CC222 – Visão Computacional

Clustering

Instituto Tecnológico de Aeronáutica

Prof. Carlos Henrique Q. Forster – Sala 121 IEC

forster@ita.br

ramal 5981

Tópicos da aula

- Transformada de Hough
- Clustering de formas
- Hashing geométrico

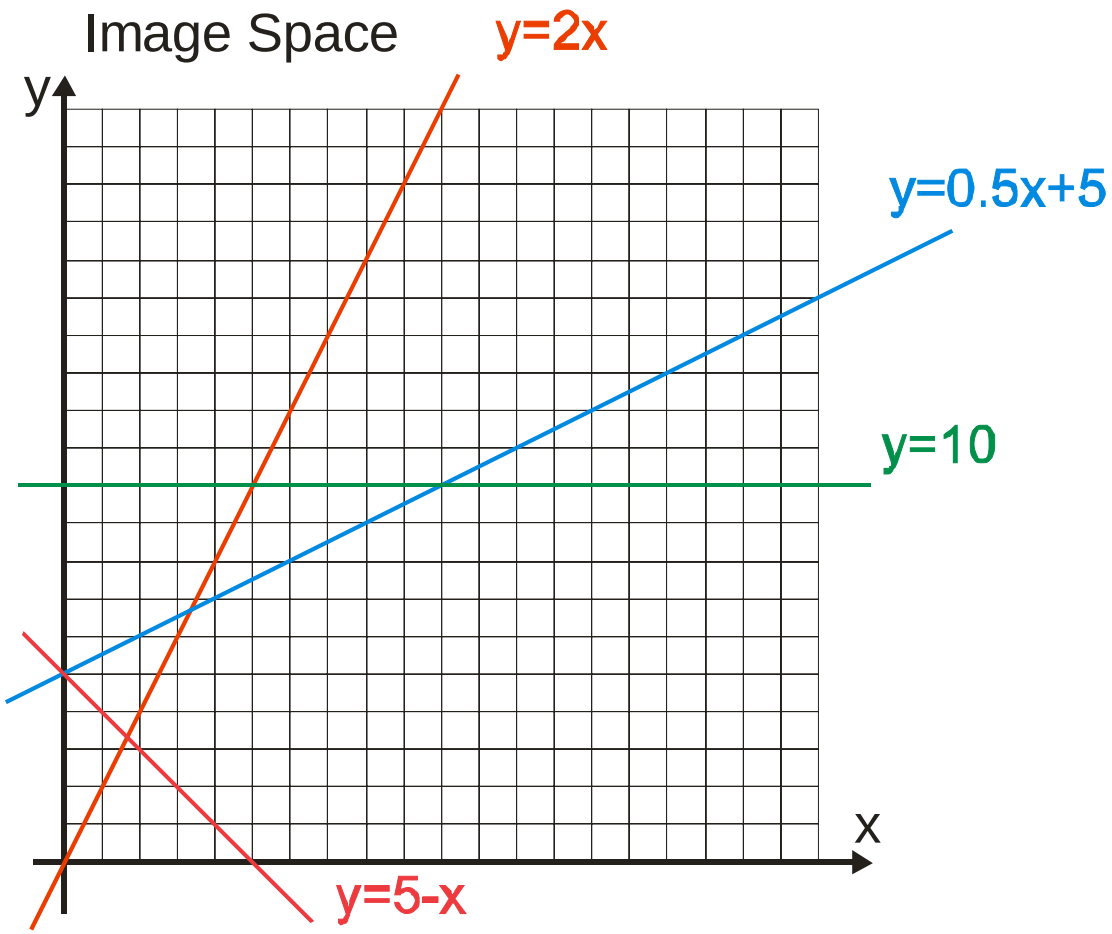
Livro para acompanhar essa aula

Gonzalez e Woods

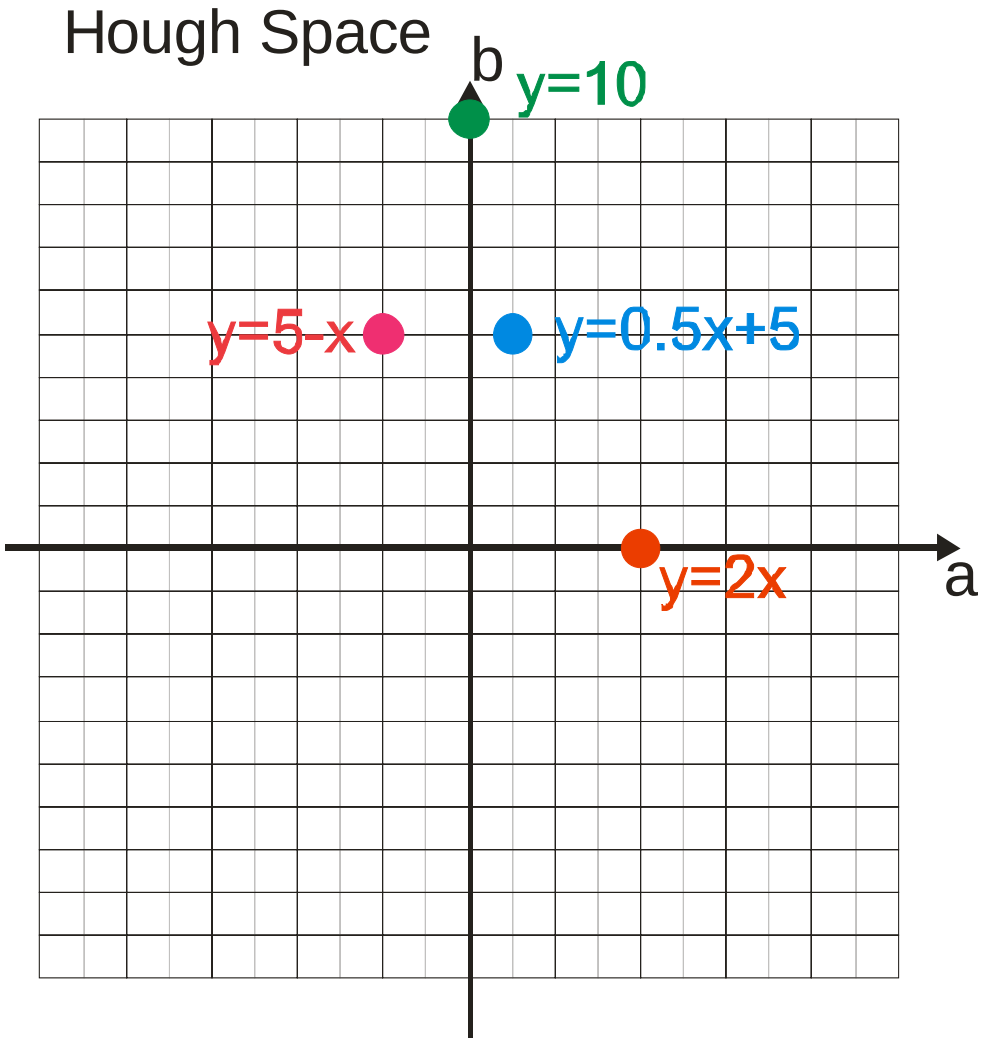
Shapiro e Stockman

Transformada de Hough

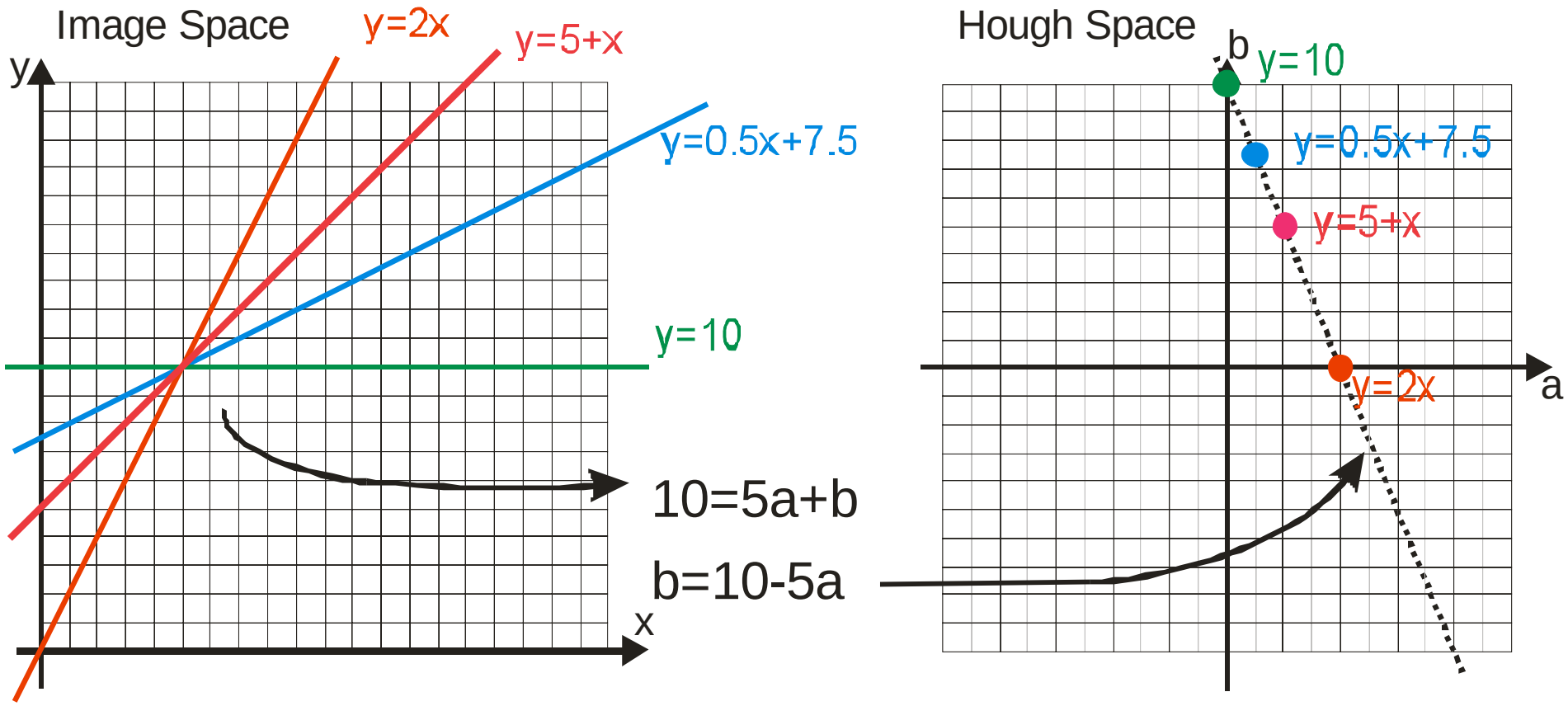
Parametização da reta $y = ax + b$



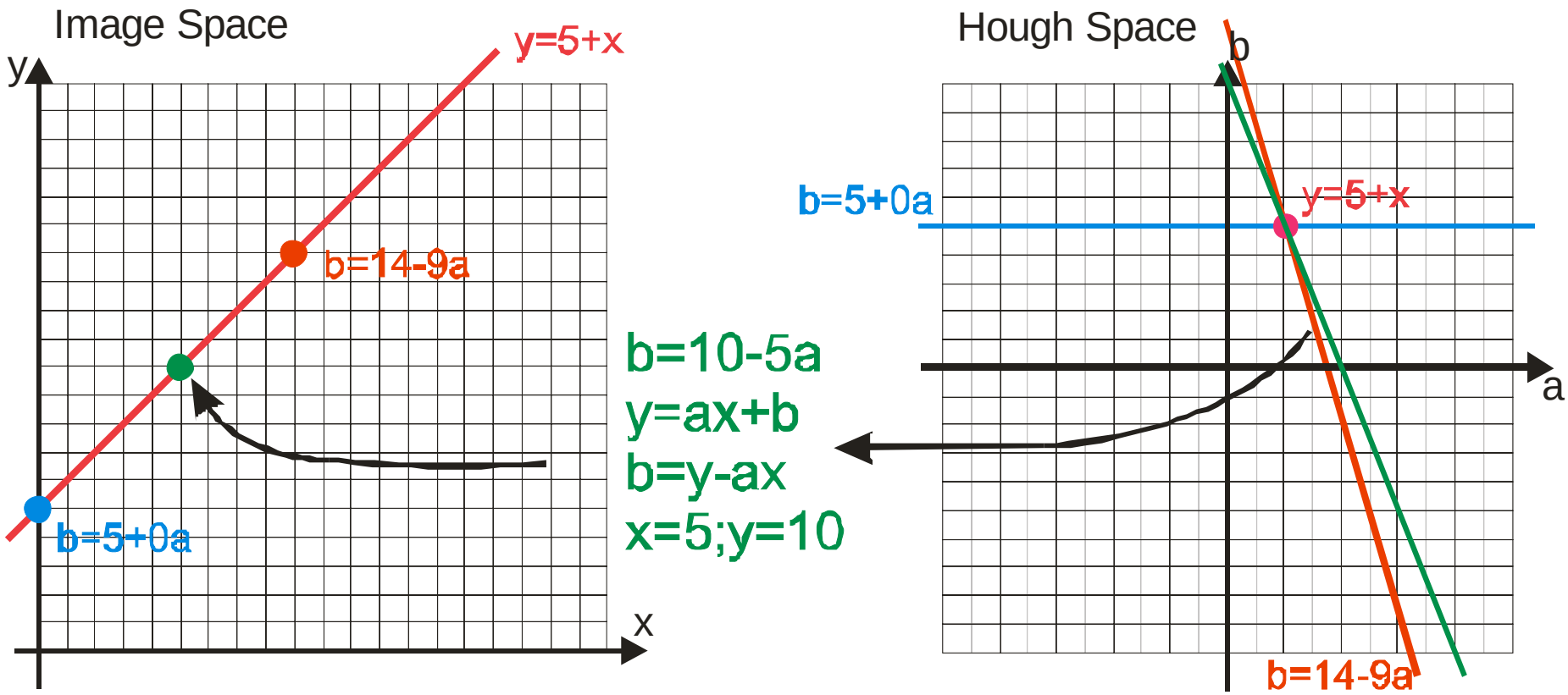
Retas no espaço de Hough (espaço de parâmetros)



Todas as retas que passam por um ponto



Dual: pontos pertencentes a uma reta



Matriz de acumulação de Hough

Para cada ponto encontrado na imagem

- Determinar os parâmetros da reta no espaço de Hough
- Adicionar “um” a cada célula da matriz de acumulação por que passa a reta

Selecionar as células com mais pontos acumulados

Para cada célula selecionada

- Identificar os parâmetros da reta sobre a imagem
- Verificar na imagem os segmentos de reta encontrados

Parametrização polar

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

Cada ponto x,y corresponde a uma linha sinusoidal em função de theta no espaço de Hough. Intersecções dessas linhas sinusoidais correspondem aos parâmetros das retas que passam pelos pontos x,y.

Exemplo

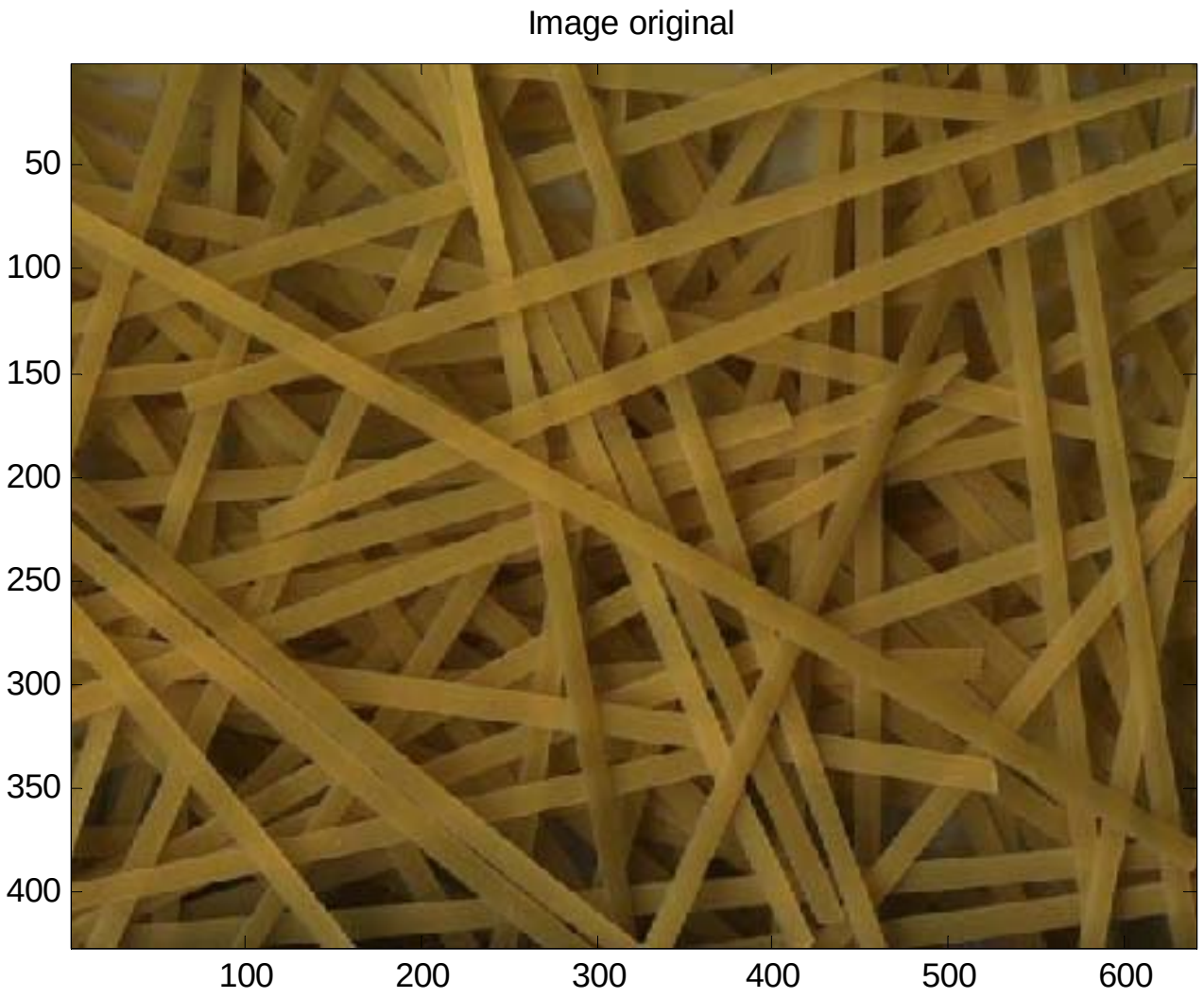
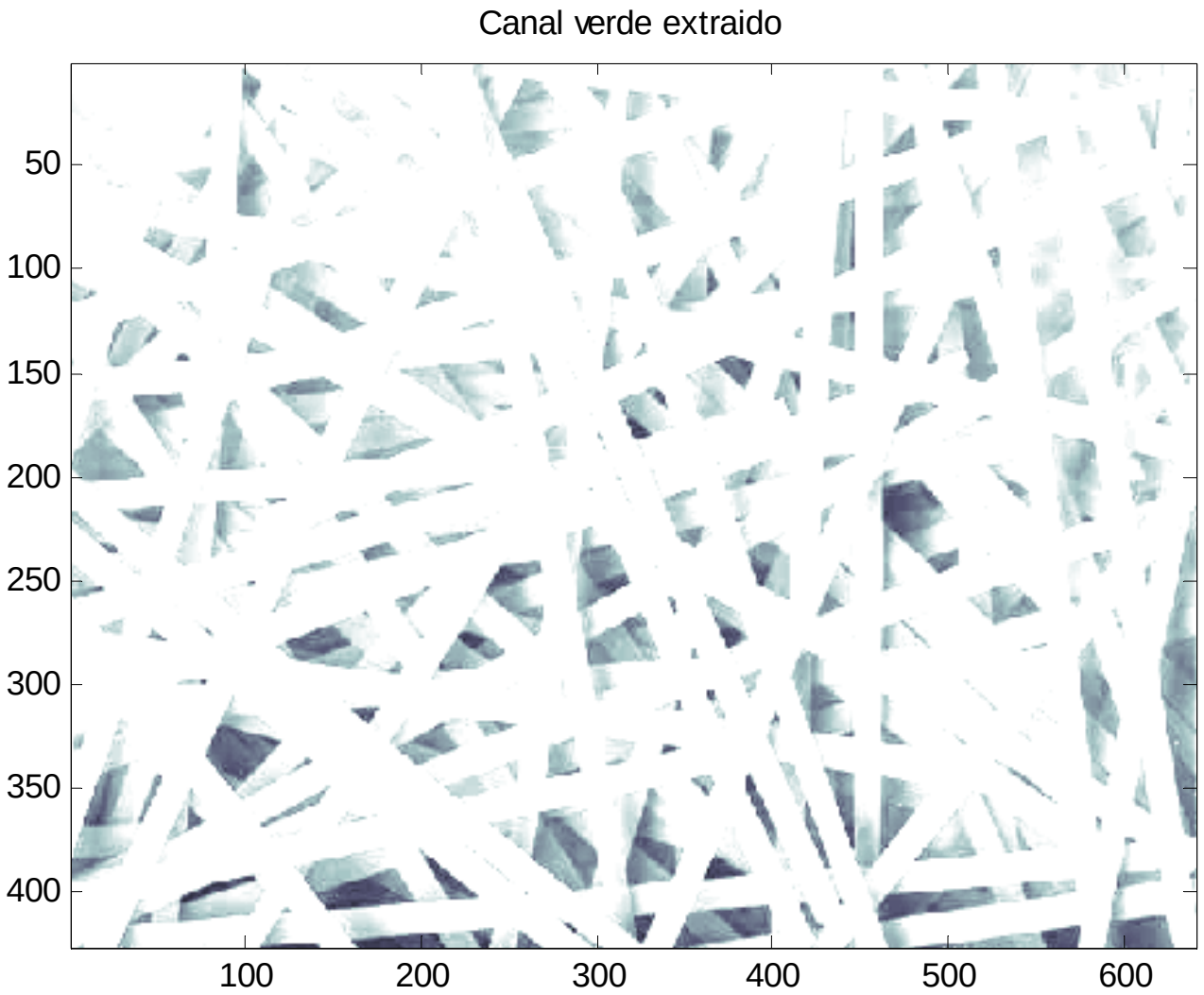
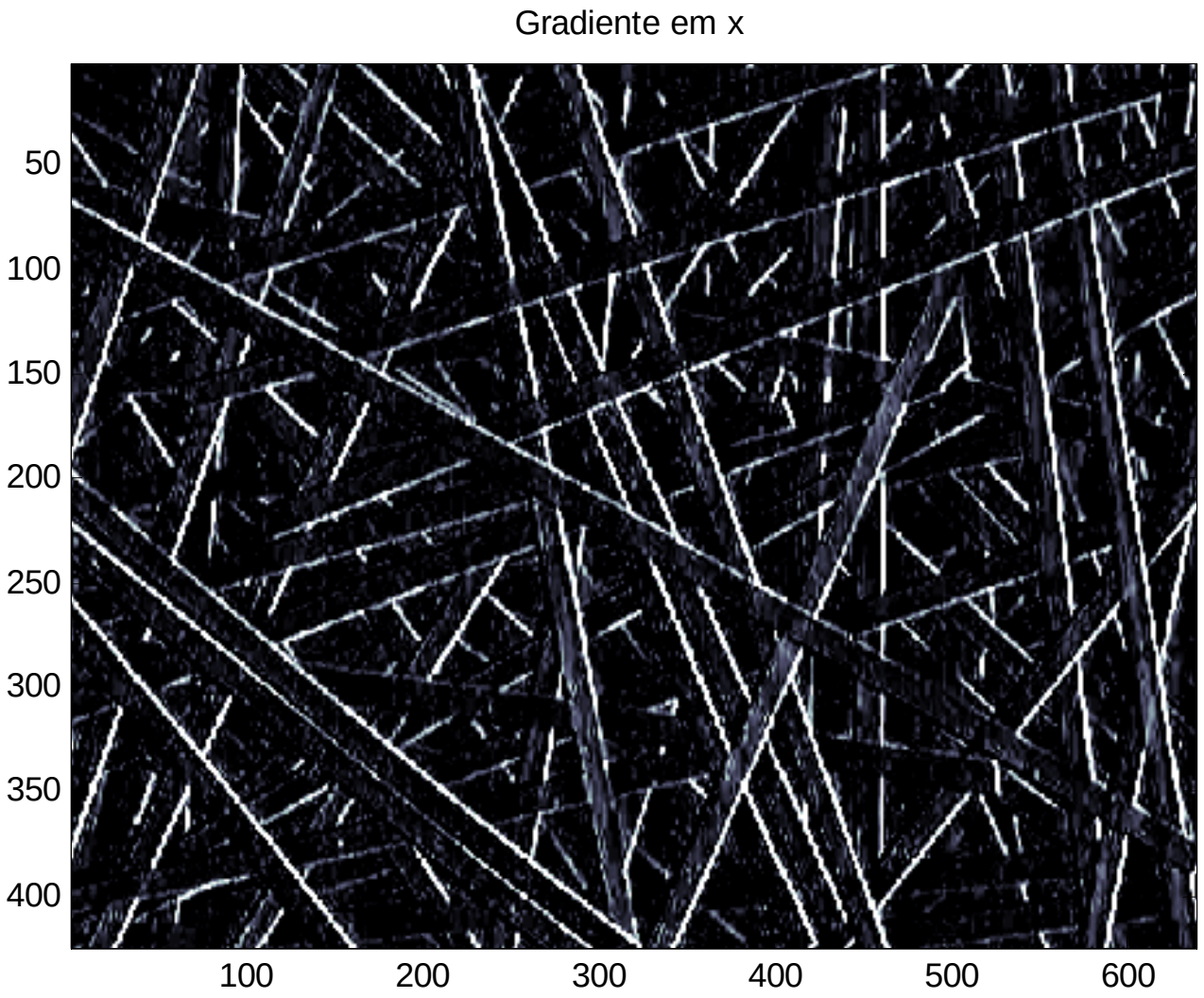
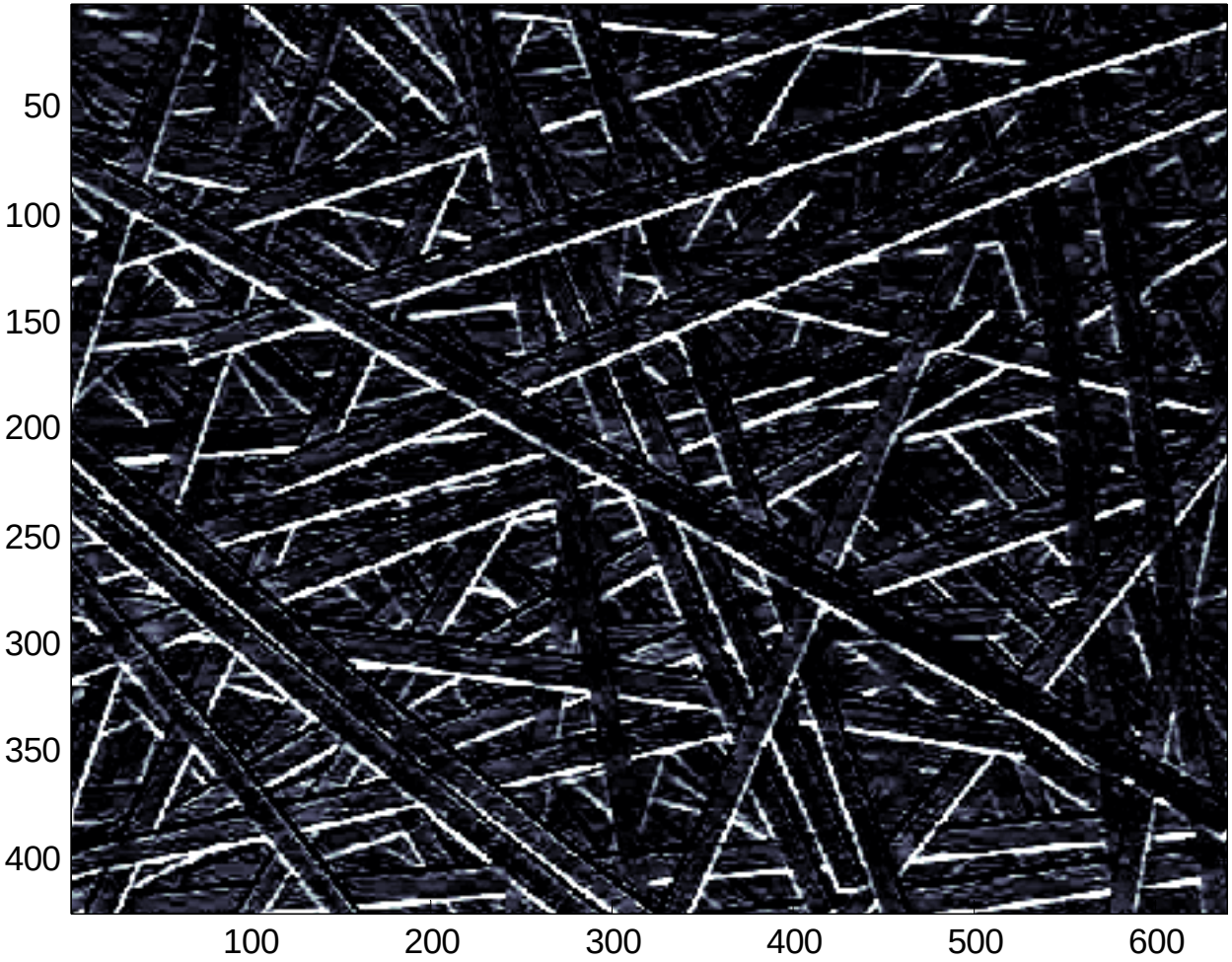


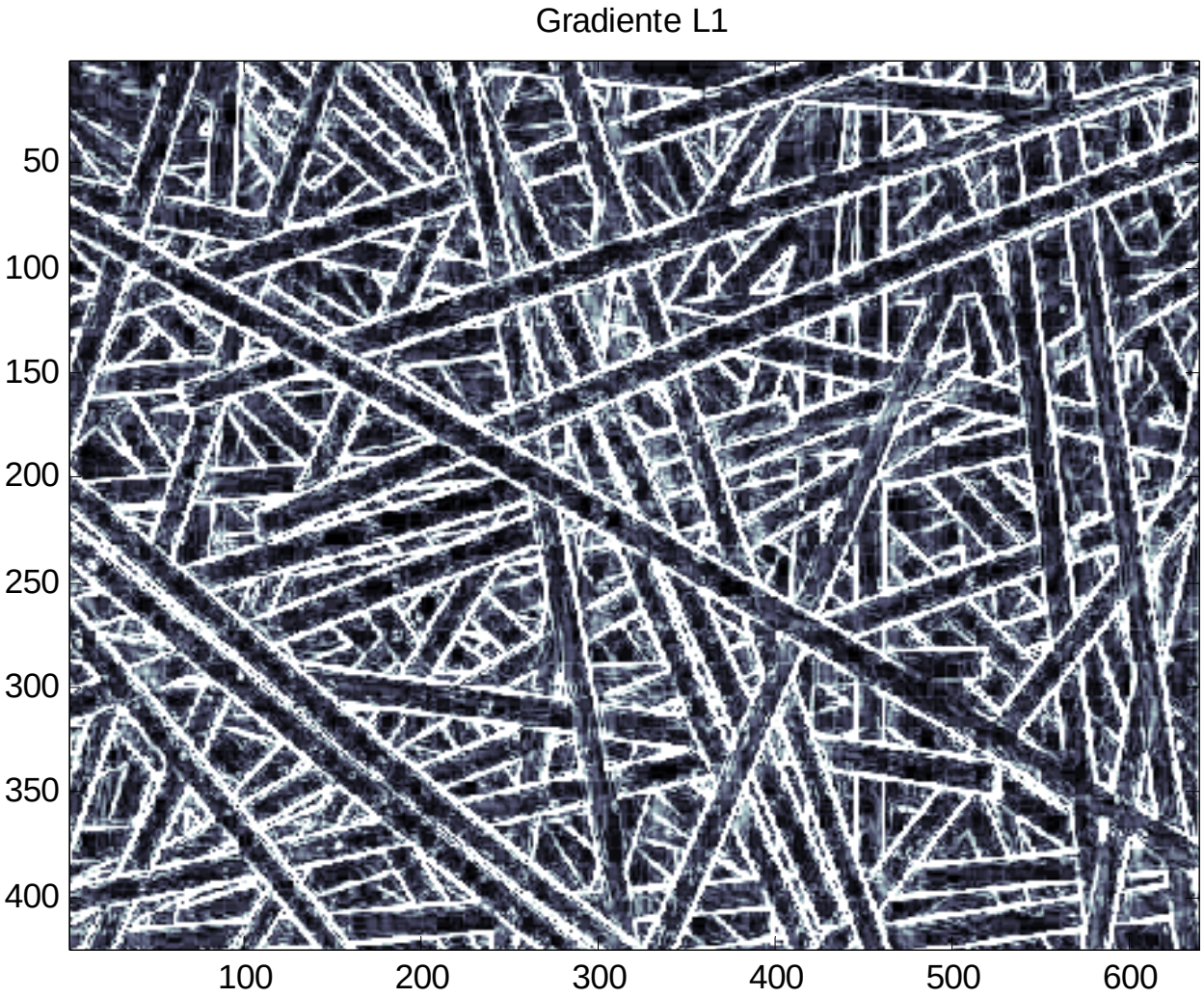
Image original
50
100
150
200
250
300
350
400
100
200
300
400
500
600

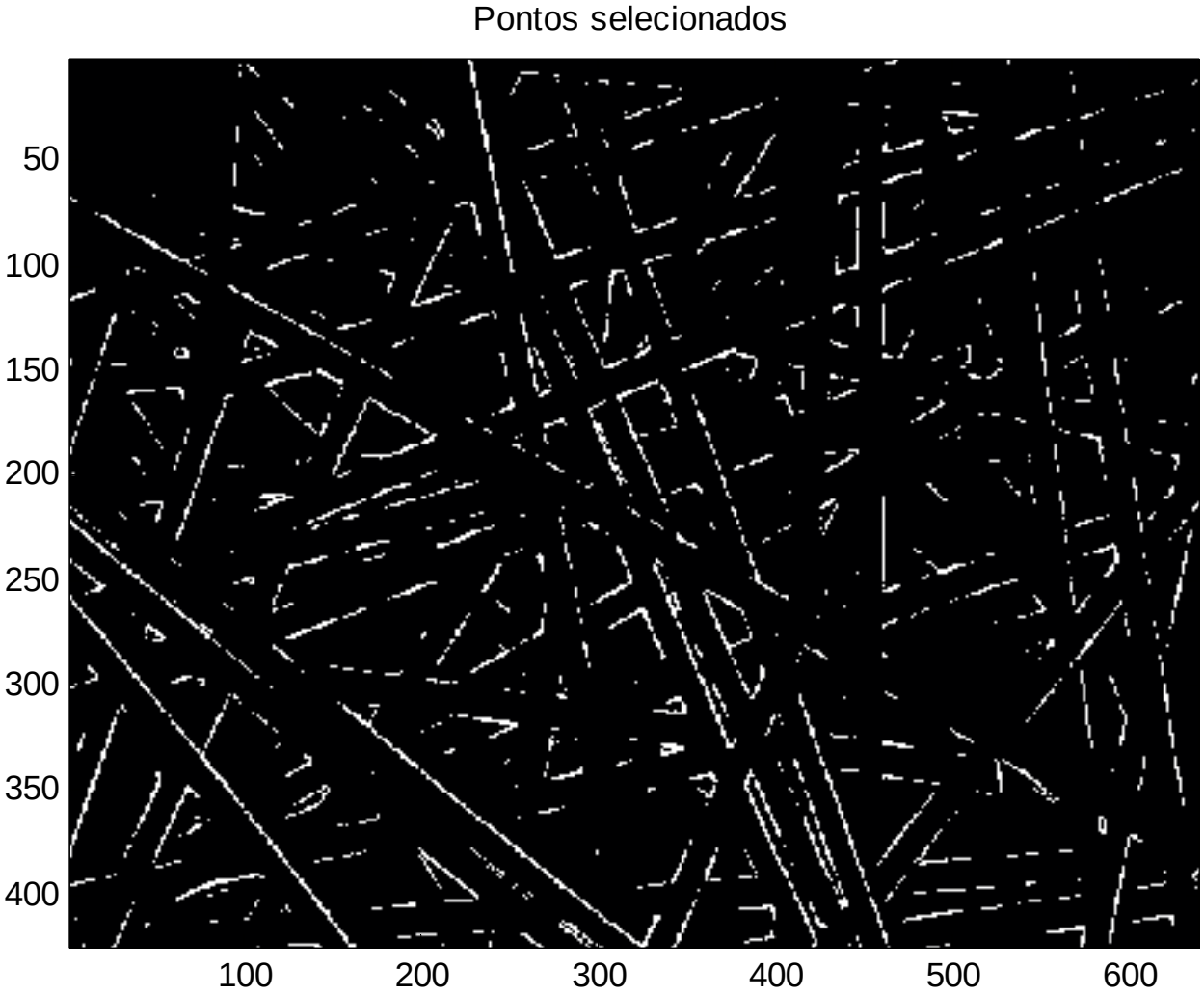

Canal verde extraido
50
100
150
200
250
300
350
400
100
200
300
400
500
600

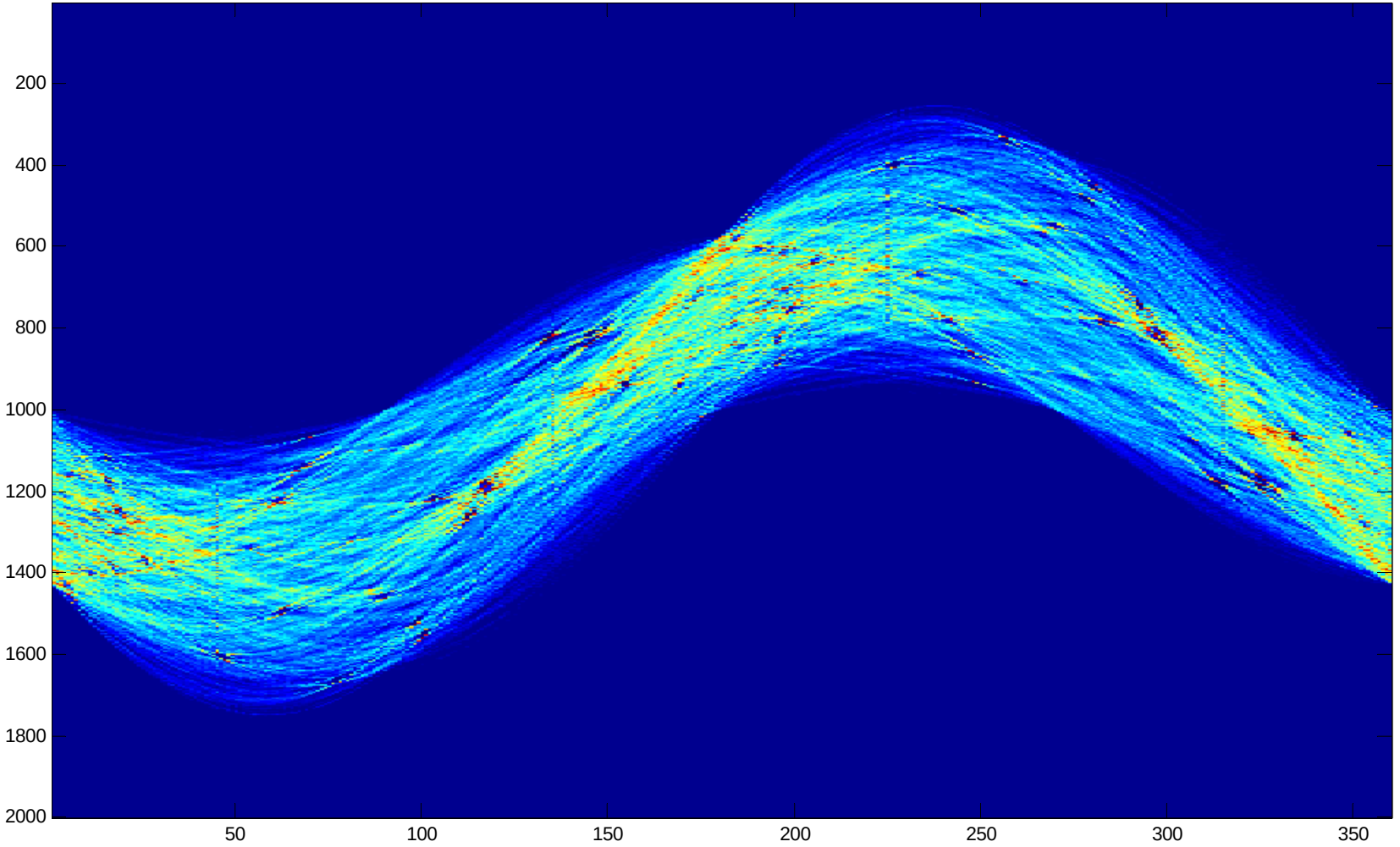

Gradiente em x
50
100
150
200
250
300
350
400
100
200
300
400
500
600

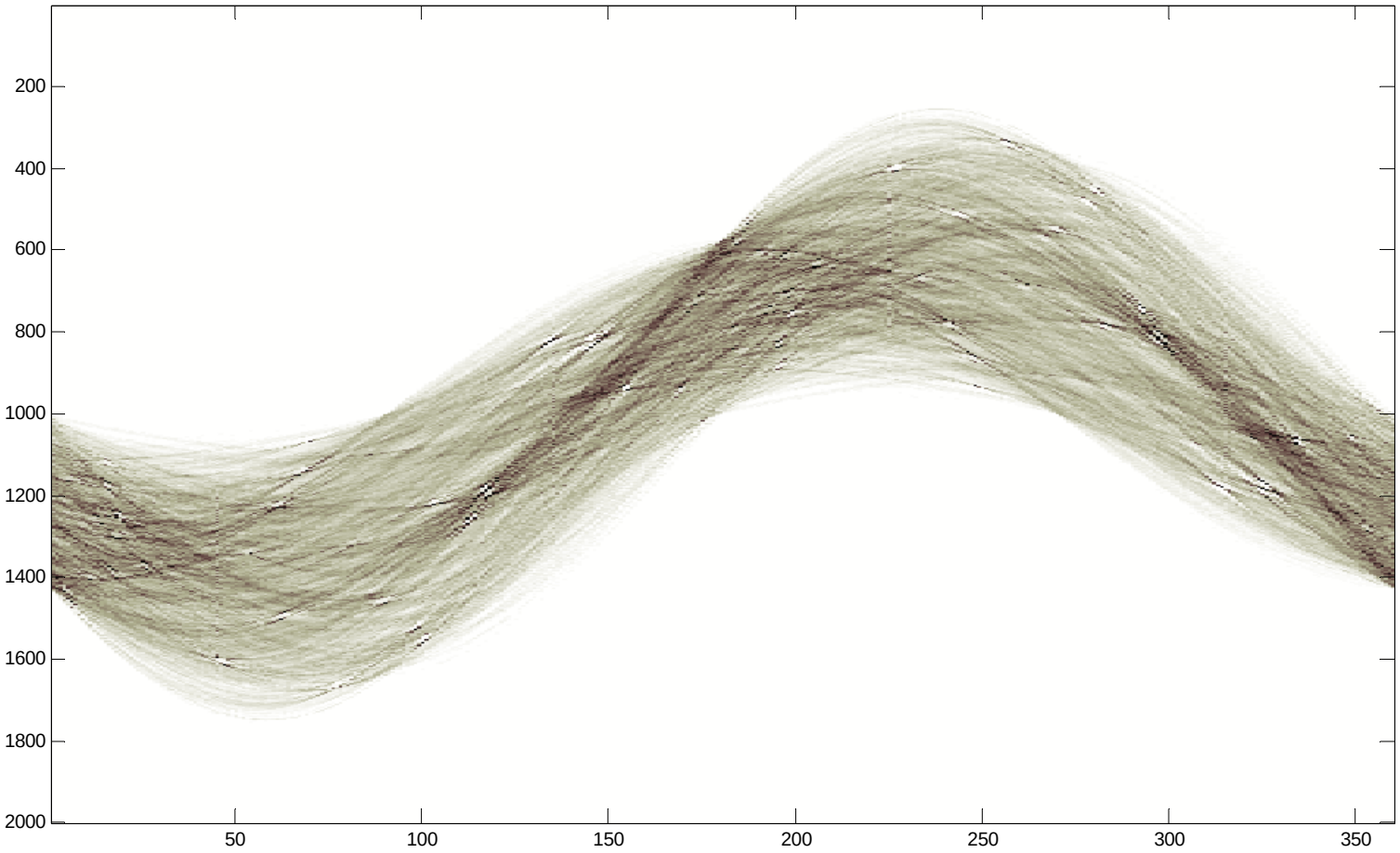
Gradiente em y

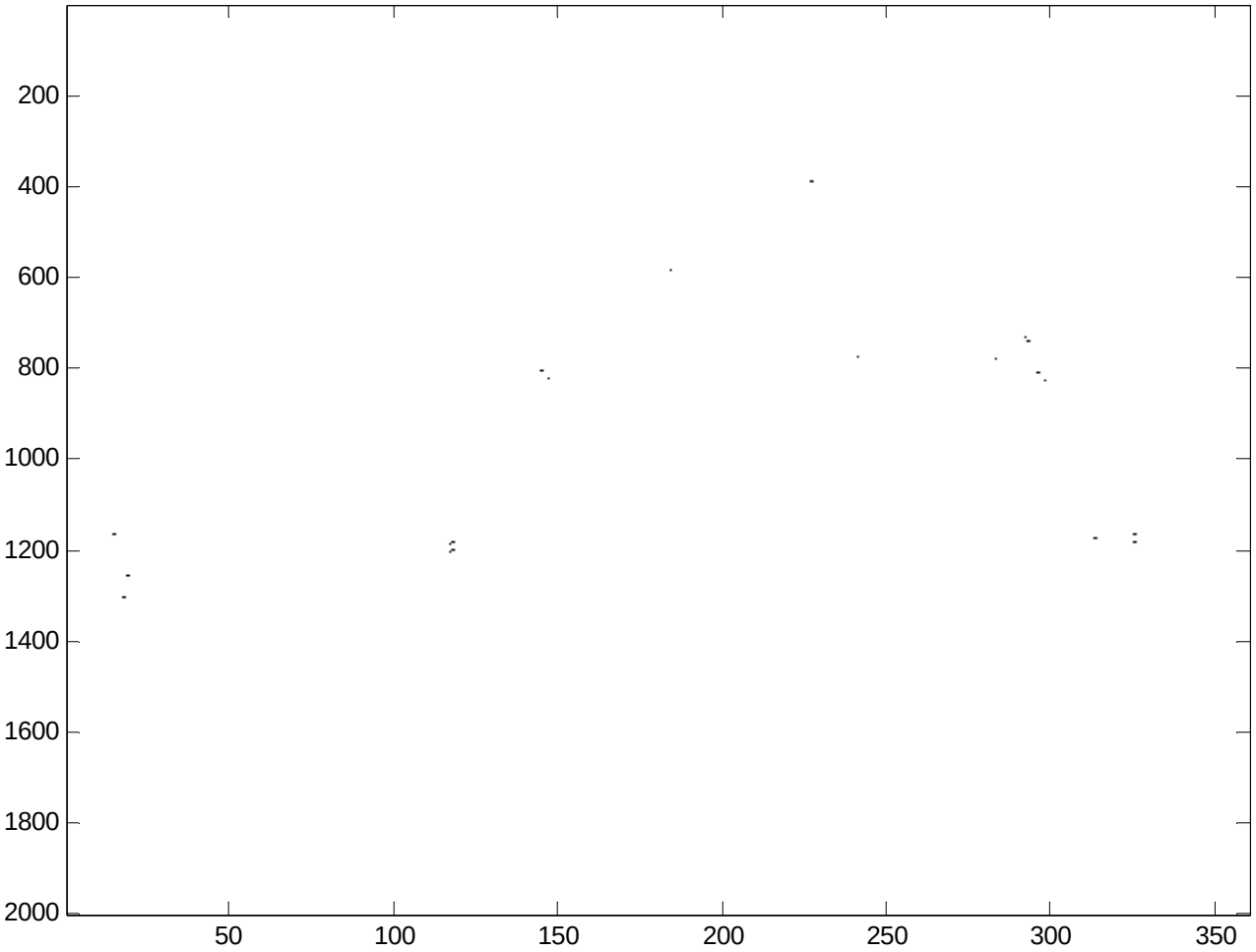


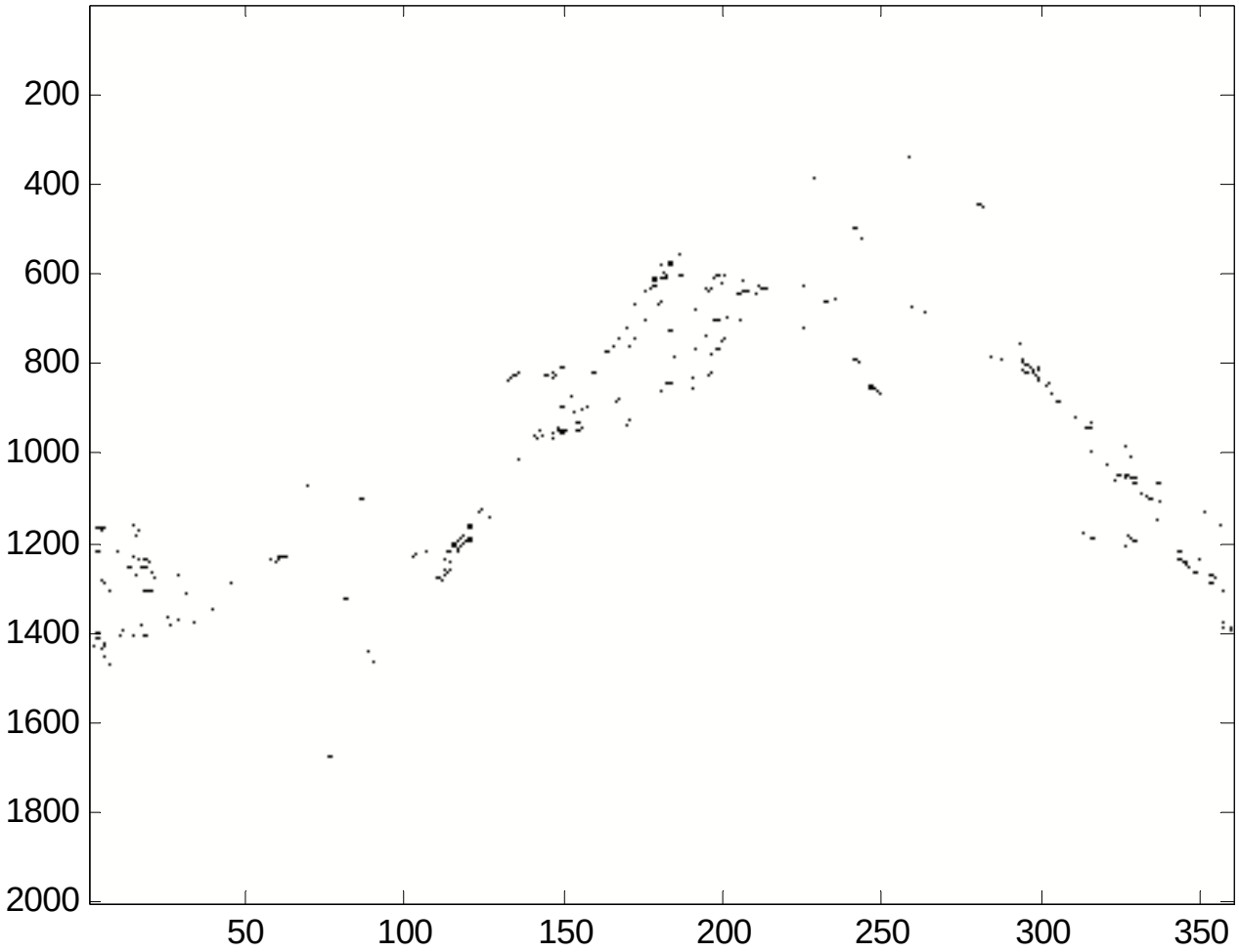

Gradiente L1
50
100
150
200
250
300
350
400
100
200
300
400
500
600


Pontos selecionados
50
100
150
200
250
300
350
400
100
200
300
400
500
600


200
400
600
800
1000
1200
1400
1600
1800
2000
50
100
150
200
250
300
350


200
400
600
800
1000
1200
1400
1600
1800
2000
50
100
150
200
250
300
350


200
400
600
800
1000
1200
1400
1600
1800
2000
50
100
150
200
250
300
350


200
400
600
800
1000
1200
1400
1600
1800
2000
50
100
150
200
250
300
350

```
function h=houghlines(imgname)

% parameter: imgname

img=imread(imgname);
g=img(:,:,2);
figure; image(img); title('Image original');
figure; colormap(bone); image(g); title('Canal verde extraido');
sobel=[-1 0 1;-2 0 2;-1 0 1];
gradx=conv2(g,sobel,'valid');
grady=conv2(g,sobel','valid');
figure; colormap(bone); image(gradx); title('Gradiente em x');
figure; colormap(bone); image(grady); title('Gradiente em y');
border=(abs(gradx)+abs(grady));
figure; colormap(bone); image(border); title('Gradiente L1');
border=(border>200)*255;
```

```
figure; colormap(bone); image(border); title('Pontos selecionados');

tsize=360;
rsize=2000;
h=zeros(rsize,tsize);

[height,width]=size(border);
for i=1:height
    for j=1:width
        if(border(i,j)>0) %para cada pixel de valor 1
            for t=1:tsize
                r=floor(i*cos(t*pi/180)+j*sin(t*pi/180));
                if(r<1000) & (r>-1000)
                    h(r+1000,t)=h(r+1000,t)+1;
                end
            end
```

```
        end
    end
end

figure; colormap(1-bone); image(255*h/max(max(h))); title('Hough');
```

Transformada de Hough para detecção de círculos

Utilizar a parametrização $(x - x_0)^2 + (y - y_0)^2 = r^2$

Matriz tridimensional em função dos parâmetros x_0, y_o, r

Cada r é uma imagem em que se procura os centros de circunferências de raio r que passam pelos pontos x,y. A posição desses centros está a uma distância r de x,y, portanto, formando uma circunferência de raio r.

Utilização da direção do gradiente

Cada ponto da imagem só contribui votos para circunferências centradas em algum ponto na direção do gradiente

Transformada probabilística amostrando pares de pontos

Para cada par de pontos considerar a mediatriz como possíveis posições de centros de circunferências. As direções do gradiente podem ajudar a descartar pares não consistentes.

Hashing Geométrico

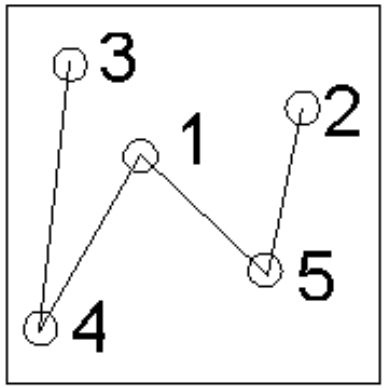
A construção da tabela *hash* ocorre numa fase *off-line* e todas possíveis combinações de feições contribuem com itens a adicionar na tabela. Escolhendo-se três pontos e_0, e_1, e_2 para formar uma base, todos pontos do plano podem ser representados da forma $e_0 + \kappa(e_1 - e_0) + \lambda(e_2 - e_0)$, o plano (κ, λ) é quantizado numa tabela bidimensional com par de índices inteiros (k, l).

Sejam N conjuntos de pontos alvos B_i. Para cada conjuntos de pontos alvos, faz-se o seguinte. Para cada três pontos não colineares e_0, e_1, e_2, do conjunto de pontos, expresse os demais pontos da forma $e_0 + \kappa(e_1 - e_0) + \lambda(e_2 - e_0)$ e adicione a tupla (i, e_0, e_1, e_2) a entrada (k, l) da tabela. Para $O(m)$ pontos em cada conjunto B_i, a construção da tabela *hash* tem complexidade $O(Nm^4)$.

Referência à tabela

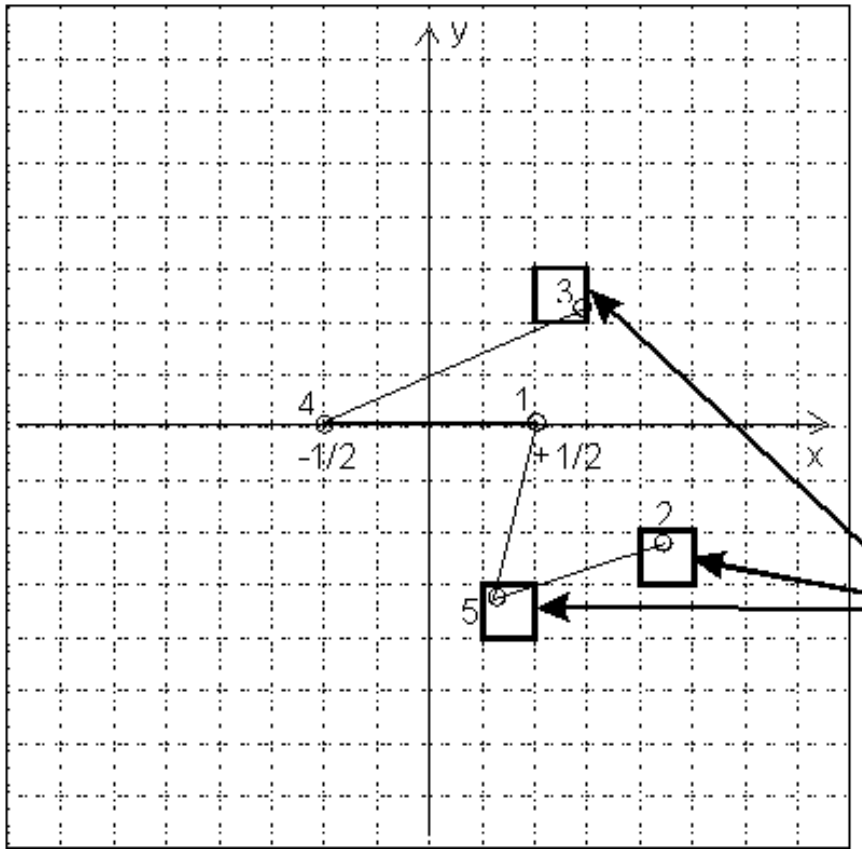
A tabela é indexada *on-line* de forma muito eficiente. Dado um conjunto de pontos A, que se deseja identificar e determinar a transformação afim, escolhem-se três pontos não colineares e'_0, e'_1, e'_2 do conjunto de pontos e se expressam os demais pontos como $e'_0 + \kappa(e'_1 - e'_0) + \lambda(e'_2 - e'_0)$ e se acumula um voto para cada tupla (i, e_0, e_1, e_2) na entrada (k,l) correspondente da tabela. A tupla (i, e_0, e_1, e_2) que receber mais votos indica o conjunto de pontos alvo T_i contendo o conjunto de pontos procurado. A transformação afim que mapeia e'_0, e'_1, e'_2 na base vencedora e_0, e_1, e_2 é assumidamente a transformação entre as duas formas. A complexidade de realizar o *matching* para um conjunto de n pontos é $O(n)$. A fim de obter a pose, deve-se combinar a transformação de normalização com a transformação tabelada.

Modelo



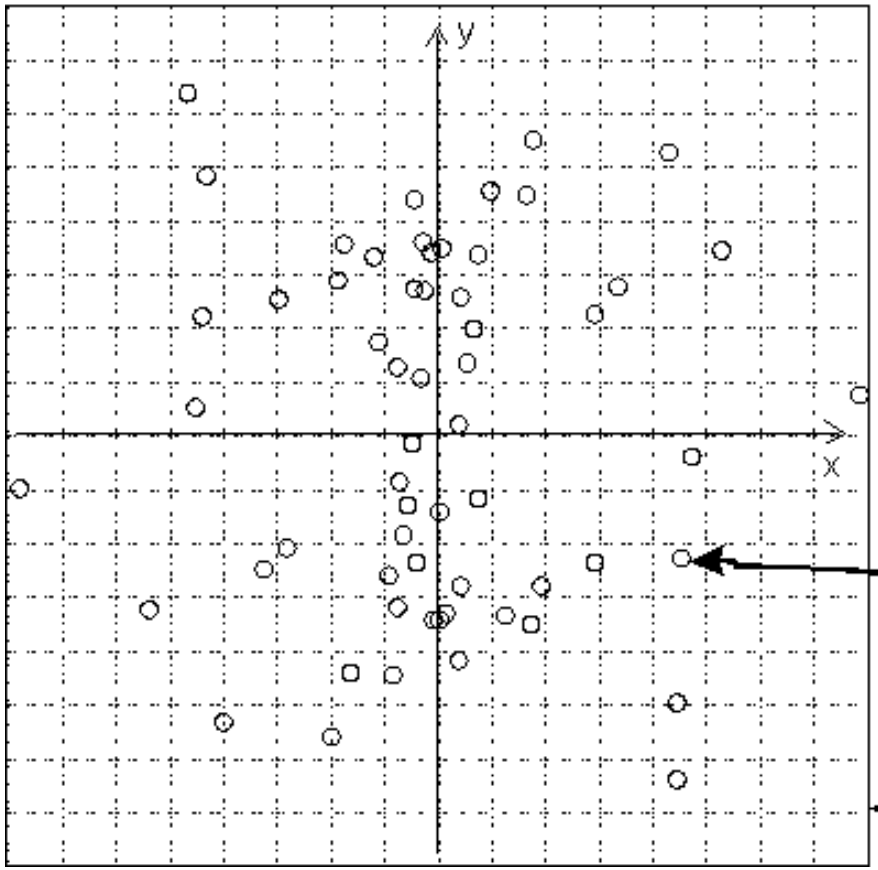
Os pontos 1 e 4 são utilizados para formar uma base. Aplicando rotação, translação e escala, esses pontos são mapeados em (+1/2,0) e (-1/2,0) respectivamente. Os demais pontos sofrem a mesma transformação.

Base (1,4)



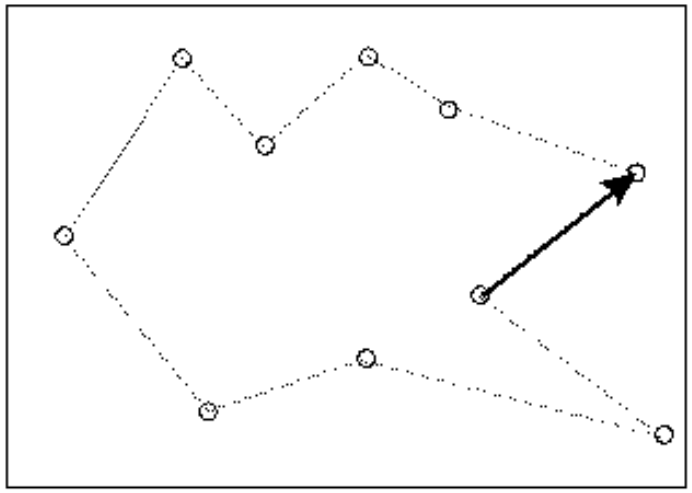
Acrescente (1,4) no final da lista dessas células da tabela hash.

Células da tabela hash para esse modelo



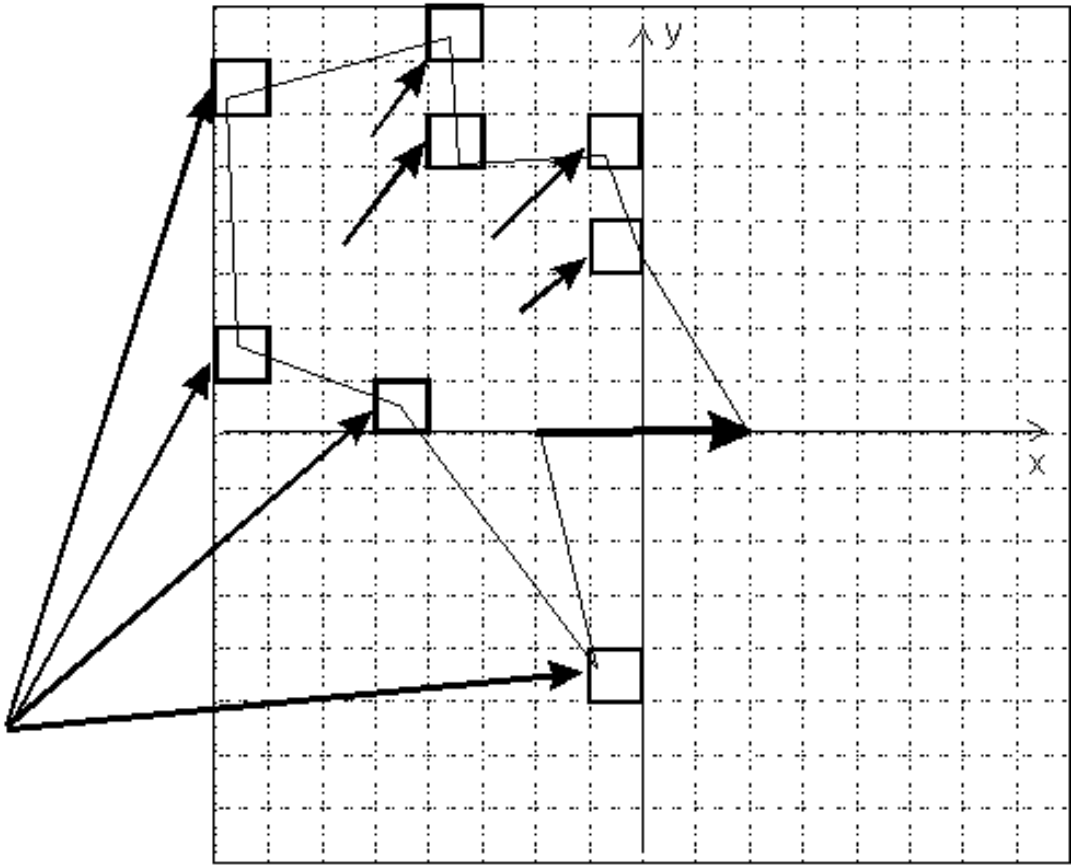
Utilizando esse procedimento para transformar os pontos do modelo para todas as possíveis bases (1,2), (1,3), ..., (5,1) e acrescentando a identificação da base na lista de cada célula da tabela hash indexada pelas coordenadas dos pontos transformados (de acordo com essa base), é construída a tabela hash para identificação desse modelo invariante a transformações de rotação translação e escala uniforme.

Identificação do modelo dada uma imagem



Imagem

Some um voto para cada item nas células da tabela hash indexadas pelas feições da imagem, transformadas segundo uma base qualquer formada por duas de suas feições.
No final, crie um histograma de votos considerando os itens que tem pelo menos um voto.



Clustering de formas

O algoritmo K-Means

Faça

    Para cada centro de cluster J

        Para cada amostra I

            Associar I ao cluster J mais próximo

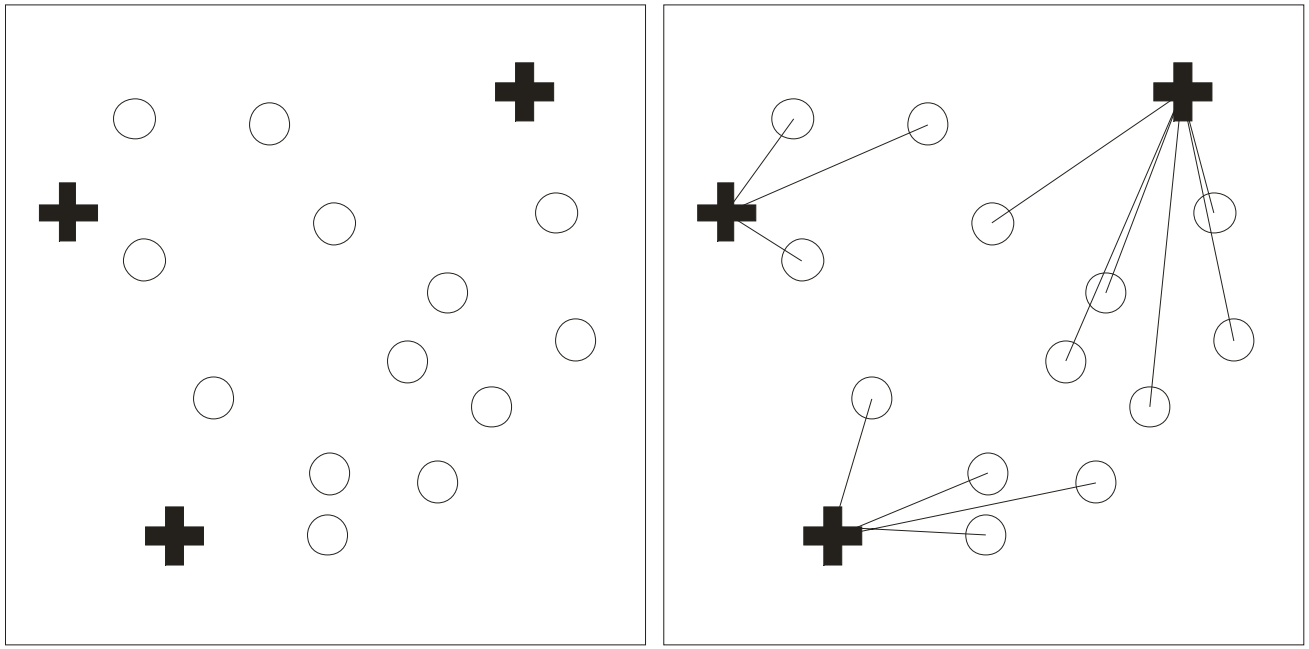
    Se houver alteração

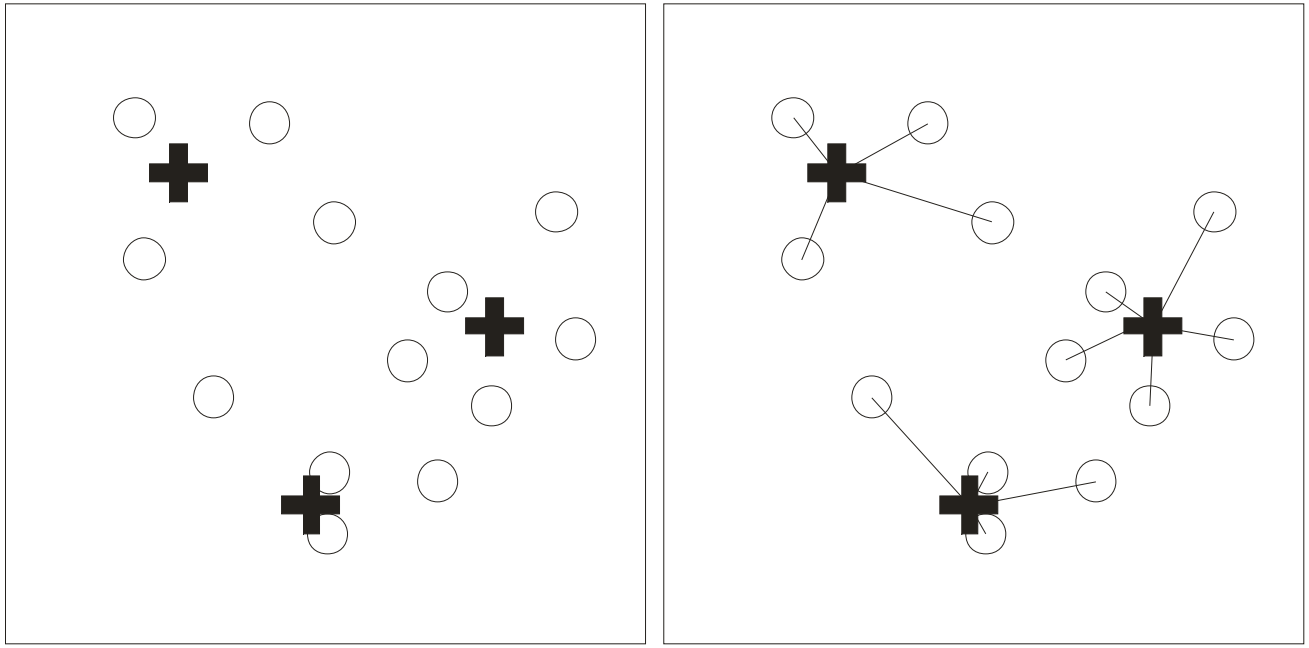
        Para cada cluster J

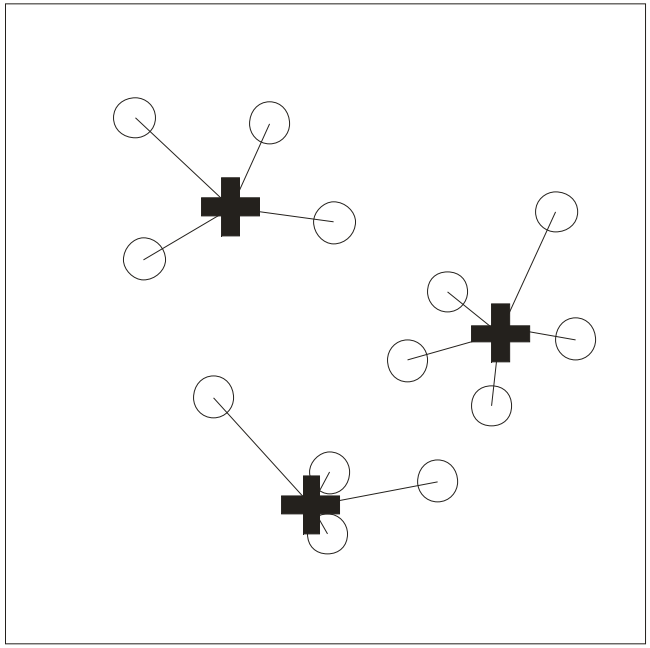
            Arrume o centro do cluster para ser a média das amostras I relacionadas

Até que não ocorram mais alterações dos clusters

Exemplo







Fuzzy C-Means

Diferente do K-means no sentido de que cada amostra pertence em algum grau a cada cluster.

II. Prototype-Based Fuzzy Clustering

Let $\mathbf{X} = \{\mathbf{x}_j \mid j = 1 \cdots N\}$ be a set of feature vectors in an n-dimensional feature space with coordinate-axis labels $[x_1, x_2, \cdots, x_n]$, where $\mathbf{x}_j = [x_{j1}, x_{j2}, \cdots, x_{jn}]^T$. Let $B = (\beta_1, \cdots, \beta_C)$ represent a C-tuple of prototypes each of which characterizes one of the C clusters. Each β_i consists of a set of parameters. In the following, we use β_i to denote both cluster i and its prototype. Let u_{ij} represent the grade of membership of feature point $\mathbf{x}_j$ in cluster β_i. The $C \times N$ matrix $\mathbf{U} = [u_{ij}]$ is called a constrained fuzzy C-partition matrix if it satisfies the following conditions [4], [25]

$$u_{ij} \in [0, 1] \text{ for all } i, \quad 0 < \sum_{j=1}^{N} u_{ij} < N \text{ for all } i, j \quad \text{and}$$

$$\sum_{i=1}^{C} u_{ij} = 1 \text{ for all } j. \tag{1}$$

The problem of fuzzily partitioning the feature vectors into C clusters can be formulated as the minimization of an objective function $J(B, \mathbf{U}; \mathbf{X})$ of the form

$$J(B, \mathbf{U}; \mathbf{X}) = \sum_{i=1}^{C} \sum_{j=1}^{N} (u_{ij})^m d^2(\mathbf{x}_j, \beta_i). \tag{2}$$

In the above equation, $m \in [1, \infty)$ is a weighting exponent called the fuzzifier, and $d^2(\mathbf{x}_j, \beta_i)$ represents the distance from a feature point $\mathbf{x}_j$ to the prototype β_i. Minimization of the objective function with respect to $\mathbf{U}$ subject to the constraints in (1) gives us [4]

$$\left.\begin{array}{ll} u_{ij} = \dfrac{1}{\sum_{k=1}^{C} \left(\frac{d^2(\mathbf{x}_j, \beta_i)}{d^2(\mathbf{x}_j, \beta_k)} \right)^{\frac{1}{m-1}}} & \text{if } I_j = \emptyset \\ \left.\begin{array}{ll} u_{ij} = 0 & i \notin I_j \\ \sum_{i \in I_j} u_{ij} = 1 & i \in I_j \end{array}\right\} & \text{if } I_j \neq \emptyset \end{array}\right\} \tag{3}$$

where $I_j = \{i \mid 1 \leq i \leq C, d^2(\mathbf{x}_j, \beta_i) = 0\}$. Minimization of

PROTOTYPE-BASED FUZZY CLUSTERING

Fix the number of clusters C*; fix* $m, m \in [1, \infty)$*;*
Initialize the fuzzy C*-partition* $\mathbf{U}$*;*
REPEAT
Update the parameters of each cluster prototype;
Update the partition matrix $\mathbf{U}$ *by using* (3)*;*
UNTIL$(\|\Delta \mathbf{U}\| < \varepsilon)$;

The hard (crisp) versions of these algorithms are easily obtained by changing the updating rule for the memberships so that they are always binary. In other words, one uses

$$u_{ij} = \begin{cases} 1 & \text{if } d^2(\mathbf{x}_j, \beta_i) < d^2(\mathbf{x}_j, \beta_k) \quad \text{for all } k \\ 0 & \text{otherwise.} \end{cases}$$

Ties are broken arbitrarily. In practice, the hard versions do not perform as well as their fuzzy counterparts. Due to space limitation, we do not deal with the hard algorithms in this paper.

Distância de um ponto a um protótipo

III. The Fuzzy C Quadric Shells (FCQS) Algorithm

This algorithm [31], [32] assumes that each cluster resembles a hyperquadric surface, and the prototypes β_i consist of parameter vectors $\mathbf{p}_i$ which define the equations of the hyperquadric surfaces. The general equation for such a hyperquadric surface is

$$\mathbf{p}_i^T \mathbf{q} = 0 \tag{7}$$

where

$$\begin{aligned} \mathbf{p}_i^T &= \left[p_{i1}, p_{i2}, \cdots, p_{in}, p_{i(n+1)}, \cdots, \right. \\ &\qquad \left. p_{ir}, p_{i(r+1)}, p_{i(r+2)}, \cdots, p_{i(r+n)}, p_{is}\right]. \\ \mathbf{q}^T &= \left[x_1^2, x_2^2, \cdots, x_n^2, x_1 x_2, \cdots, \right. \\ &\qquad \left. x_{n-1} x_n, x_1, x_2, \cdots, x_n, 1\right] \end{aligned} \tag{8}$$

and $s = \frac{n(n+1)}{2} + n + 1 = r + n + 1$. We may define the algebraic (or residual) distance from a point $\mathbf{x}_j$ to a prototype β_i as

$$d^2(\mathbf{x}_j, \beta_i) = d_{Qij}^2 = \mathbf{p}_i^T \mathbf{q}_j \mathbf{q}_j^T \mathbf{p}_i = \mathbf{p}_i^T \mathbf{M}_j \mathbf{p}_i \tag{9}$$

Aproximação de um protótipo a um conjunto de pontos

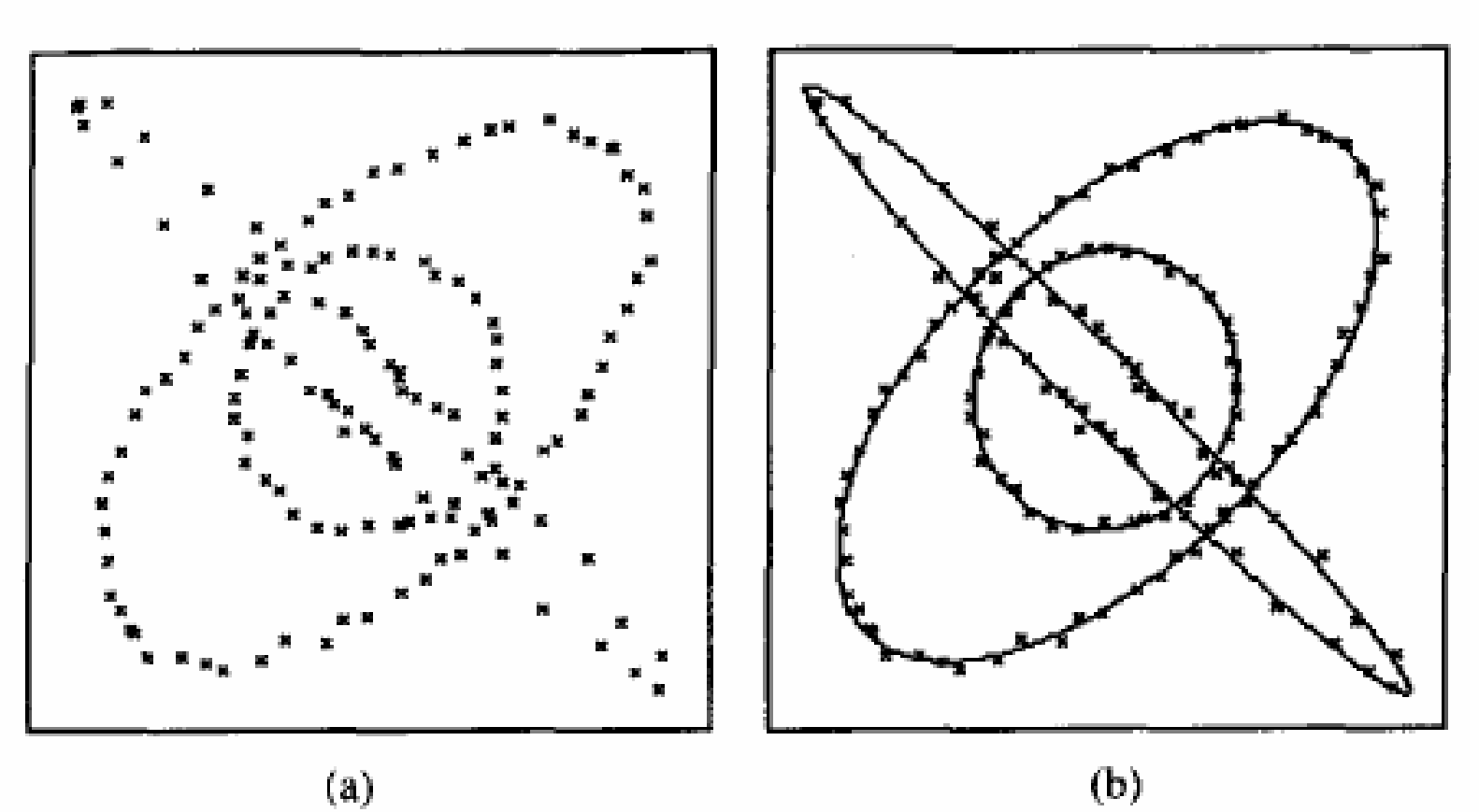


Fig. 1. (a) A data set consisting of two ellipses and a circle. (b) The prototypes found by the FCQS algorithm superimposed on the dataset.