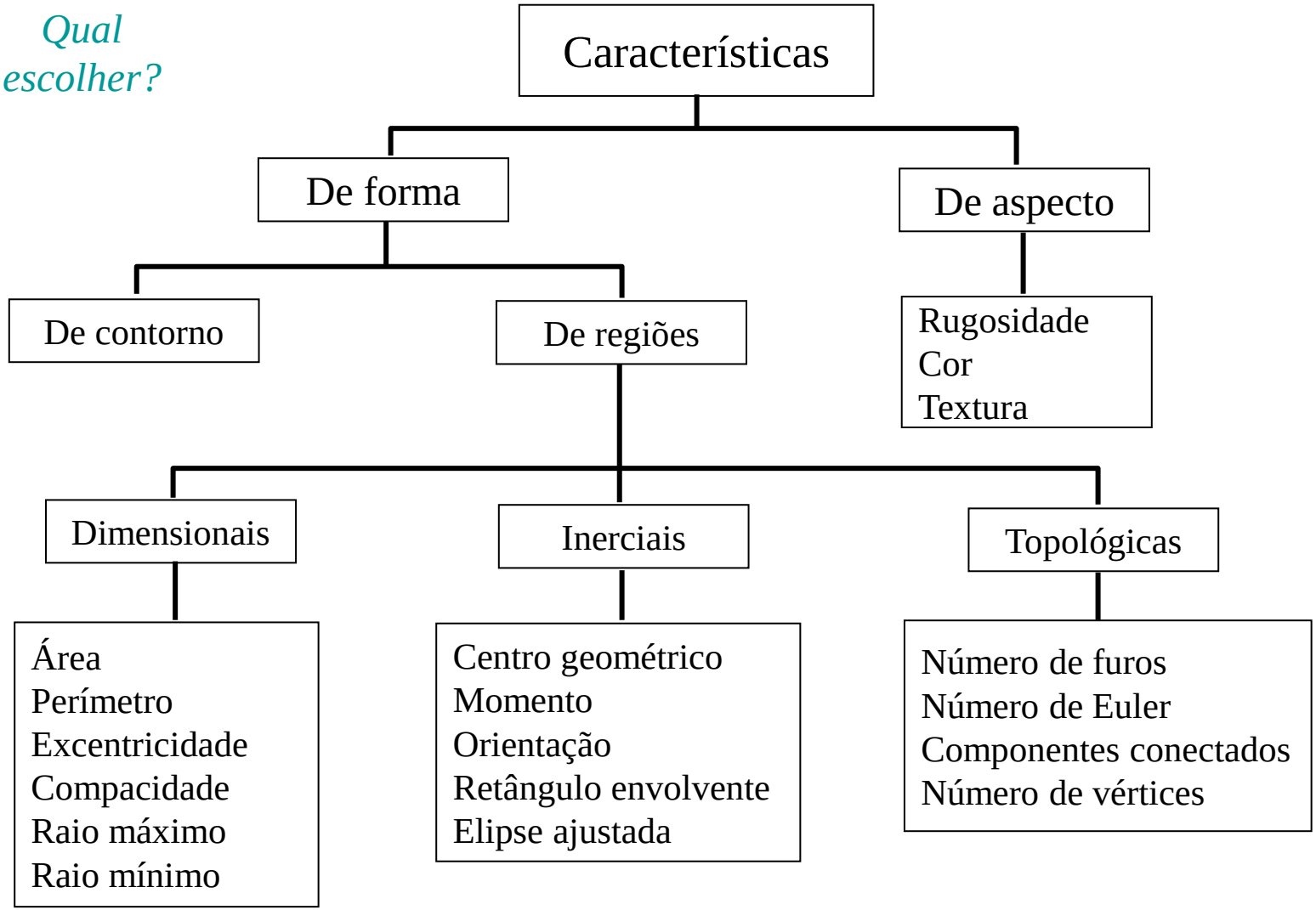
Níveis de conectividade:

§ **Conectividade de 4:** q está em $N_4(p)$ e atributos iguais.

§ **Conectividade de 8**: q está em $N_8(p)$ e atributos iguais.

Tipos de características

Qual escolher?



Análise de Componentes Principais (PCA)

- **Componentes**
  - **principal** representa melhor a distribuição dos dado
  - **secundária** é perpendicular à componente principal.
- Passos:
  - Obter as n amostras
  - Calcular a média
  - Calcular a matriz de covariância
  - Calcular os autovalores e autovetores da matriz de covariância
  - Componente principal e secundaria: autovetores de maior e menor autovalor, respectivamente.

Matriz de covariância

A matriz de covariância para *M* amostras de vetores p_i, com vetor médio *m* pode ser calculada de acordo com:

$$C_x = \frac{1}{M}\sum_{i=1}^{M}(x_i - m_x)(y_i - m_y)$$

O vetor médio pode ser calculado:

$$m = \frac{1}{M}\sum_{i=1}^{M} p_i$$

```
c_real32 *calcula_autovalores(c_triple_real32 *cov)
{
        c_uint16 x, y;   c_real32 *lambda;   c_real32 m;
        lambda = aloca_array_real32((c_uint16)2);
        x =0;   y =1;
        m =  sqrt(pow((cov[x][x] - cov[y][y]), 2)
                    + 4.0f*pow(cov[x][y], 2));
        lambda[0] = ((cov[x][x] + cov[y][y] + m) / 2.0f);
        lambda[1] = ((cov[x][x] + cov[y][y] - m) / 2.0f);
        return lambda;
}
```

```
Vetor caixa_retangular(float [][] cov)
{
        calcula_matriz_covariancia_area(vetor pts)
        lambda = calcula_autovalores(cov);
        ordena_autovetores_para_eixos(lambda);
        xy = calcula_autovetores(cov, lambda);
}
```

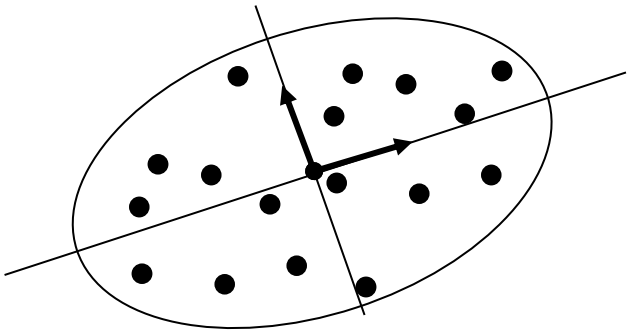
```
c_triple_real32 *obbtree_calcula_autovetores(c_triple_real32 *cov, c_real32
*lambda)
{
        c_uint16 x, y;   c_triple_real32 *xy;   c_real32 m1, m2;
        x =0;   y =1;
        xy = aloca_array_triple_real32((c_uint16)3);
        if((lambda[0] == 0.0f) && (lambda[1] == 0.0f))    // se for circulo
        {
                ca_scala_triple_real32(v_un_x, 1.0f, xy[0]);  // vetor unitario (1,0)
                ca_scala_triple_real32(v_un_y, 1.0f, xy[1]);  // vetor unitario (0,1)
        }   else   {
                xy[0][0] = (c_real32)(-1.0f * cov[x][y]);
                xy[0][1] = cov[x][x] - lambda[0];
                xy[0][2] = 0.0f;
                m1 = ca_modulo_triple_real32(xy[0]);
                xy[1][0] = (c_real32)(-1.0f * cov[x][y]);
                xy[1][1] = cov[x][x] - lambda[1];
                xy[1][2] = 0.0f;
                m2 = ca_modulo_triple_real32(xy[1]);
                if(m1 > m2)      {
                    m2 = (c_real32)1.0f / m1;
                    ca_scala_triple_real32(xy[0], m2, xy[0]);
                    xy[1][0] = (c_real32)(-1.0f * xy[0][1]);
                    xy[1][1] = xy[0][0];
                }      else      {
                    m1 = (c_real32)1.0f / m2;
                    ca_scala_triple_real32(xy[1], m1, xy[1]);
                    xy[0][0] = (c_real32)(-1.0f * xy[1][1]);
                    xy[0][1] = xy[1][0];
                }
        }
        return xy;
}
```

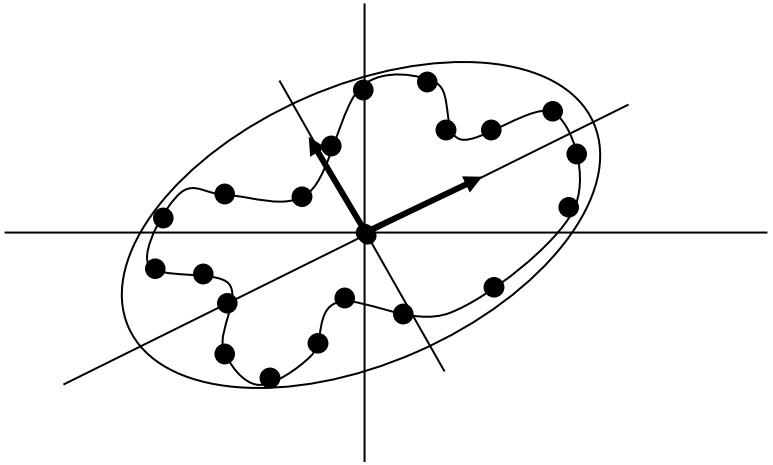
Autoespaços , autovetores e autovalores

Um vetor v é um **autovetor** de uma matriz quadrada M se

$$M\,v = \lambda\,v$$

Escalar λ é **autovalor** de M associado ao autovetor v.

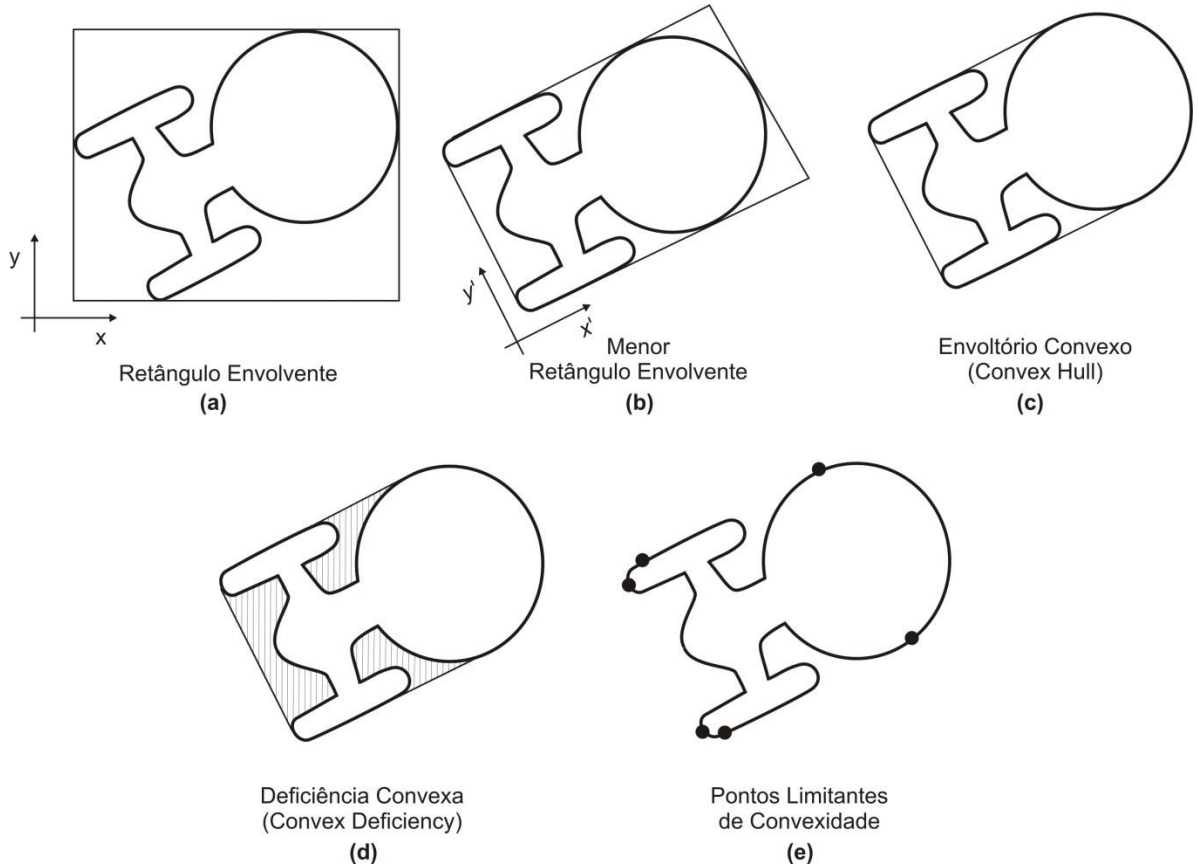




Descritores de forma

Área e Retângulos envolventes

É necessário que a imagem tinha sido segmentada



Perímetro, Alongamento e Retangularidade

P**erímetro -** número de *pixels* conexos que constituem o contorno da região.

Alongamento - relação de lados do menor retângulo que envolve o objeto.

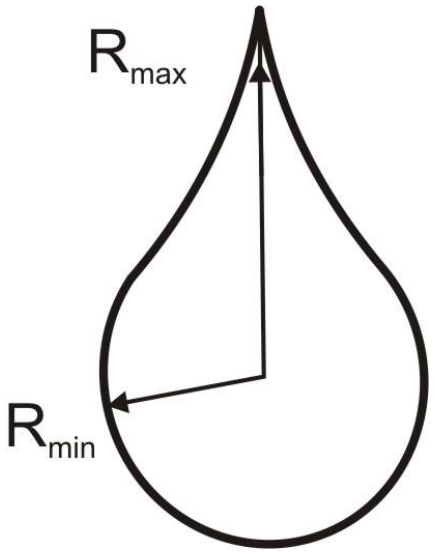
Retangularidade - relação entre a área do objeto e área do menor retângulo que o envolve.

Excentricidade, diâmetro, raio máximo e **mínimo do objeto**

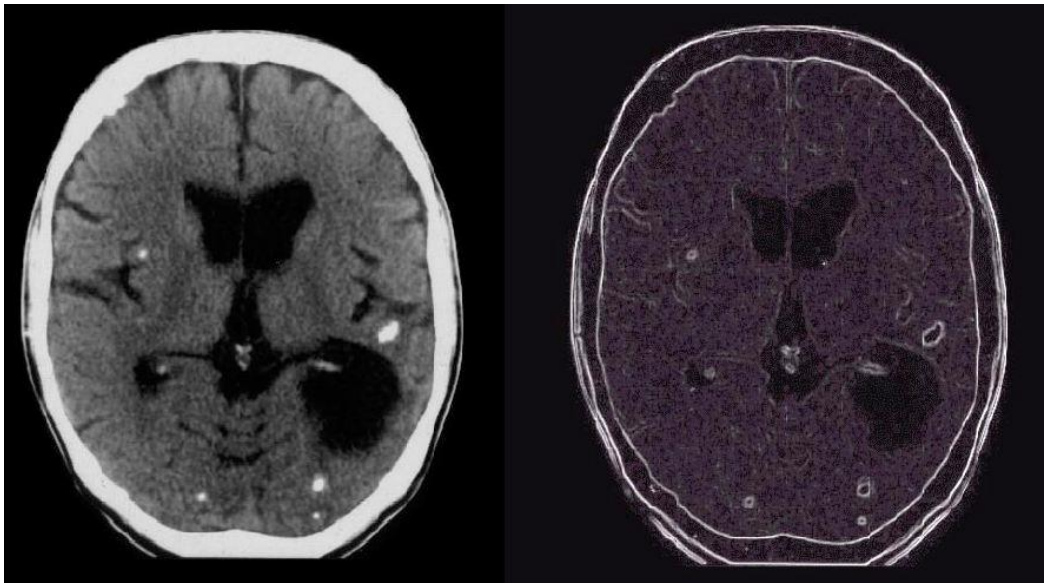
Diâmetro de um objeto - maior distância entre 2 pontos deste objeto.

Excentricidade - relação entre dois pontos extremos do objeto que passem pelo **eixo maior** e **eixo ortogonal**.

Raio máximo e mínimo do objeto - distâncias máxima e mínima, respectivamente, da borda ao centro geométrico.



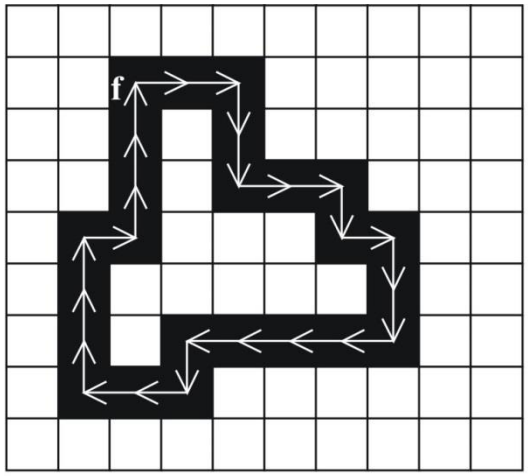
Contornos

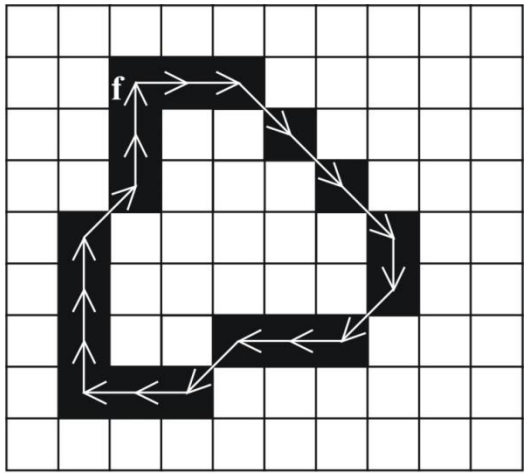


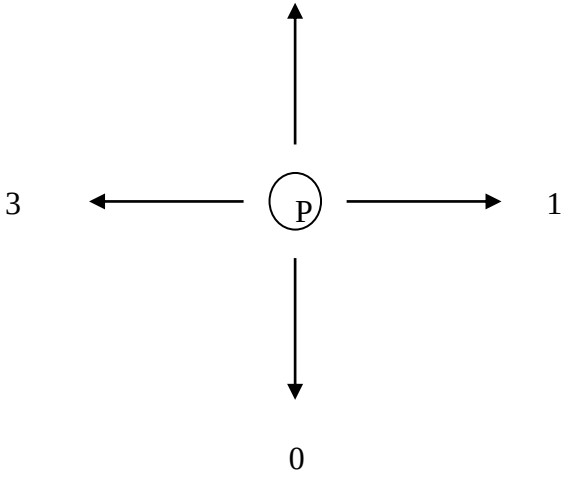
(a) (b)

Exemplo de aplicação do filtro de gradiente (b) para acentuar o contorno em uma imagem de tomografia (a). Neste exemplo foram realizados procedimentos para ligação de bordas.

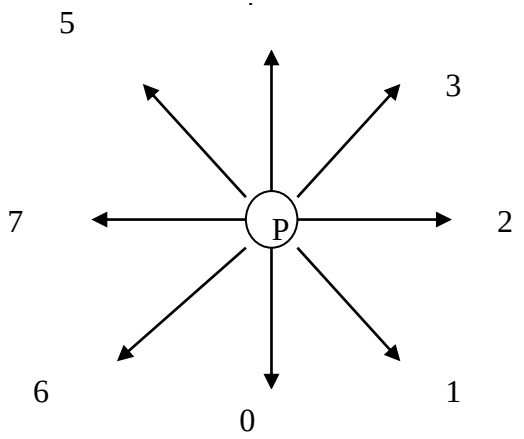
Código da Cadeia



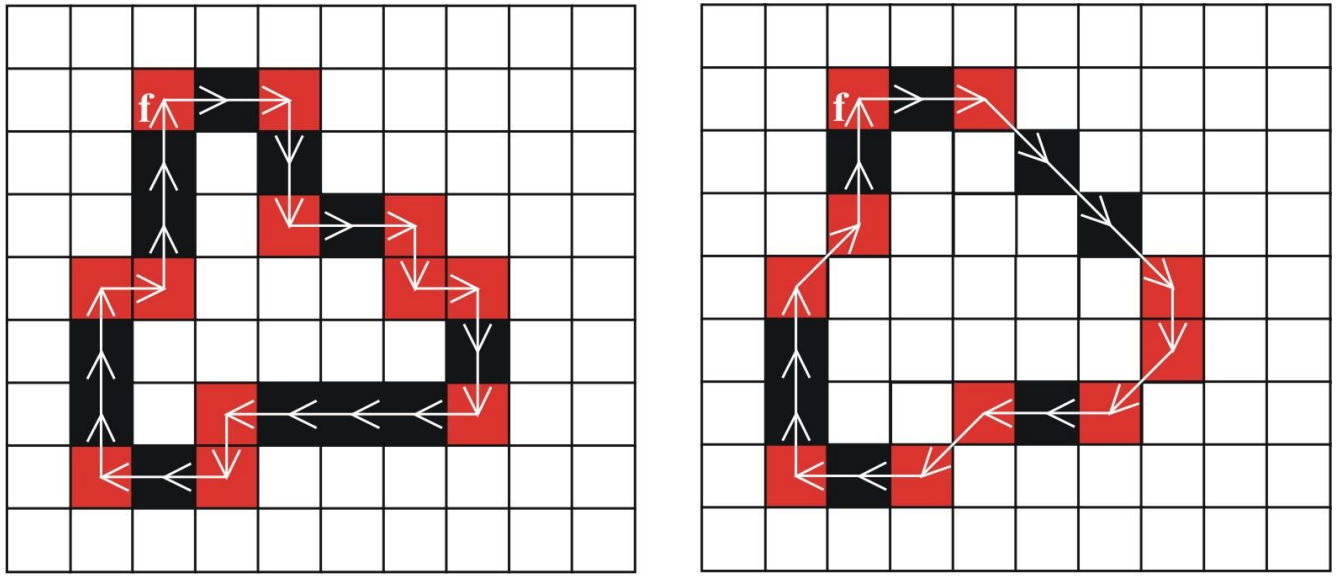




Vizinhança-4 de *p*, *N4(p)*



Vizinhança-8 de *p*, $N_8(p)$



(a) (b)

Pontos onde o código se diferencia do vizinho.

- 10/10
- Maria
- Luis
- Yuri
- Javier

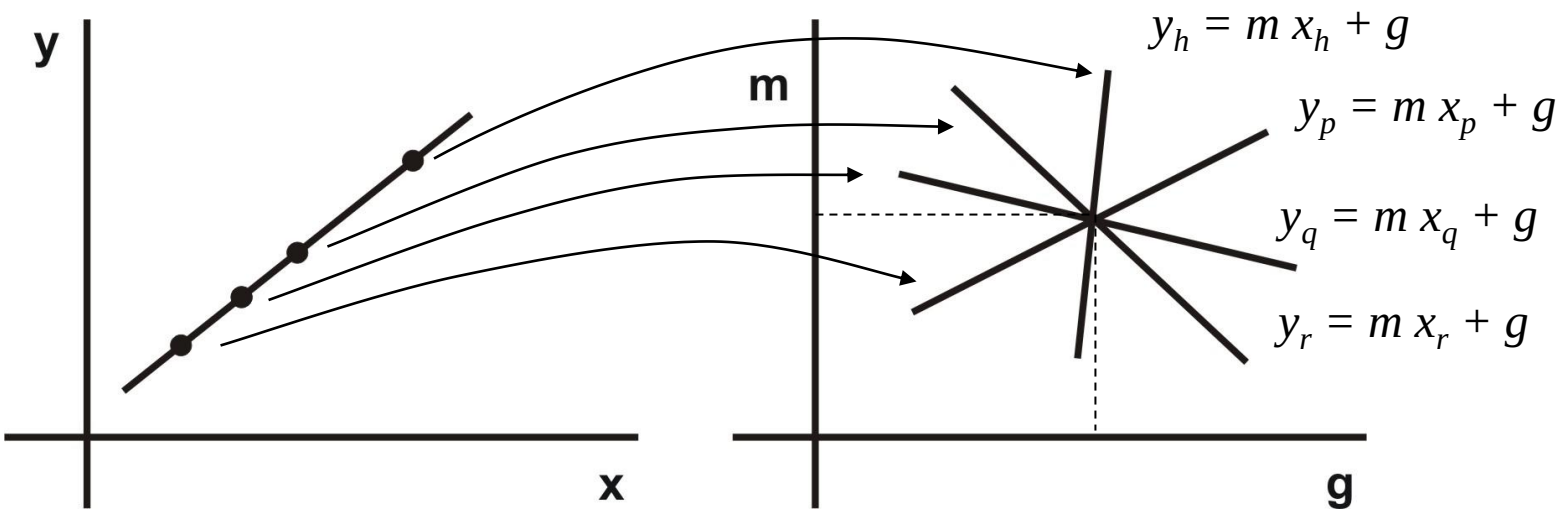
Transformada de Hough

Transformar a imagem do espaço digital *(x,y)* para uma representação na forma dos parâmetros descritos pela curva que se deseja encontrar na imagem

Etapas da aplicação da transformada de Hough para qualquer forma geométrica.

Retas:

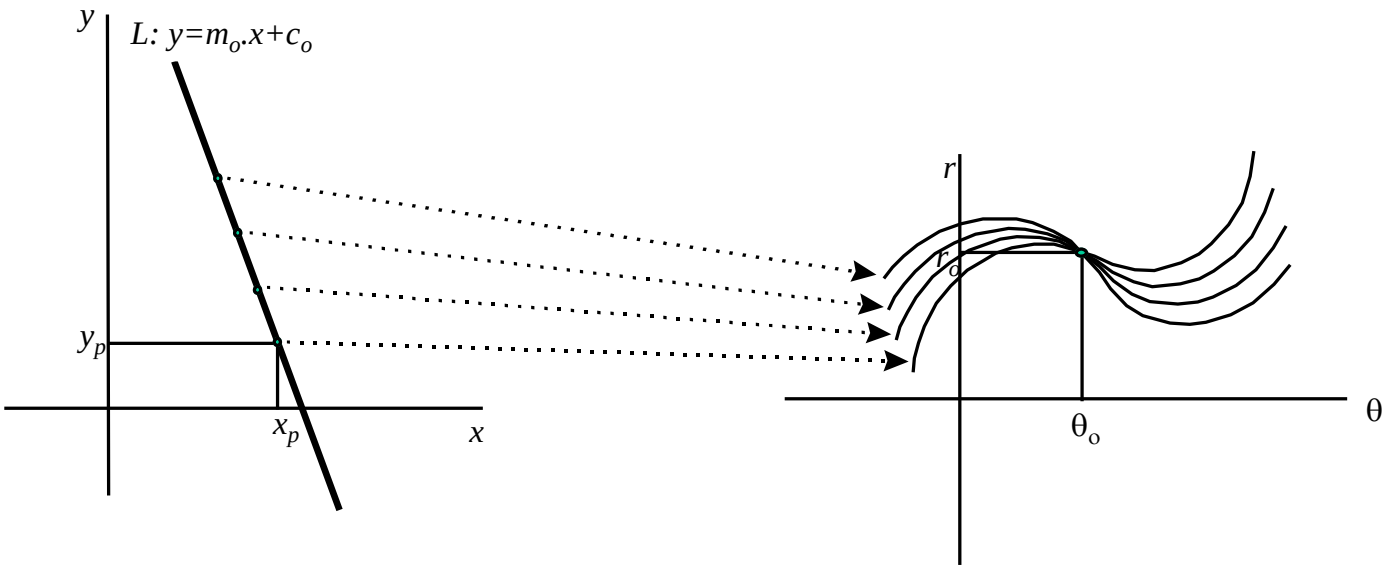
$$y = mx + g$$



espaço (x,y) espaço de parâmetros(m,g)

- Cada ponto no espaço da imagem transforma-se em uma reta no espaço de parâmetro: $g = -mx + y$.
- Para reta vertical m = 0 ➔ infinita (não funciona)

Retas – forma polar: $r = x\cos\theta + y\sin\theta$



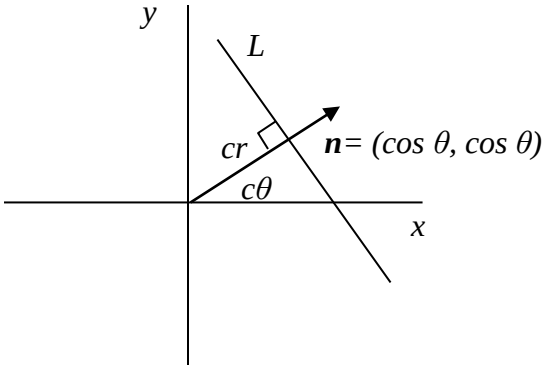
Plano *xy* (espaço de imagem)

Plano θ*r* (espaço de parâmetros)

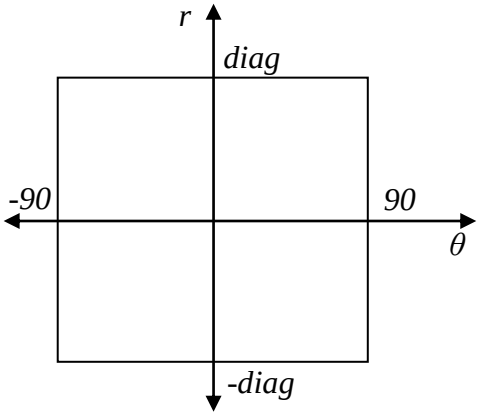
Cada ponto P(x,y) no espaço da imagem, corresponde a uma senóide S(ρ,θ) no espaço de parâmetros.

Algoritmo de Hough (para retas)

1. *Discretizar espaço de parâmetros S(θ, r) em (θ_{min} , θ_{max}) x (r_{min}, r_{max})*
   - *Matriz acumulador A de inteiros*
2. *Zerar A (valor inicial)*
3. *Para cada pixel (x,y), com gradiente maior que o limiar zero*
   - *Calcular as coordenadas (cθ, cr) de A restrita à linha desejada*
   - *Incrementar: A (cθ, cr) += 1*
4. *Buscar o máximo local em A → (cθ, cr)*
5. *Converter (cθ, cr) para espaço de imagem*

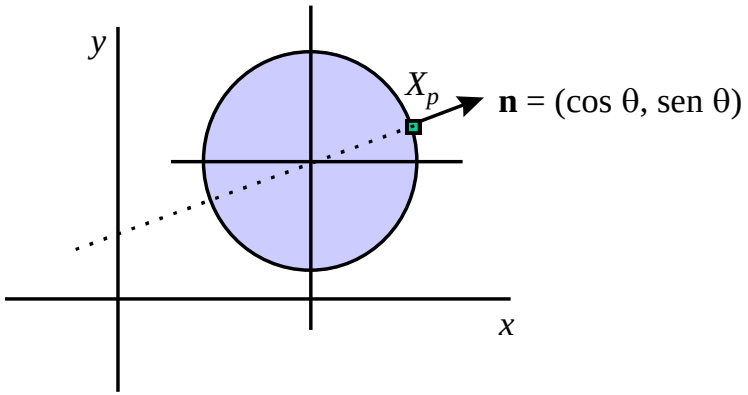


Espaço de imagem

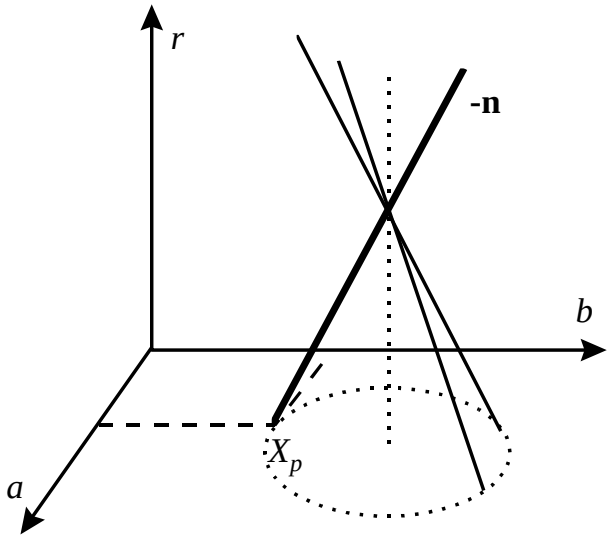


Espaço de parâmetros

Circulo: $(x - a)^2 + (y - b)^2 = r^2$



Espaço de imagem



Espaço de parâmetros

- *Acumulador A(_,_,_)*
- $0 \leq r \leq diag$ *// diag diagonal do plano da imagem*
- $0 \leq a, b \leq diag.$

Usando gradiente

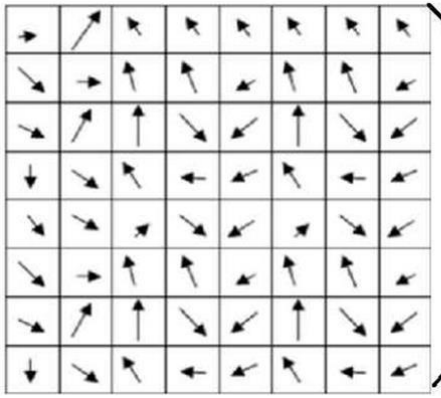
$$\frac{\partial}{\partial x}\left[(x-a)^2 + (y-b)^2 = r^2\right]$$

$$b = y \pm \frac{r}{\sqrt{1+\tan^2(\theta)}} = y \pm r.\cos(\theta)$$

$$a = x \pm \frac{r}{\sqrt{1+\ 1/\tan^2(\theta)}} = x \pm r.\sin(\theta)$$

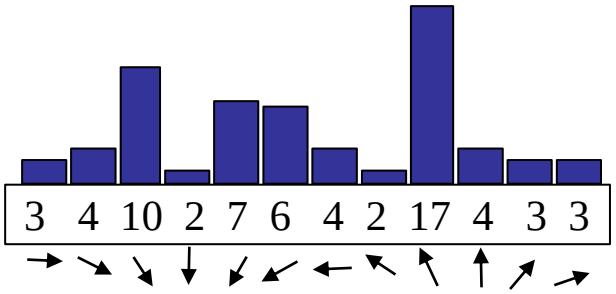
Histograma de Gradientes Orientados

- HOG (Histogram of Oriented Gradients)
  - Histograma local de gradientes
    - Áreas regulares na imagem



Células
8 x 8 pixels

Imagem: 10 x 10
Células



← (x orientação)

Descritor global

$x = (x_1,.., x_1)$

Gradiente

Kernel:
[-1 0 1]

Gradiente no eixo X

Kernel:
[-1 0 1]^T

Gradiente no eixo Y

Kernel:

No Eixo X: $[-1 \quad 0 \quad 1]$

No Eixo Y: $\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix}$

Variação de intensidades (Altas frequências)

Magnitude

$$m(x_i, y_i) = \sqrt{\partial_x(x_i, y_i)^2 + \partial_y(x_i, y_i)^2}$$

Orientação

$$ang(x_i, y_i) = aTan\left(\frac{\partial_y(x_i, y_i)}{\partial_x(x_i, y_i)}\right)$$

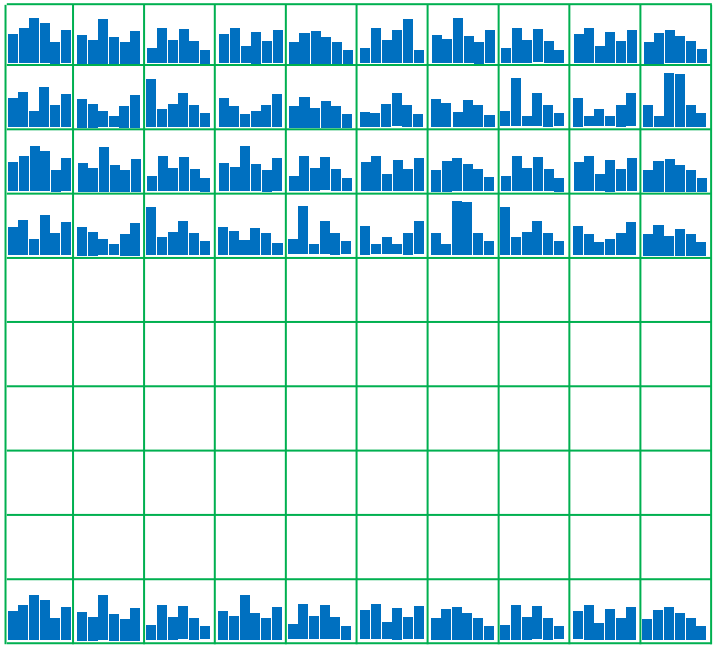
Magnitude

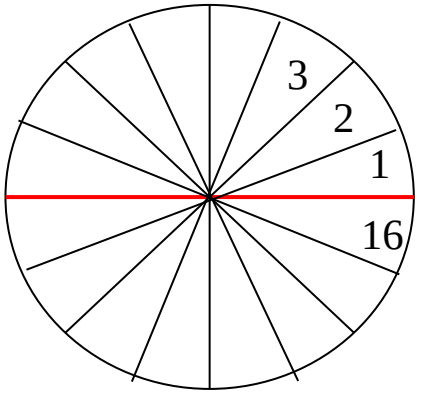
Orientação (ang)

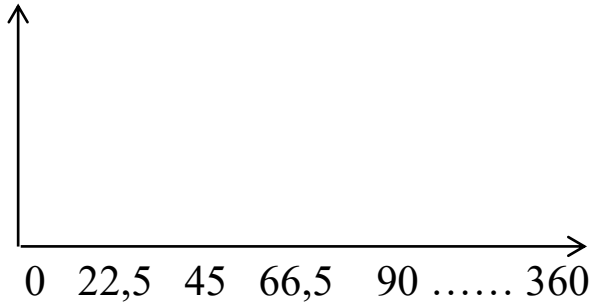
Processo

Calcular as HOG de todas as células

ang.

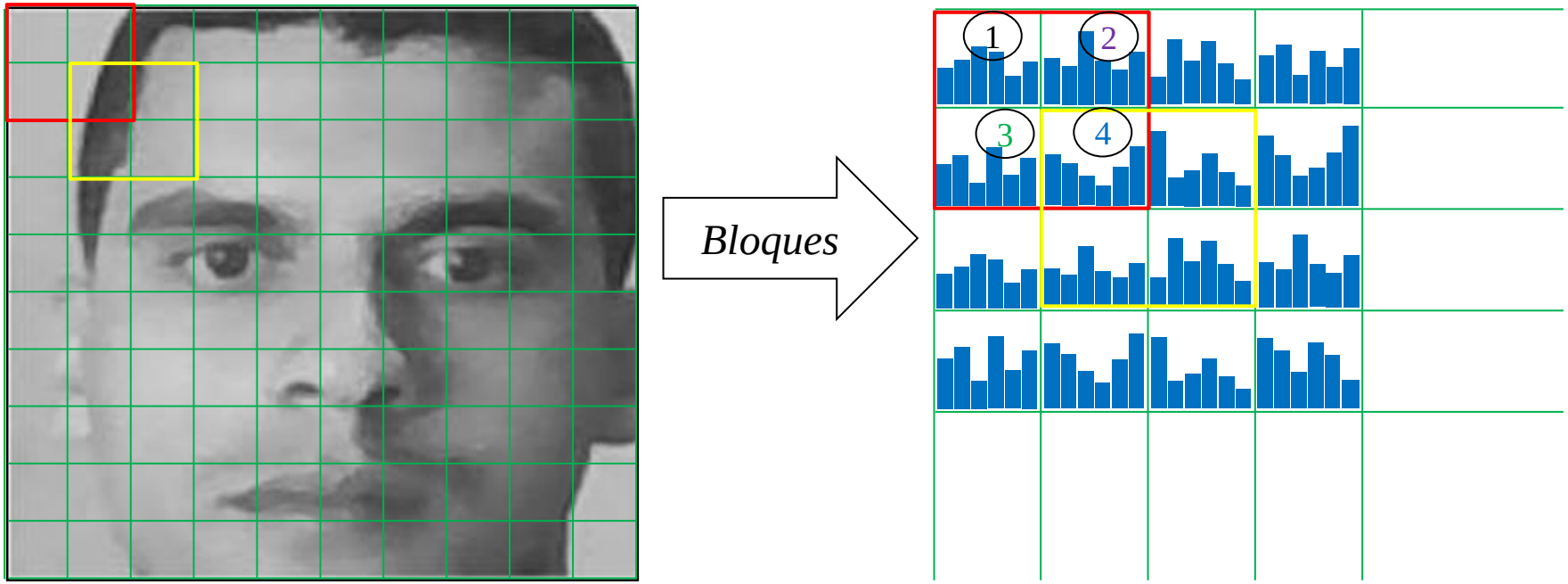






Normalização

Variações de iluminação → variam os gradientes

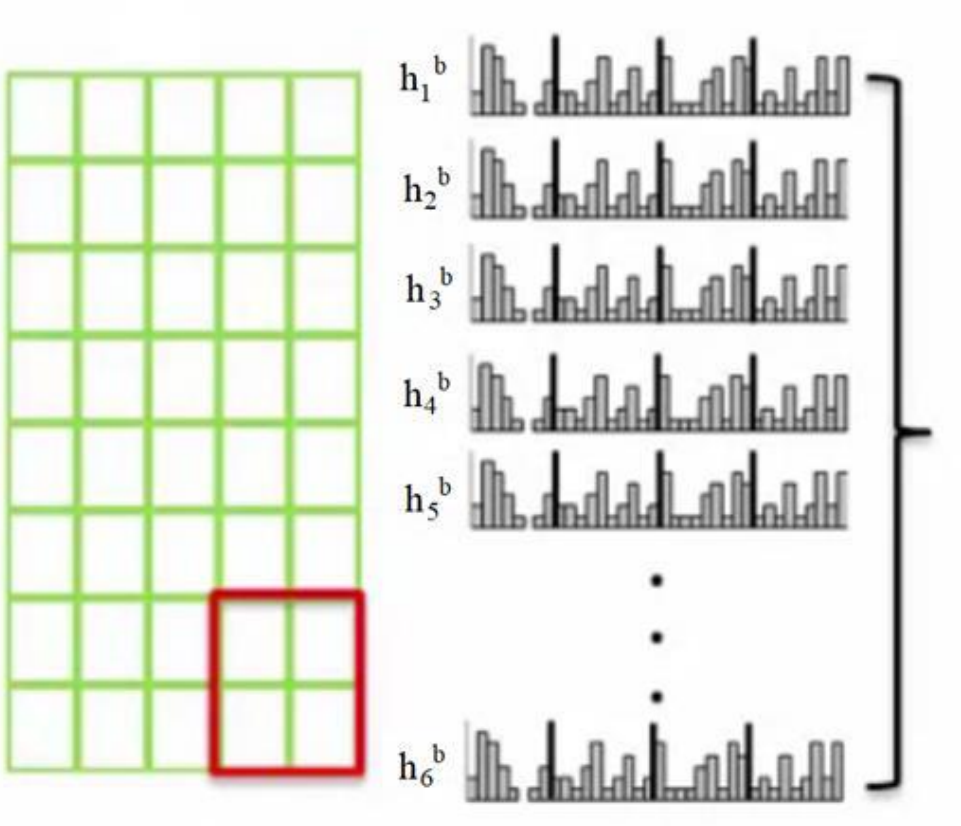


$V_1 = (x_{11}, \ldots, x_{1d}, x_{21}, \ldots, x_{2d}, x_{31}, \ldots, x_{3d}, x_{41}, \ldots, x_{4d})$

$Vn_1 = (x_{11}, \ldots, x_{1d}, x_{21}, \ldots, x_{2d}, x_{31}, \ldots, x_{3d}, x_{41}, \ldots, x_{4d}) / \| V_1 \|$ Norma L2

Norma L2: $\|v\| = (\varepsilon + \sum(x_i)^2)^{\frac{1}{2}}$

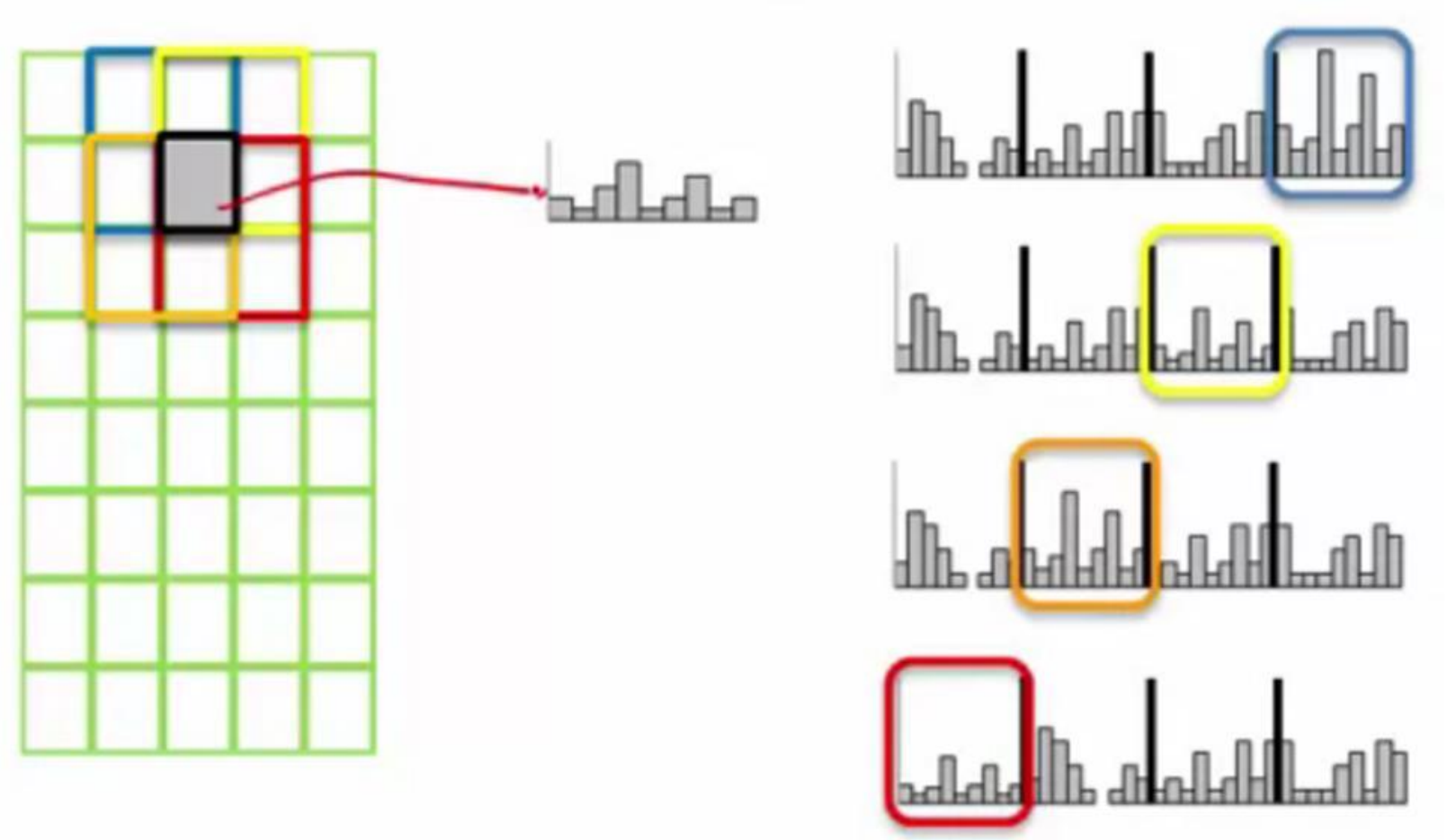
Vetor HOG



Concatenação

$HOG = (x_1, \ldots, x_n)$

Efeitos de Superposição



Reconhecimento de Padrões em Imagens

- **Reconhecimento de Padrões**
- **Classificação Supervisionada**
- **Classificação Não Supervisionada**
- **Redes Neurais Artificiais**
- **Lógica Fuzzy**