Segmentação de Imagens

Carlos Alexandre Mello

Pós-Graduação em Ciência da Computação

Segmentação

- **Segmentação** é o processo de dividir ou separar uma imagem em **objetos** ou **elementos** que são **coerentes** sob algum **critério**.
  - Esse **processo** de divisão **deve ocorrer** até que a informação desejada seja **corretamente separada**.
- **Classicamente**, se **divide** as técnicas de segmentação em:
  - **Seg. por Descontinuidade**
    - Ex.: detecção de bordas, de contornos, etc.
  - **Seg. por Similaridade**
    - Ex.: segmentação de cor, crescimento de regiões, etc.

Segmentação

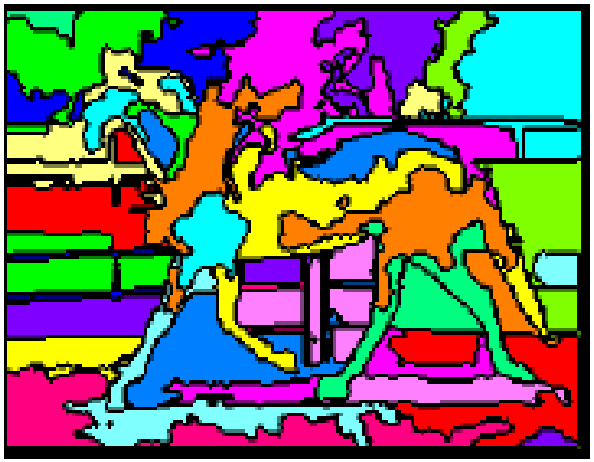
- A grande dificuldade é ter automaticamente definido o que é objeto de interesse na cena
  - Às vezes, definido em cima da aplicação
- Por esse motivo, temos diferentes abordagens para tratar a questão de segmentação
- Mas nosso sistema visual segmenta mesmo?
  - http://www.ted.com/talks/lang/pt/pawan_sinha_on_how_brains_learn_to_see.html

Segmentação

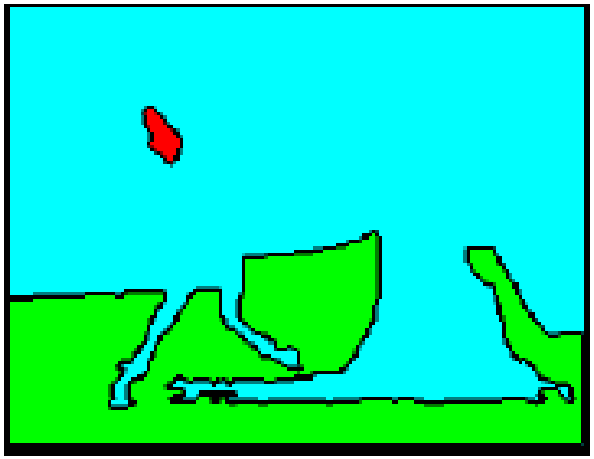
- Tipos de segmentação
  - Detecção de bordas
  - Detecção de contornos
  - Segmentação de objetos
    - Por similaridade
    - Por dissimilaridade
      - Ambas, em algum sentido

Segmentação

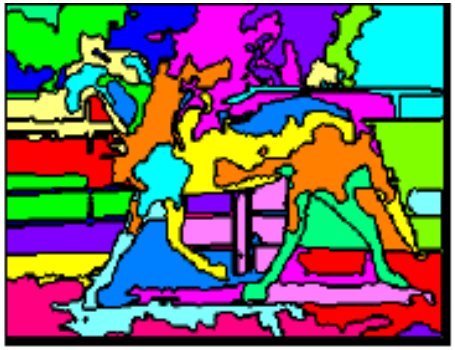
- Quanto ao resultado final

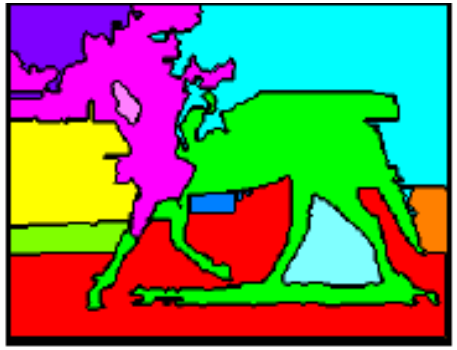


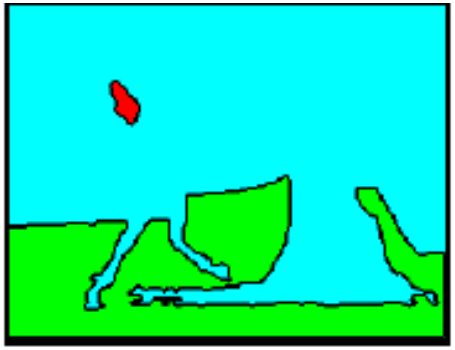
Sobre-segmentação



Sub-segmentação



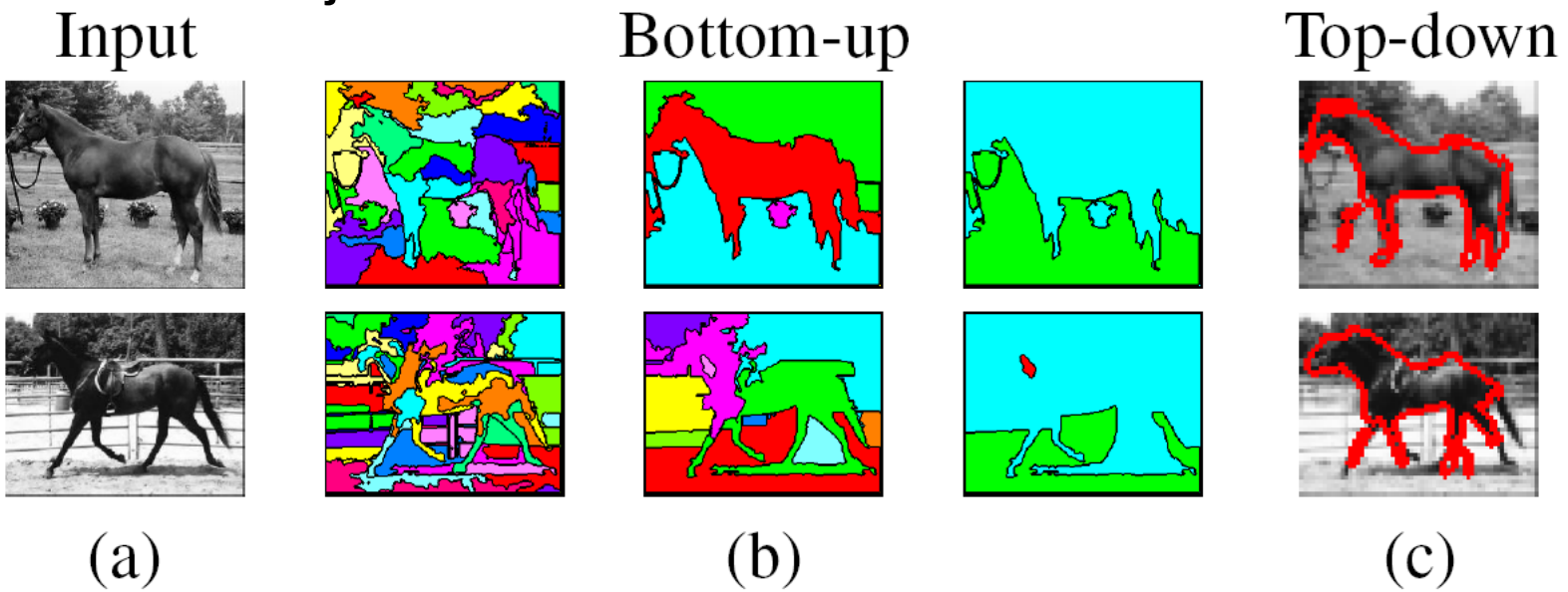




Múltiplas segmentações

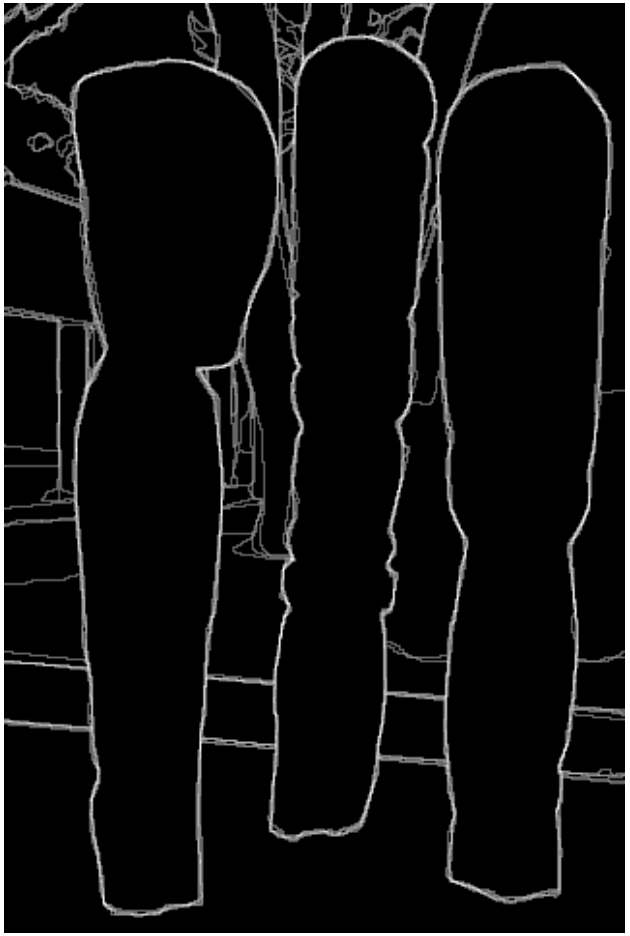
Segmentação

- Processos principais
  - Bottom-up: agrupa tokens com características semelhantes
  - Top-down: agrupa tokens que parecem pertencer ao mesmo objeto



[Levin and Weiss 2006]

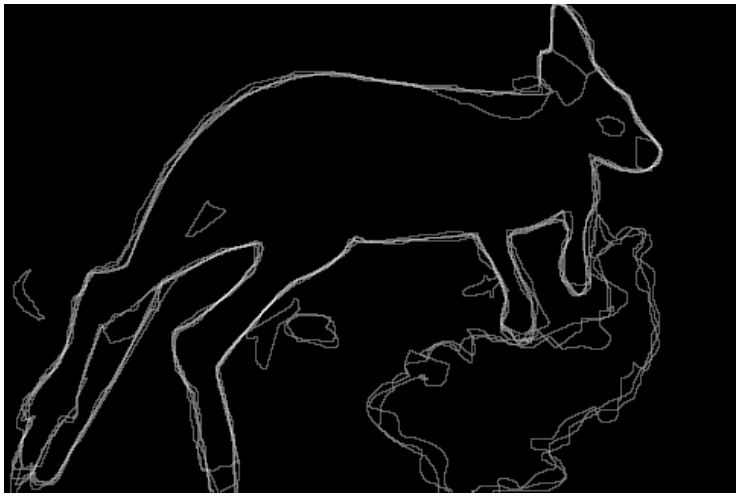
Segmentação – Padrão ouro



Fonte: The Berkeley Segmentation Dataset and Benchmark
https://www.eecs.berkeley.edu/Research/Projects/CS/vision/bsds/

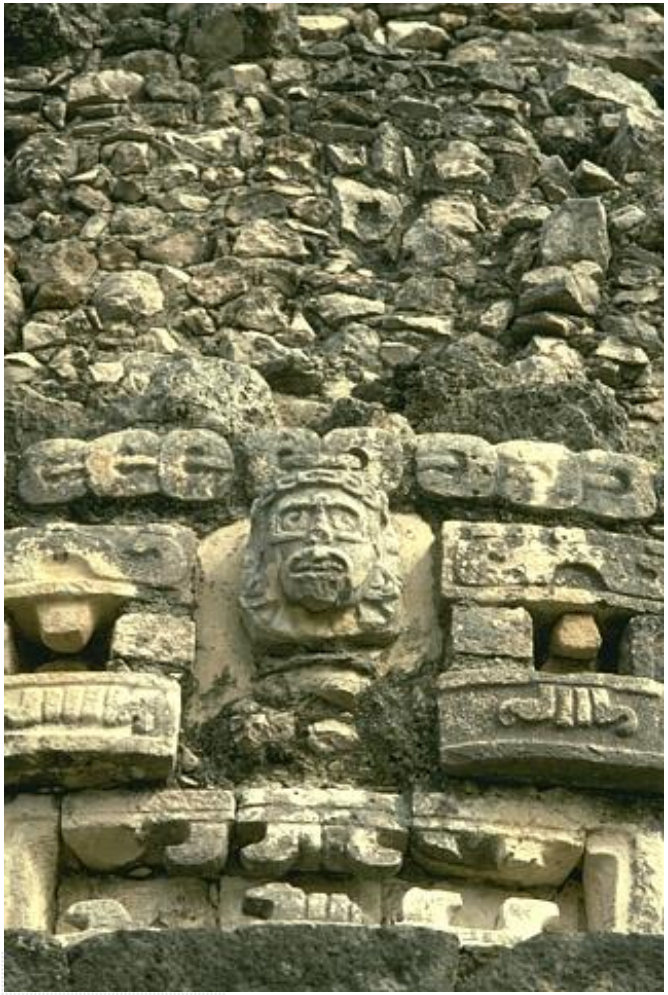
Segmentação – Padrão ouro

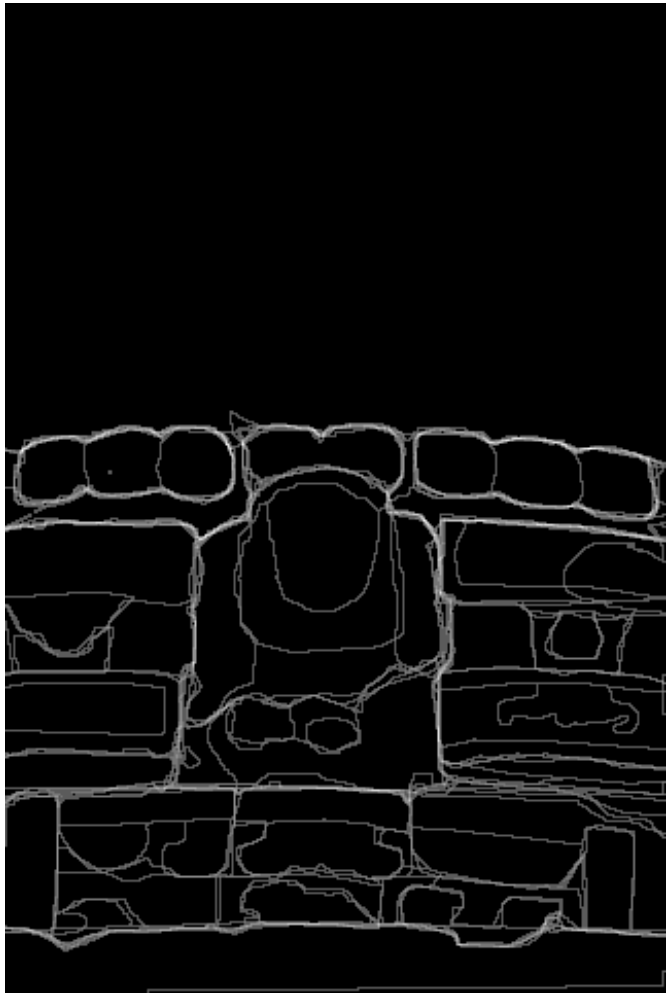
Segmentação – Padrão ouro



Segmentação – Padrão ouro

Segmentação – Padrão ouro



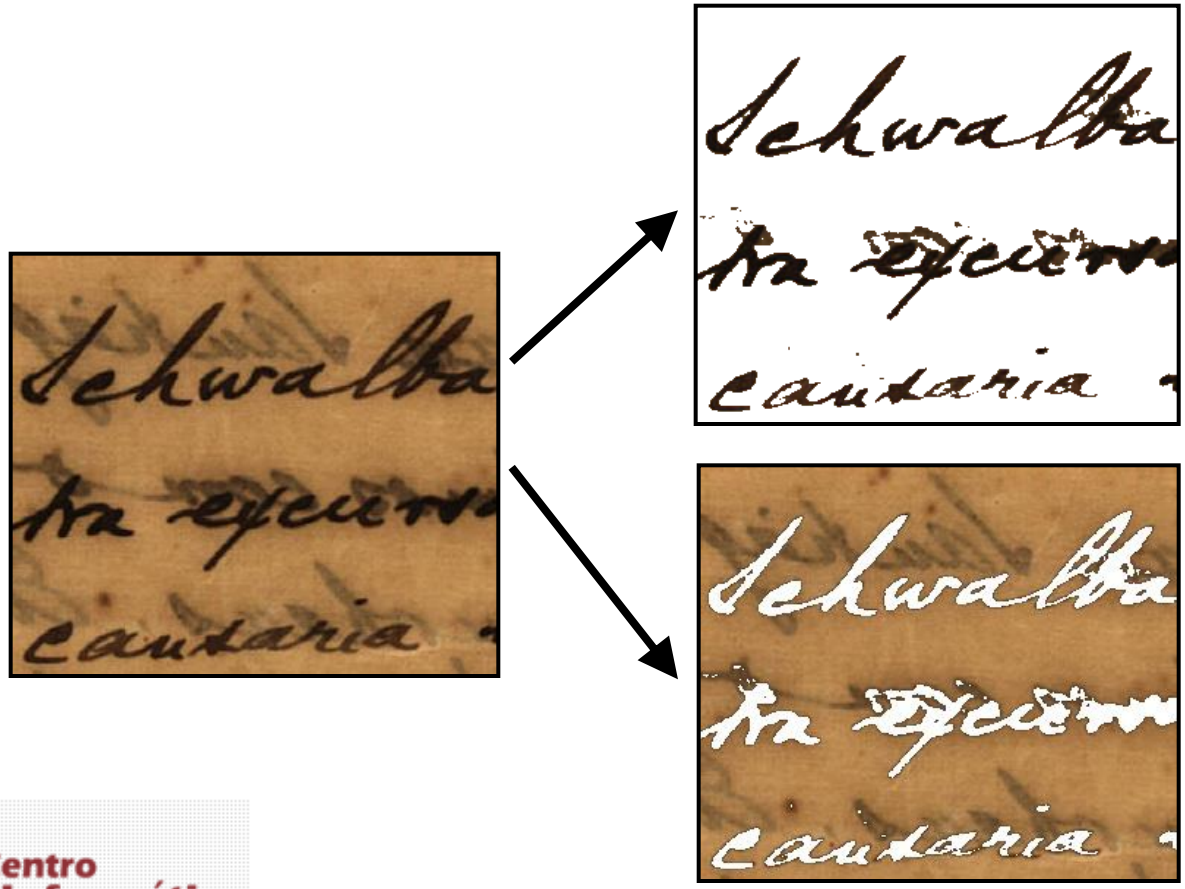


Segmentação por Limiarização

Carlos Alexandre Mello

Limiarização

- Conversão de uma imagem para dois tons a partir de um dado ponto de corte (limiar)



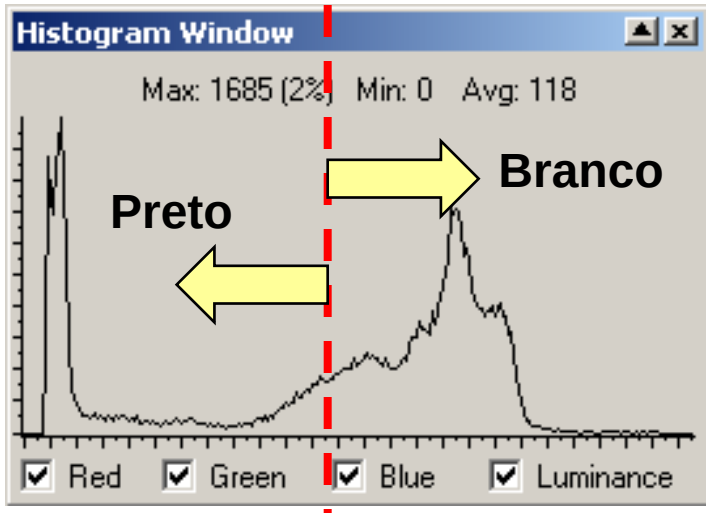
“Objeto”

Background

Limiarização

- Algoritmo de Recorte

Se cor(i) <= 127
Então cor(i) = Preto
Senão cor(i) = Branco

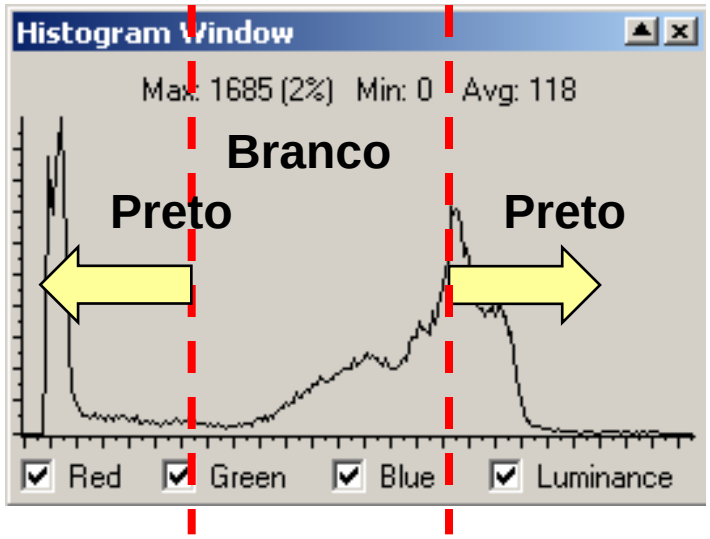


Valor de Corte = 127
(threshold, limiar)

Limiarização

- Algoritmo de Recorte (faixa de corte)

Se 50 < cor(i) < 100
Então cor(i) = Branco
Senão cor(i) = Preto



Branco entre 50 e 100

Limiarização

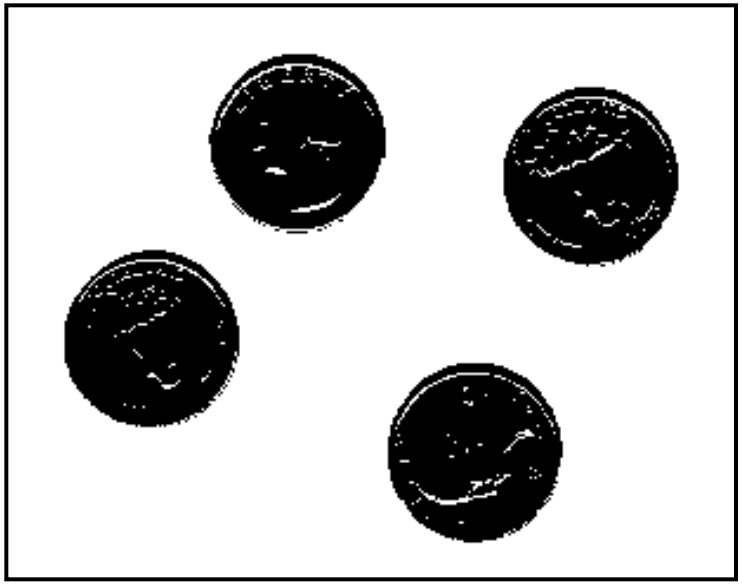
- Tipos de Limiarização
  - Global
    - Um único valor de corte é definido para toda a imagem
      - Vantagem: Velocidade
      - Desvantagem: Qualidade
  - Local
    - Diferentes pontos de corte são definidos para diferentes regiões da imagem
      - Vantagem: Qualidade
      - Desvantagem: Velocidade

Limiarização

- Tipos de Limiarização
  - Global
    - Mean Grey Level, Dois Picos, Otsu, Renyi, Pun, Kapur, Johannsen, etc
  - Local
    - Niblack, White, Bernsen, etc

Limiarização

- Limiarização Ótima



Objetos formados por tons bem diferentes do background:
Fácil redução de cores com qualidade

Limiarização

- Limiarização Ótima

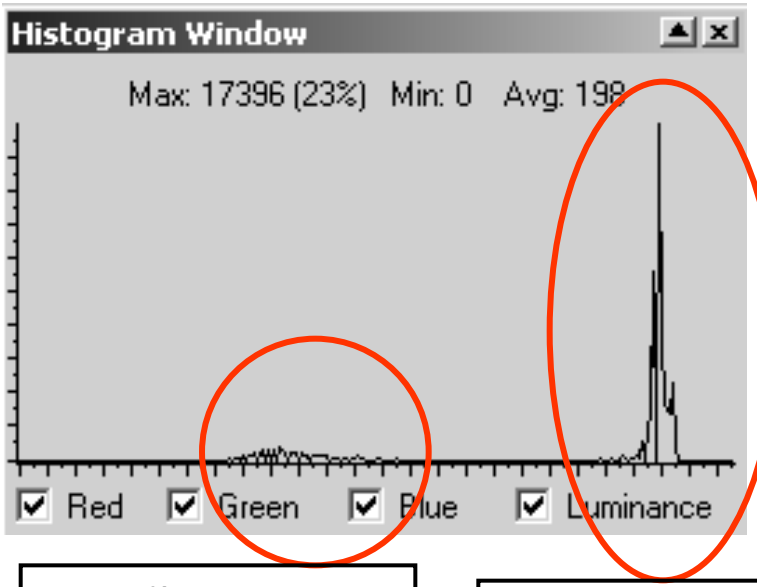


Imagem original com cores bem distintas no Histograma

Região com os tons da moeda

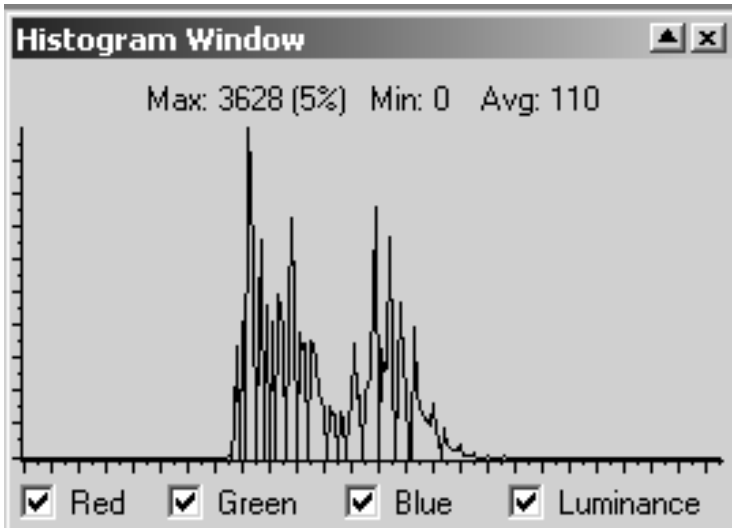
Região com os tons do background

Regiões bem separadas

Limiarização

- Problemas na Limiarização

Problema: Imagem com baixo contraste

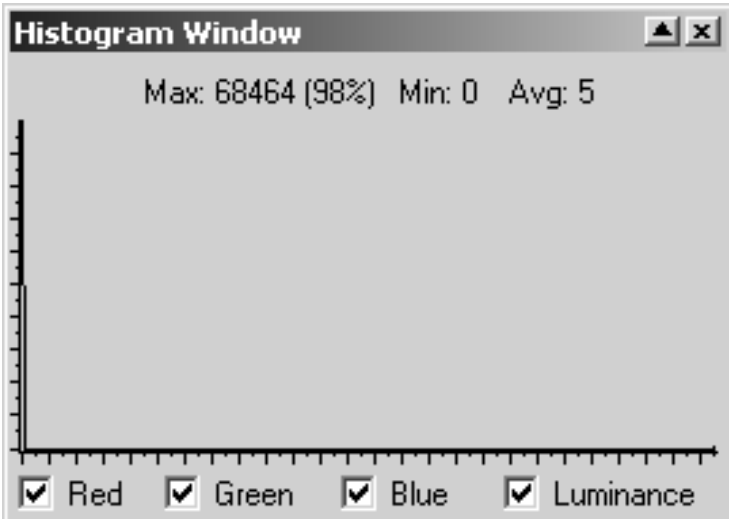


É possível, por exemplo, separar automaticamente a menina do background?

Limiarização

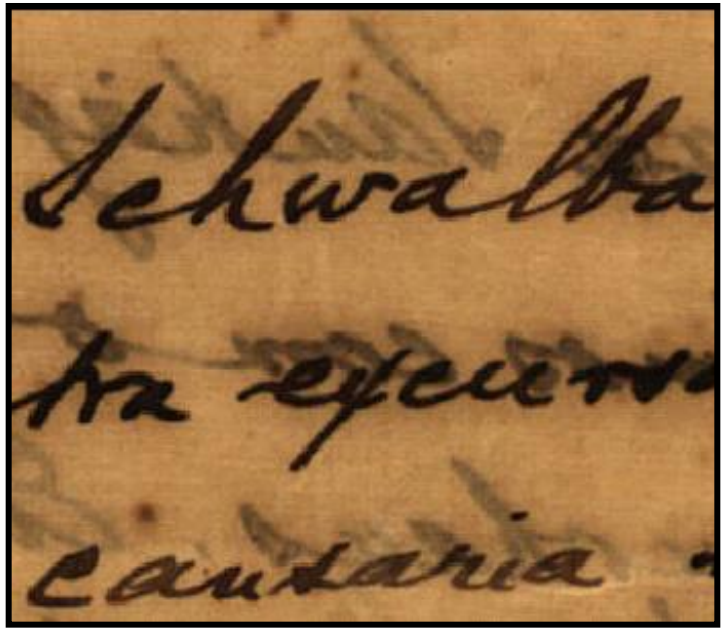
- Problemas na Limiarização

Resultado da Limiarização



Limiarização

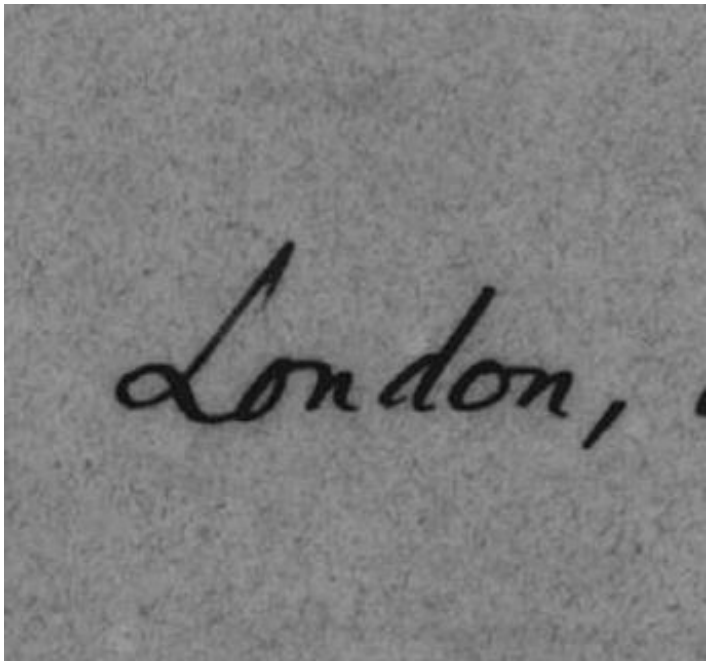
- Problemas na Limiarização

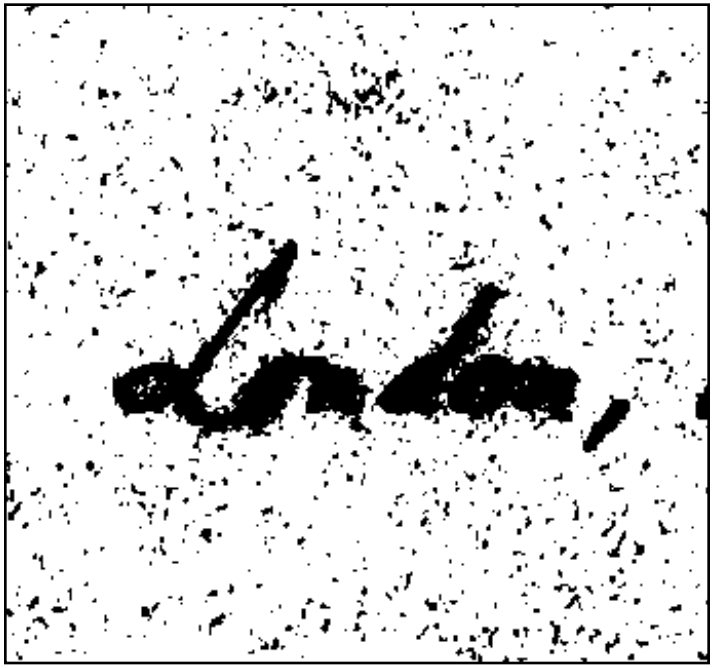


Algoritmo de threshold com valor de corte padrão

Limiarização Global

- Porcentagem de Preto

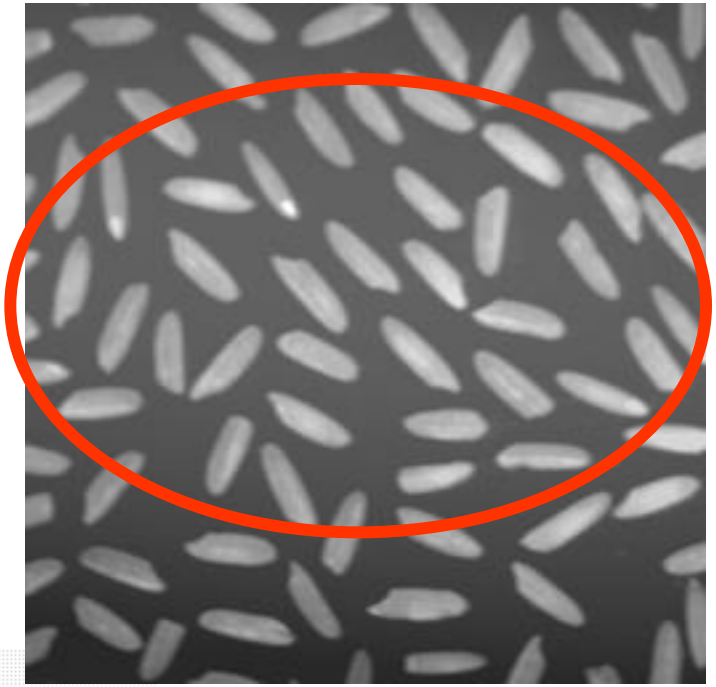




th = 119 (10% de preto)

Limiarização Local

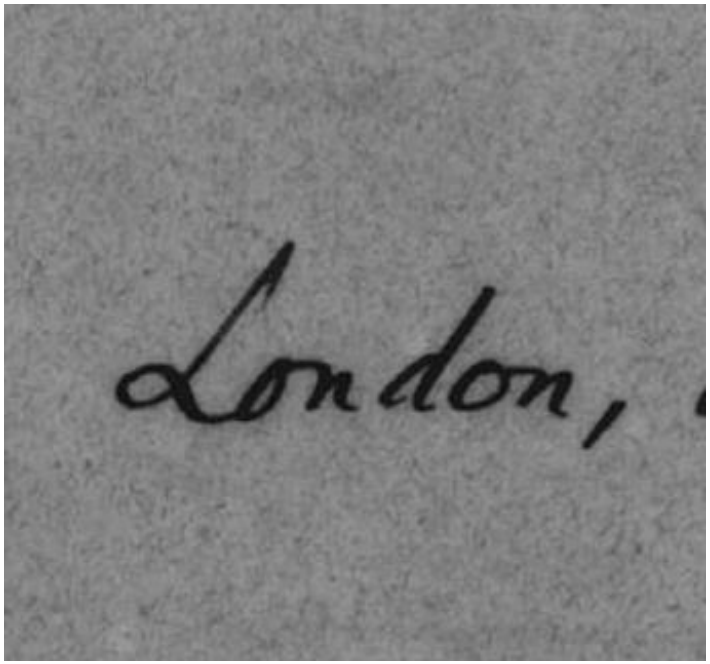
- O valor de corte varia dentro de uma mesma imagem, dependendo de fatores da imagem
  - Exemplo: Imagem com diferentes focos de iluminação

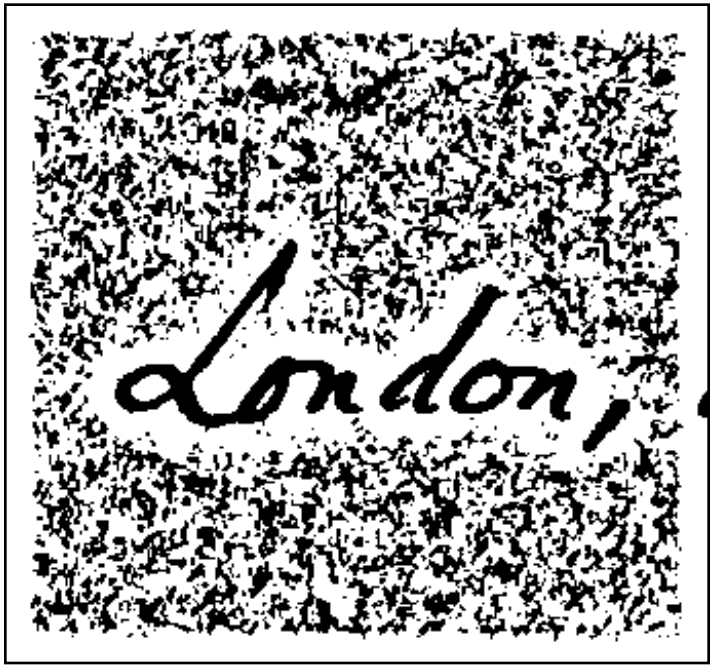


Área iluminada

Limiarização Local

- Niblack: Exemplo





Janela = 31; Bias = -0.8

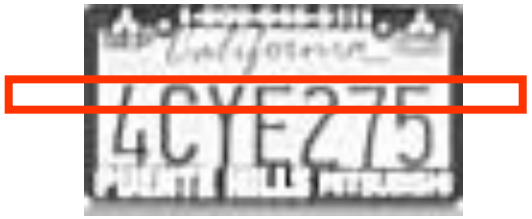
th = média + bias*desvio

Segmentação de Bordas

Carlos Alexandre Mello

Segmentação de Imagens

- Principais algoritmos
  - Baseados em Descontinuidade
    - A imagem é particionada, tomando como base mudanças bruscas em níveis de cinza

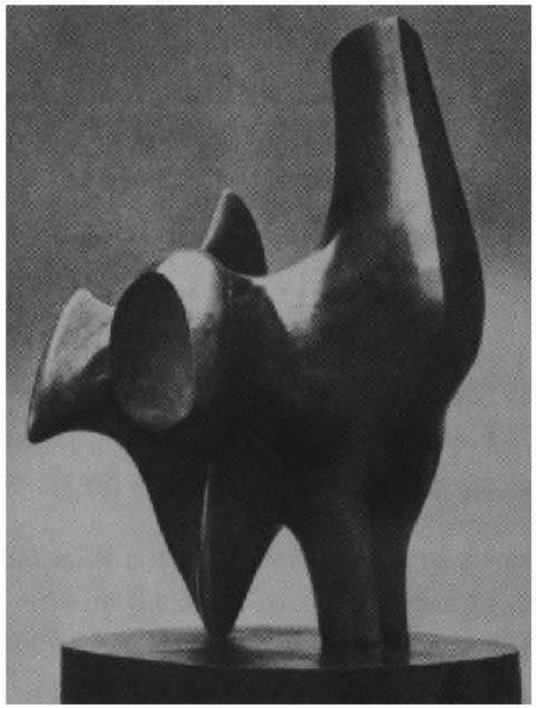


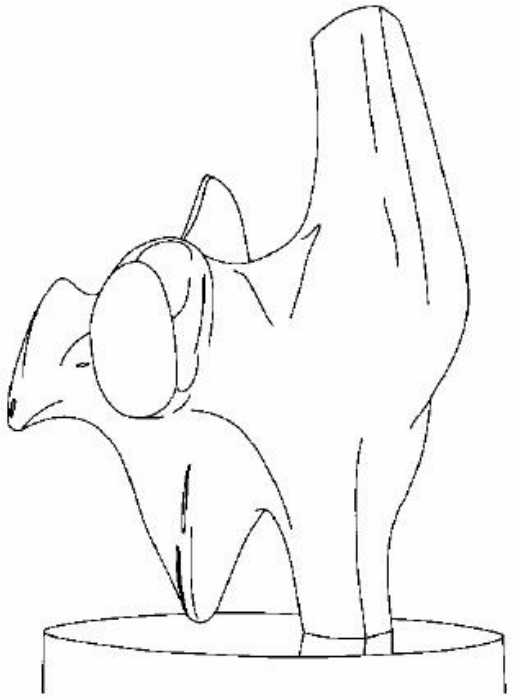
Mudanças entre tons claros e escuros

  - Baseados em Similaridades
    - Baseada em limiarização, crescimento de regiões e divisão e fusão de regiões
- Válidos para imagens estáticas e dinâmicas

Edge Detection – Detecção de Bordas

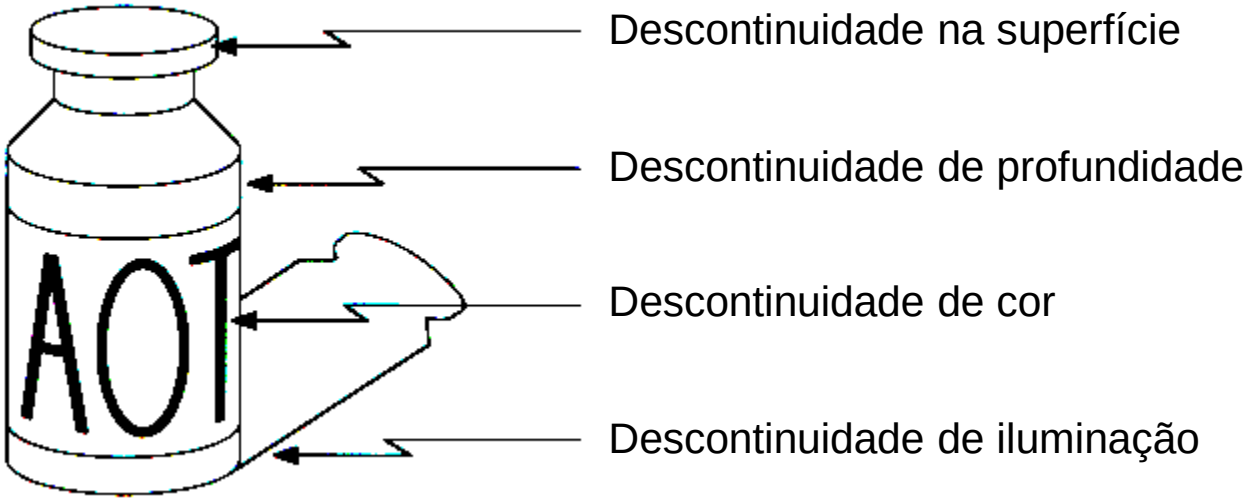
- Converte uma imagem 2D em um conjunto de curvas
  - Extrai características salientes da cena





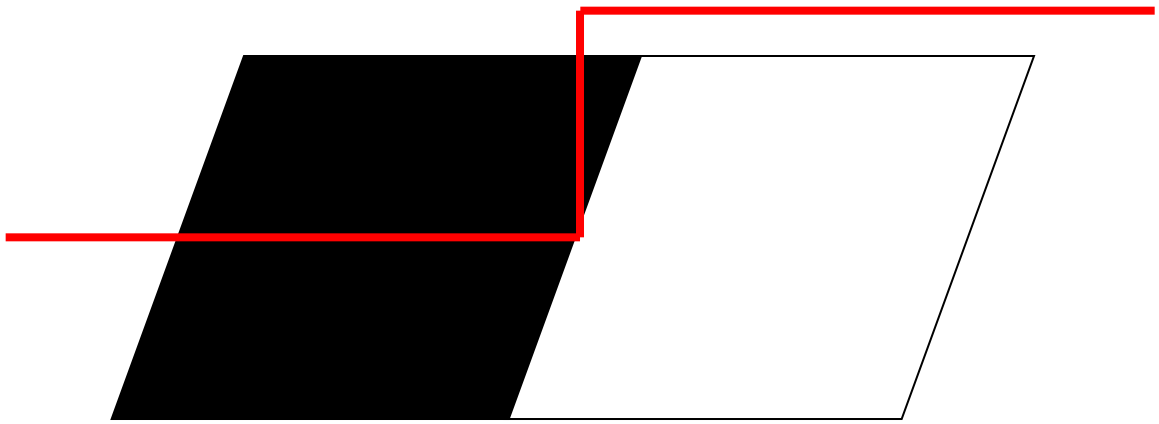
Edge Detection – Detecção de Bordas

- As bordas são causadas por uma variedade de fatores



Edge Detection – Detecção de Bordas

- As bordas ocorrem junto com mudanças



Edge Detection – Detecção de Bordas

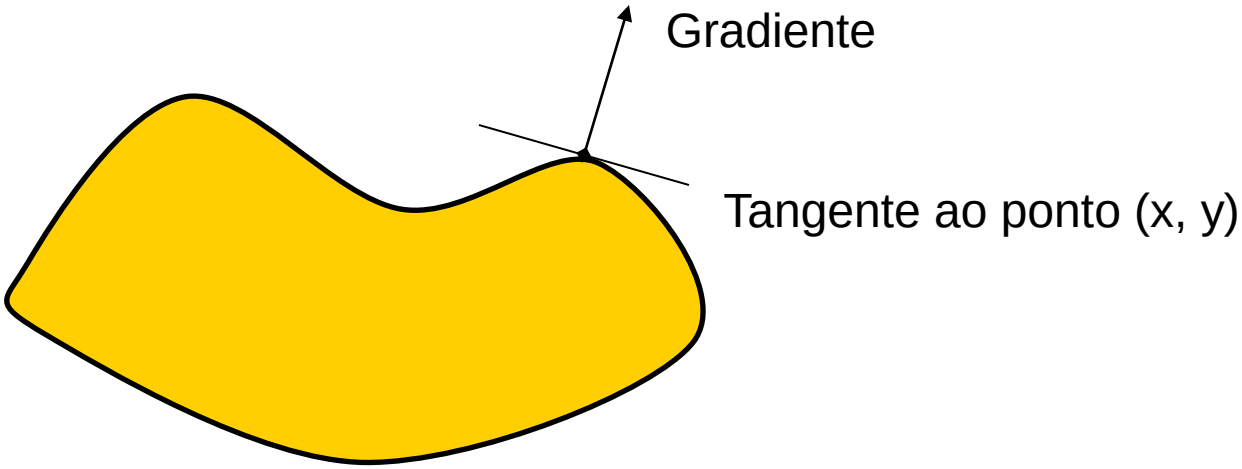
- Derivada de Primeira Ordem / Métodos de Gradiente
  - Operadores de Roberts
  - Operadores de Sobel
  - Operadores de Prewitt
- Derivada de Segunda Ordem
  - Laplaciano
  - Laplaciano de Gaussiana (LoG)
  - Diferença de Gaussianas (DoG)
- Detecção de Bordas Ótimo
  - Canny Edge Detection

Edge Detection – Detecção de Bordas

- Gradiente
  - A detecção de bordas é essencialmente uma operação de identificação de mudanças locais significativas nos tons de uma imagem
  - Essas mudanças podem ser descritas através de derivadas
  - Como uma imagem depende de duas coordenadas espaciais, as bordas podem ser expressas por meio de derivadas parciais
  - Uso do Gradiente: um vetor cuja direção indica os locais nos quais os níveis de cinza sofrem maior variação

Edge Detection – Detecção de Bordas

- Gradiente
  - O gradiente tem direção sempre perpendicular à tangente da borda



Edge Detection – Detecção de Bordas

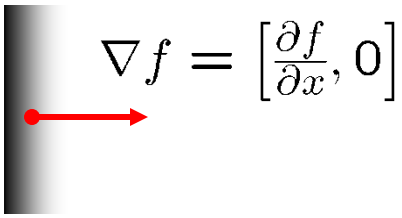
- Gradiente
  - Para detecção de bordas, duas medidas são importantes:
    - A magnitude do vetor gradiente
      - $\nabla f = mag(\nabla \mathbf{f}) = (Gx^2 + Gy^2)^{1/2} = ((\partial f/\partial x)^2 + (\partial f/\partial y)^2)^{1/2}$
      - ou
      - $\nabla f \approx |Gx| + |Gy|$
    - A direção do vetor gradiente
      - Seja $\theta(x, y)$ o ângulo da direção do vetor ∇f na posição (x,y), então $\theta(x, y)=\tan^{-1}(Gy/Gx)$

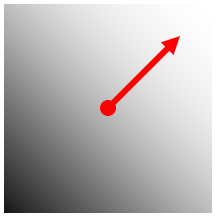
Edge Detection – Detecção de Bordas

- Gradiente de uma imagem
  - O gradiente de uma imagem

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

  - O gradiente aponta na direção da mudança mais rápida de intensidade





Edge Detection – Detecção de Bordas

- Gradiente Discreto
  - Como podemos calcular a derivada de uma imagem f(x, y)
    - Opção 1: reconstruir uma imagem contínua e então tomar o gradiente
    - Opção 2: tomar a derivada discreta (diferença finita)

$$\frac{\partial f}{\partial x}[x, y] \approx f[x+1, y] - f[x, y]$$

Edge Detection – Detecção de Bordas

- Gradiente Discreto
  - Uma mudança de intensidade pode ser detectada pela diferença entre os valores de pixels adjacentes
  - Bordas verticais podem ser detectadas pela diferença horizontal entre pontos, enquanto bordas horizontais podem ser detectadas pela diferença vertical entre pontos adjacentes da imagem

Edge Detection – Detecção de Bordas

- Gradiente Discreto

f(x-1,y-1)	f(x-1,y)	f(x-1,y+1)
f(x,y-1)	f(x,y)	f(x,y+1)
f(x+1,y-1)	f(x+1,y)	f(x+1,y+1)

  - Uma outra abordagem é o cálculo aproximado do gradiente por diferenças cruzadas:

$$\nabla f \approx \sqrt{[f(x,y) - f(x+1,y+1)]^2 + [f(x+1,y) - f(x,y+1)]^2}$$

Edge Detection – Detecção de Bordas

- Gradiente Discreto

f(x-1,y-1)	f(x-1,y)	f(x-1,y+1)
f(x,y-1)	f(x,y)	f(x,y+1)
f(x+1,y-1)	f(x+1,y)	f(x+1,y+1)

  - Ou uso de valores absolutos:

$$\nabla f \approx |f(x,y) - f(x+1,y+1)| + |f(x+1,y) - f(x,y+1)|$$

Edge Detection – Detecção de Bordas

- Gradiente Discreto
  - Essa aproximação por valores absolutos pode ser implementada por máscaras
  - Toma-se o valor absoluto das duas máscaras e soma-se os resultados:

Gx=

1	0
0	-1

Gy=

0	-1
1	0

Edge Detection – Detecção de Bordas

- Operadores de Kirsh (1971)
  - Consiste em 8 máscaras de convolução orientadas em 45º
  - Para cada pixel da imagem, o operador aplica cada uma das 8 orientações da máscara de derivadas e mantém o valor máximo
  - O gradiente é então obtido pela maior resposta do conjunto de máscaras

Edge Detection – Detecção de Bordas

- Operadores de Kirsh (1971)

5	-3	-3
5	0	-3
5	-3	-3

-3	-3	-3
5	0	-3
5	5	-3

-3	-3	-3
-3	0	-3
5	5	5

-3	-3	-3
-3	0	5
-3	5	5

-3	-3	5
-3	0	5
-3	-3	5

-3	5	5
-3	0	5
-3	-3	-3

5	5	5
-3	0	-3
-3	-3	-3

5	5	-3
5	0	-3
-3	-3	-3

Edge Detection – Detecção de Bordas

- Operadores de Sobel (1990)
  - Aproxima a magnitude do gradiente como a diferença de valores ponderados dos níveis de cinza como:
    - $Gx \approx [f(x - 1, y + 1) + 2f(x, y + 1) + f(x + 1,y + 1)] – [f(x - 1, y - 1) +2f(x, y - 1) + f(x + 1, y - 1)]$
    - $Gy \approx [f(x +1, y - 1) + 2f(x + 1, y) + f(x + 1, y + 1)] – [f(x - 1, y - 1) +2f(x - 1, y) + f(x - 1, y + 1)]$

Edge Detection – Detecção de Bordas

- Operadores de Sobel (1990)
  - Operadores mais comumente usados:

G_x

$\frac{1}{p}$

-1	0	1
$2-p$	0	$p-2$
-1	0	1

G_y

$\frac{1}{p}$

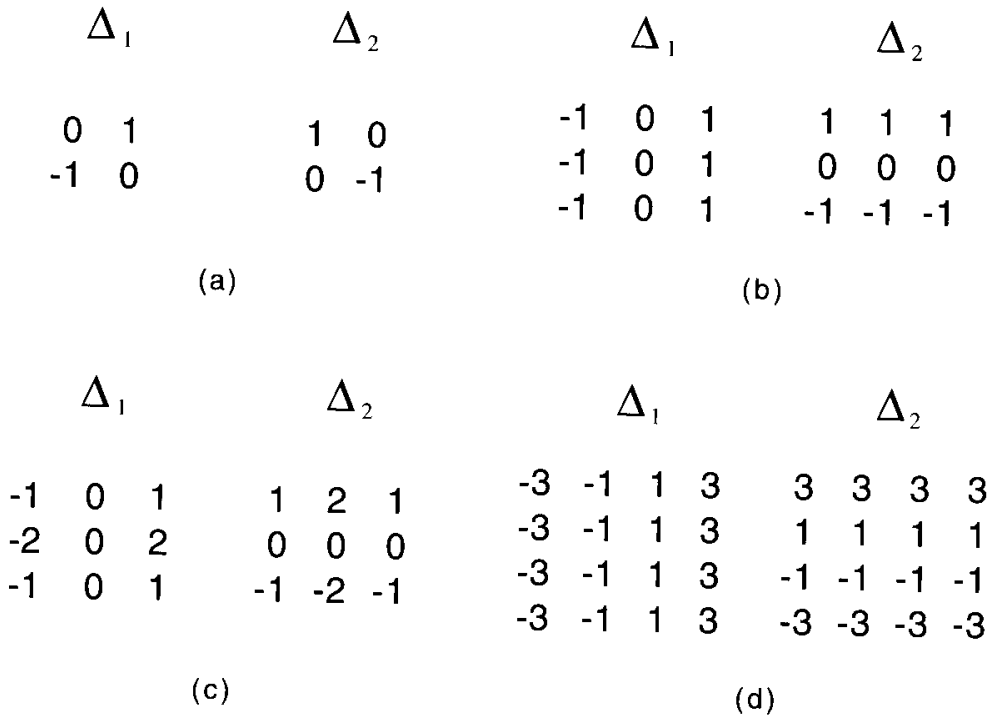
-1	$2-p$	-1
0	0	0
1	$p-2$	1

	Prewitt	Sobel	Isotropic
p	3	4	$2+\sqrt{2}$

  - A definição padrão dos operadores omite o 1/p
    - Não faz diferença para a detecção de bordas
    - O termo 1/p **é** necessário para ter o valor correto do gradiente

Edge Detection – Detecção de Bordas

- Outros Operadores



(a): operador de Roberts (b): 3x3 operador de Prewitt
(c): operador de Sobel (d) 4x4 operador de Prewitt

Detecção de Descontinuidades

- Detecção de Linhas

-1	-1	-1
2	2	2
-1	-1	-1

Horizontal

-1	-1	2
-1	2	-1
2	-1	-1

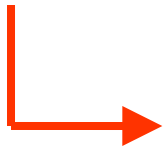
+45º

-1	2	-1
-1	2	-1
-1	2	-1

Vertical

2	-1	-1
-1	2	-1
-1	-1	2

-45º



Intensifica regiões horizontais em uma imagem

Detecção de Descontinuidades

- Detecção de Bordas
  - Abordagem mais comum para detecção de descontinuidades
  - Uma borda é o limite entre duas regiões com propriedades relativamente distintas de nível de cinza

Detecção de Descontinuidades

- Detecção de Bordas
  - Operadores de Gradiente (Sobel)

-1	-2	-1
0	0	0
1	2	1

-1	0	1
-2	0	2
-1	0	1

Detecção de Descontinuidades

- Detecção de Bordas
  - Máscaras Ortogonais (Frei e Chen)

0	1	0
-1	0	-1
0	1	0

Detecção de Descontinuidades

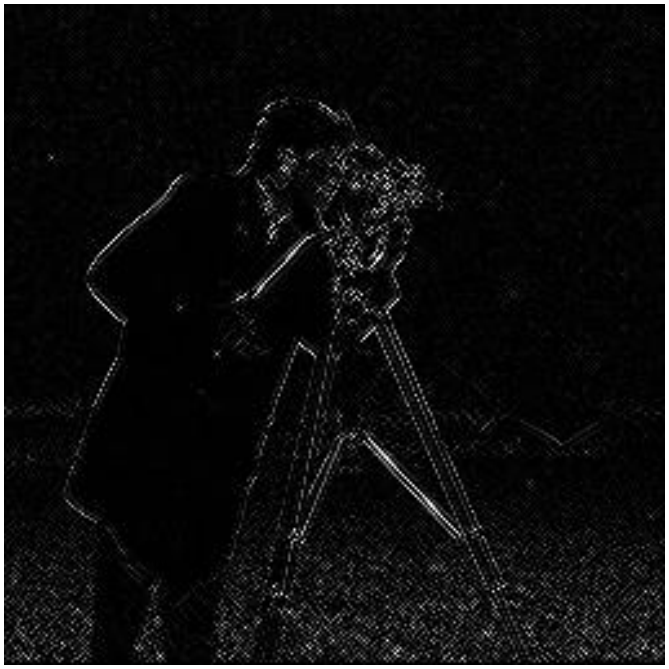
- Detecção de Bordas
  - Máscaras Ortogonais (Frei e Chen)

-1	0	1
0	0	0
1	0	-1

Detecção de Descontinuidades

- Detecção de Bordas
  - Máscaras Ortogonais (Frei e Chen)

1	-2	1
-2	4	-2
1	-2	1



Detecção de Descontinuidades

- Detecção de Bordas
  - Máscaras Ortogonais (Frei e Chen)

-2	1	-2
1	4	1
-2	1	-2

Operador de Roberts - Exemplo

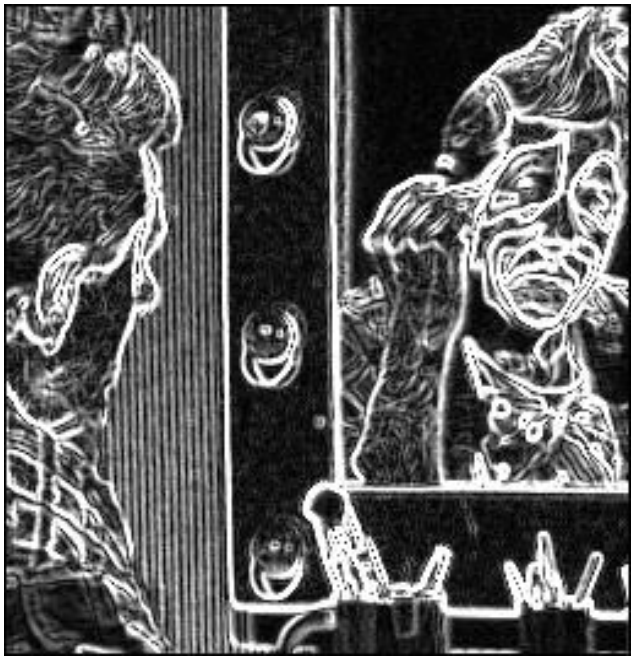
- Pontos espúrios indicam que o operador é susceptível a ruído

Original

Roberts

Operador de Sobel - Exemplo

- Compare a saída do operador de Sobel com a de Roberts:
  - O ruído ainda está presente mas com menor intensidade
  - O operador de Roberts perdeu algumas bordas
  - O operador de Sobel detecta bordas mais grossas



Sobel

Roberts

Detecção de Descontinuidades

- Derivada de Segunda Ordem
  - Laplaciano

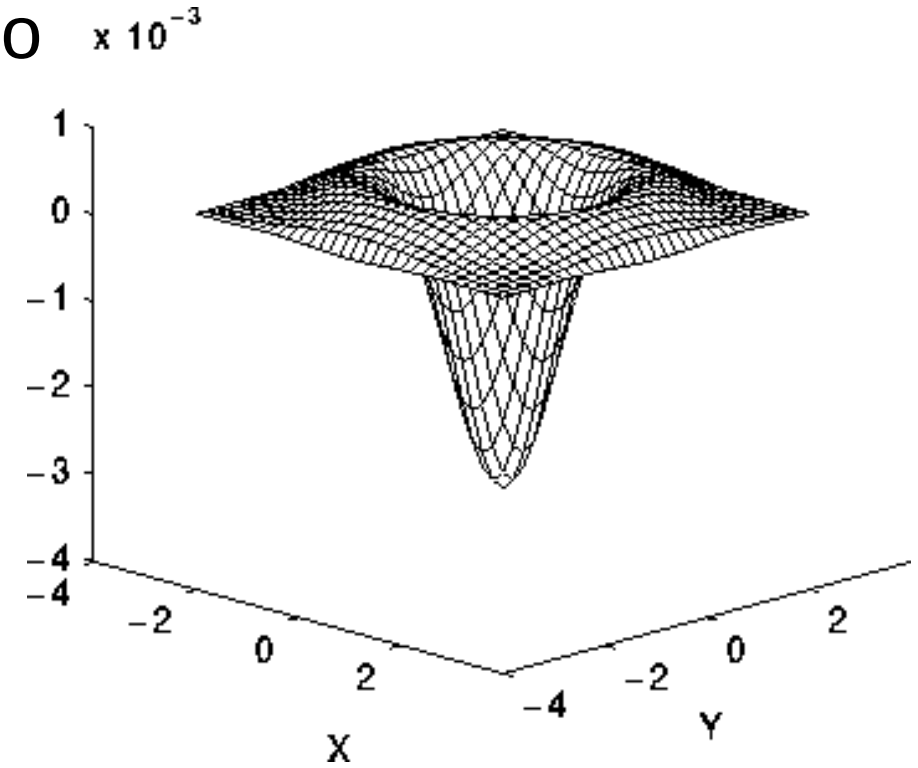
0	1	0
1	-4	1
0	1	0

Detecção de Descontinuidades

- Derivada de Segunda Ordem
- Laplaciano da Gaussiana
  - Também chamado de Marr-Hildreth Edge Detector
- Passos
  - Suavizar a imagem usando filtro Gaussiano
  - Intensificar as bordas usando o operador Laplaciano
  - Passagens pelo Zero denotam a localização de bordas
  - Uso de interpolação linear para determinar a localização dos pixels na borda

Detecção de Descontinuidades

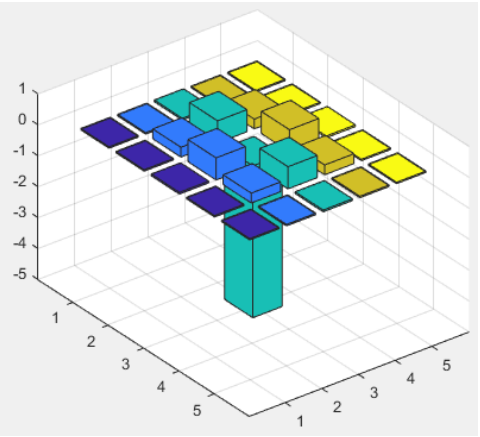
- Laplaciano de Gaussiana
  - Também chamado de operador de Chapéu Mexicano



Detecção de Descontinuidades

- Derivada de Segunda Ordem
  - Laplaciano de Gaussiana

```
>> h = fspecial ('log')
h =
    0.0448    0.0468    0.0564    0.0468    0.0448
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0564    0.7146   -4.9048    0.7146    0.0564
    0.0468    0.3167    0.7146    0.3167    0.0468
    0.0448    0.0468    0.0564    0.0468    0.0448
>> im_filt = imfilter (im, h);
```

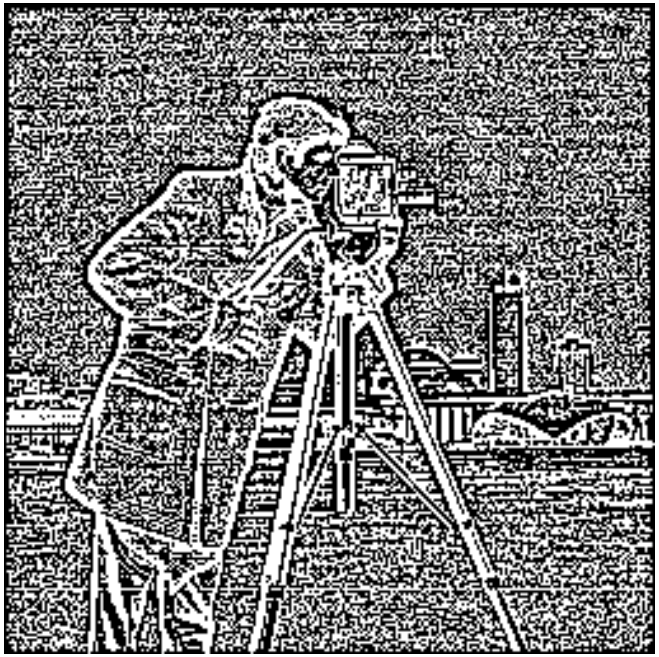


Detecção de Descontinuidades

- Derivada de Segunda Ordem
  - Laplaciano de Gaussiana

Detecção de Descontinuidades

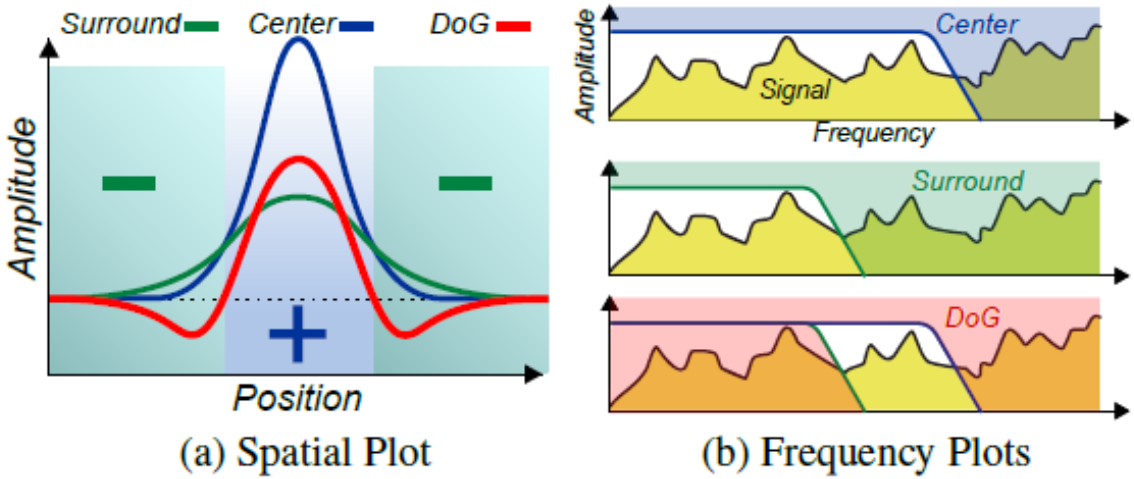
- Derivada de Segunda Ordem
  - Laplaciano de Gaussiana (binarizada)



Detecção de Descontinuidades

- Derivada de Segunda Ordem
- Diferença de Gaussianas (DoG)

```
>> h1 = fspecial ('gaussian', 5, 0.5);
>> h2 = fspecial ('gaussian', 5, 0.3);
>> h = h1 - h2;
>> im_filt = imfilter (im, h);
```



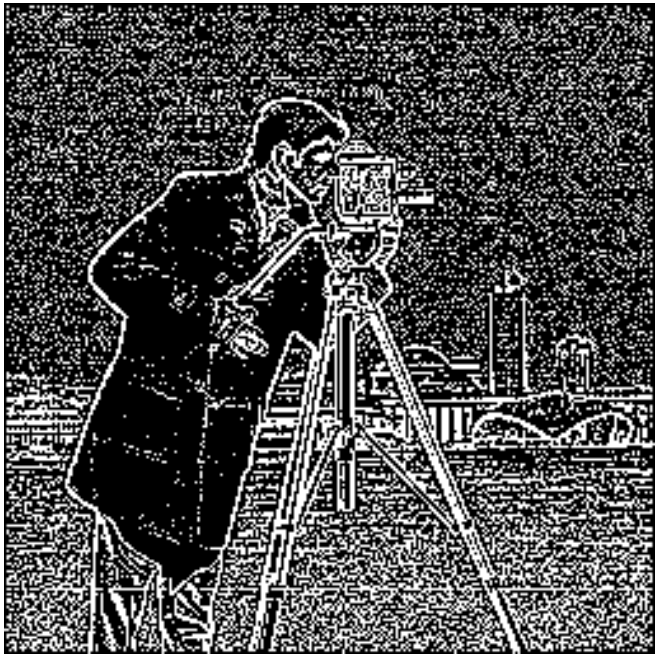
(a) Spatial Plot (b) Frequency Plots

Detecção de Descontinuidades

- Derivada de Segunda Ordem
  - Diferença de Gaussianas

Detecção de Descontinuidades

- Derivada de Segunda Ordem
  - Diferença de Gaussianas (binarizada)



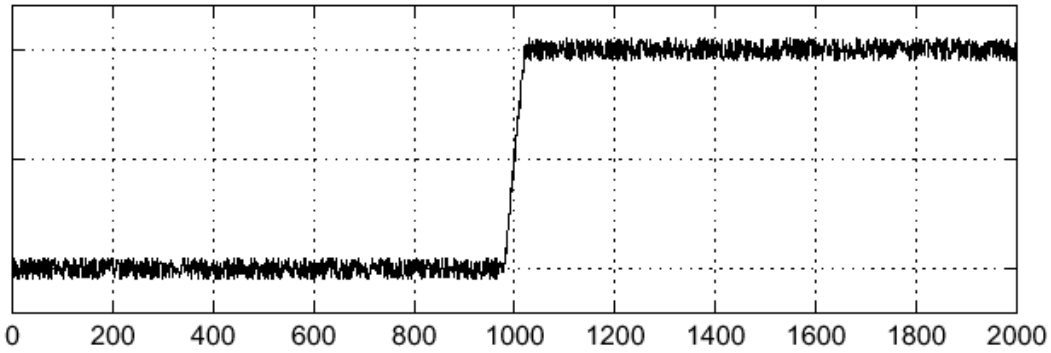
Detecção de Descontinuidades

- Derivada de Segunda Ordem
  - Diferença de Gaussianas (*enhanced*)

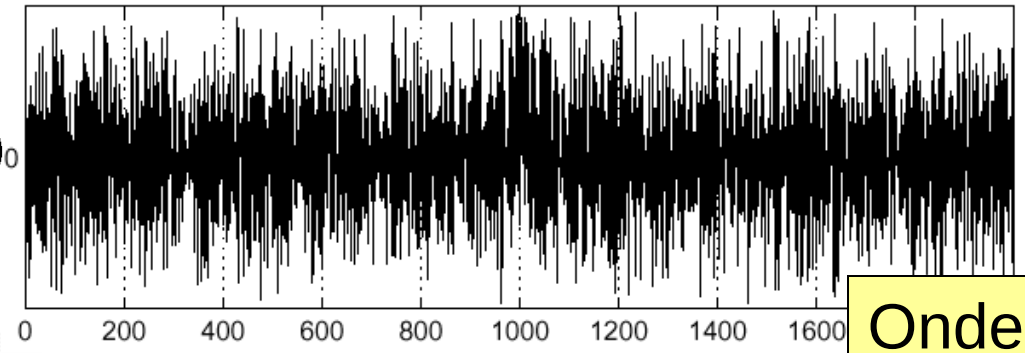
Efeito do ruído

- Considere uma única linha ou coluna da imagem
  - Suponha um sinal ruidoso

$f(x)$



$\frac{d}{dx} f(x)$



Onde está a borda?

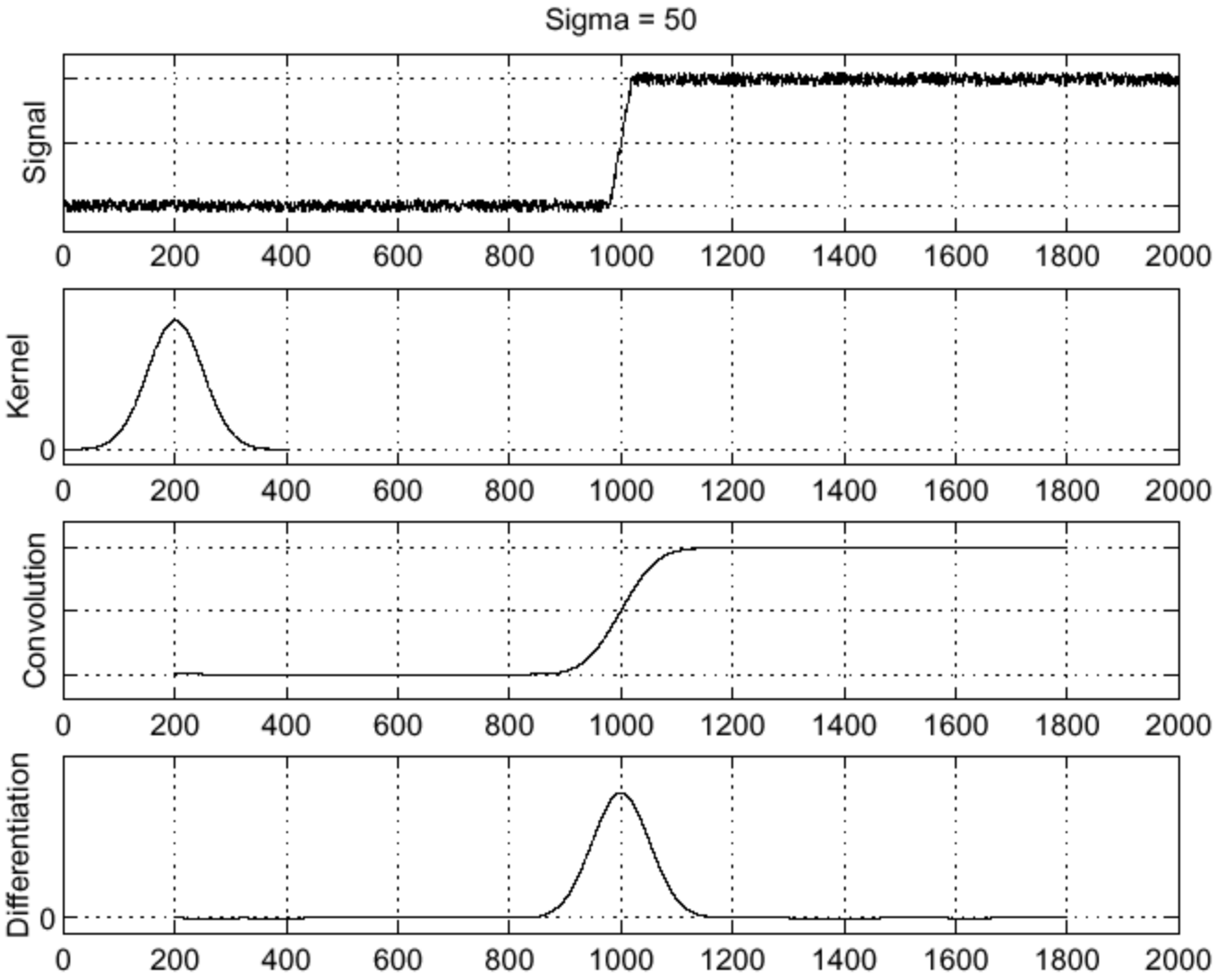
Solução: suavizar primeiro

f

h

$h \star f$

$\frac{\partial}{\partial x}(h \star f)$



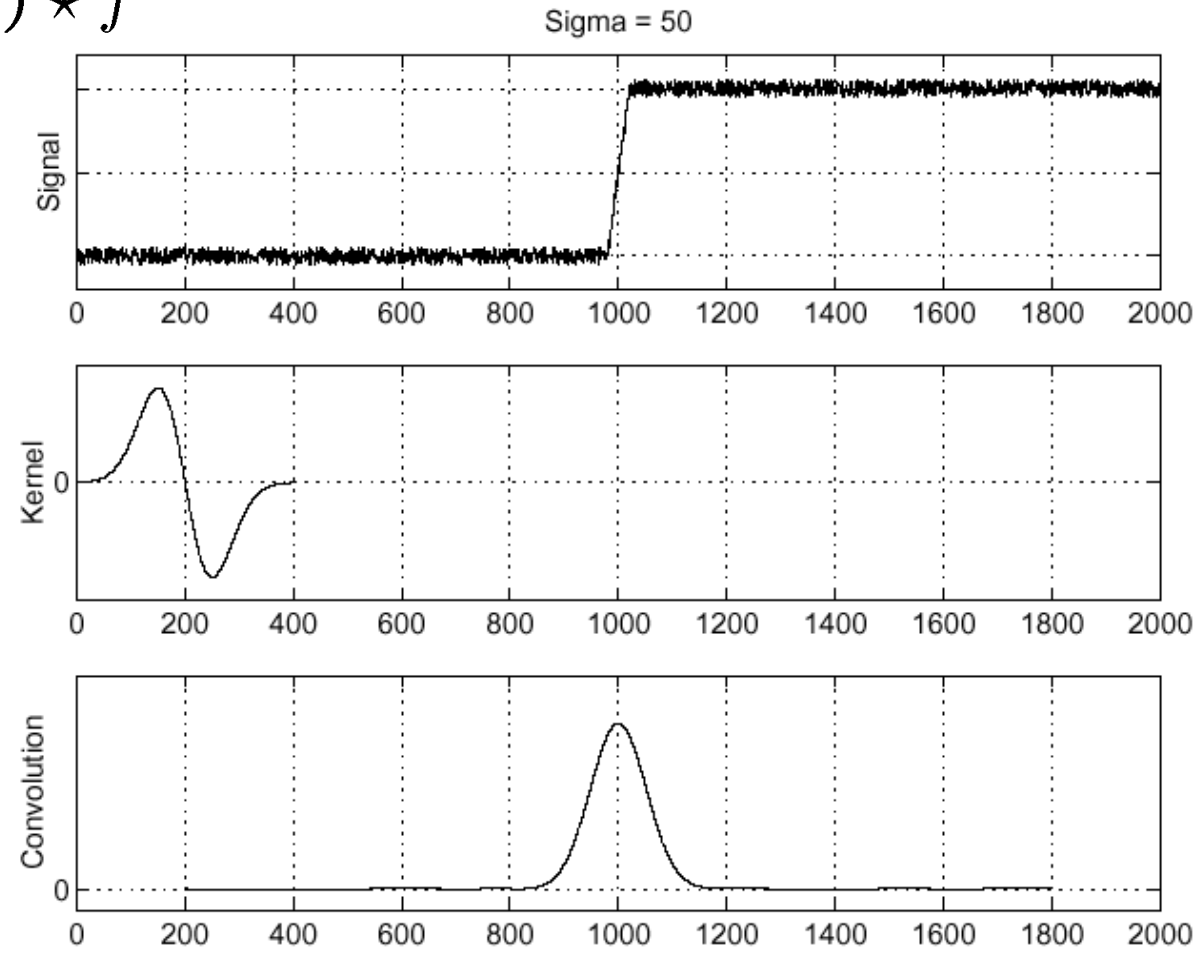
Teorema da Derivada da Convolução

$$\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$$

f

$\frac{\partial}{\partial x}h$

$(\frac{\partial}{\partial x}h) \star f$



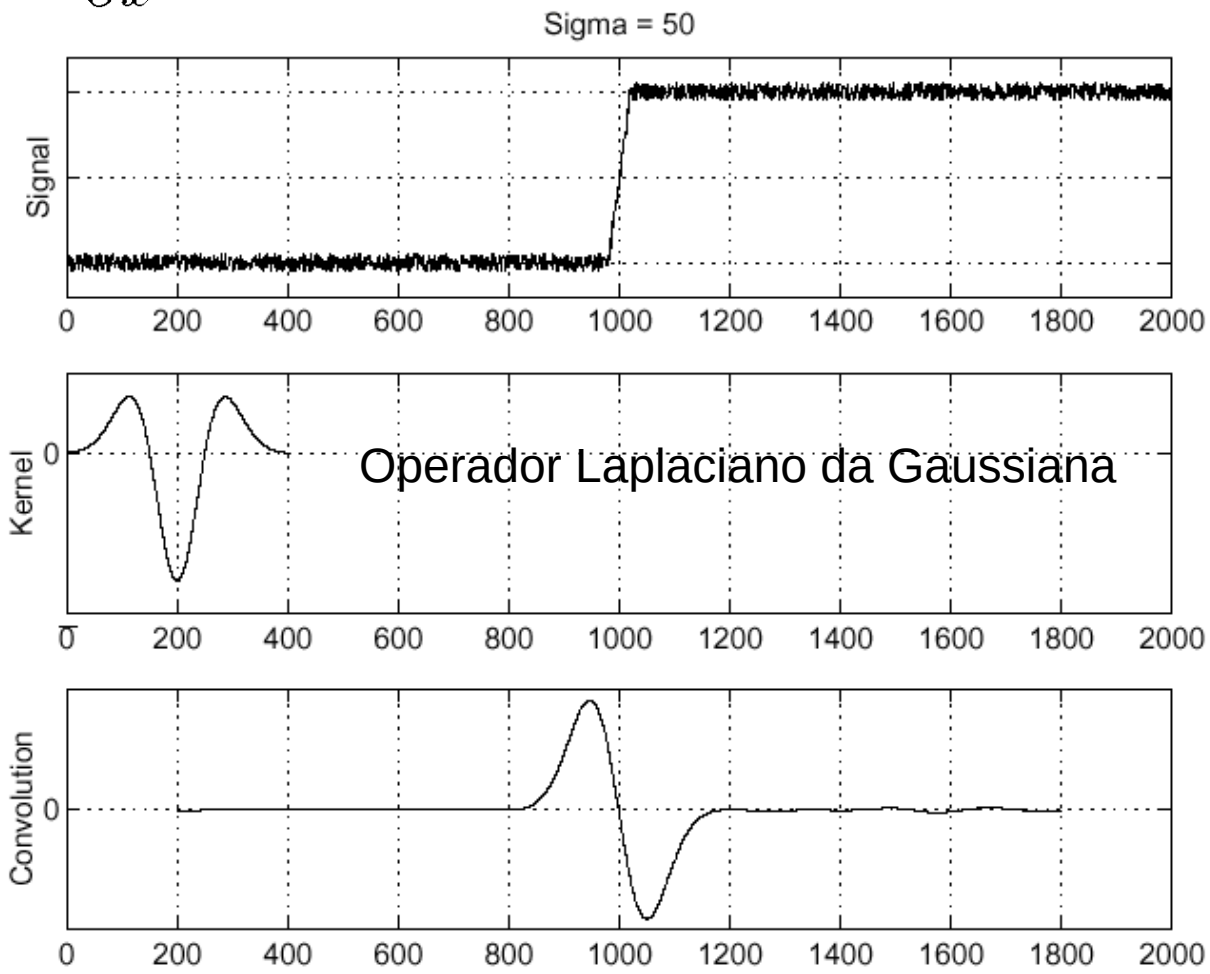
Laplaciano da Gaussiana (LoG)

- Considere $\frac{\partial^2}{\partial x^2}(h \star f)$

f

$\frac{\partial^2}{\partial x^2}h$

$(\frac{\partial^2}{\partial x^2}h) \star f$



Detecção de Borda Ótimo: Canny

- John Canny, 1986
- Um bom detector deve se preocupar com:
  - Taxa de erro: O detector deve responder apenas a bordas, encontrando todas
  - Localização: A distância entre os pixels de borda encontrados e as bordas reais deve ser a menor possível
  - Resposta: O detector não deve identificar múltiplos pixels de borda onde só existir uma borda

Detecção de Borda Ótimo: Canny

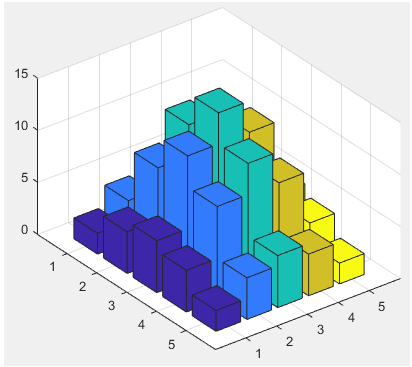
- Algoritmo:
  - 1. Suavização da imagem
    - Objetivo: Eliminar ruído
  - 2. Busca por gradientes
    - Intensifica regiões com maiores derivadas
  - 3. Encontra a direção das bordas
  - 4. Aproxima as direções para 0, 45, 90 ou 135 graus
  - 5. *Non-maximum Suppression*
    - Apenas máximos locais são considerados bordas
  - 6. Limiarização por histerese + Edge Tracking
    - Bordas finais são determinadas, suprimindo as bordas que não estão conectadas a bordas “fortes”

Detecção de Borda Ótimo: Canny

- Algoritmo (na prática)
  - Suavização
    - Aplicação de um filtro Gaussiano
    - Exemplo: desvio = 1,4:

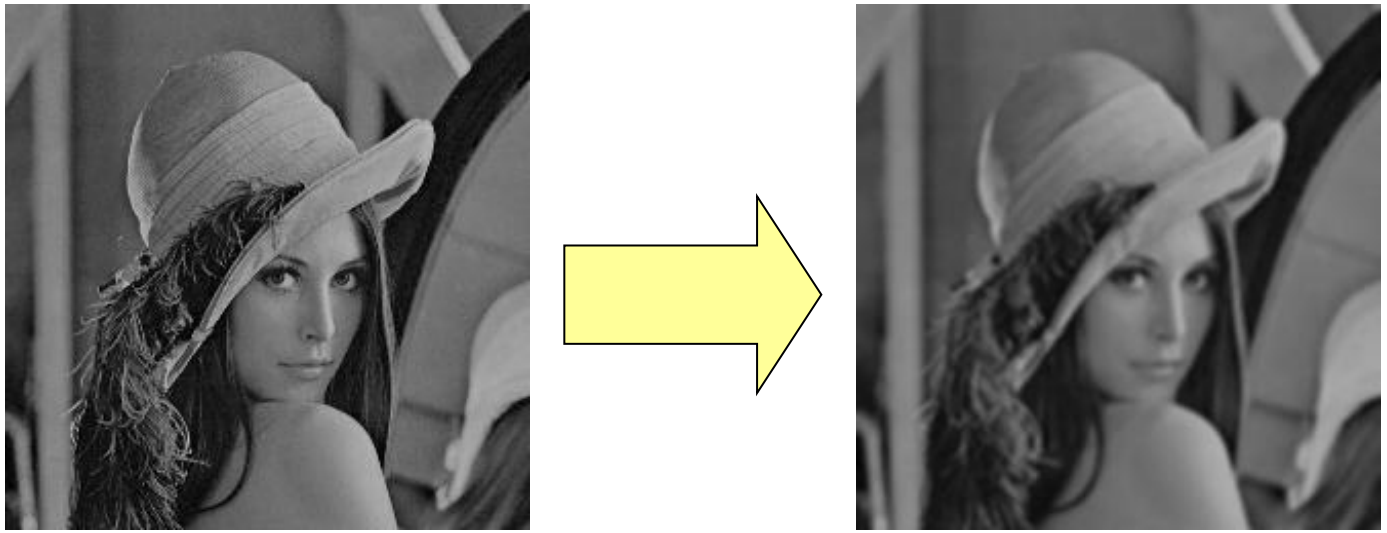
$$B = \frac{1}{159} \cdot \begin{bmatrix} 2 & 4 & 5 & 4 & 2 \\ 4 & 9 & 12 & 9 & 4 \\ 5 & 12 & 15 & 12 & 5 \\ 4 & 9 & 12 & 9 & 4 \\ 2 & 4 & 5 & 4 & 2 \end{bmatrix}$$

Fator de Normalização



Detecção de Borda Ótimo: Canny

- Algoritmo (na prática)
  - Suavização

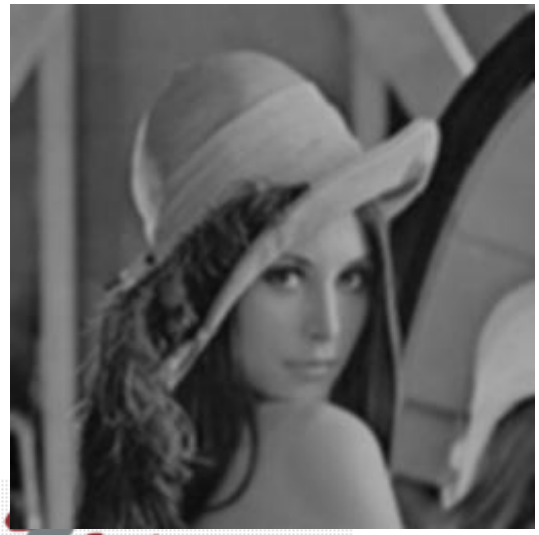


Detecção de Borda Ótimo: Canny

- Algoritmo (na prática)
  - Busca por gradientes

$$K_{GX} = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \qquad K_{GY} = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

Gx Gy



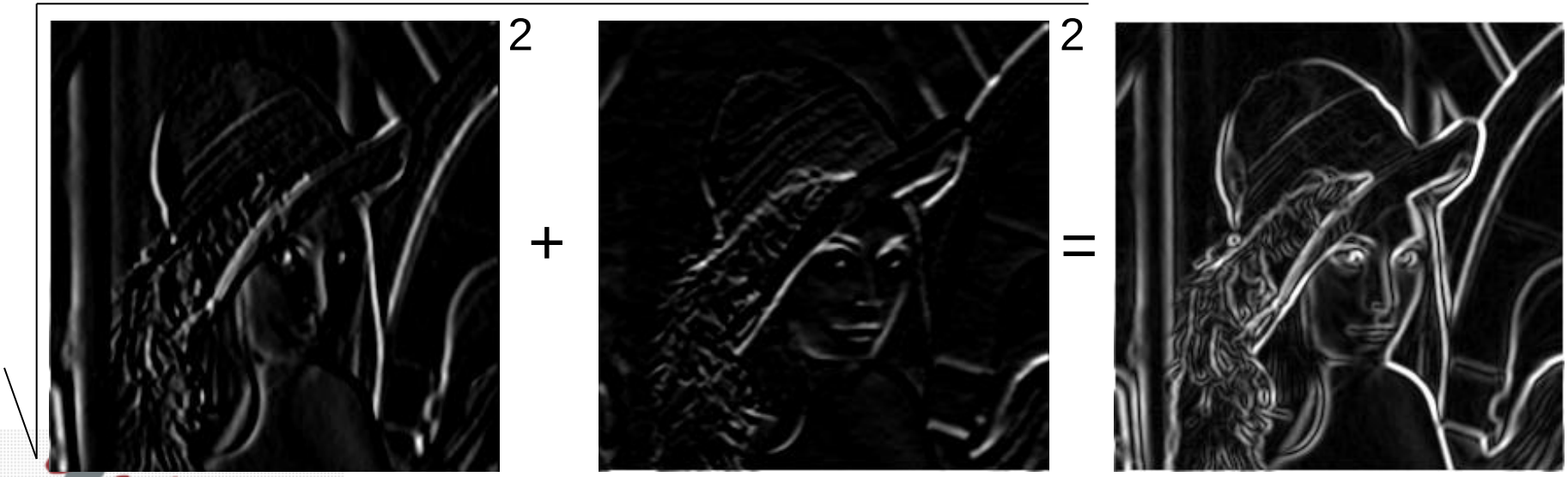
Detecção de Borda Ótimo: Canny

- Algoritmo (na prática)
  - A magnitude do gradiente pode ser encontrada como:

$$|G| = \sqrt{G_x^2 + G_y^2}$$

ou

$$|G| = |G_x| + |G_y|$$

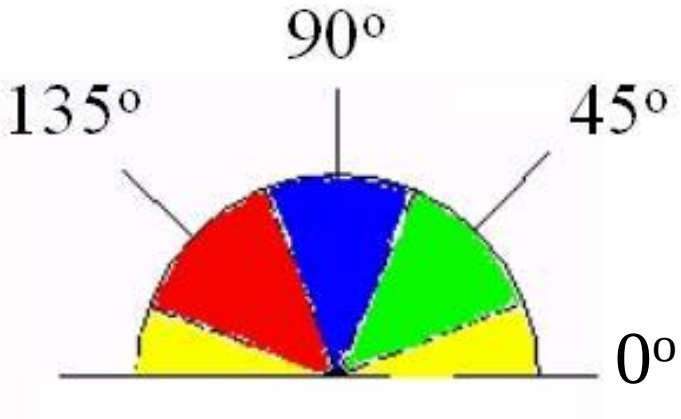


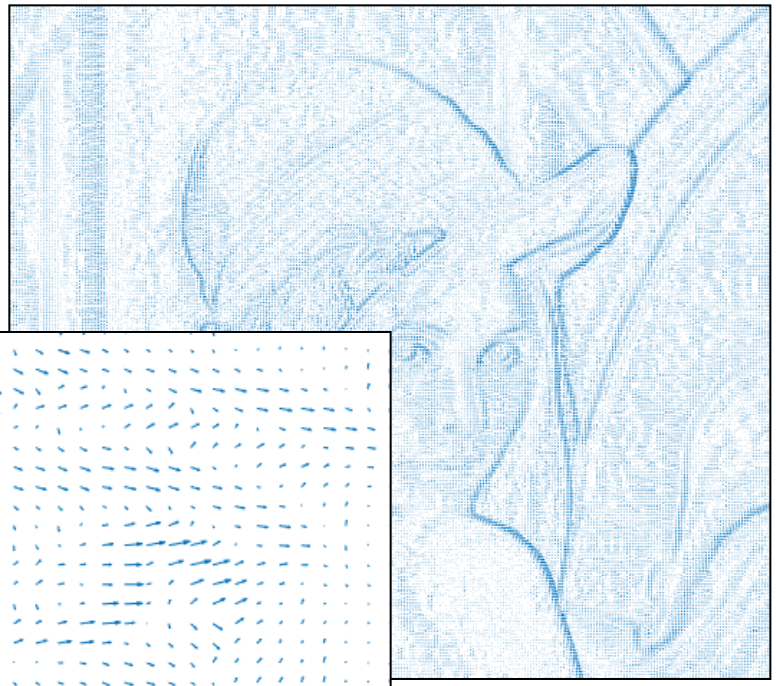
Detecção de Borda Ótimo: Canny

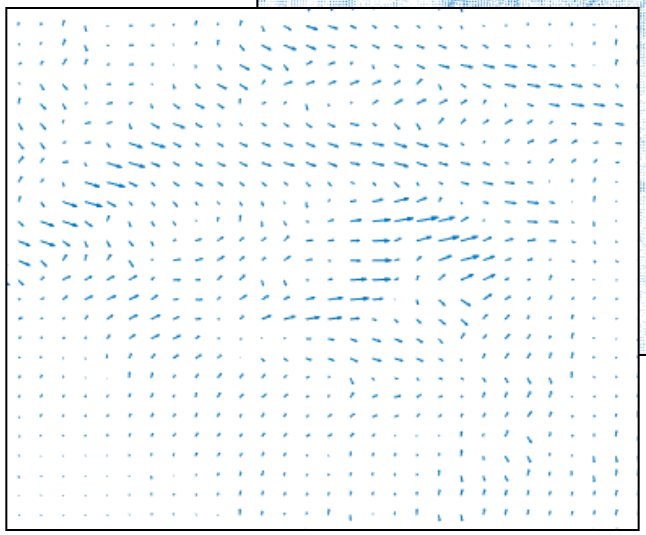
- Algoritmo (na prática)
  - A direção das arestas é calculada e armazenada:

$$\theta = \arctan\left(\frac{|G_y|}{|G_x|}\right)$$

  - e aproximada para





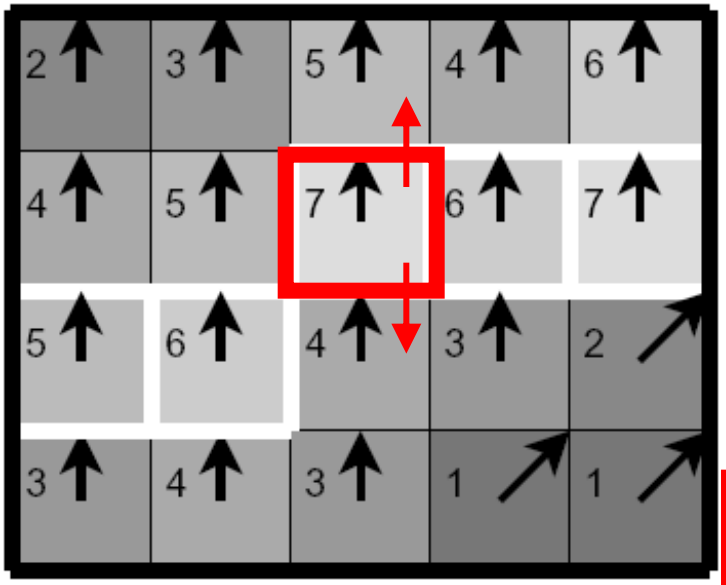


Detecção de Borda Ótimo: Canny

- Algoritmo (na prática)
  - *Non-Maximum Suppression*:
    - O propósito deste passo é converter arestas suavizadas na imagem da magnitude em arestas mais “sharp”
    - Isso é feito preservando os máximos locais na imagem do gradiente e removendo o resto
    - A comparação é feita na direção do ângulo
      - Se a direção for de 0^o, compara o elemento anterior e posterior da mesma linha. Se o valor do pixel for menor em um dos dois casos, torna-se zero (é suprimido)
      - Faz algo semelhante nas outras direções

Detecção de Borda Ótimo: Canny

- Algoritmo (na prática)
  - Non-Maximum Suppression:
    - Exemplo: pixels apontando para o norte são comparados com pixels abaixo e acima



Os que são preservados são os que estão nos quadrados brancos, pois são os maiores (nessa direção); os outros são removidos.

Exemplo: compara o 7 com os valores acima e abaixo; como ele é maior permanece.

Detecção de Borda Ótimo: Canny

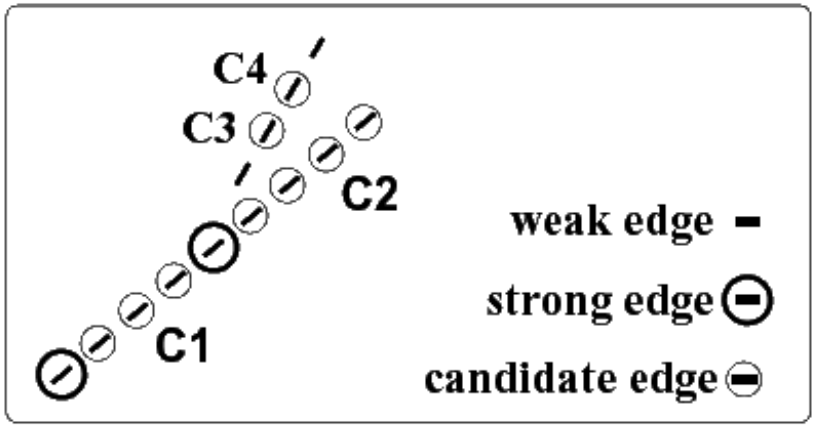
- Algoritmo (na prática)
  - Hysteresis thresholding
    - Dois limiares: um alto e um baixo
    - Qualquer pixel acima do maior limiar é convertido para branco. Os pixels ao redor são então analisados: se seus valores são maiores que o menor limiar, eles são convertidos para branco
      - Um pixel (x, y) é chamado forte se cor(x, y) > Th_{alto}
      - Um pixel (x, y) é chamado fraco se cor(x, y) <= Th_{baixo}
      - Todos os outros pixels são chamados de candidatos

Detecção de Borda Ótimo: Canny

- Algoritmo (na prática)
  - Hysteresis thresholding + Edge Tracking
    - Em cada posição (x, y), descarta o pixel (x, y) se ele é fraco; permanece na imagem de saída se for forte
    - Se um pixel é um candidato, segue a cadeia de máximos locais conectados em ambas as direções na direção da aresta, enquanto cor(i, j) > Th_{baixo}
    - Se um pixel candidato inicial está conectado a um pixel forte, ele não é suprimido; do contrário, é

Detecção de Borda Ótimo: Canny

- Algoritmo (na prática)
  - Hysteresis thresholding + Edge Tracking



As arestas candidatas C1 e C2 são preservadas na saída, enquanto as arestas C3 e C4 são suprimidas.

Detecção de Borda Ótimo: Canny

- Exemplo: Variações nos parâmetros

Gaussiana: 5x5, $\sigma = 1.4$
Thresholding:
Th_{baixo} = 10% do máximo
Th_{alto} = 30% do máximo

Detecção de Borda Ótimo: Canny

- Exemplo: Variações nos parâmetros

Gaussiana: 5x5, $\sigma = 5$
Thresholding:
Th_{baixo} = 10% do máximo
Th_{alto} = 30% do máximo

Detecção de Borda Ótimo: Canny

- Exemplo: Variações nos parâmetros

Gaussiana: 5x5, $\sigma = 1.4$
Thresholding:
Th_{baixo} = 10% do máximo
Th_{alto} = 20% do máximo

Detecção de Borda Ótimo: Canny

- Exemplo: Variações nos parâmetros

Gaussiana: 5x5, $\sigma = 1.4$
Thresholding:
Th_{baixo} = 5% do máximo
Th_{alto} = 10% do máximo

Detecção de Borda Ótimo: Canny

- Exemplo: Variações nos parâmetros

Gaussiana: 5x5, $\sigma = 0.5$
Thresholding:
Th_{baixo} = 5% do máximo
Th_{alto} = 10% do máximo

Detecção de Borda Ótimo: Canny

- Exemplo: Variações nos parâmetros

Gaussiana: 5x5, $\sigma = 0.5$
Thresholding:
Th_{baixo} = 10% do máximo
Th_{alto} = 30% do máximo

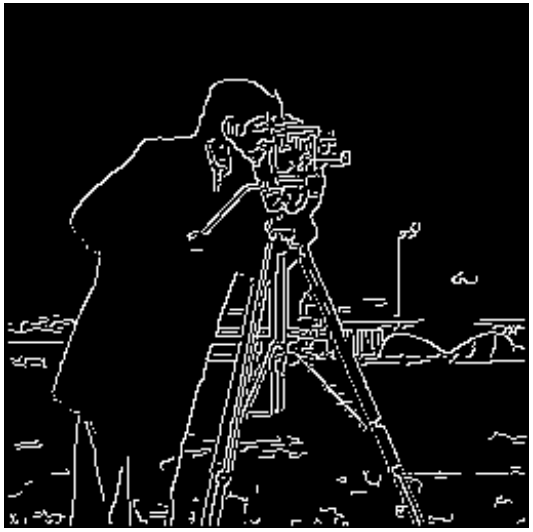
Detecção de Borda Ótimo: Canny

- Exemplo: Variações nos parâmetros

Gaussiana: 5x5, $\sigma = 0.5$
Thresholding:
Th_{baixo} = 10% do máximo
Th_{alto} = 20% do máximo

Detecção de Borda Ótimo: Canny

- Exemplo:

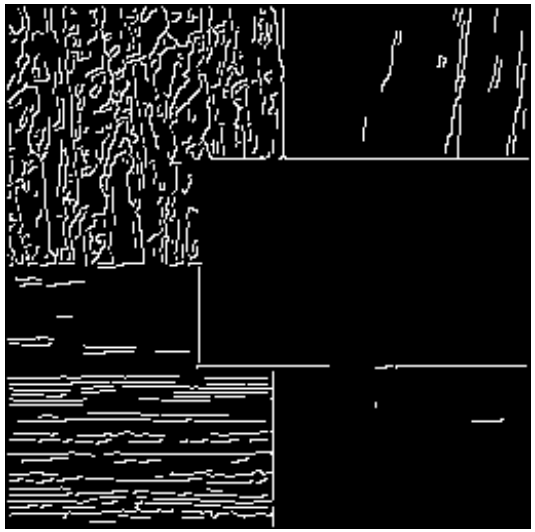


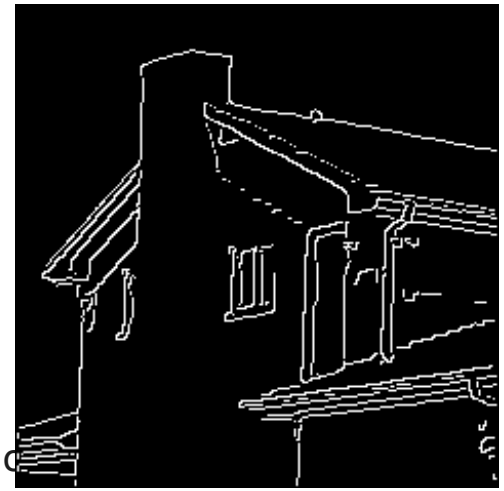
Gaussiana: 5x5, $\sigma = 0.5$
Thresholding:
Th_{baixo} = 10% do máximo
Th_{alto} = 20% do máximo

Detecção de Borda Ótimo: Canny

- Exemplo:

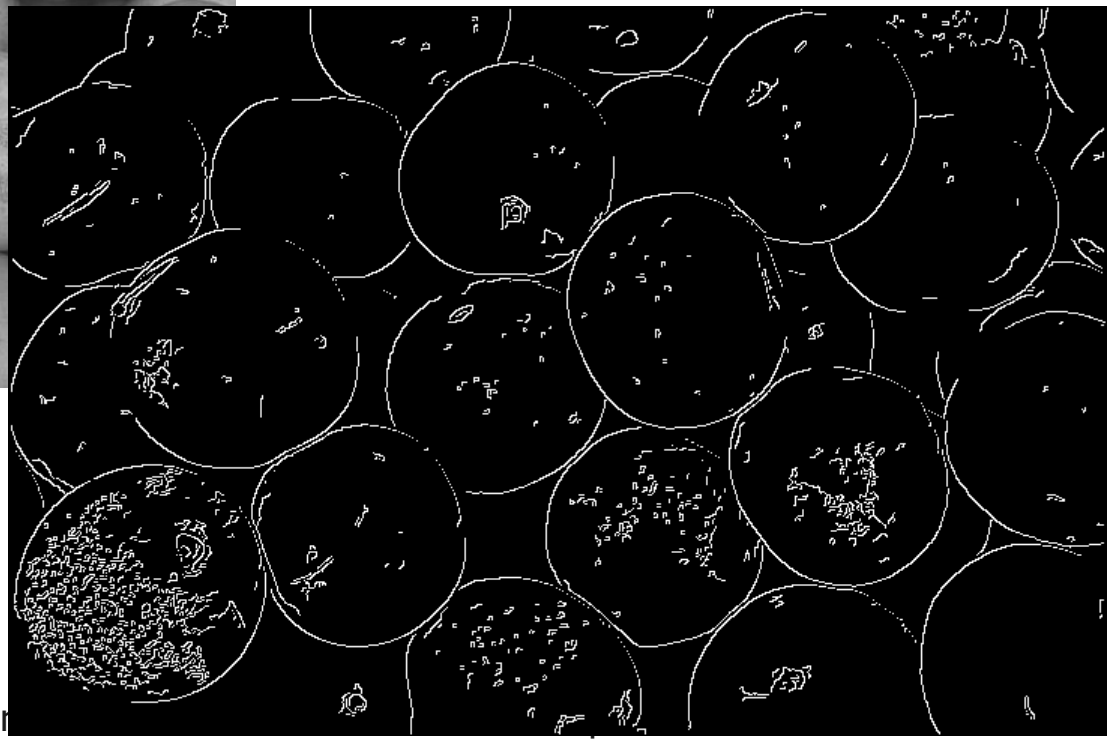
Gaussiana: 5x5, $\sigma = 0.5$
Thresholding:
Th_{baixo} = 10% do máximo
Th_{alto} = 20% do máximo





Detecção de Borda Ótimo: Canny

- Exemplo:



Detecção de Borda

Detecção de Borda

Prewitt

Detecção de Borda

Roberts

Detecção de Borda

Sobel

Detecção de Borda

Canny

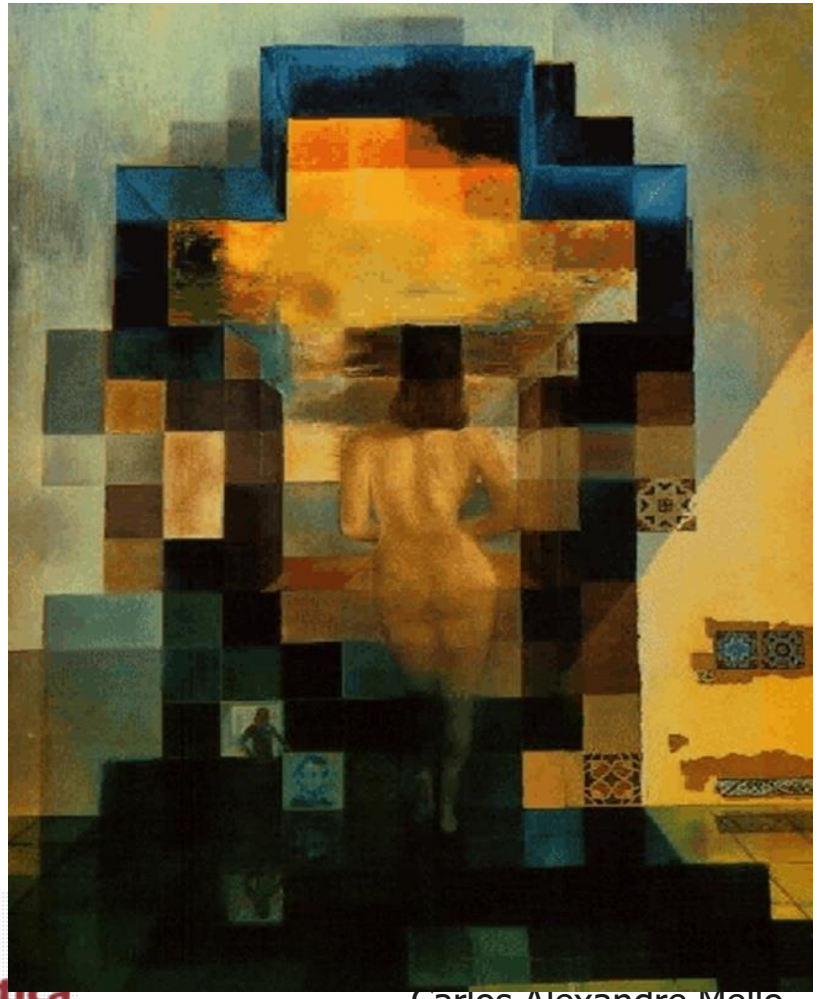
Detecção de Borda

DoG

```
>> h1 = fspecial ('gaussian', [5 5], 0.5);
>> h2 = fspecial ('gaussian', [5 5], 0.7);
>> h3 = h2 – h1;
>> im2 = imfilter (im, h3);
```

Análise Scale-Space
(para auxiliar a detecção de bordas)

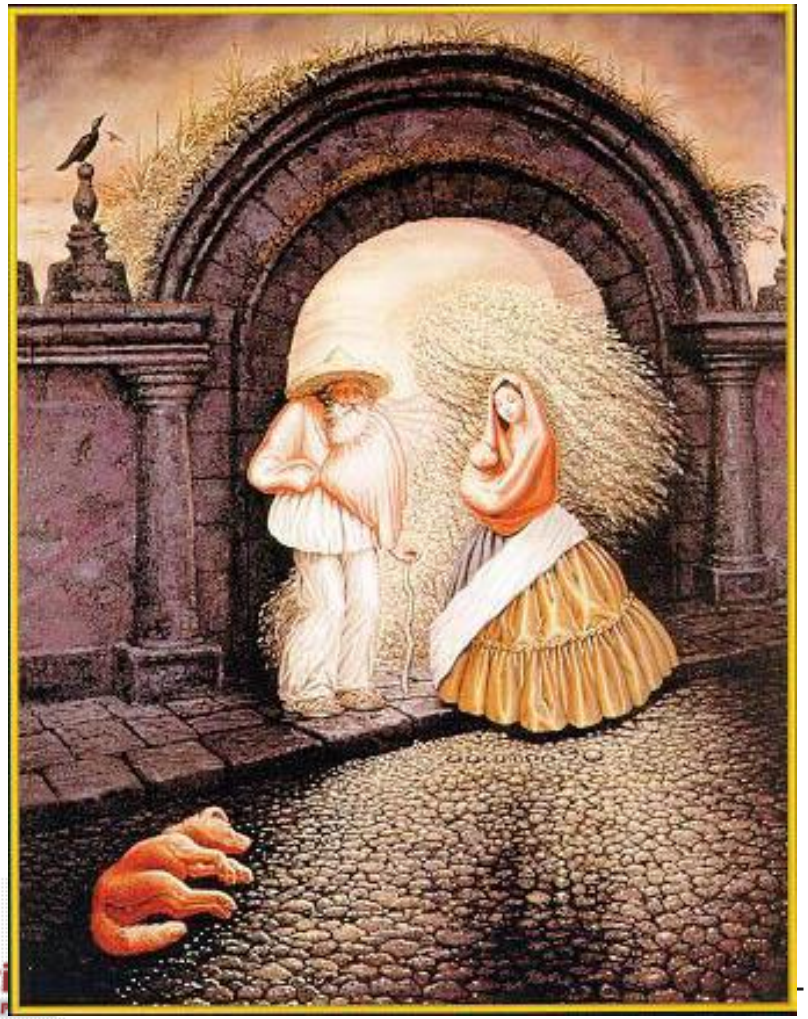
- Importância da escala

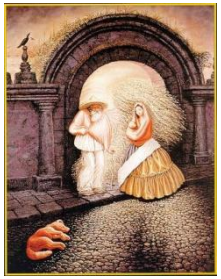


•Objetos existem em certa escala;
•Não se sabe *a priori* que escala olhar

Análise Scale-Space

- Importância da escala

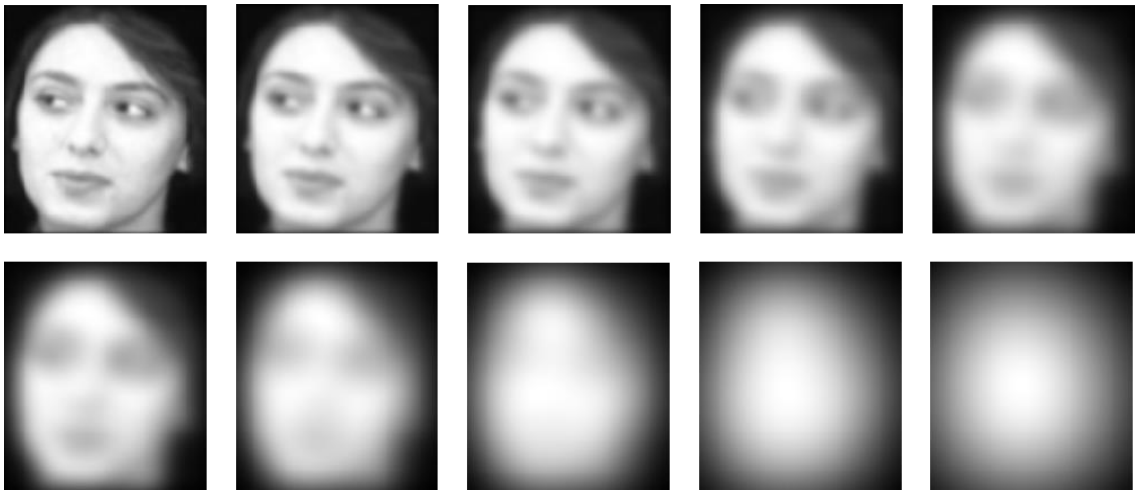




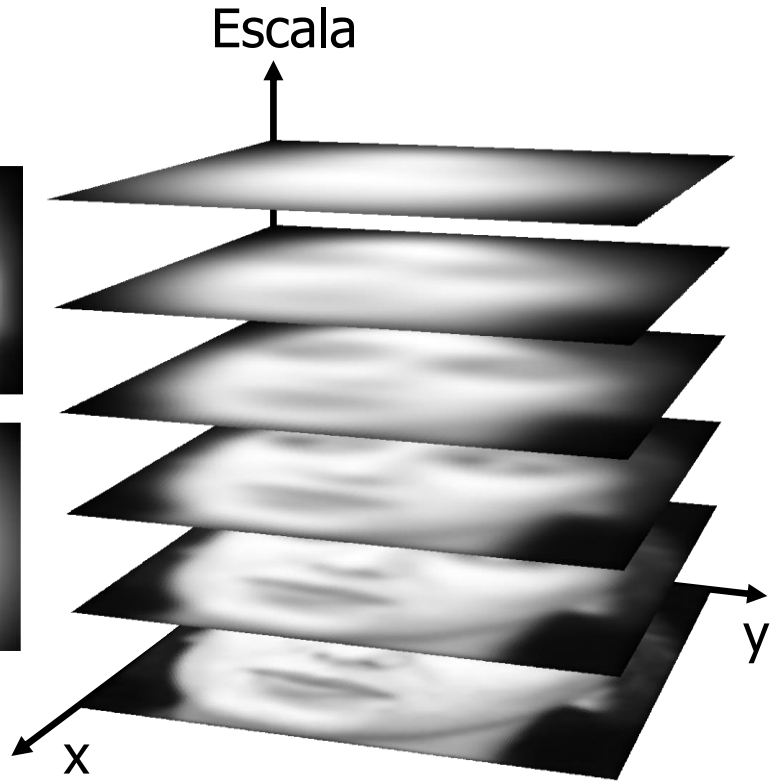
Análise Scale-Space

Solução?

- Olhar todas as escalas simultaneamente!

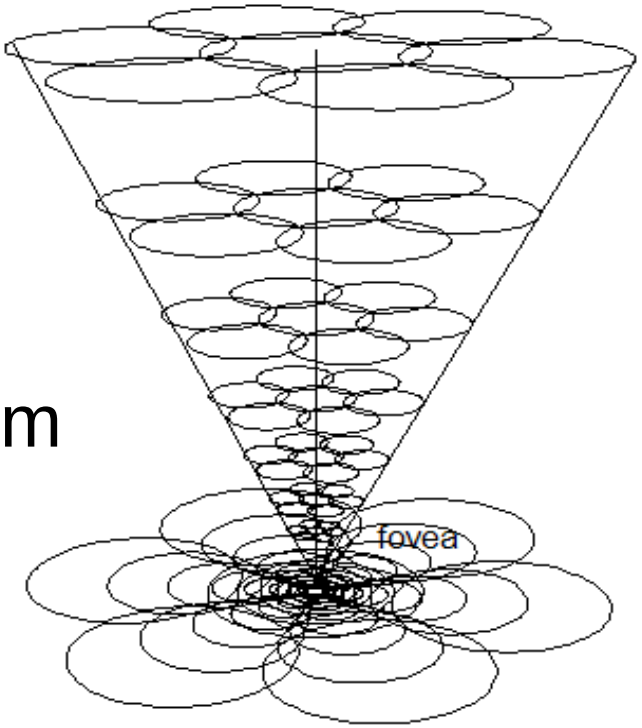


Scale Space



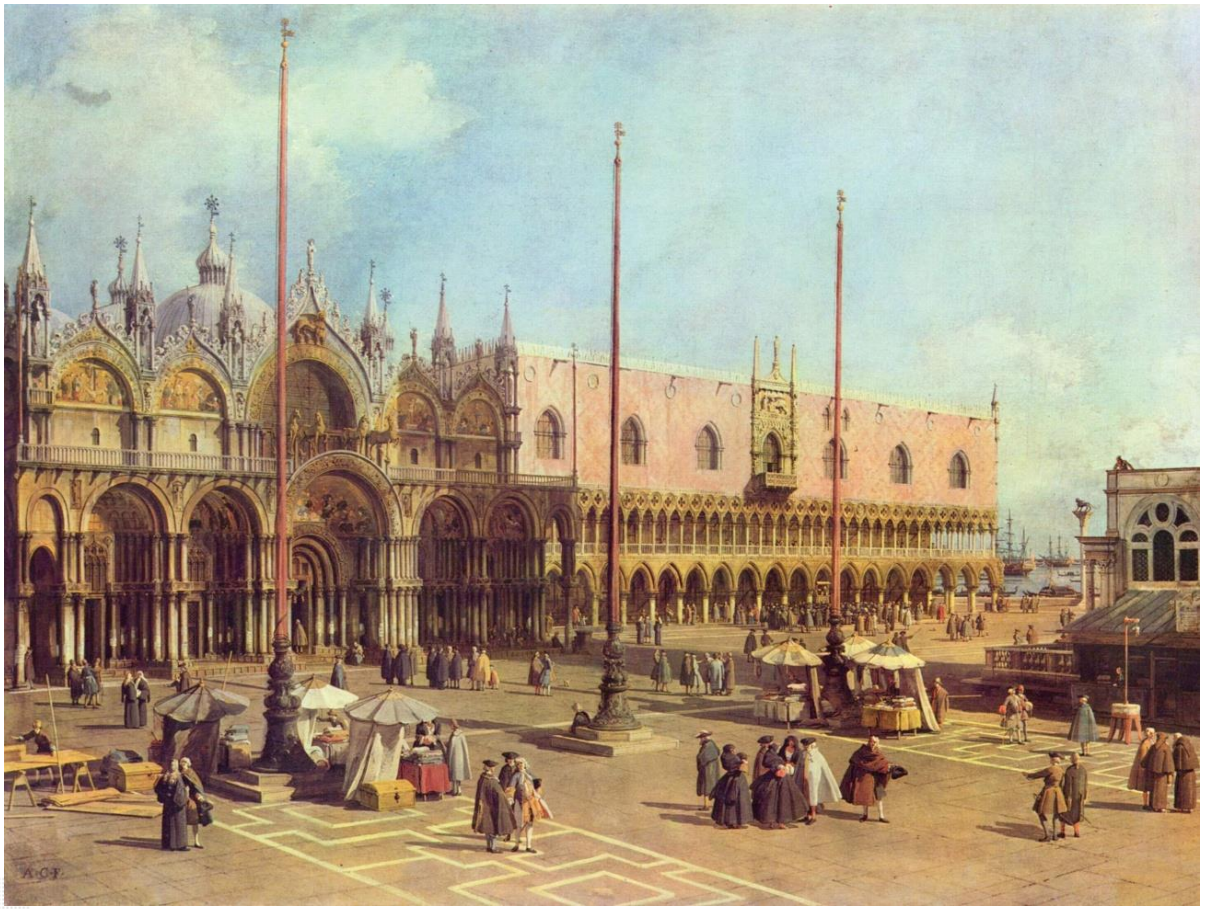
Análise Scale-Space

- O sistema visual humano é um *dispositivo de amostragem multi-escala*
- A retina contém *campos receptivos*; grupos de receptores agrupados de forma que eles criam um conjunto de aberturas de tamanho variável.



Análise Scale-Space

- Nós já vemos o mundo em diversas escalas!

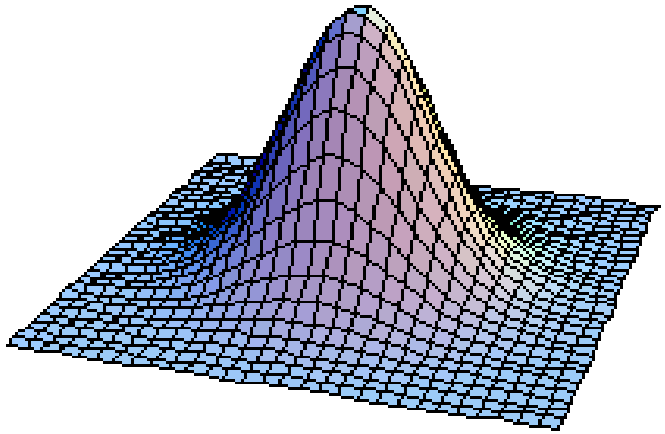


La Piazza San Marco - Veneza (Canaletto)

Análise Scale-Space

- Implementação prática
  - Convoluir a imagem com filtros Gaussianos

$$G(x,\sigma) = \frac{1}{(2\pi\sigma^2)^{D/2}} e^{-\frac{x_1^2 + \ldots + x_D^2}{2\sigma^2}}$$



Análise Scale-Space

- Imagem original

Análise Scale-Space

- Filtro Gaussiano, $\sigma = 1,7$

Análise Scale-Space

- Filtro Gaussiano, $\sigma = 2,5$

Análise Scale-Space

- Filtro Gaussiano, $\sigma = 5$

Análise Scale-Space

- Filtro Gaussiano, $\sigma = 1,7$

Análise Scale-Space

- Filtro Gaussiano, $\sigma = 2,5$

Análise Scale-Space

- Filtro Gaussiano, $\sigma = 5$

Análise Scale-Space

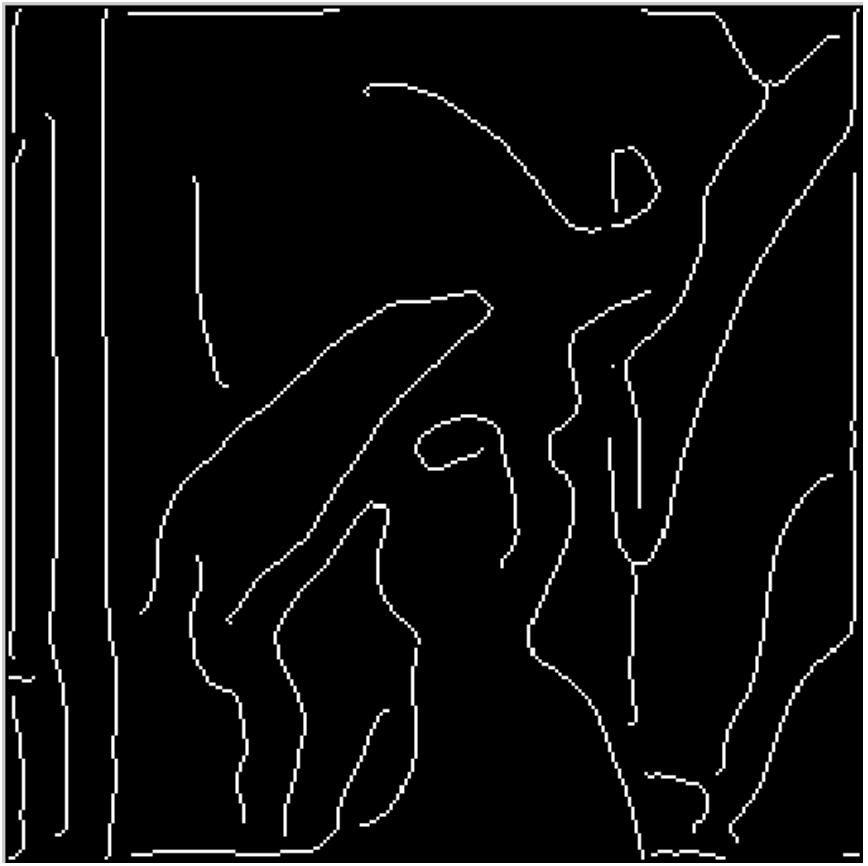
- Filtro Gaussiano, $\sigma = 1,7$

Análise Scale-Space

- Filtro Gaussiano, $\sigma = 2,5$

Análise Scale-Space

- Filtro Gaussiano, $\sigma = 5$



Análise Scale-Space

- Junção (soma) das 3 imagens

Há bordas que só aparecem em um nível de resolução.

Segmentação de Regiões

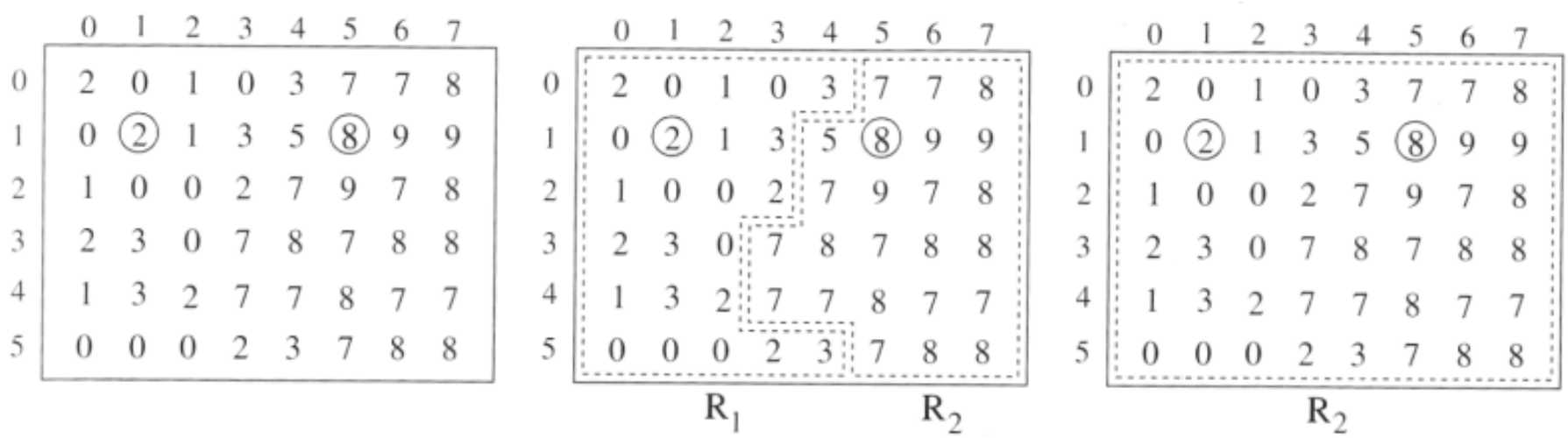
Carlos Alexandre Mello

Segmentação

- Métodos de Crescimento de Regiões
  - Agrega em regiões pixels com propriedades similares
  - Inicia com um conjunto de pixels (sementes) e, a partir deles, começa a crescer as regiões anexando a cada semente outros pixels que possuam propriedades similares
  - Os pixels sementes podem ser escolhidos de forma aleatória, determinística ou selecionados pelo usuário

Segmentação

- Métodos de Crescimento de Regiões



  - Critério (exemplo): $|f(x, y) - f(r, s)| \leq T$

Segmentação

- Métodos *Split-and-Merge* (Divisão e União)
  - Divide a imagem em regiões
    - *split*
  - Agrupa regiões com algum grau de "similaridade"
    - *merge*
  - Problemas ("só" dois):
    - Como dividir?
    - Como agrupar?

Segmentação

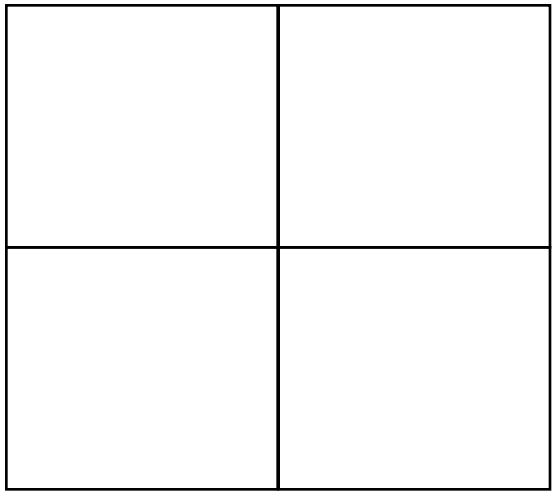
- Métodos de Divisão de Regiões
  - Quadtree
  - Watershed
  - Fluxo de água

Segmentação

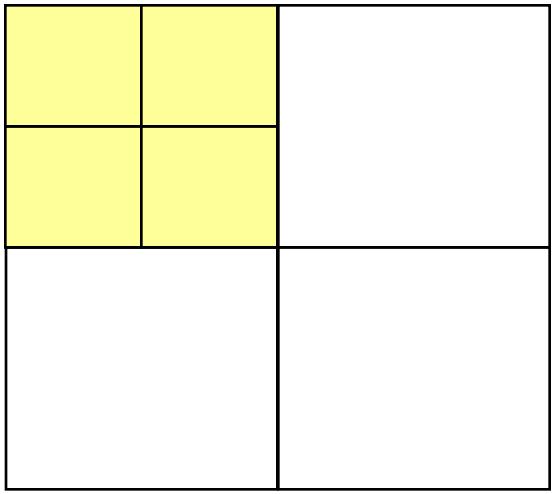
- Métodos de Divisão de Regiões
  - Algoritmo de decomposição Quadtree
    - O algoritmo divide a imagem em blocos
    - Cada bloco é então testado para um dado critério de homogeneidade
      - Se o bloco atende o critério, ele não é mais dividido
      - Se não atende, ele é subdividido e cada parte é testada sob o mesmo critério

Segmentação

- Decomposição Quadtree



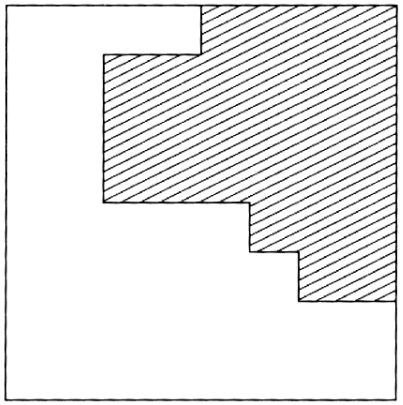
•1ª Divisão da imagem
•Cada bloco é analisado



•Os blocos que não atendem o critério são divididos de novo

Segmentação

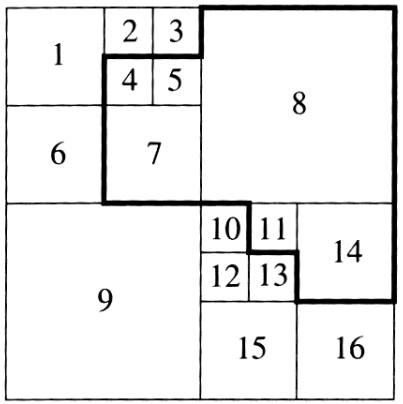
- Decomposição Quadtree



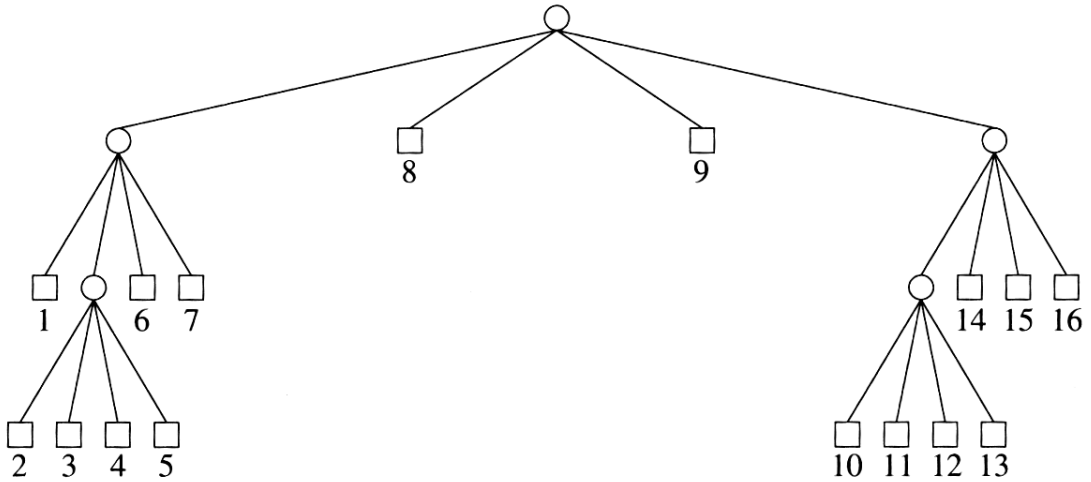
(a)

0	0	0	0	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	1	1	1	1	1	1
0	0	0	0	0	1	1	1
0	0	0	0	0	0	1	1
0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0

(b)



(c)



(d)

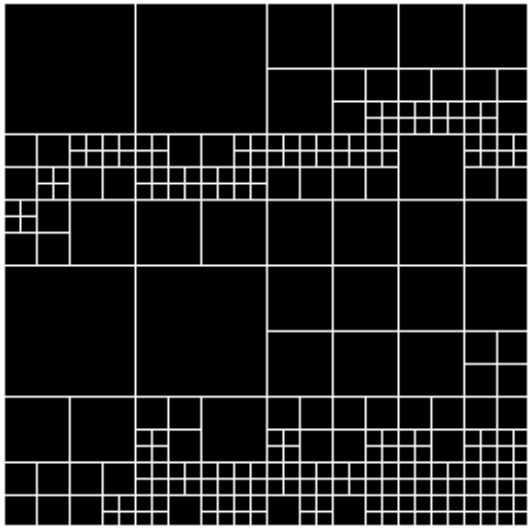
Segmentação

- Decomposição Quadtree
  - Exemplo de critério:
    - (a maior cor do bloco – a menor cor do bloco) <= 20
  - No MatLab
    - Função qtdecomp
    - Divide a imagem em quadrados de tamanhos: 512, 256, 128, 64, 32, 16, 8, 4, 2, e 1
    - Critério como estabelecido acima
    - Restrição: tamanho da imagem deve ser proporcional às dimensões possíveis
    - Exemplo: qtdecomp (imagem, 0.2)
      - 0.2 é o ponto de corte normalizado por 255

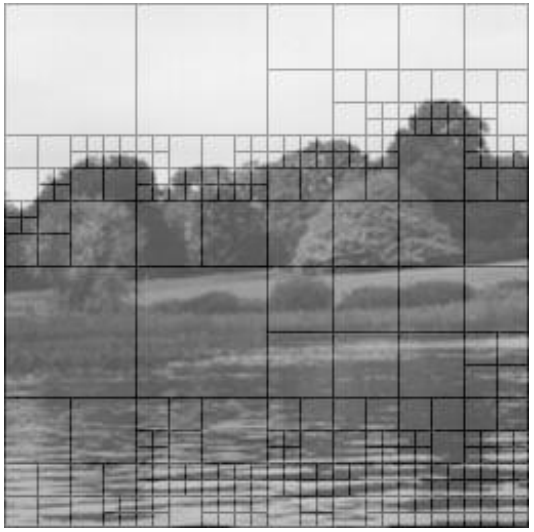
Segmentação

- Decomposição Quadtree
  - Exemplo:

Imagem original



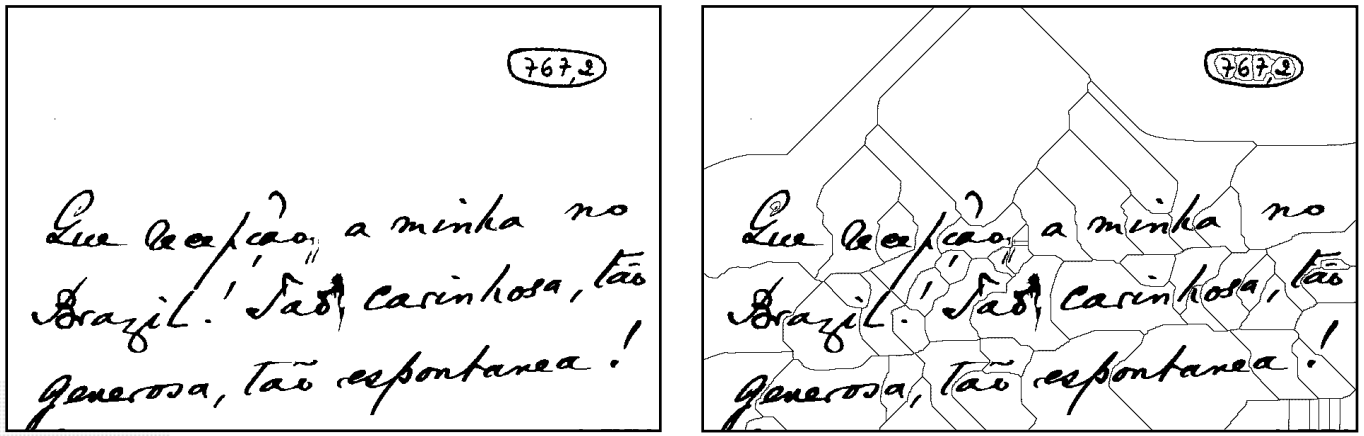
Decomposição



Visualização das regiões

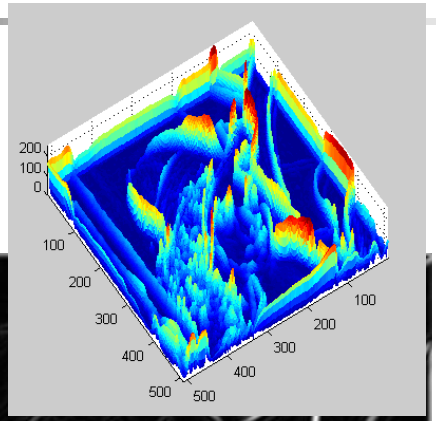
Segmentação

- Da Morfologia Matemática
  - Watershed
    - Provoca sobre-segmentação
    - Como vimos antes, as regiões de tinta são consideradas como vales que são inundados
    - Aplicado a imagem de gradiente

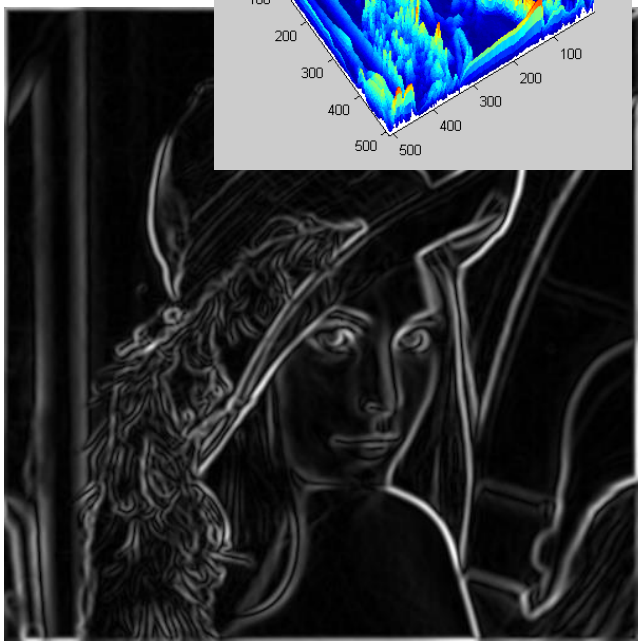


Segmentação

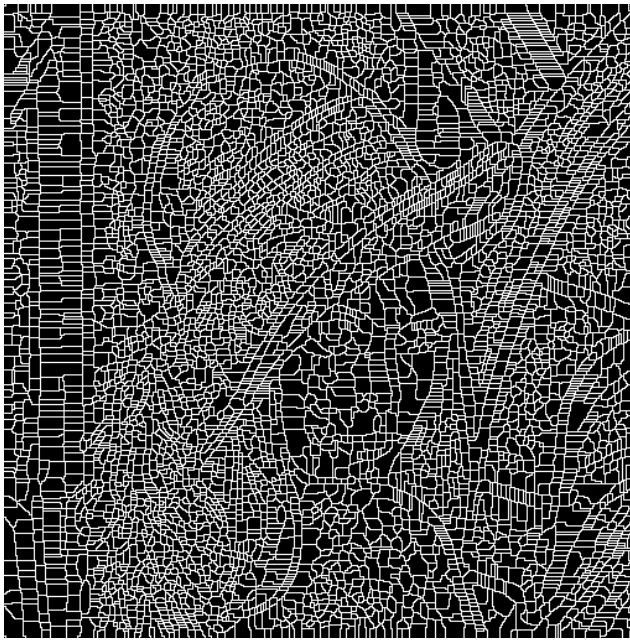
- Watershed



Image



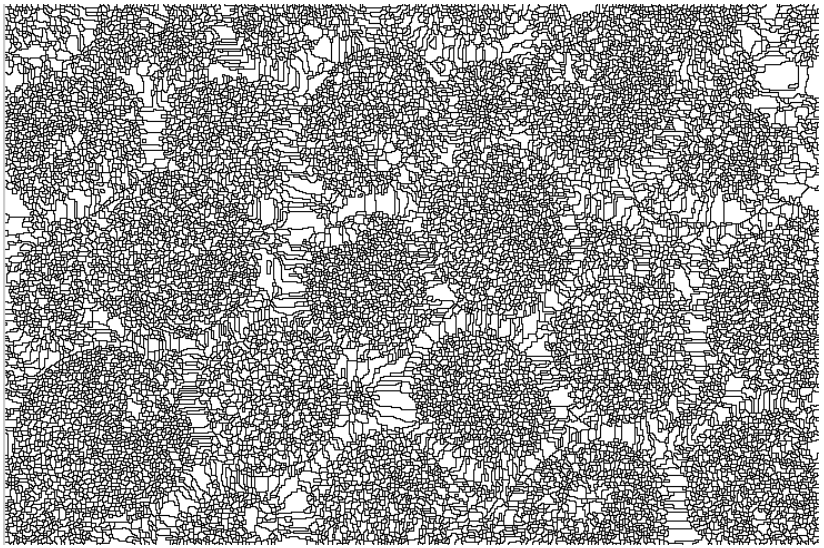
Gradient



Watershed boundaries

Segmentação

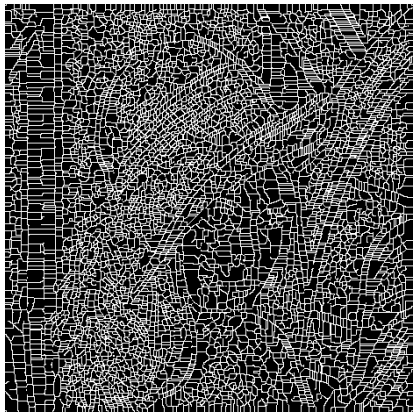
- Watershed



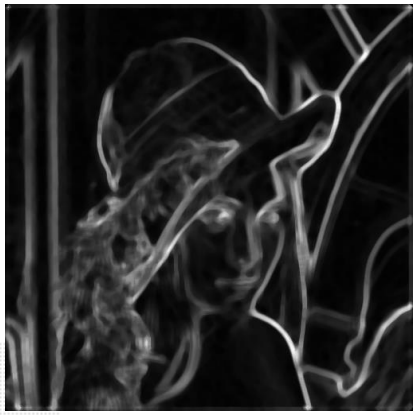
15.294 regiões encontradas!!!

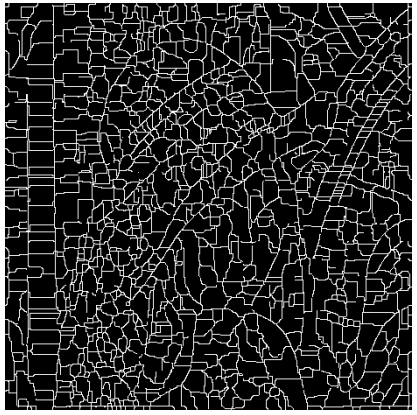
Segmentação

- Watershed: sobre-segmentação pode ser evitada com filtros (gaussiano, por exemplo)



Sem filtro.....





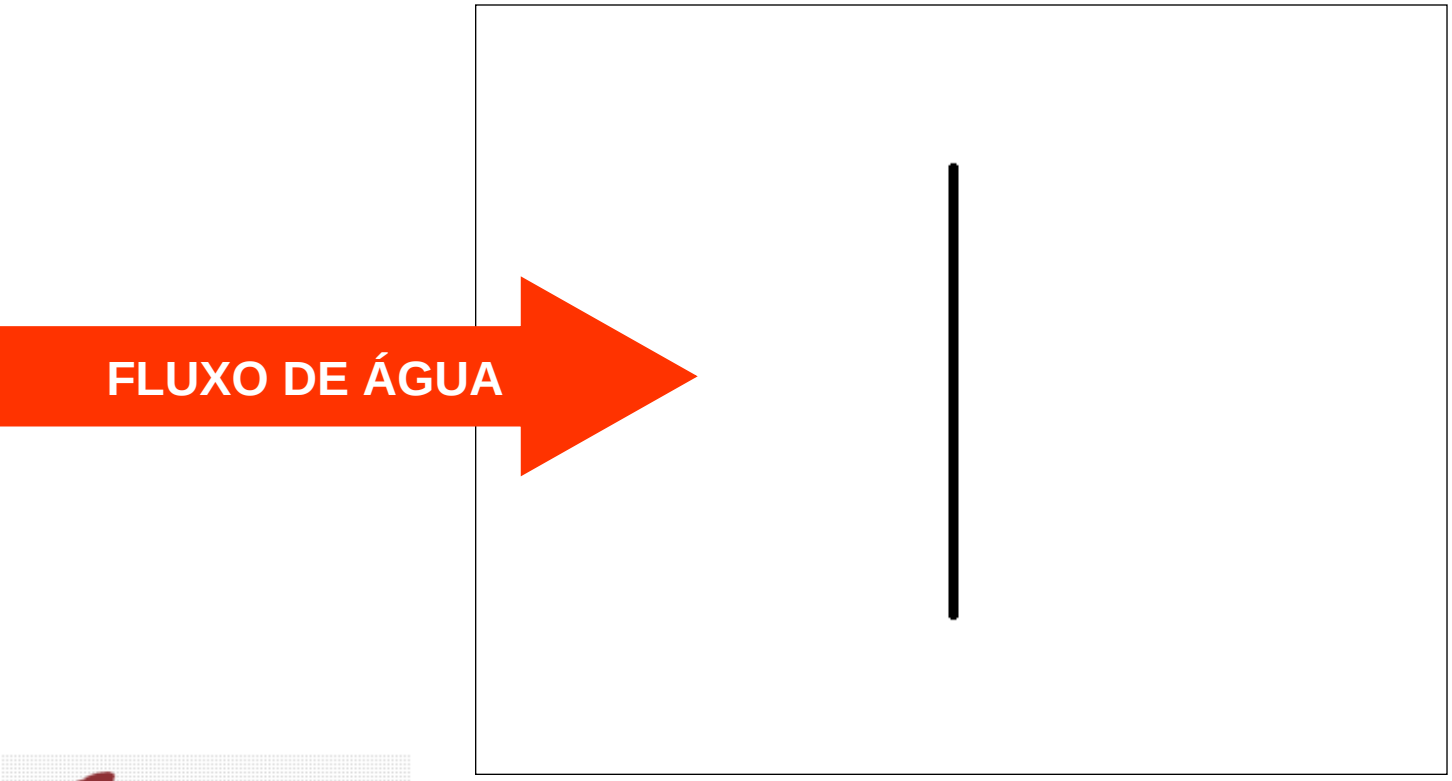
Com filtro.....

Segmentação

- Fluxo hipotético de água (Water Flow)
  - Definido por Basu *et al.*
  - Aplicado a segmentação de imagens de documentos
  - Sugere que a imagem de um documento é inundada por um fluxo imaginário de água pela esquerda e pela direita
  - A tinta funciona como uma barreira à água

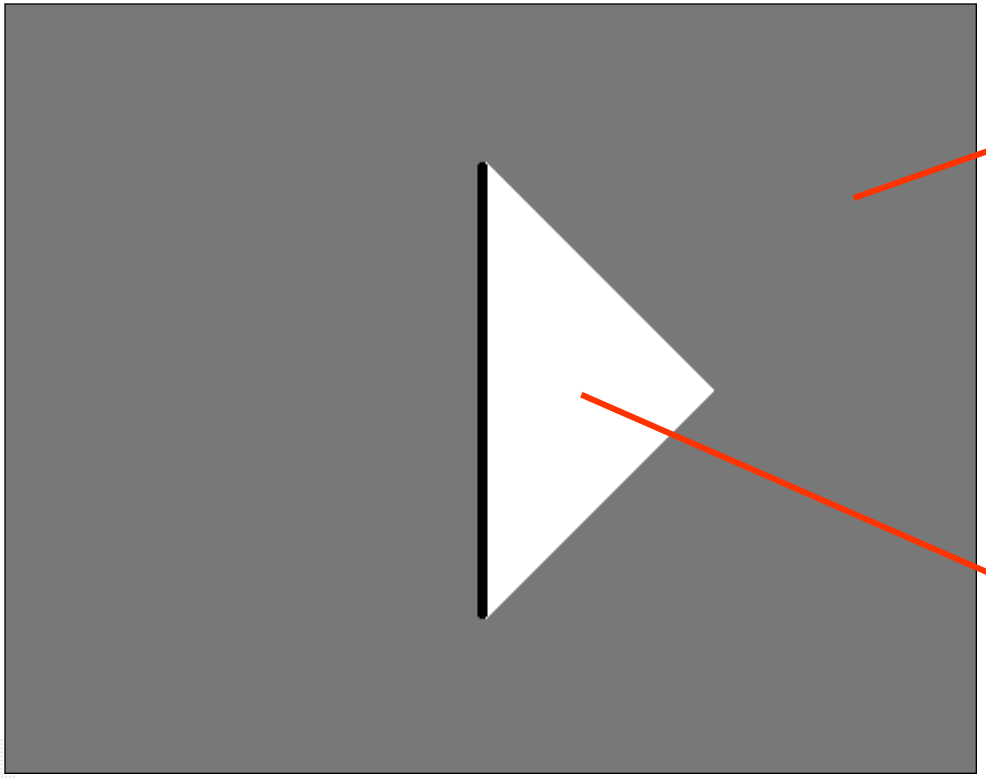
Segmentação

- Segmentação de Linhas por Fluxo hipotético de água



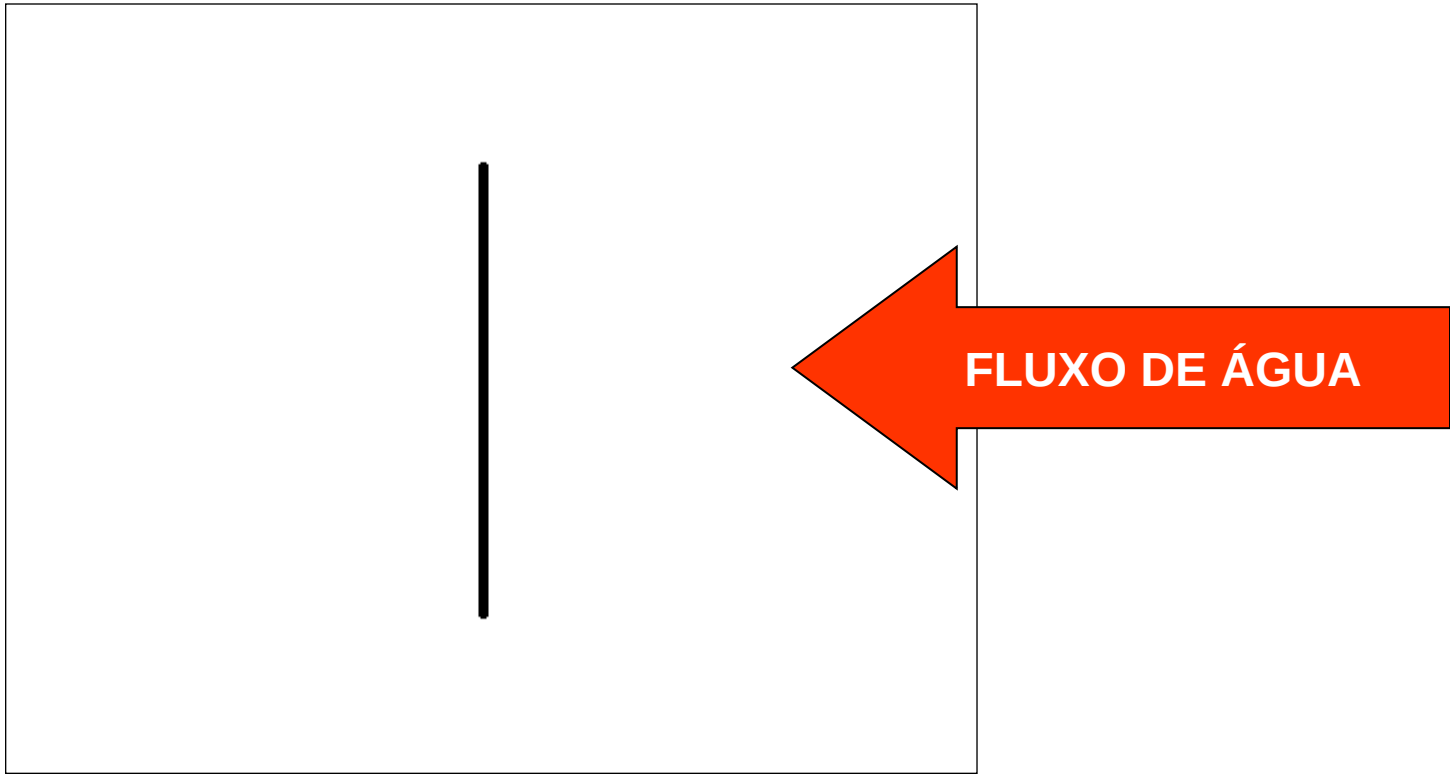
Segmentação

- Segmentação de Linhas por Fluxo hipotético de água



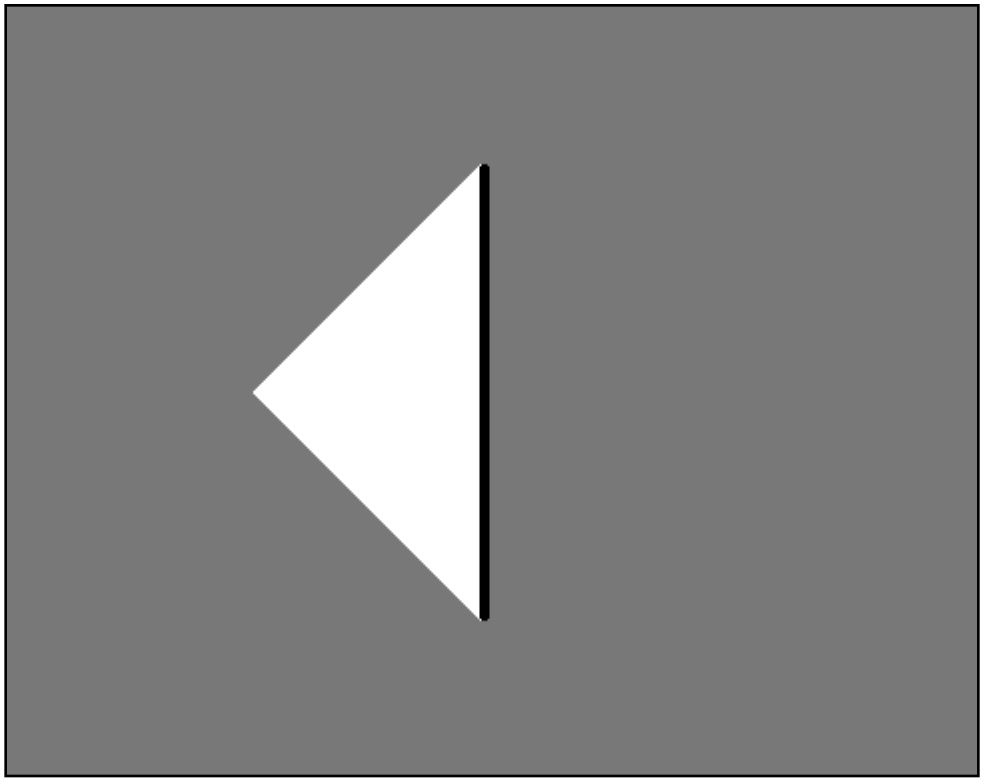
Segmentação

- Segmentação de Linhas por Fluxo hipotético de água



Segmentação

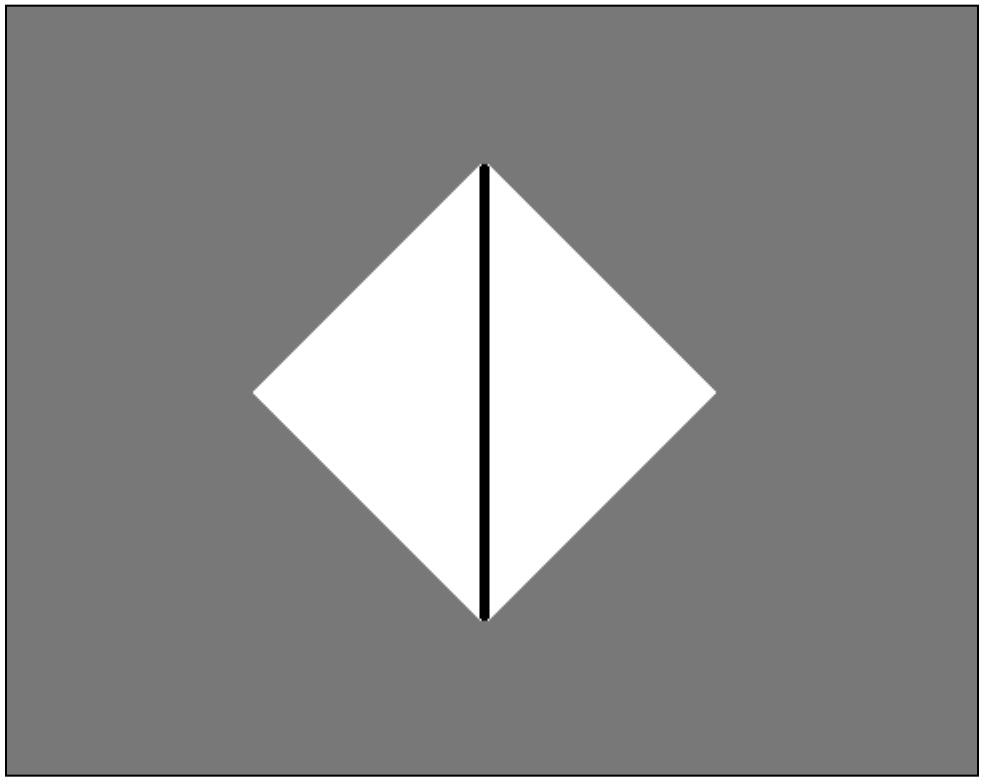
- Segmentação de Linhas por Fluxo hipotético de água



Segmentação

- Segmentação de Linhas por Fluxo hipotético de água

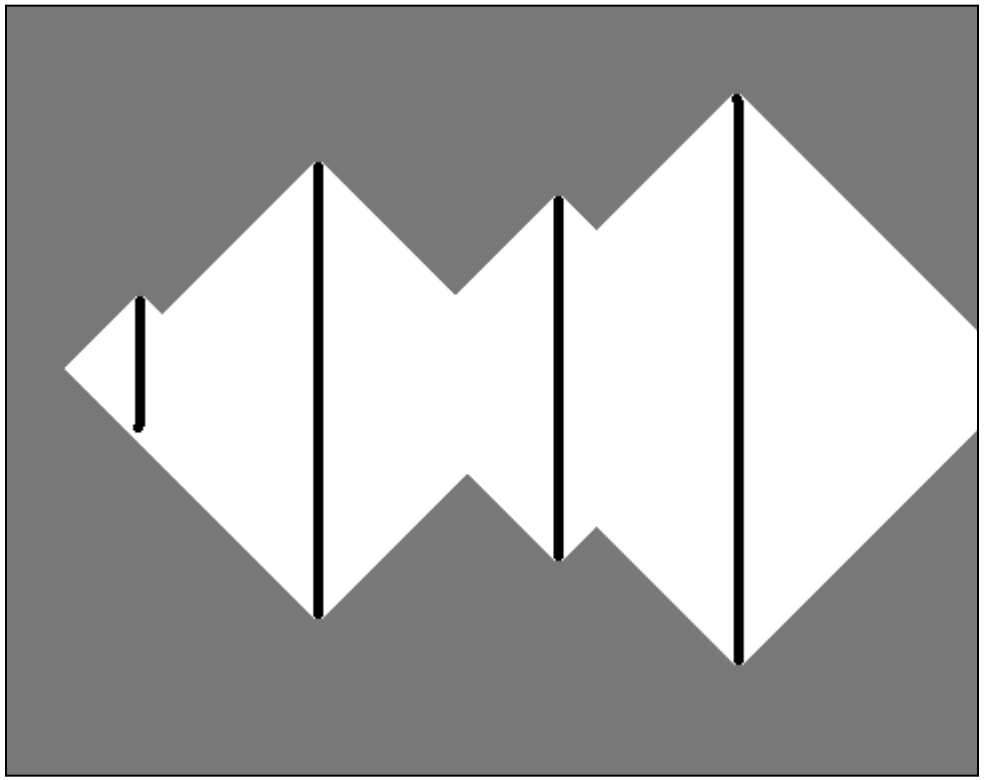
Juntando os dois fluxos...



Segmentação

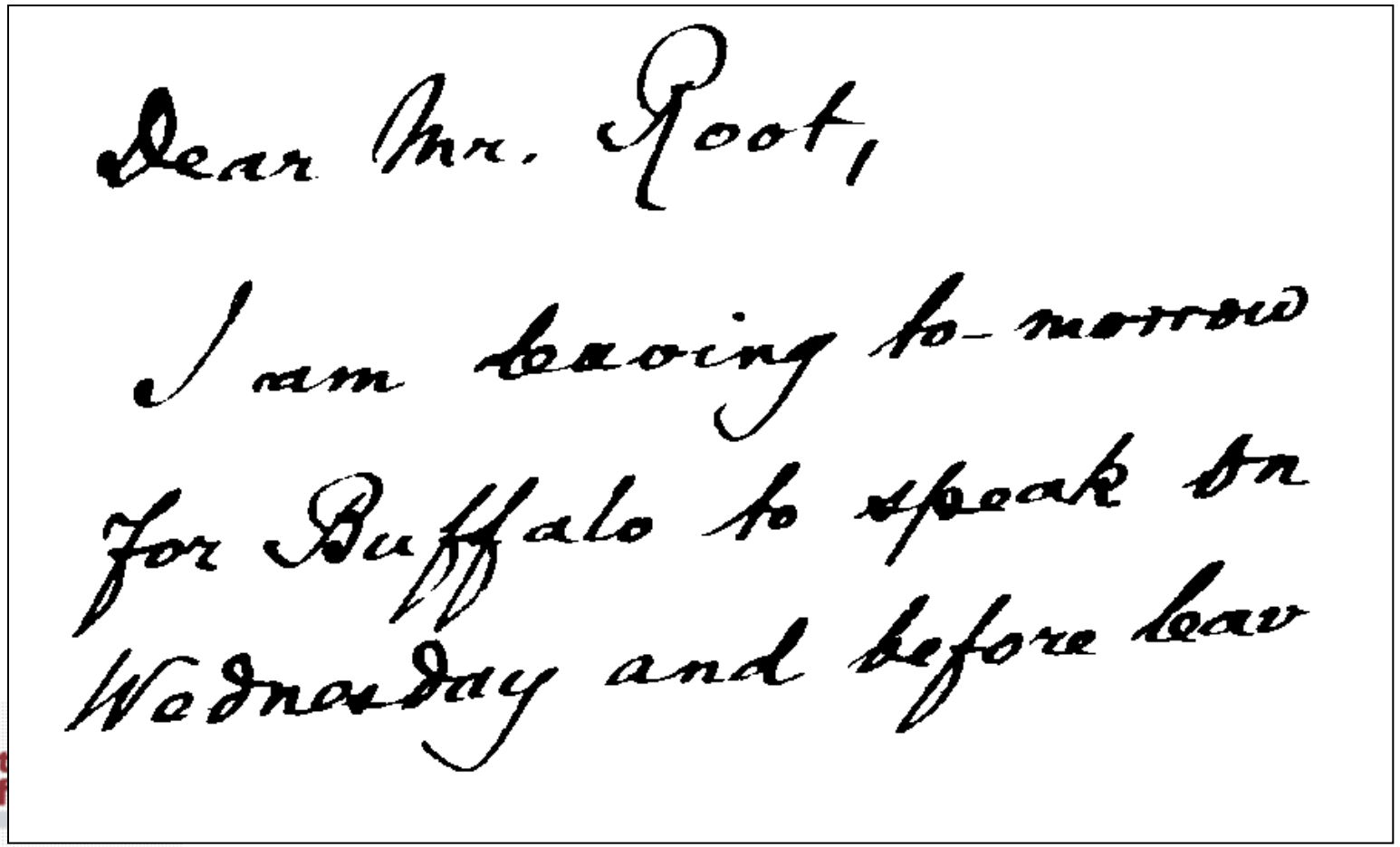
- Segmentação de Linhas por Fluxo hipotético de água

Colocando mais barreiras



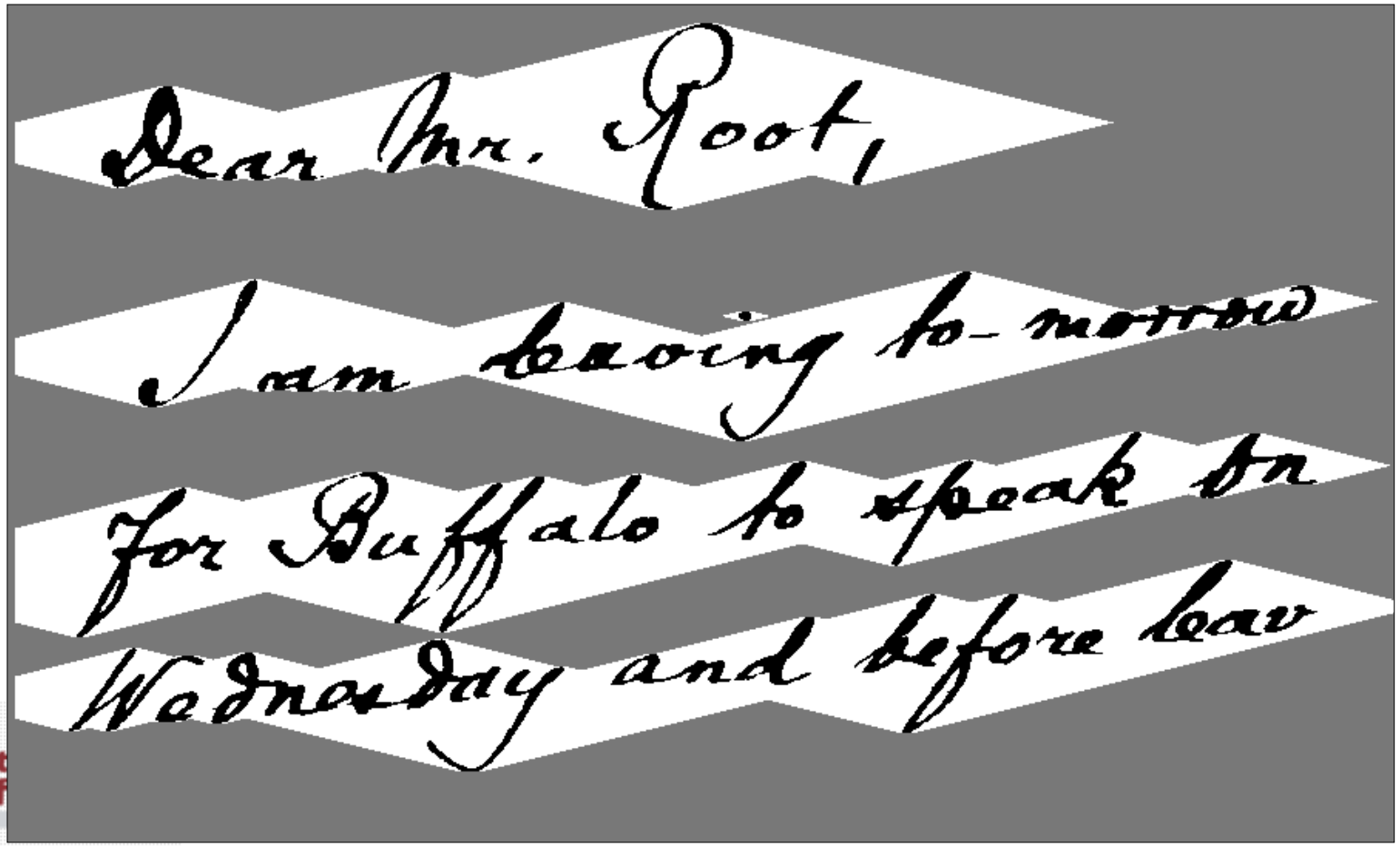
Segmentação

- Segmentação de Linhas por Fluxo hipotético de água

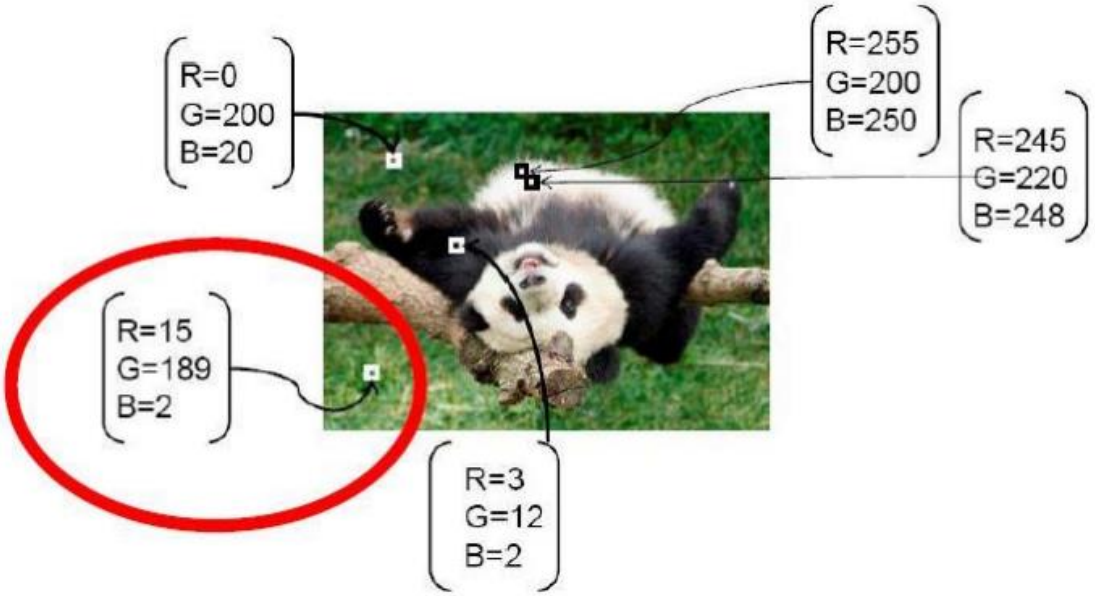


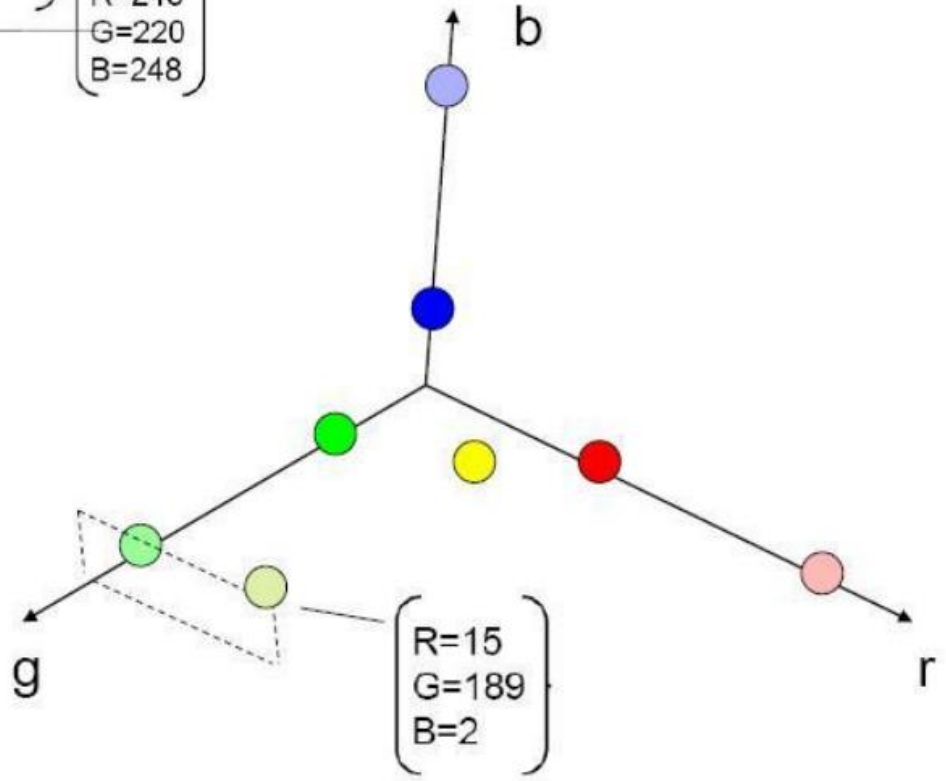
Segmentação

- Segmentação de Linhas por Fluxo hipotético de água

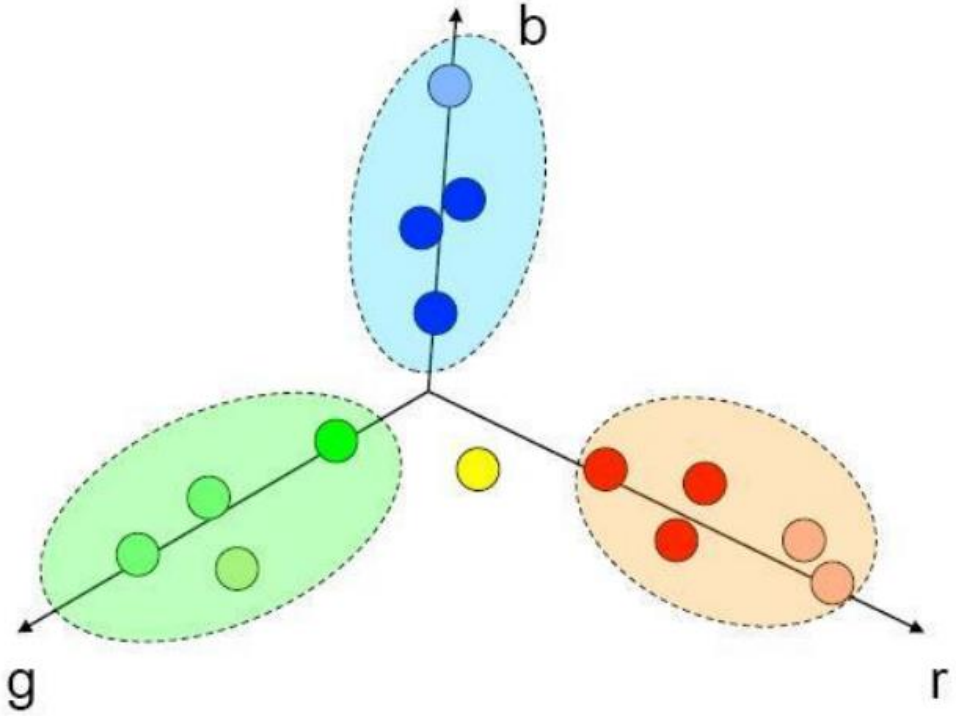


Segmentação por Agrupamento





Segmentação por Agrupamento

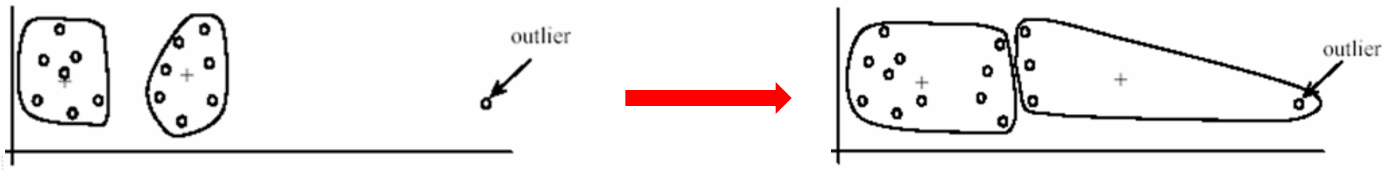


Vantagens:

- Fácil de implementar
- Simples e rápido

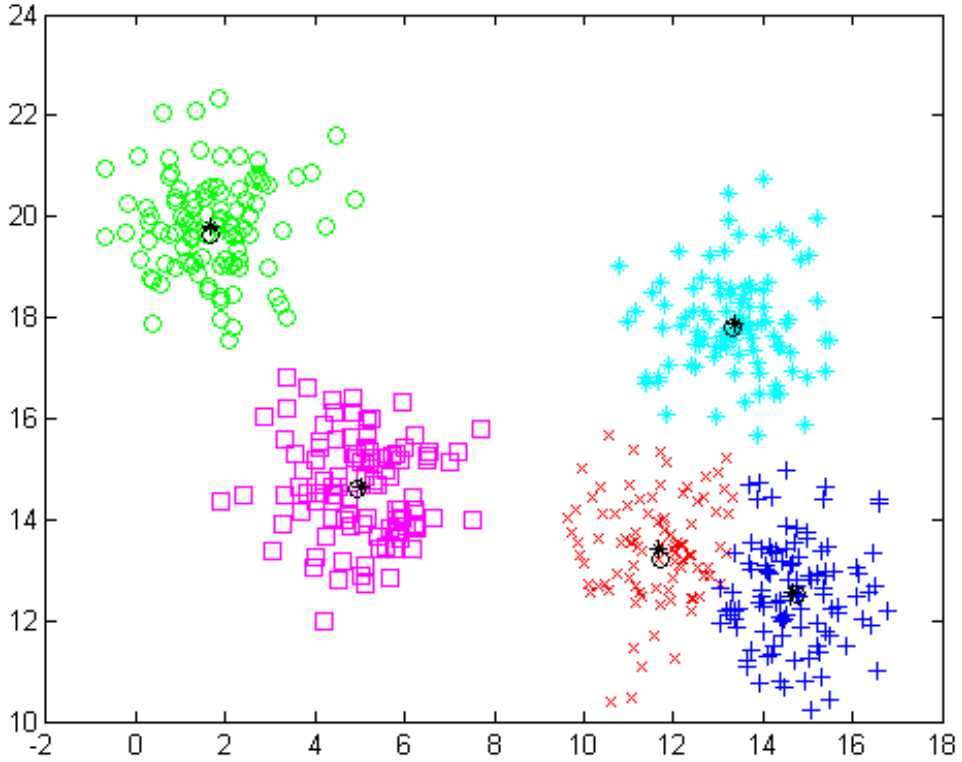
Desvantagens

- Precisa do K
- Sensível a *outliers*



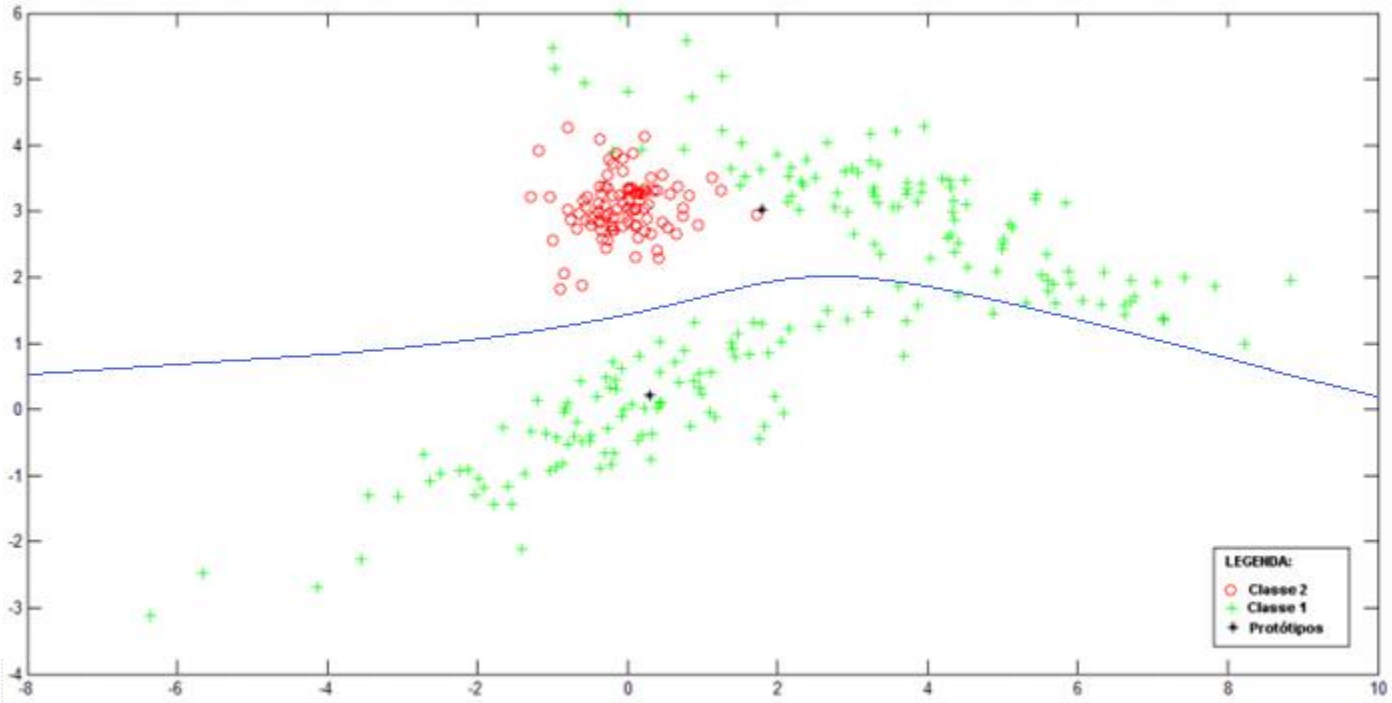
Segmentação – K-Means

- **Número de sementes** (centros) indica o número de **regiões**.

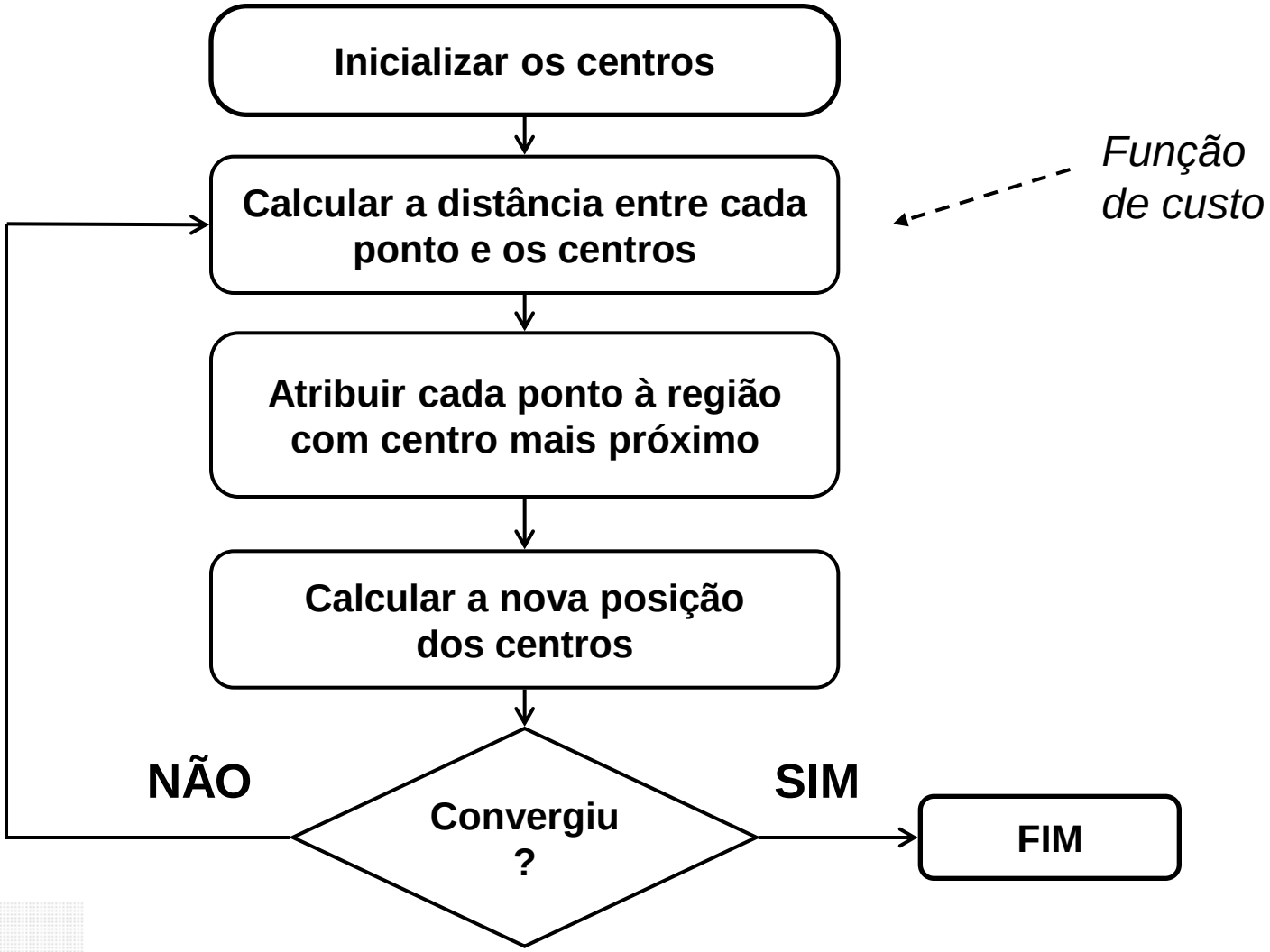


Segmentação – K-Means

- Problemas:
  - **Quantos** centros escolher (**regiões/clusters**)
  - A **posição** dos centros iniciais (**convergência**)

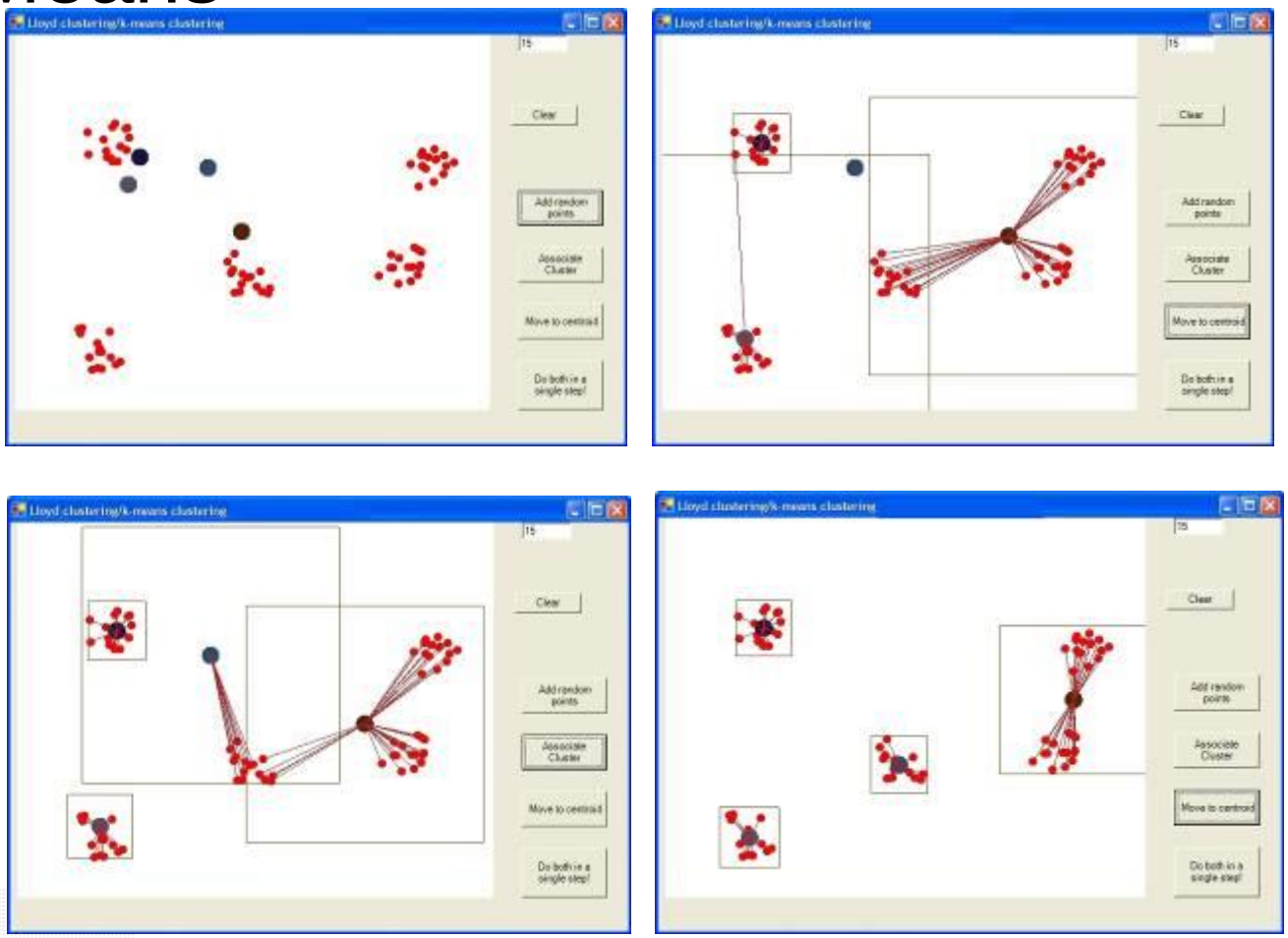


Segmentação – K-Means



Segmentação – K-Means

- K-Means



Segmentação – K-Means

Original

K = 2

K = 8

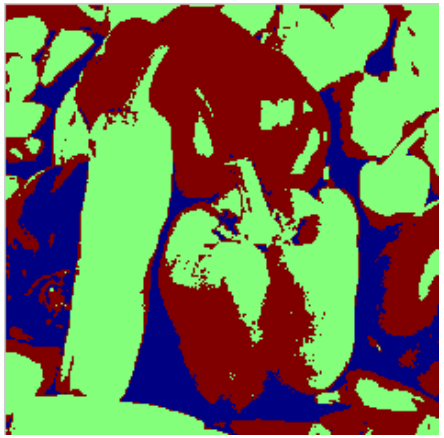
Segmentação – K-Means

K = 11

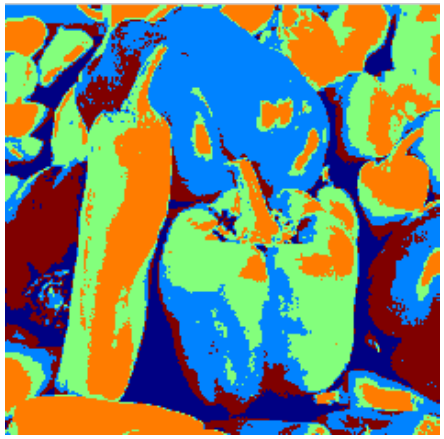
K = 14

K = 15

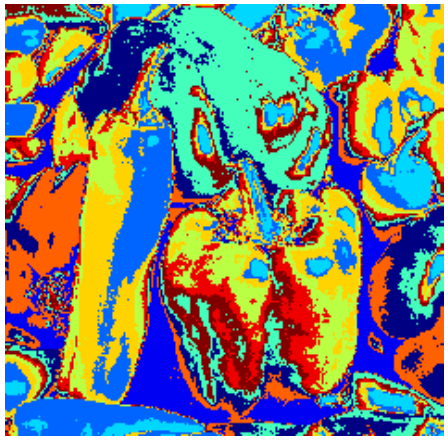
Segmentação – K-Means



k = 3



k = 4



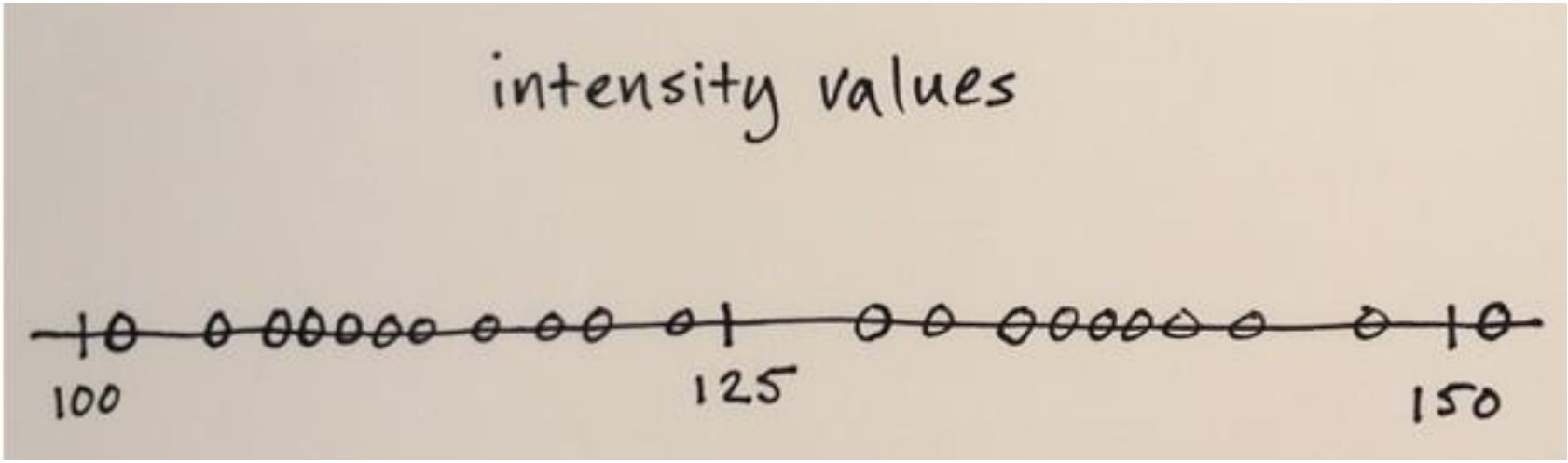
k = 10

Segmentação – Mean Shift

- Overview
  - Para cada ponto, o *mean shift* define uma janela ao seu redor e calcula a média. Então move o centro da janela para essa média e repete o processo até convergir
  - Mean shift é um algoritmo não-paramétrico (mais ou menos) e iterativo
  - Alto custo computacional

Segmentação – Mean Shift

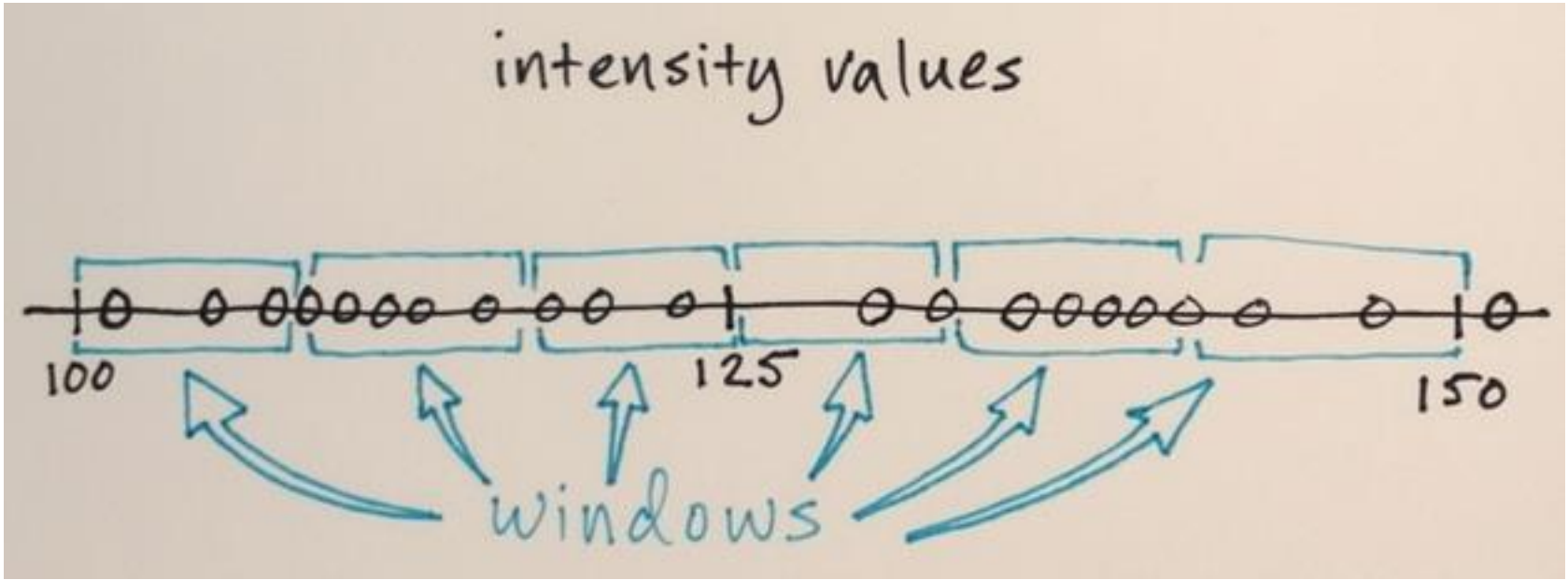
- Algoritmo
  - A imagem é convertida em um espaço de características



http://stackoverflow.com/questions/4831813/image-segmentation-using-mean-shift-explained

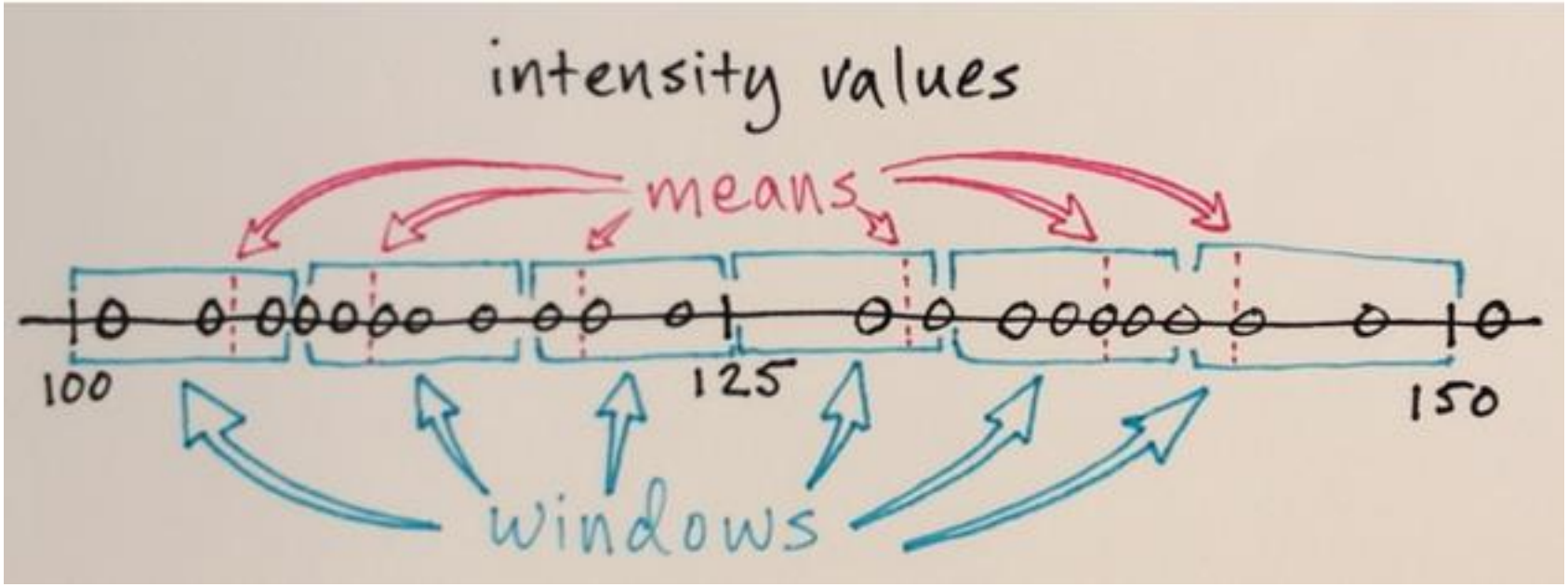
Segmentação – Mean Shift

- Algoritmo
  - Janelas de busca são distribuídas ao longo do espaço



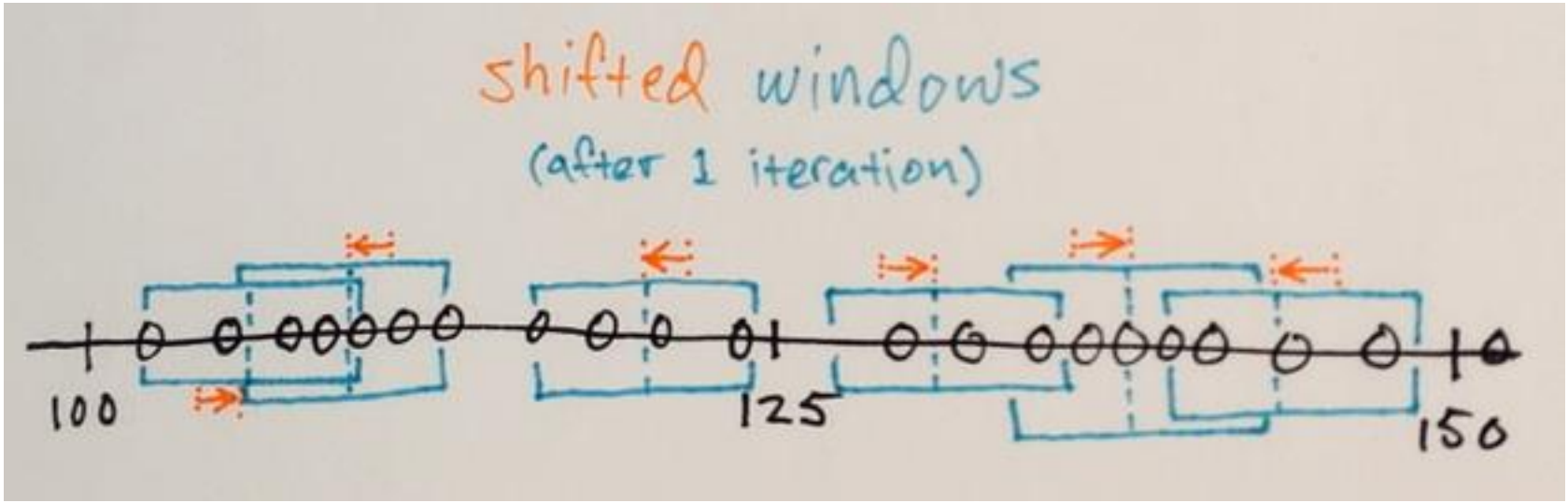
Segmentação – Mean Shift

- Algoritmo
  - 1. A média (*mean*) dentro de cada janela é calculada



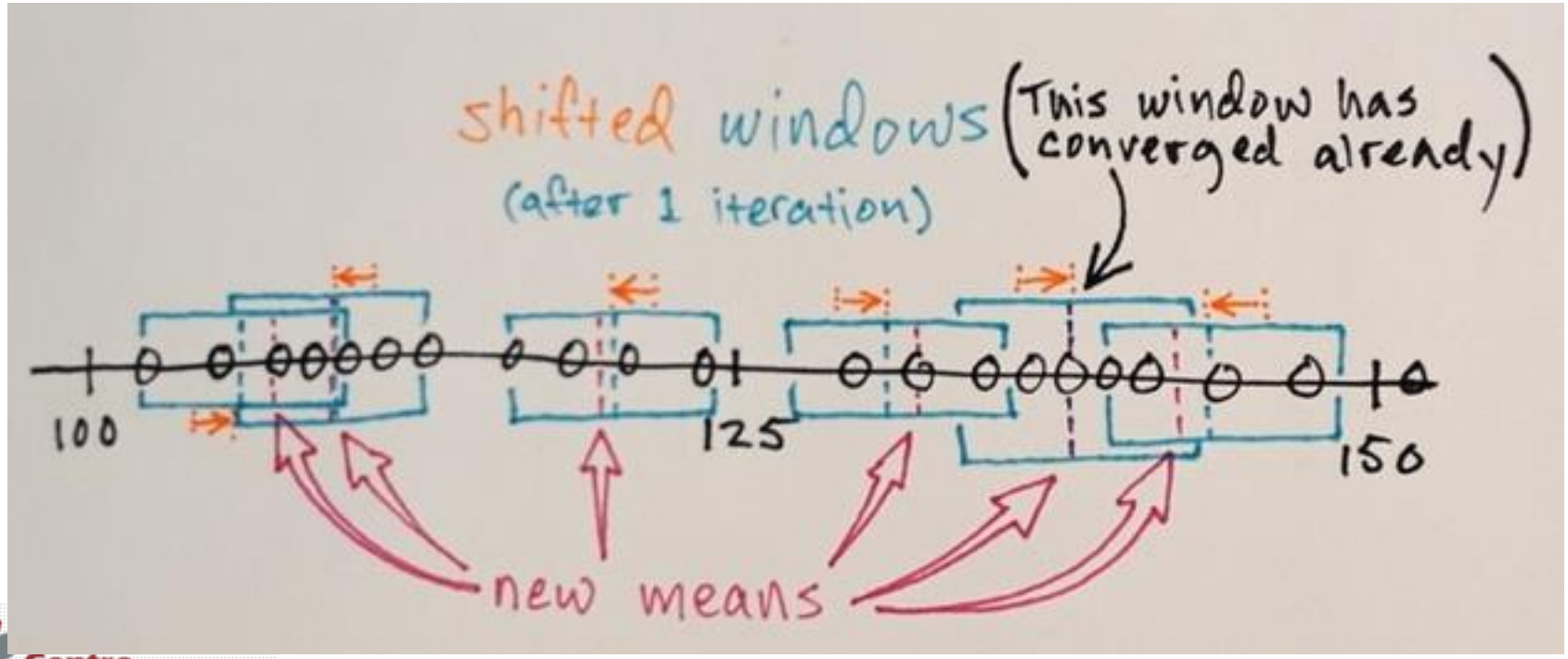
Segmentação – Mean Shift

- Algoritmo
  - 2. As janelas são deslocadas (*shifted*) para localizações iguais às médias anteriores



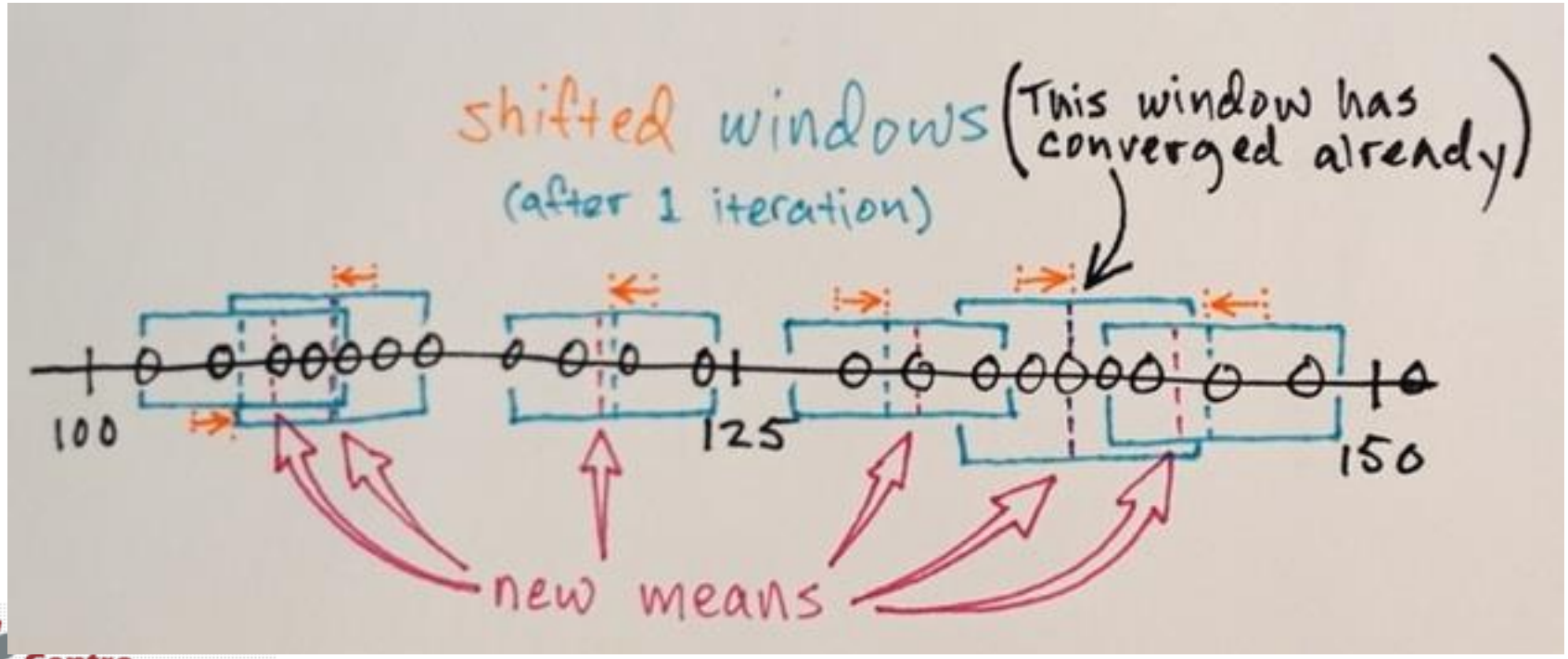
Segmentação – Mean Shift

- Algoritmo
  - 3. Repete os passos 1 e 2



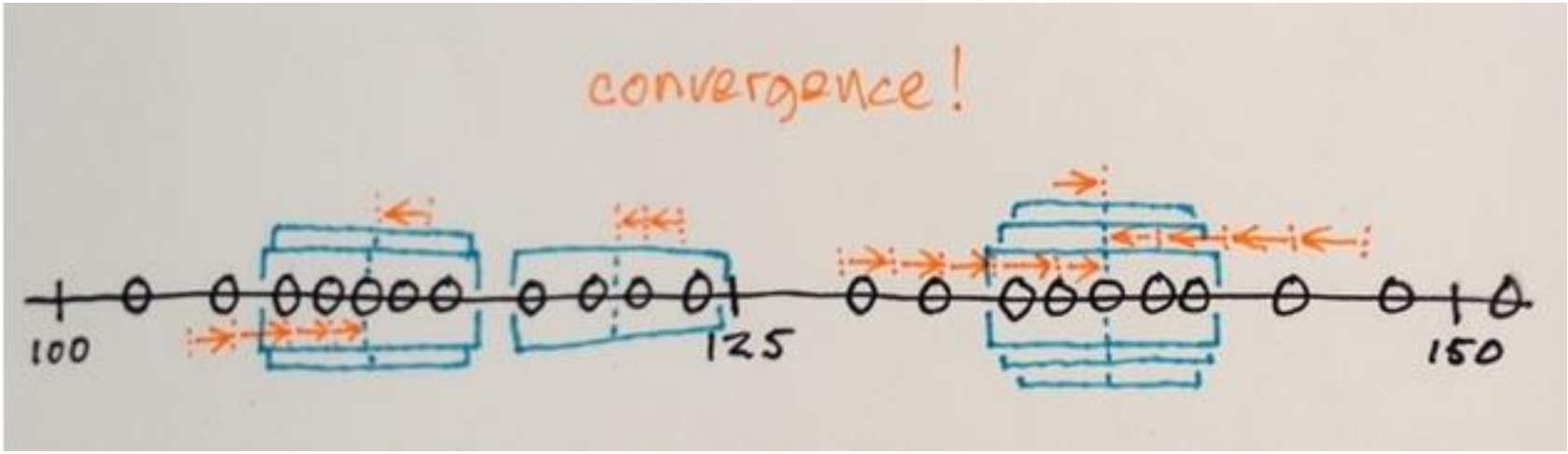
Segmentação – Mean Shift

- Algoritmo
  - 3. Repete os passos 1 e 2 até convergência, *i.e.*, as janelas estão na posição final



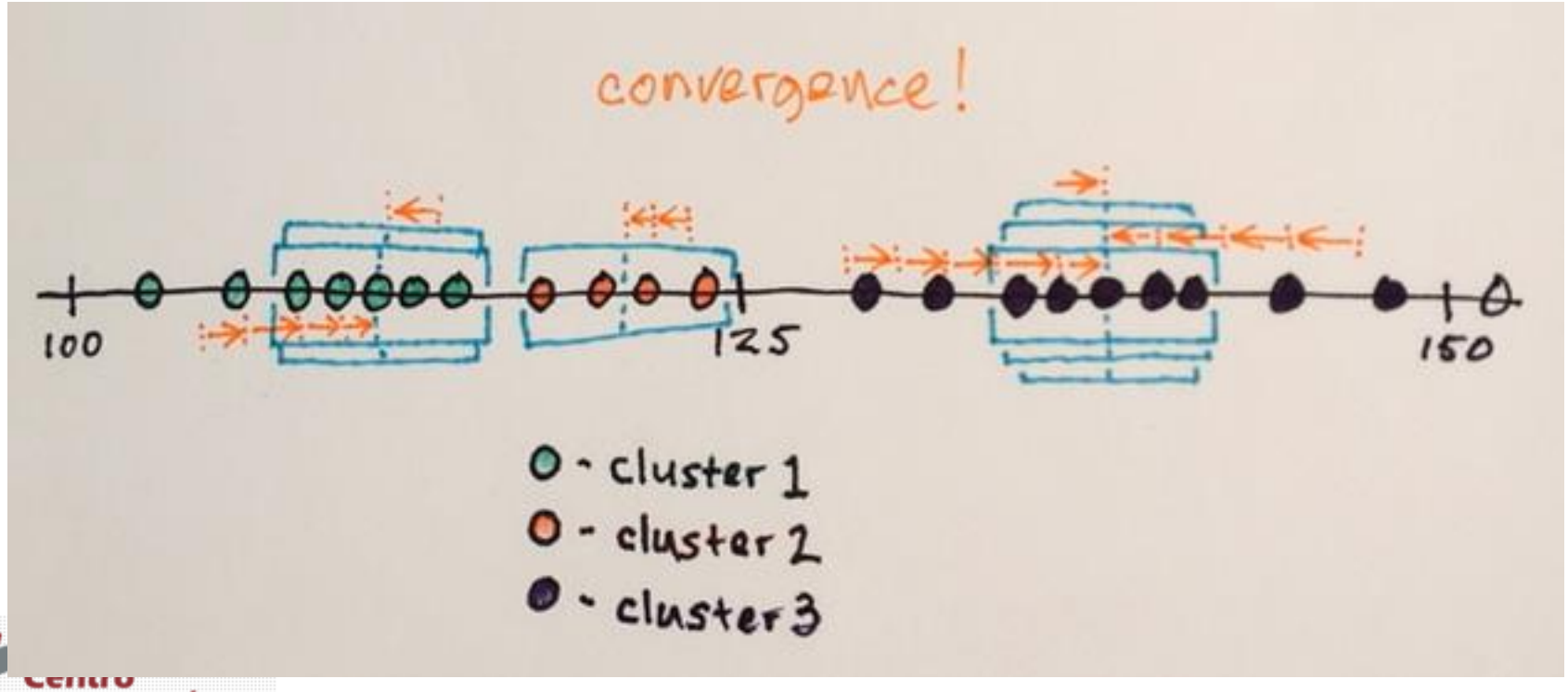
Segmentação – Mean Shift

- Algoritmo
  - 4. As janelas que estão na mesma localidade são unificadas



Segmentação – Mean Shift

- Algoritmo
  - 5. Os dados são agrupados à medida que as janelas são percorridas

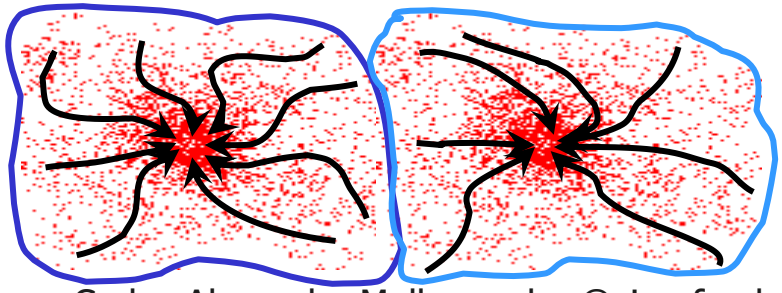


Segmentação – Mean Shift

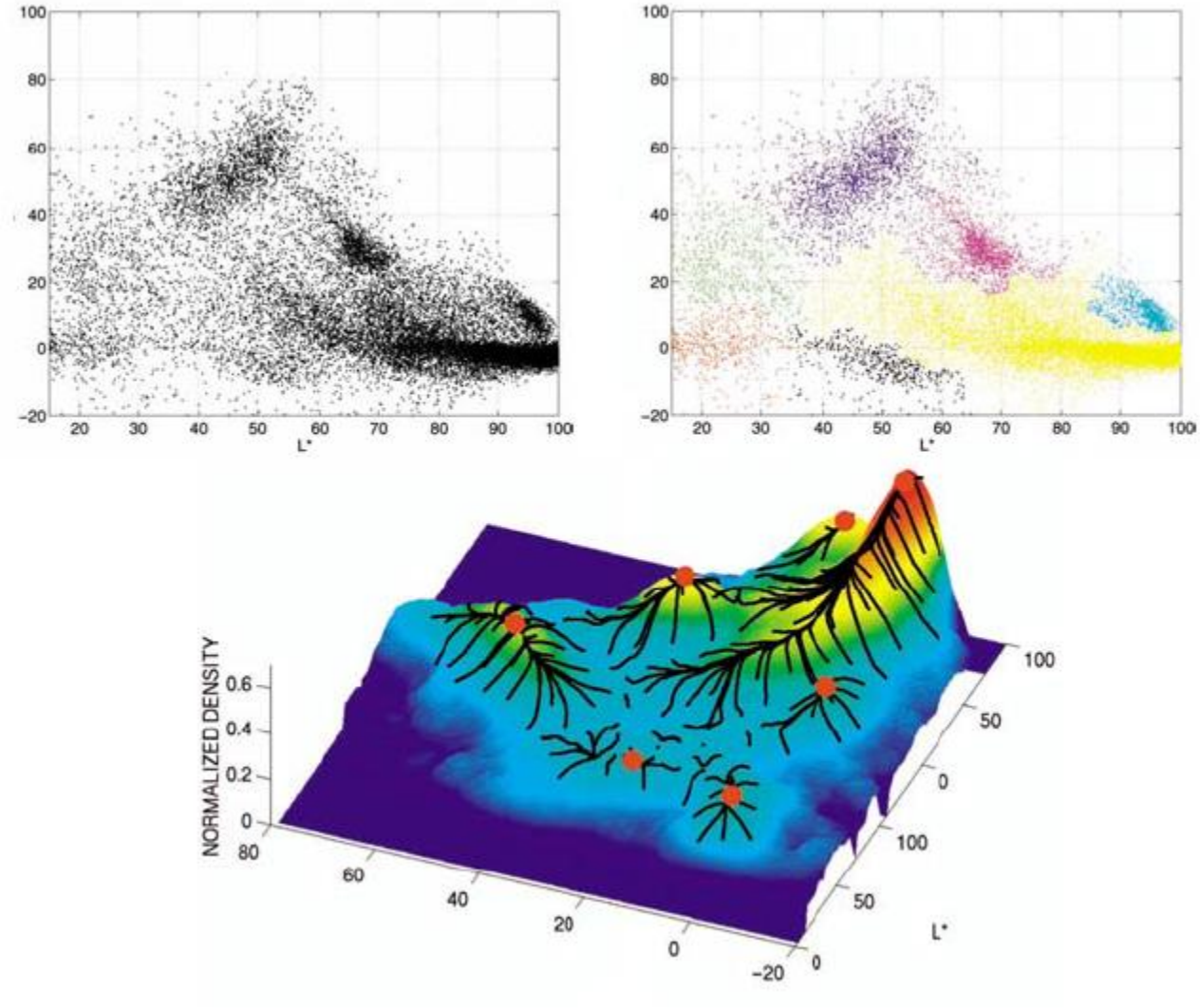
- Exemplos

Segmentação – Mean Shift

- Método: de maneira mais formal, o algoritmo de mean shift funciona assim:
  - De forma oposta ao watershed, o mean shift busca por **picos** na distribuição de dados
  - A ideia seria de buscar picos na distribuição e identificar que pontos no espaço de características “escalam” esses picos, indicando que fazem parte da mesma região



Segmentação – Mean Shift



Segmentação – Mean Shift

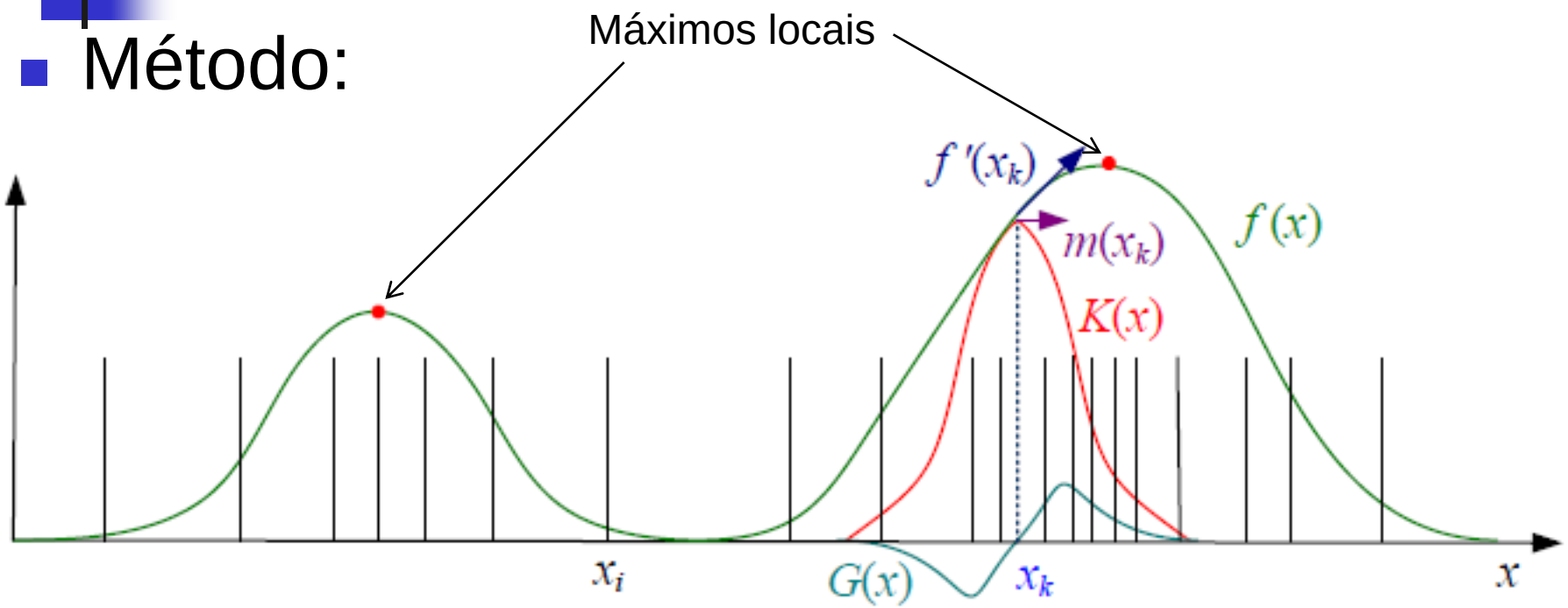
- Método:
  - O primeiro problema é como estimar a função de densidade de um conjunto de amostras
  - Primeiro, os dados são suavizados através de uma convolução com um filtro de *kernel* fixo com largura *h*:

$$f(x) = \sum_i K(x - x_i) = \sum_i k\left(\frac{\|x - x_i\|^2}{h^2}\right)$$

Em que x_i são as amostras de entrada e k(r) é a função kernel.

Segmentação – Mean Shift

- Método:



Visão geral da estimativa de densidade f(x), sua derivada e o mean shift:
f(x) é obtida pela convolução de x_i com a função de kernel K(x);
f‘(x) é obtida pela convolução de x_i com a derivada de f(x), G(x);
A estimativa de vetores de deslocamento locais ao redor de um ponto x_k resulta no vetor mean shitf $m(x_k)$.

Segmentação – Mean Shift

- Método:
  - Para diminuir o custo computacional, o mean shift trabalha com um valor aleatório para máximo local (um ponto aleatório x_i)
  - O gradiente é calculado indicando a direção da "escalada":

$$\nabla f(x) = \sum_i (x_i - x) G(x - x_i) = \sum_i (x_i - x) g\left(\frac{\|x - x_i\|^2}{h^2}\right)$$

  - que pode ser re-escrita como:

$$\nabla f(x) = \left[\sum_i G(x - x_i)\right] m(x) \quad \text{onde...} \quad m(x) = \frac{\sum_i x_i G(x - x_i)}{\sum_i G(x - x_i)} - x$$

Segmentação – Mean Shift

- Método:
  - Melhores resultados podem ser obtidos agrupando os domínios de cor e localidade
  - As coordenadas espaciais, $x_s = (x, y)$, que formam o domínio espacial, são concatenadas com a informação de cor, x_r, que forma o domínio de alcance (*range*)
  - O mean shift é então aplicado a esse espaço 5-dimensional (considerando 3 componentes de cores)
  - Como localização e cor têm diferentes escalas, elas têm seus kernels ajustados por h_s e h_r

Segmentação – Mean Shift

- Nova forma do kernel:

$$K(x_j) = k\left(\frac{\|x_r\|^2}{h_r^2}\right) k\left(\frac{\|x_s\|^2}{h_s^2}\right)$$

- Parâmetros
  - h_s: Parâmetro de resolução espacial
    - Afeta a suavização da imagem
    - Deve ser definido de acordo com o tamanho dos objetos
    - Quanto maior, mais demorada a segmentação
  - h_r: Parâmetro de resolução de alcance (*range*)
    - Afeta o número de segmentos
    - Deve ser baixo para baixo contraste
  - M (opcional): Tamanho do menor segmento

Segmentação – Mean Shift

- Exemplo:

original

$(h_s, h_r) = (8,4)$

Mesma suavização, mas com menos segmentos (h_r maior)

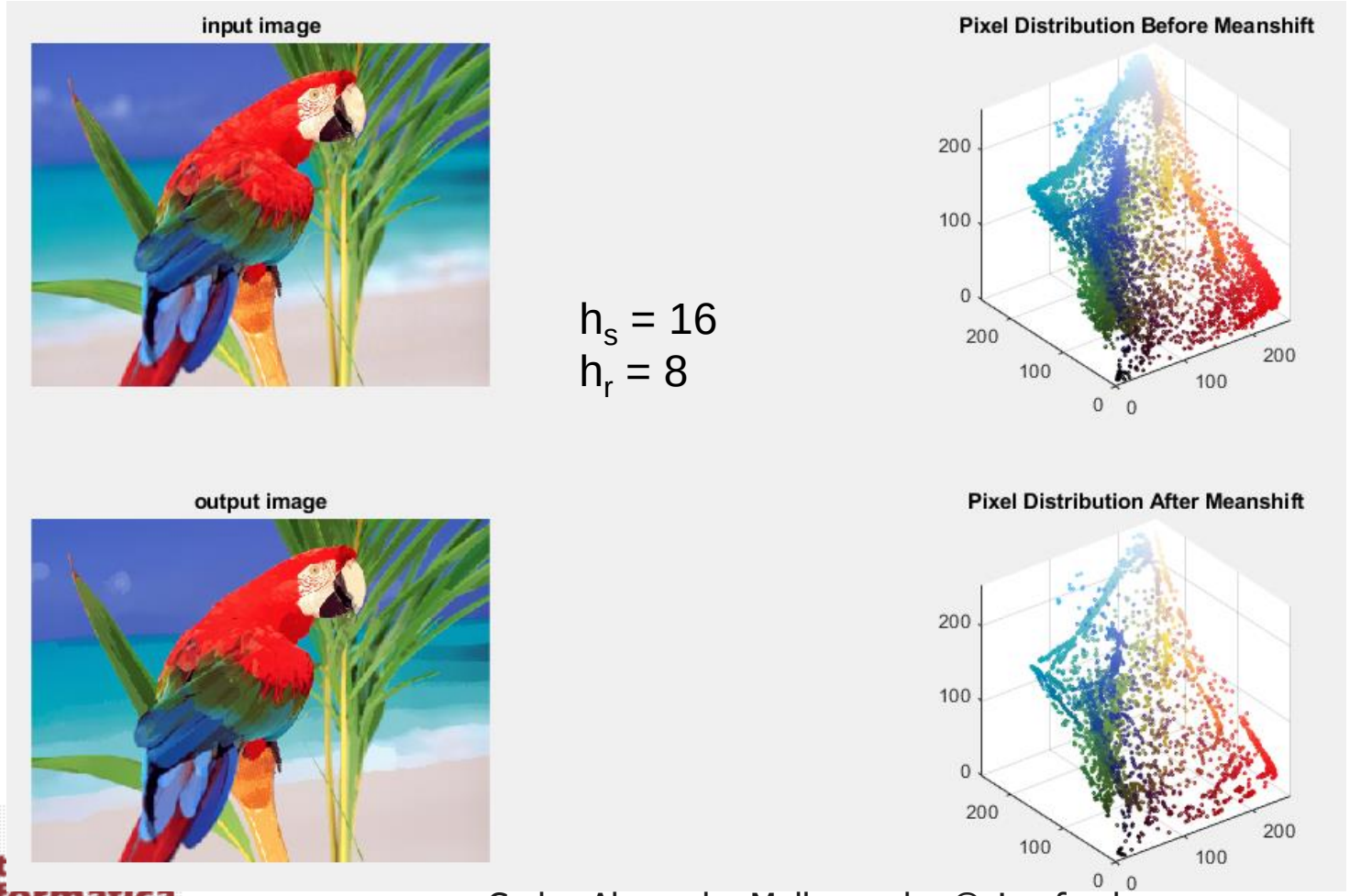
$(h_s, h_r) = (8,7)$

Segmentação – Mean Shift

- Exemplo

Segmentação – Mean Shift

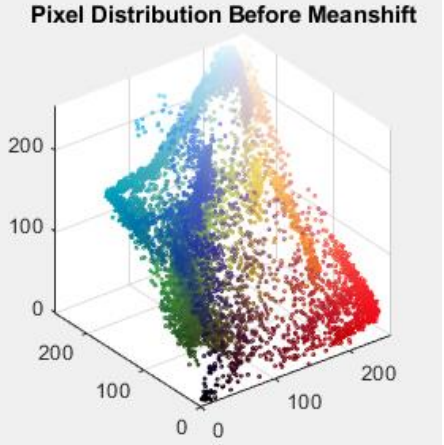
- Exemplo

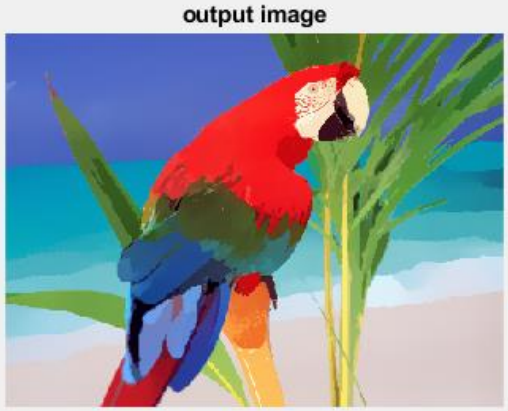


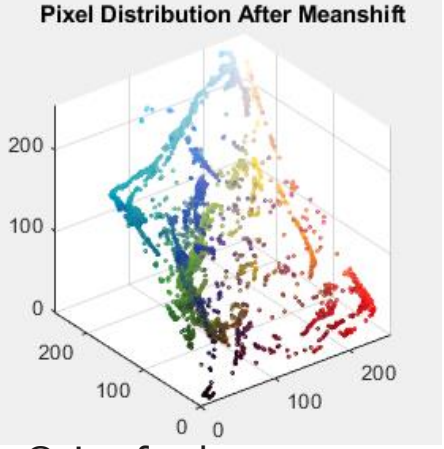
Segmentação – Mean Shift

- Exemplo

$h_s = 16$
$h_r = 15$







Segmentação – Mean Shift

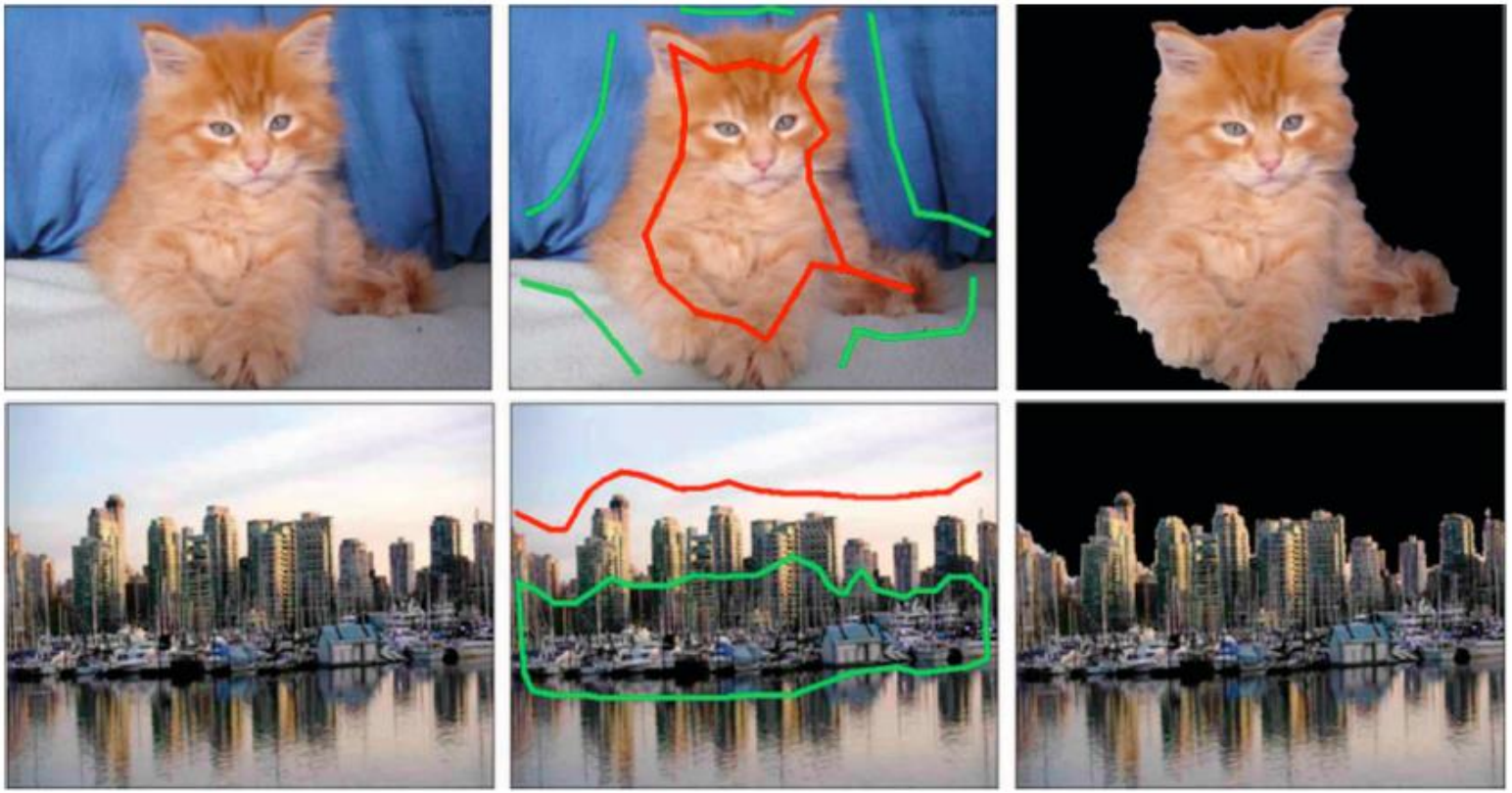
- Exemplo

$h_s = 16$
$h_r = 8$

$h_s = 16$
$h_r = 15$

Segmentação de Imagens

- Segmentação Interativa



Exemplo: Graph-Cut