

Unidade II – Imagem Digital (Parte 2)



IME 04-10842
Computação Gráfica
Professor Guilherme Mota
Professor Gilson Costa

Segmentação de Imagens

■ Objetivo

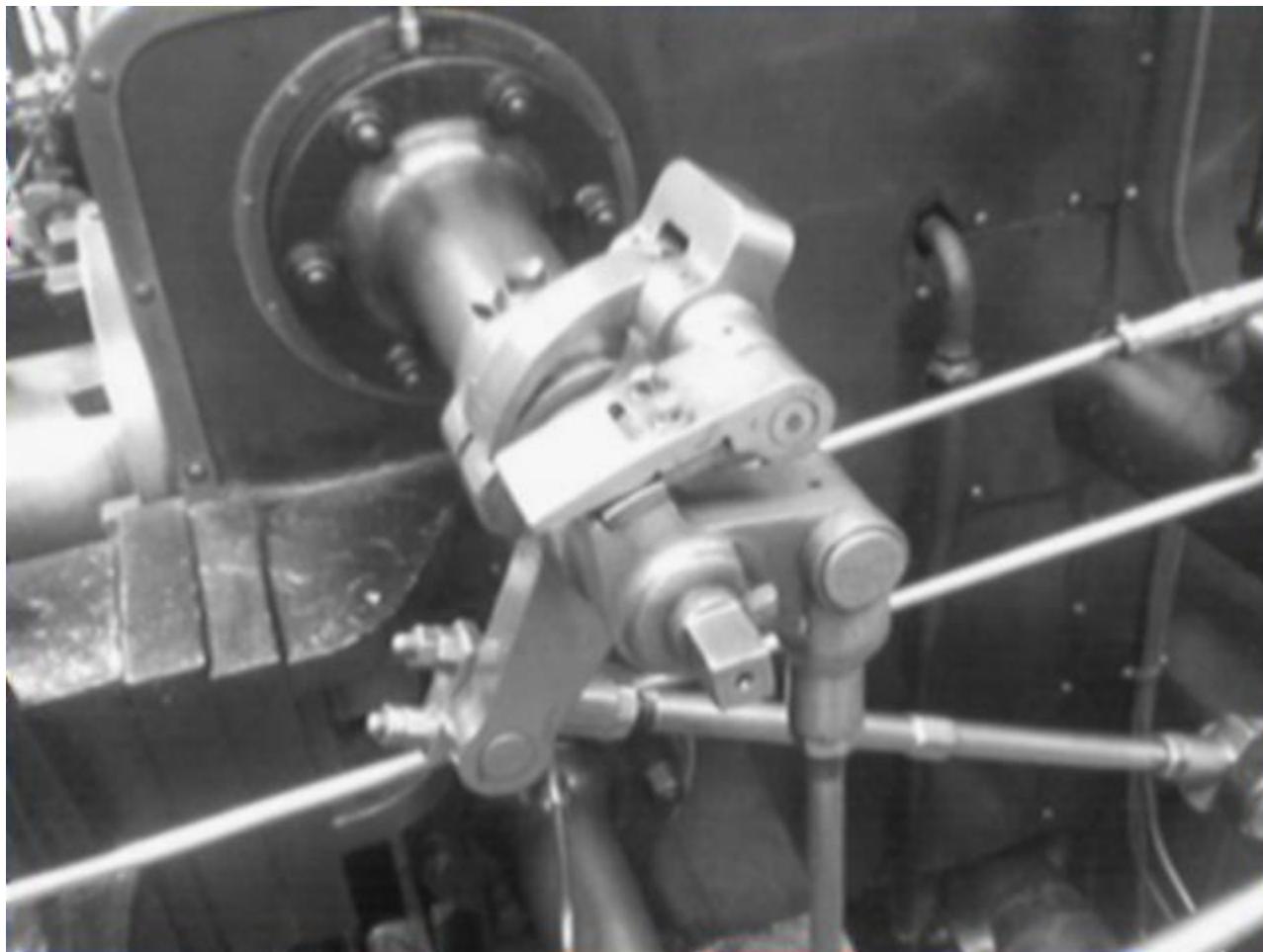
Introduzir os conceitos básicos de um conjunto de abordagens para a segmentação de imagens digitais.

Segmentação de Imagens

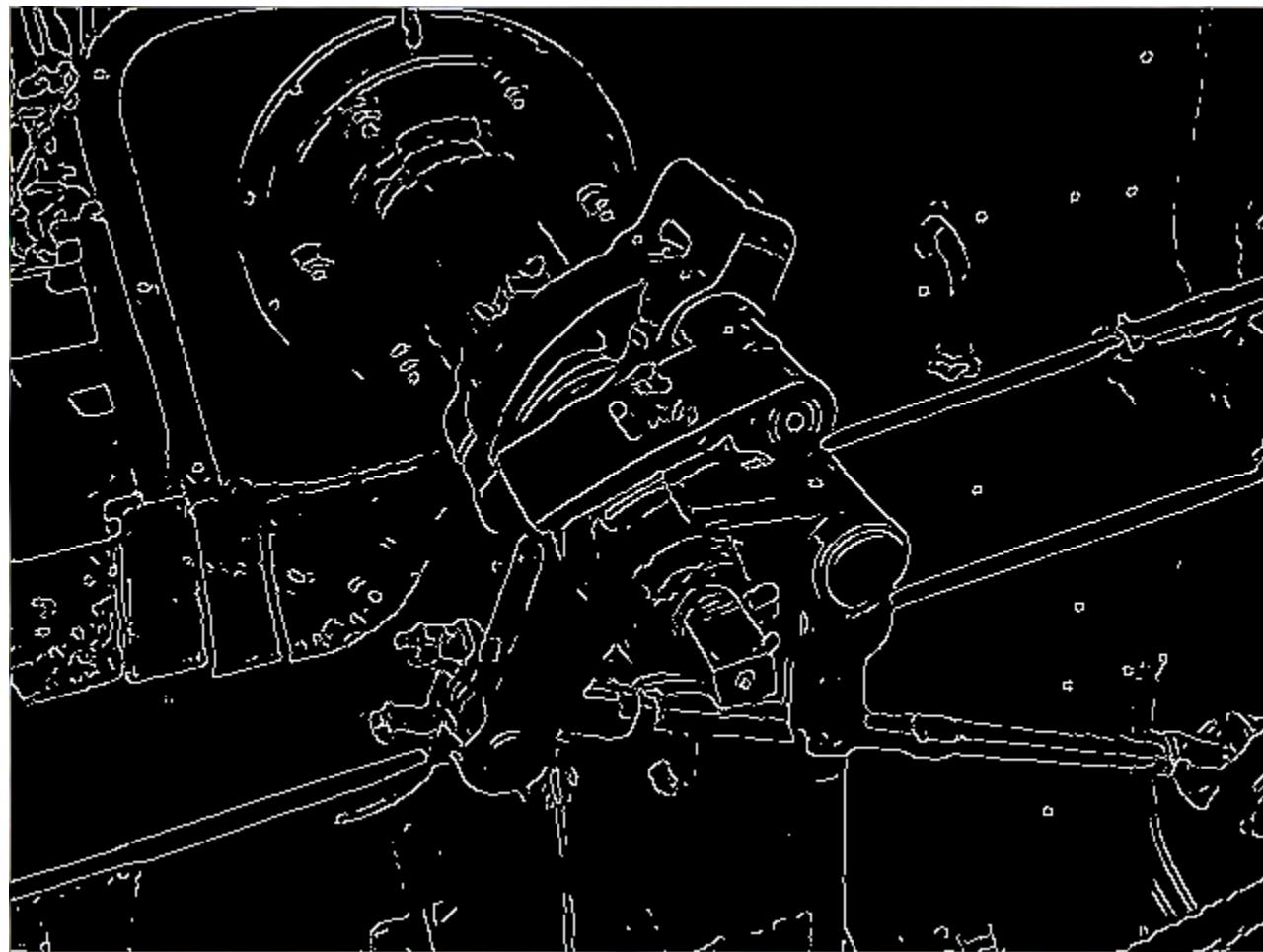
- Algoritmo de segmentação são baseados em uma ou duas propriedades básicas dos valores de intensidade dos pixels:
 - descontinuidade
 - similaridade



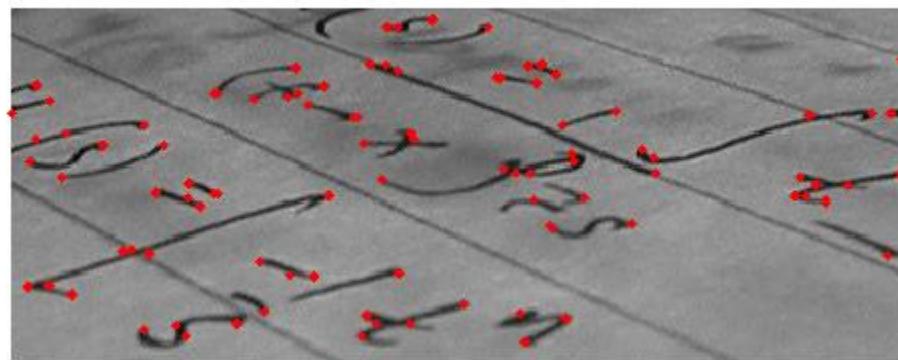
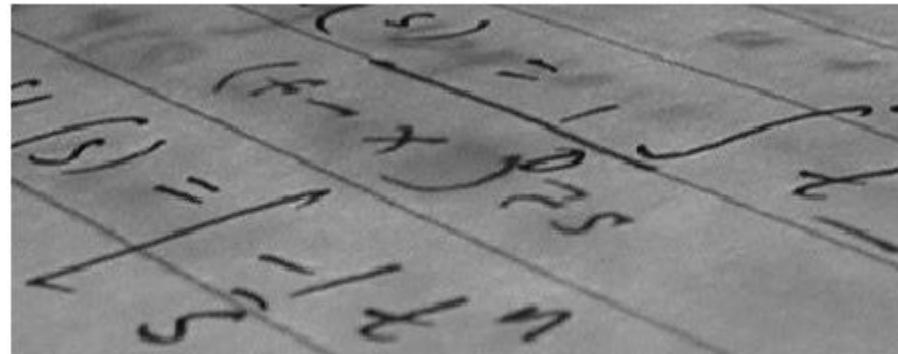
Descontinuidade (bordas)



Descontinuidade (bordas)



Descontinuidade (cantos)



Similaridade (regiões)



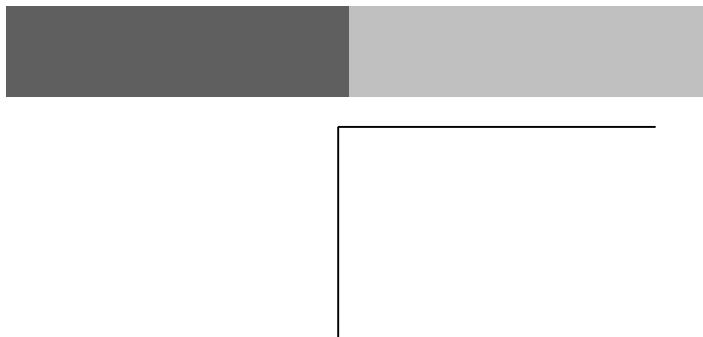
Similaridade (regiões)



Detecção de Bordas

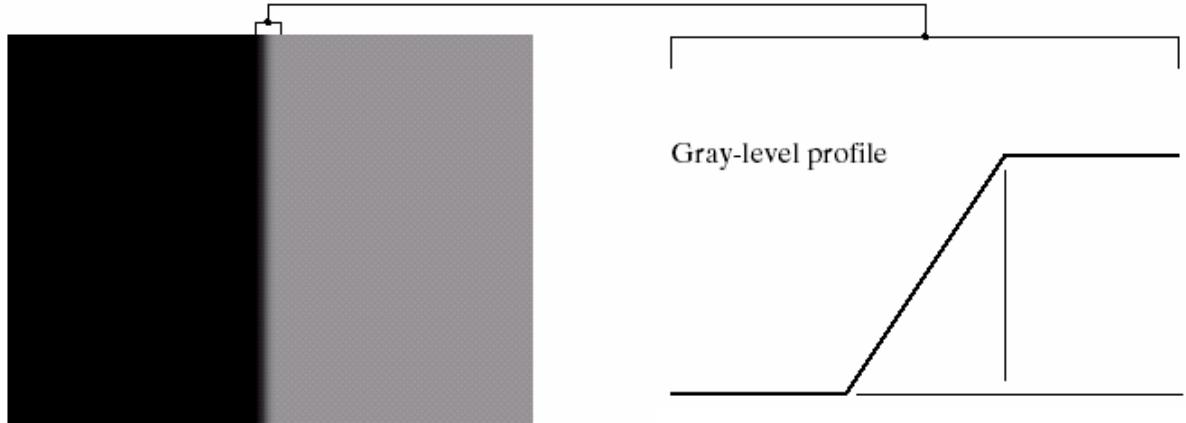
■ Formulação básica:

- *Borda ideal*: conjunto de pixels conectados, localizados em uma transição ortogonal dos valores de intensidade
- *Borda real*: tem um perfil de “rampa”



Borda ideal

Detecção de Bordas



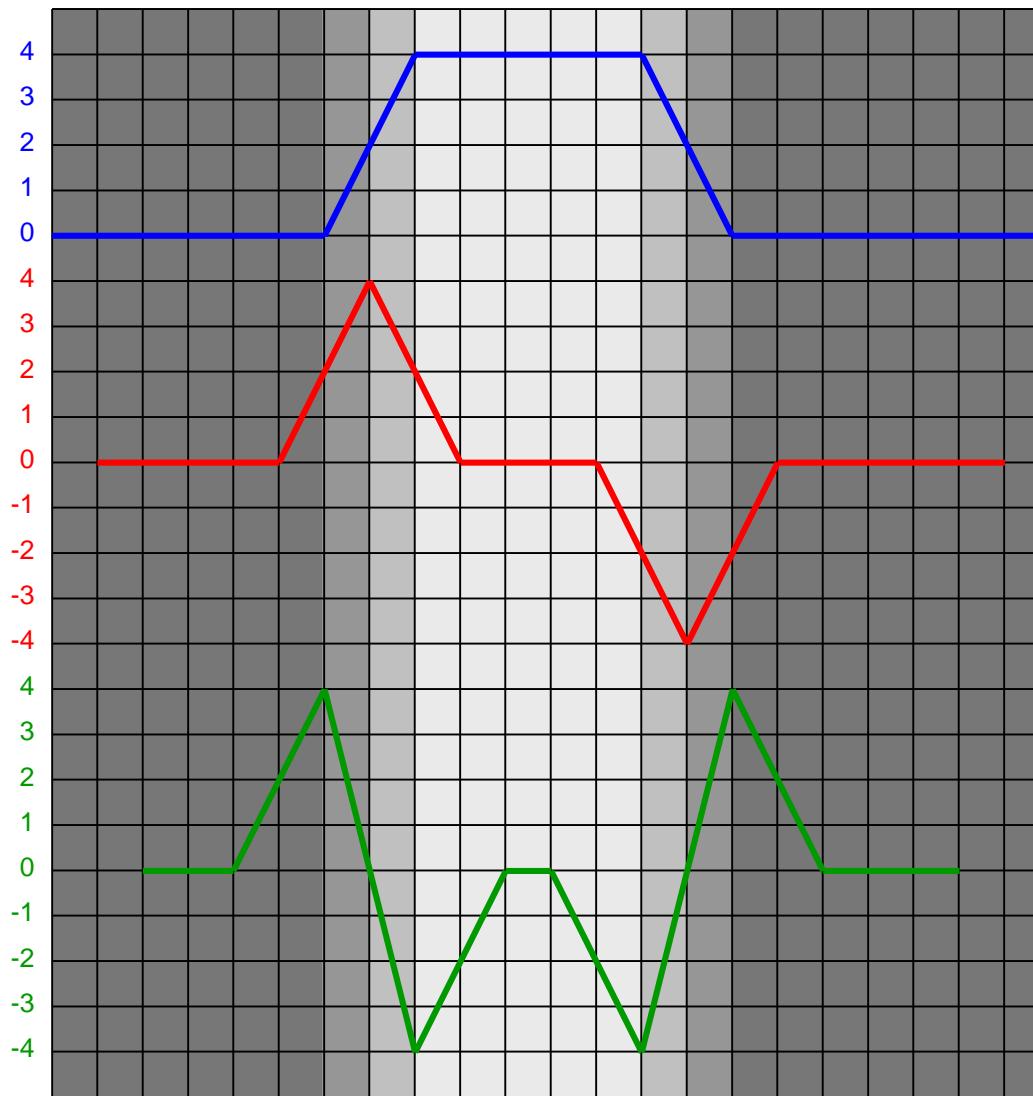
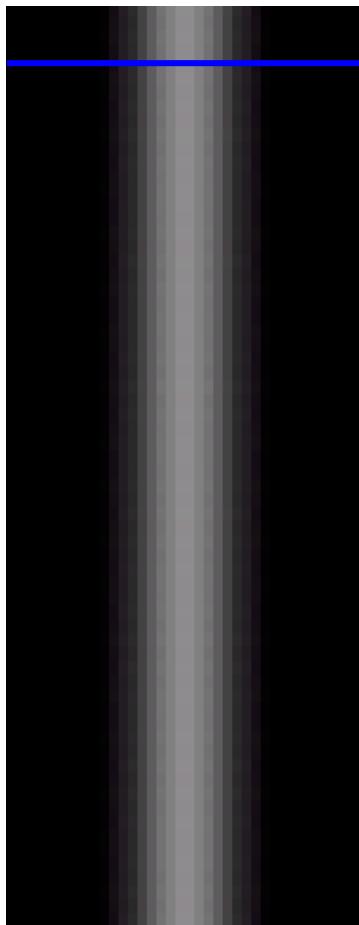
■ Conclusões:

1. Magnitude da 1^a derivada → evidencia de uma borda
2. Sinal da 2^a derivada determina lados claro/escuro
3. 2^a derivada tem 2 valores para cada borda
4. Linha ligando os valores extremos (positive e negative) da 2^a derivada cruza o zero no ponto médio

Detecção de Bordas

$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Prewitt mask



Valores dos pixels

primeira derivada

segunda derivada

Detecção de Bordas

- Ponto numa imagem é ponto de borda quando:
 - Sua derivada de primeira ordem é maior que um limiar, ou
 - Está no cruzamento por zero da segunda derivada.
- Borda: conjunto conectado de pontos de borda.
- Borda ocorre na fronteira de duas regiões.

Detecção de Bordas

■ Derivada de primeira ordem:

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

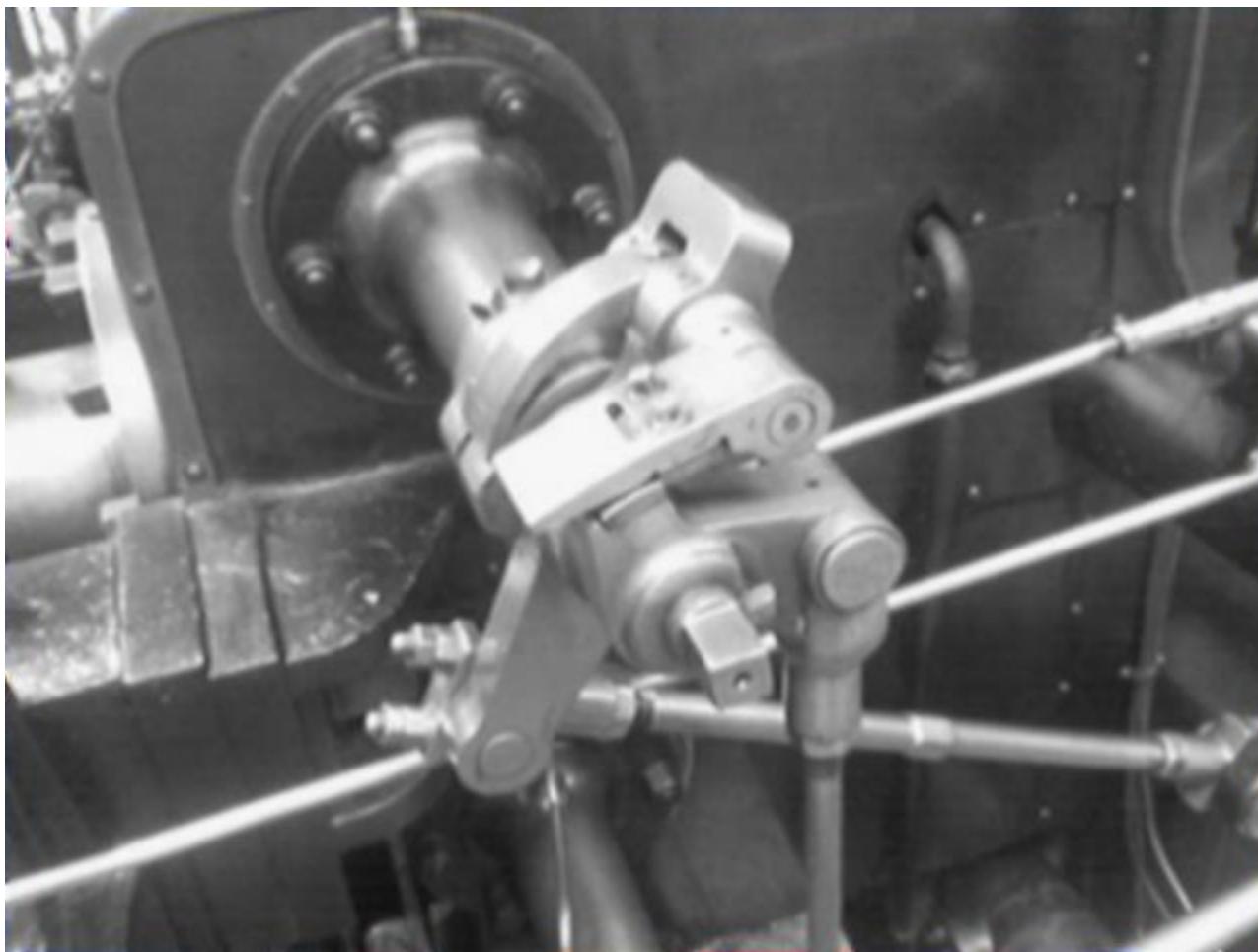
$$\nabla f = \text{mag}(\nabla \mathbf{f}) = [G_x^2 + G_y^2]^{1/2} \approx |G_x| + |G_y|$$

magnitude

$$\alpha(x, y) = \text{tag}^{-1}\left(\frac{G_y}{G_x}\right)$$

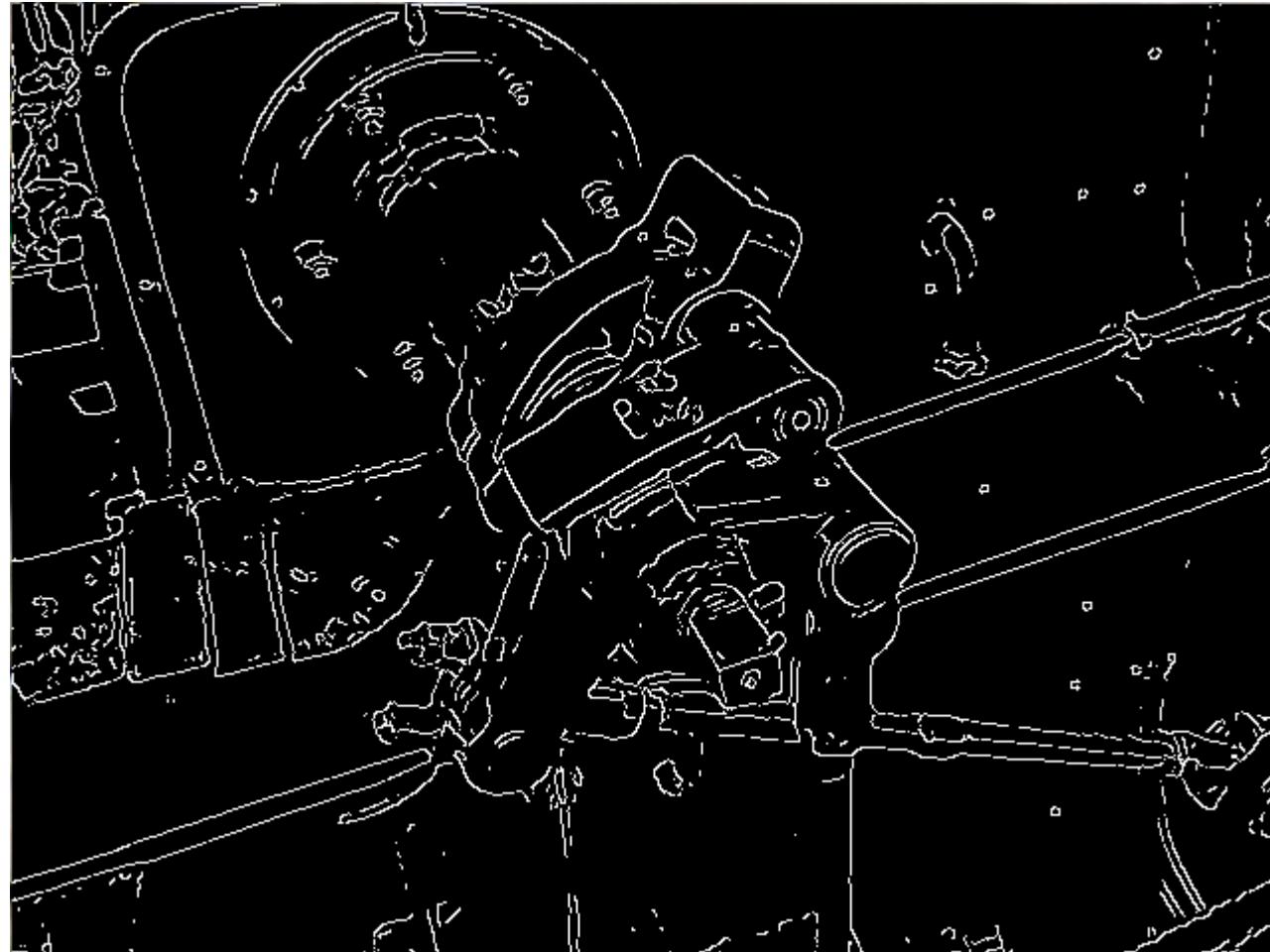
ângulo

Detecção de Bordas



Detecção de Bordas

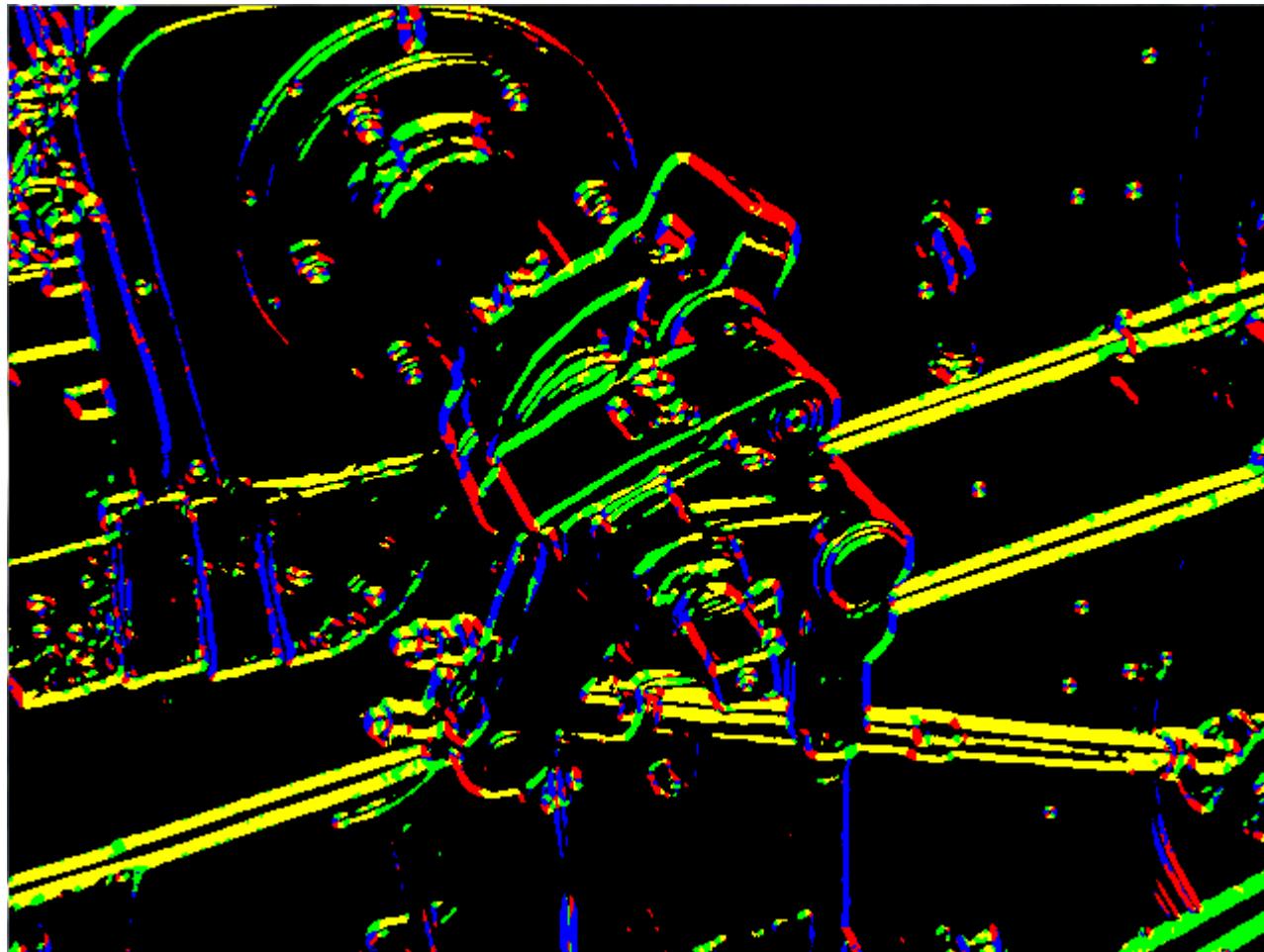
$$\nabla f = \text{mag}(\nabla \mathbf{f}) = [G_x^2 + G_y^2]^{1/2} \approx |G_x| + |G_y|$$



magnitude > 80

Detecção de Bordas

$$\alpha(x, y) = \operatorname{tag}^{-1}\left(\frac{G_y}{G_x}\right)$$



ângulo: azul ~ $0^\circ/180^\circ$; verm. ~ $45^\circ/225^\circ$; amarelo ~ $90^\circ/270^\circ$; verde ~ $135^\circ/315^\circ$

Detecção de Bordas

- # ■ Derivada de 1a. Ordem: **Gradiente**

$$\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2 \right]^{1/2} \approx |G_x| + |G_y| \quad \begin{matrix} \text{magnitude} \\ \text{magnitude} \end{matrix} \quad a(x, y) = \text{tag}^{-1} \left(\frac{G_y}{G_x} \right) \quad \begin{matrix} \text{ângulo} \\ \text{ângulo} \end{matrix}$$

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

Máscaras de Sobel para G_y e G_x

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

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

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

Máscaras de Sobel diagonais

Detecção de Bordas

- Detecção de bordas horizontais/verticais



Detecção de Bordas

■ Detecção de bordas diagonais



Detecção de Bordas

- Derivada de Segunda Ordem:
Laplaciano

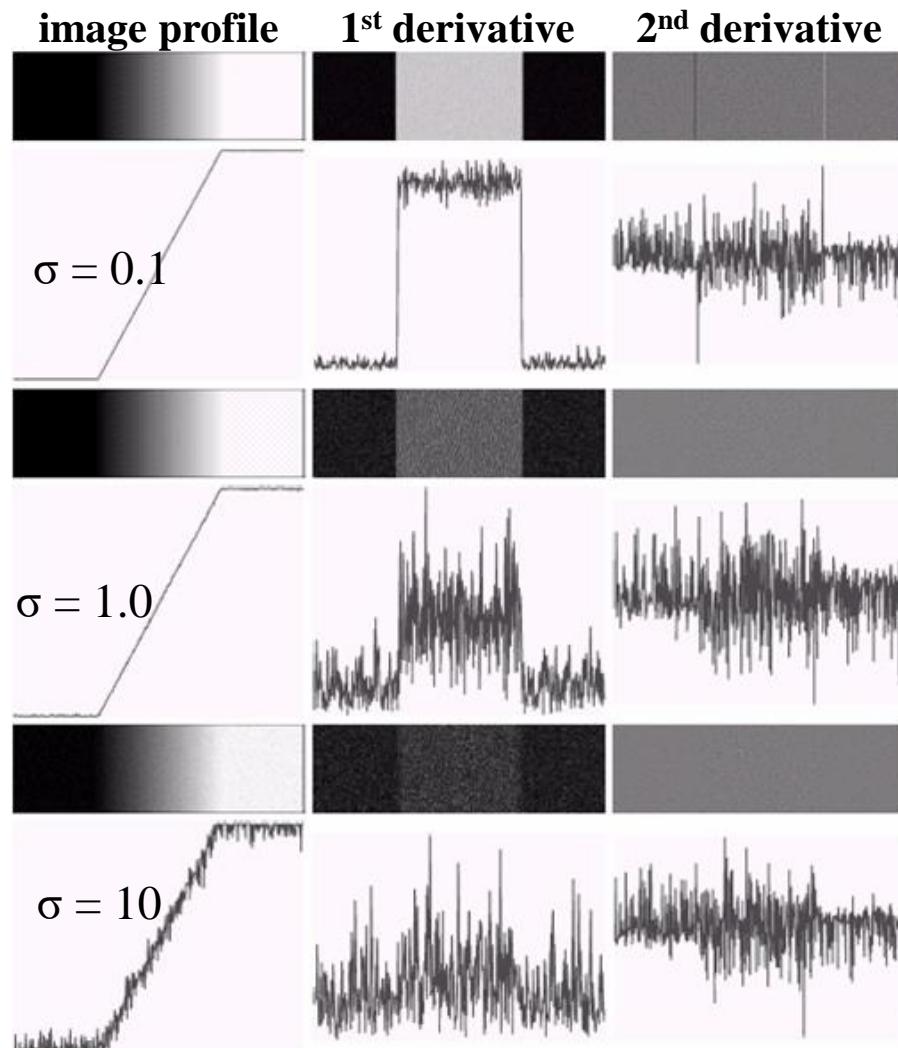
$$\nabla^2 \mathbf{f} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

0	-1	0
-1	4	-1
0	-1	0

-1	-1	-1
-1	8	-1
-1	-1	-1

Máscaras para o Laplaciano

Detecção de Bordas



Detecção de Bordas

■ Laplaciano (em relação ao Gradiente)

□ Contras:

- Muito sensível ao ruído.
- Magnitude produz picos duplos.
- Não identifica a direção da borda.

□ Prós:

- Cruzamento pelo zero: posição da borda.
- Possibilidade de indicar lado escuro e lado claro da borda.

Detecção de Bordas

- Filtro LoG: suavisa antes do Laplaciano

$$\underbrace{[h_L(x,y) * h_G(x,y)] * f(x,y)}_{h_{LoG}(x,y) * f(x,y)}$$

Máscara para: *Laplacian of Gaussian*

$f(x,y)$ – imagem

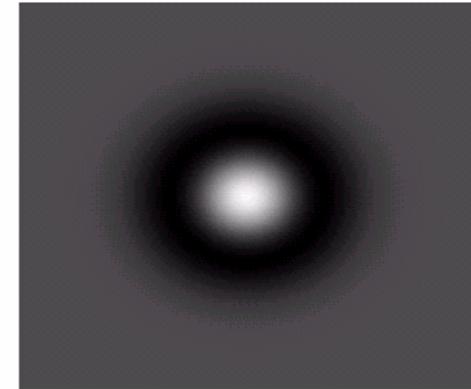
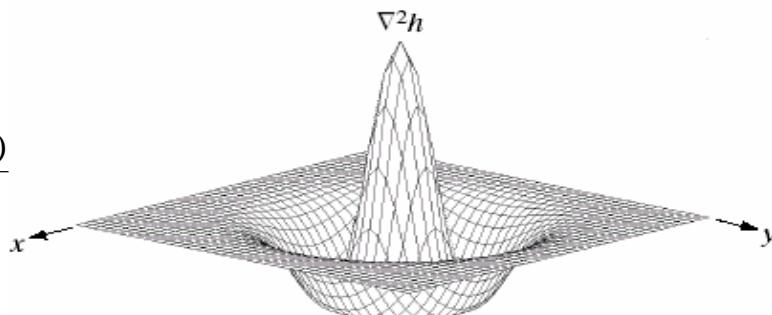
$h_L(x,y)$ – máscara laplaciana

$h_G(x,y)$ – máscara gaussiana (suavização)

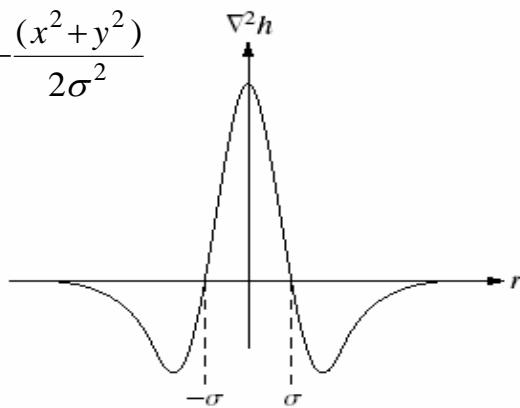
Detecção de Bordas

■ Filtro LoG

$$h_G(x, y) = -e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



$$\nabla^2 h_G(x, y) = -\left[\frac{x^2 + y^2 - \sigma^2}{\sigma^4}\right] e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$



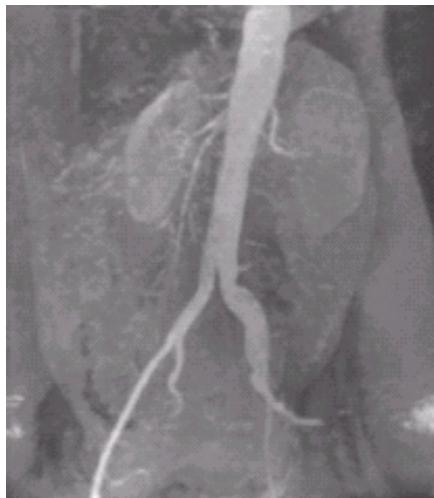
seção transversal

0	0	-1	0	0
0	-1	-2	-1	0
-1	-2	16	-2	-1
0	-1	-2	-1	0
0	0	-1	0	0

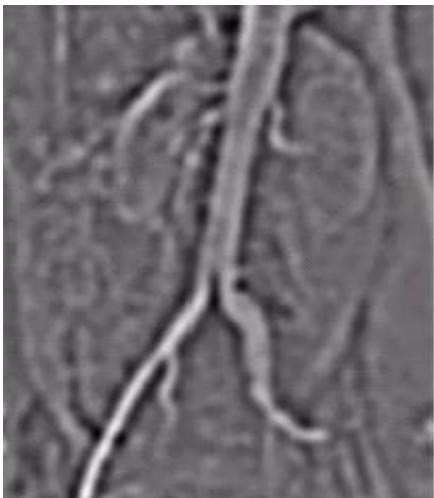
máscara 5×5

Detecção de Bordas

■ Filtro LoG



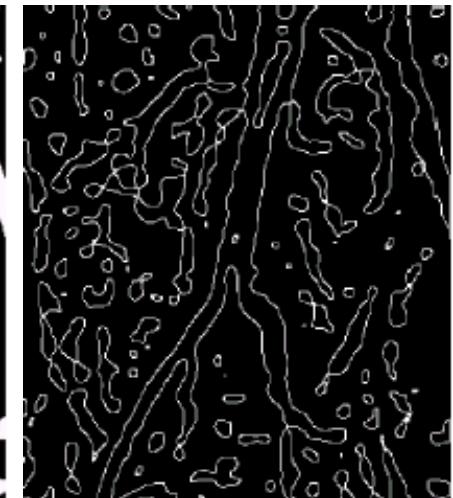
Angiograma



LoG



LoG limiarizado



Cruzamento por
zero

Ligaçāo de Pontos de Borda

■ Motivação

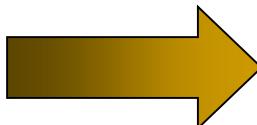
- Muitos dos pontos de borda identificados anteriormente não indicam exatamente a localização de uma borda.
- A detecção de pontos de borda deve ser seguida de uma ligação de pontos para representar as bordas corretamente.

Canny Algorithm

1. Suavise a imagem $f(x,y)$ com um filtro Gaussiano.
2. Calcule a magnitude $\nabla f(x,y)$ e a direção $\alpha(x,y)$ do gradiente.
3. Pixels com a magnitude maior que a de seus vizinhos na direção do gradient são considerados pixels de borda, outros pixels são descartados (*non maximal suppression*).

0	0	0	0	0	0	0	0	0	0	0
1	2	1	1	2	3	2	1	3	3	3
6	5	6	5	4	4	5	6	5	4	4
9	8	9	7	6	5	7	8	9	9	9
6	6	7	6	5	4	5	6	7	7	7
3	2	1	1	1	2	1	2	2	1	1
0	0	0	0	0	0	0	0	0	0	0
2	2	3	3	3	2	0	0	0	0	0
2	3	4	5	4	3	0	0	0	0	0
1	2	3	3	2	1	0	0	0	0	0

magnitude

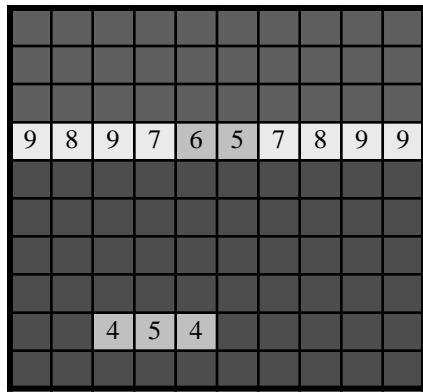


9	8	9	7	6	5	7	8	9	9
4	5	4							

pixels de borda

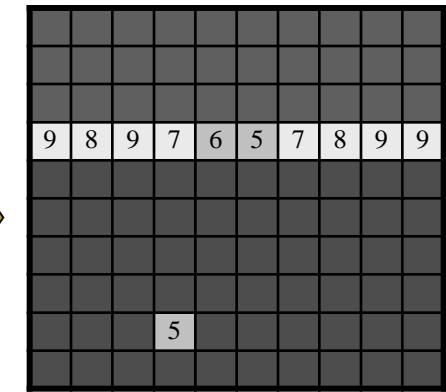
Canny Algorithm

4. Pontos de borda com a magnitude maior que um limiar alto T_H são pontos “fortes”, outros pontos de borda cuja magnitude é maior que um limiar mais baixo T_L ($T_L < T_H$) são pontos de borda “fracos”.
5. Borda é formada pelos pontos “fortes” mais os pontos “fracos” conectados aos pontos “fortes”.

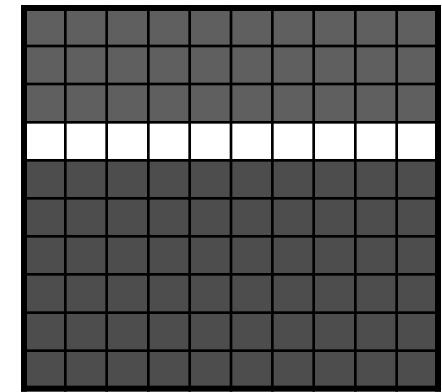
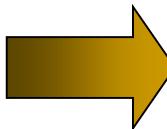


pontos de borda

$T_H=6$
 $T_L=4$

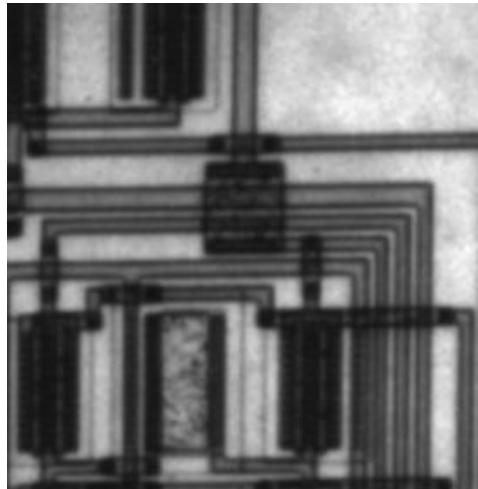


pontos fortes e fracos

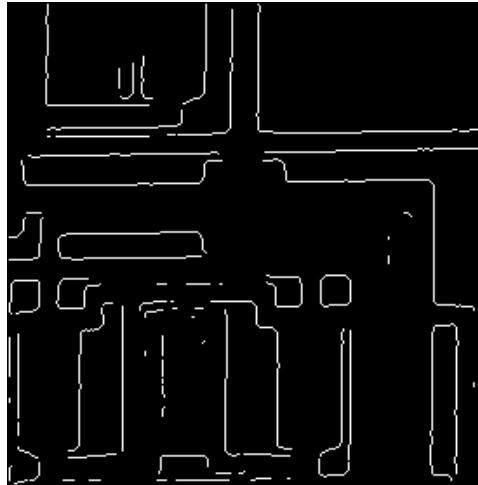


borda final

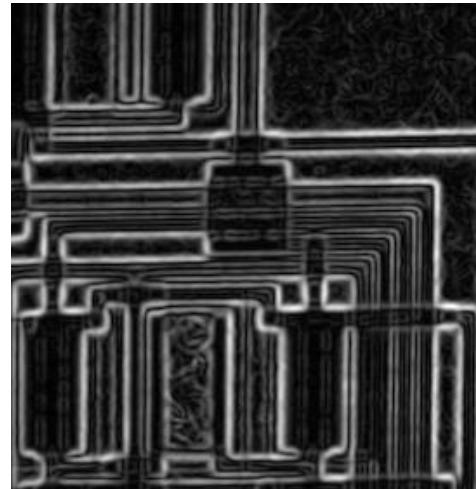
Canny Algorithm



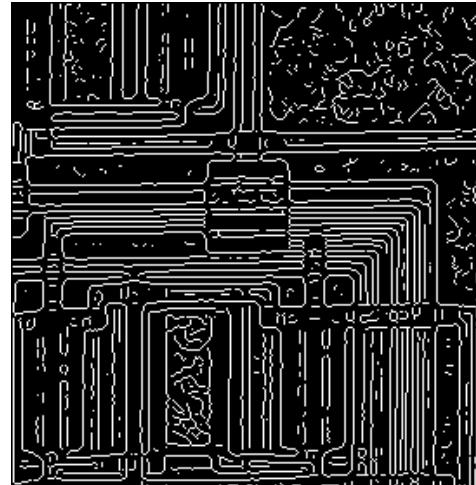
depois de filtro Gaussiano



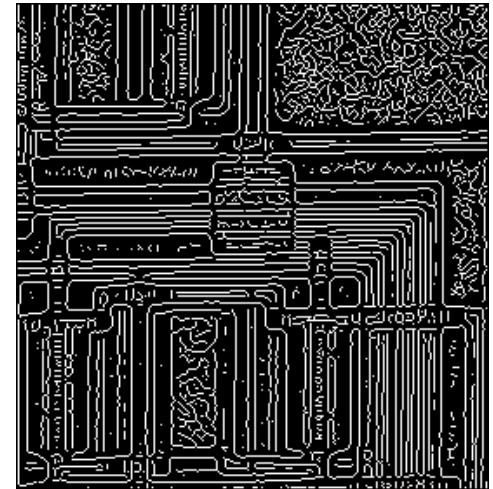
pontos $> T_H$



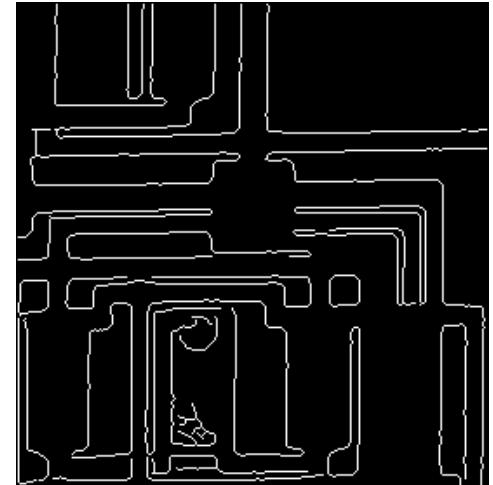
magnitude do gradiente



$T_L <$ pontos $< T_H$

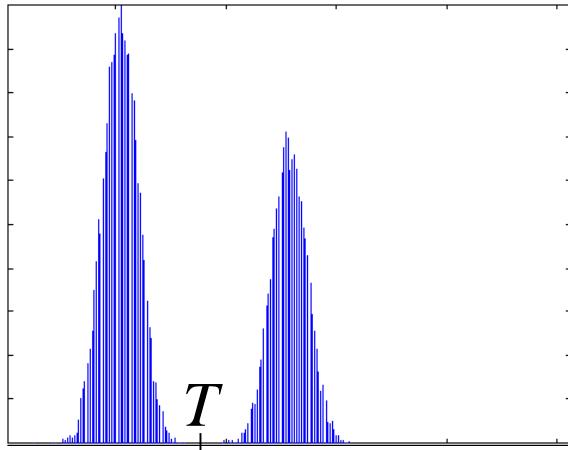


non maximal suppression

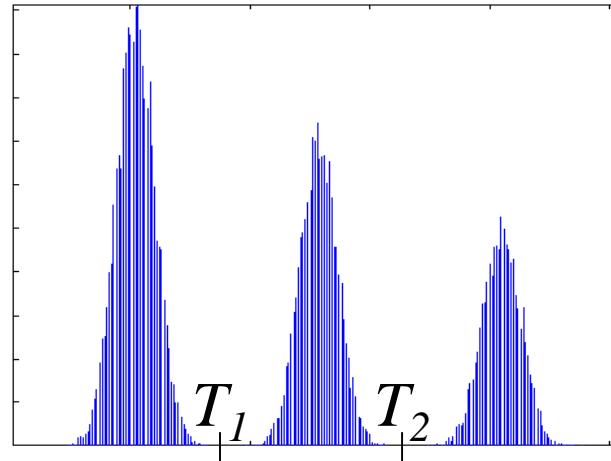


resultado final

Limiarização



Limiar



Multiplos limiares

$$T = T [x, y, p(x,y), f(x,y)] \rightarrow g(x, y) = \begin{cases} 1 & \text{if } f(x, y) > T \\ 0 & \text{if } f(x, y) \leq T \end{cases}$$

$$T = T [f(x,y)]$$

global

$$T = T [p(x,y), f(x,y)]$$

local

$$T = T [x, y, p(x,y), f(x,y)]$$

dinâmico ou adaptativo

Limiarização Global

Determinação automática do limiar: algoritmo simples

1. Selecione um limiar inicial T
2. Limiarize a imagem usando T :

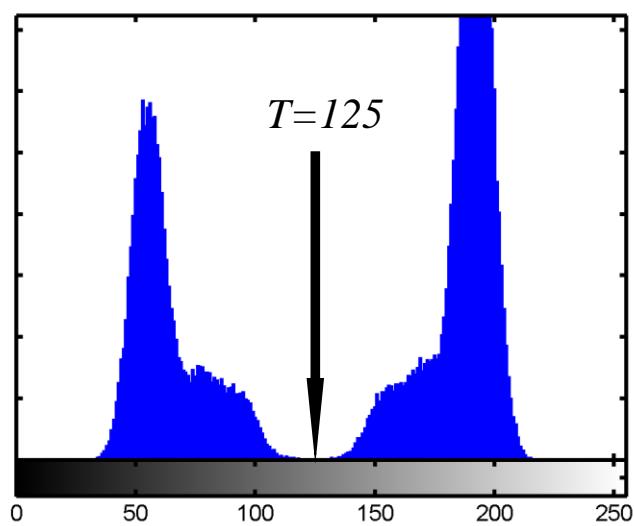
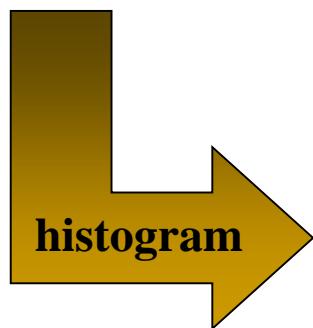
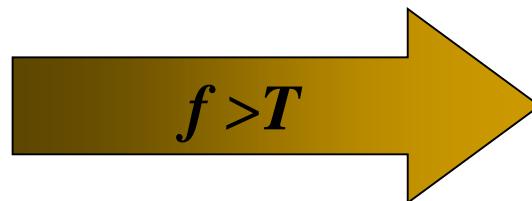
Vai gerar dois grupos de pixels: G_1 (pixels com valores $< T$) e G_2 (pixels com valores $\geq T$)

3. Calcule a intensidade media de cada grupo (μ_1 e μ_2)
4. Calcule um novo limiar:

$$T = \frac{1}{2} (\mu_1 + \mu_2)$$

5. Repita os passos 2 a 4 até que não ocorram mudanças significativas de T em duas iterações consecutivas

Limiarização Global



Imagens de Sensoriamento Remoto

Blue Band



Green Band



Red Band



Near infra-red Band



IKONOS

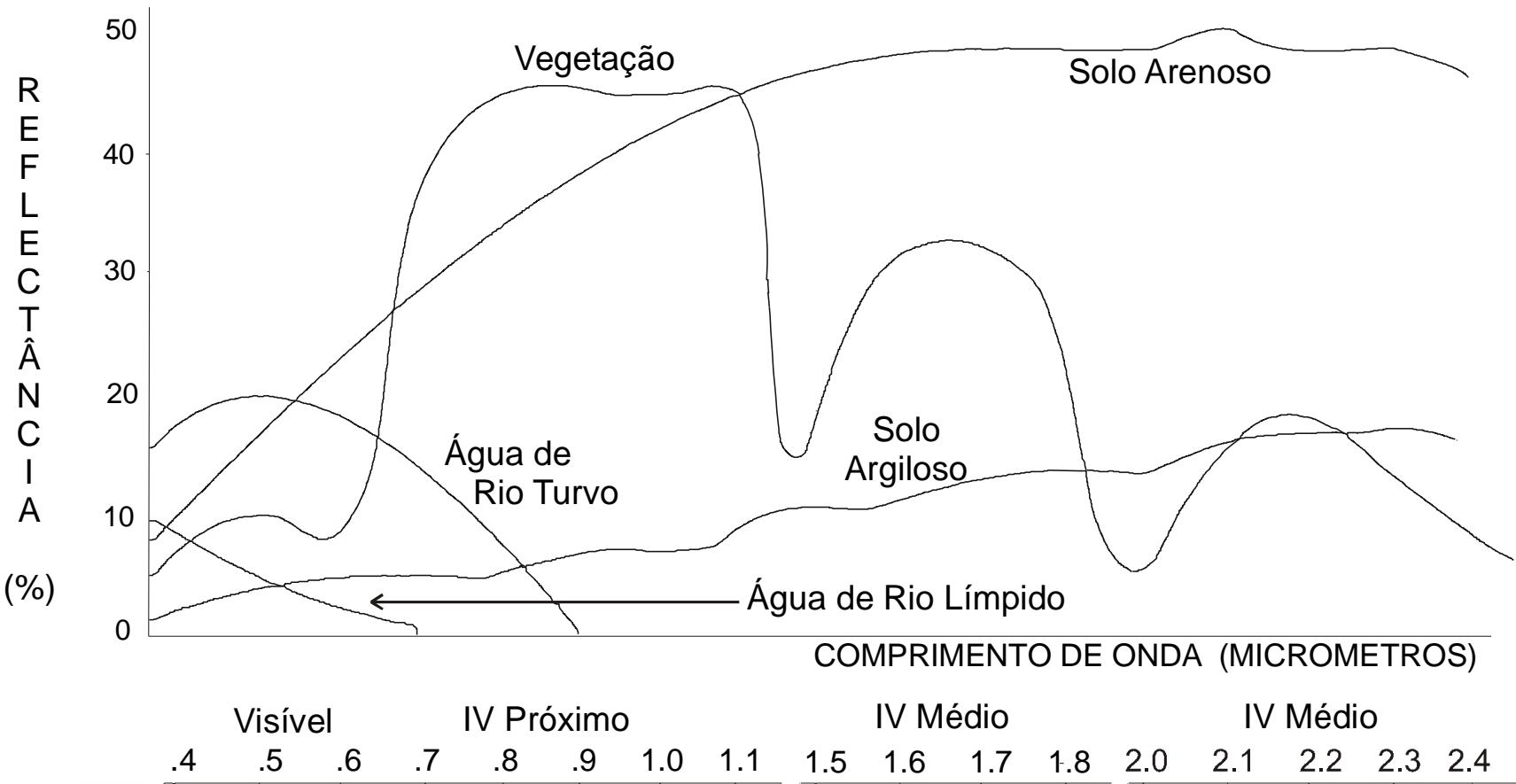
Imagens de Sensoriamento Remoto

Color composition
R (Intra-red) – G (red) – B (green)



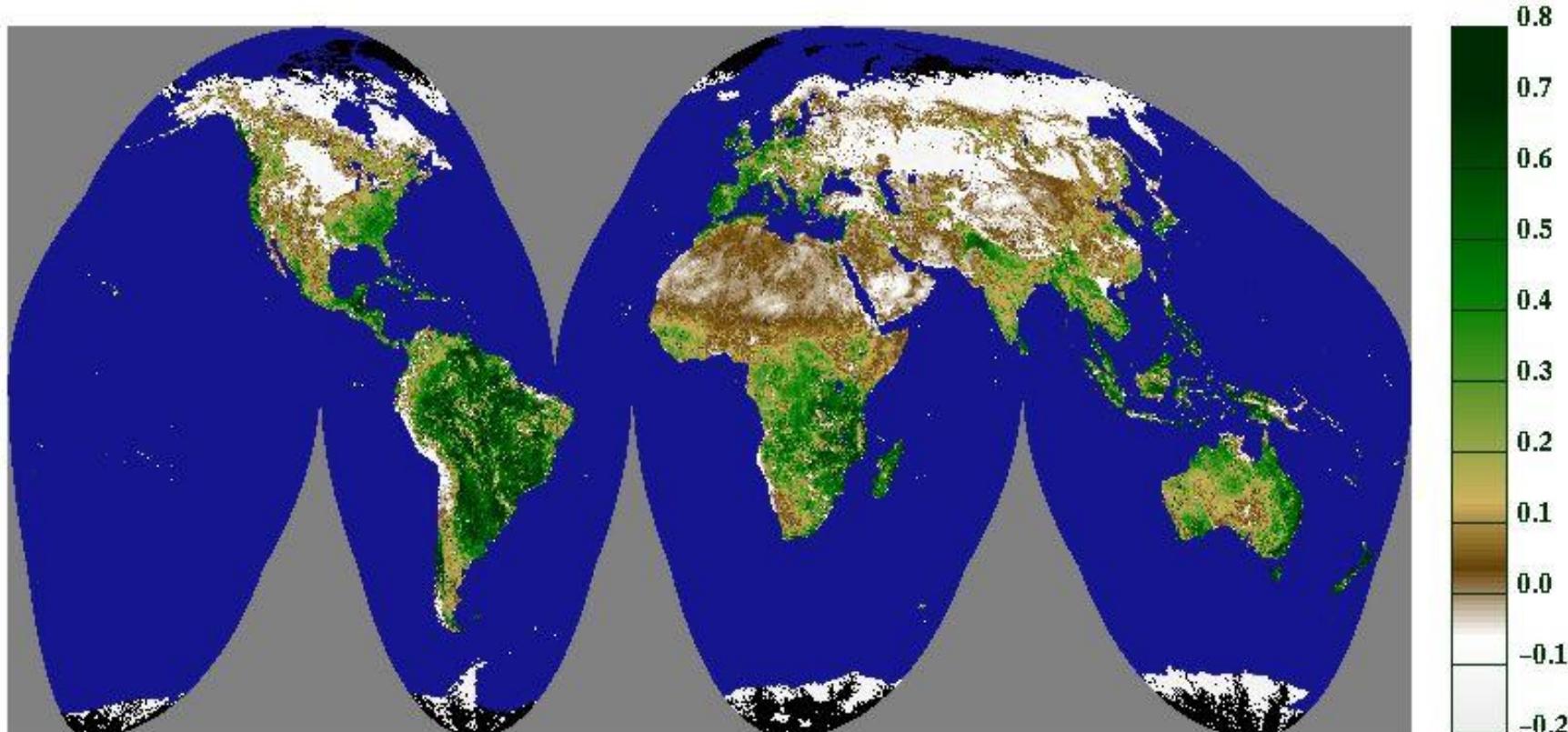
IKONOS

Imagens de Sensoamento Remoto



Imagens de Sensoamento Remoto

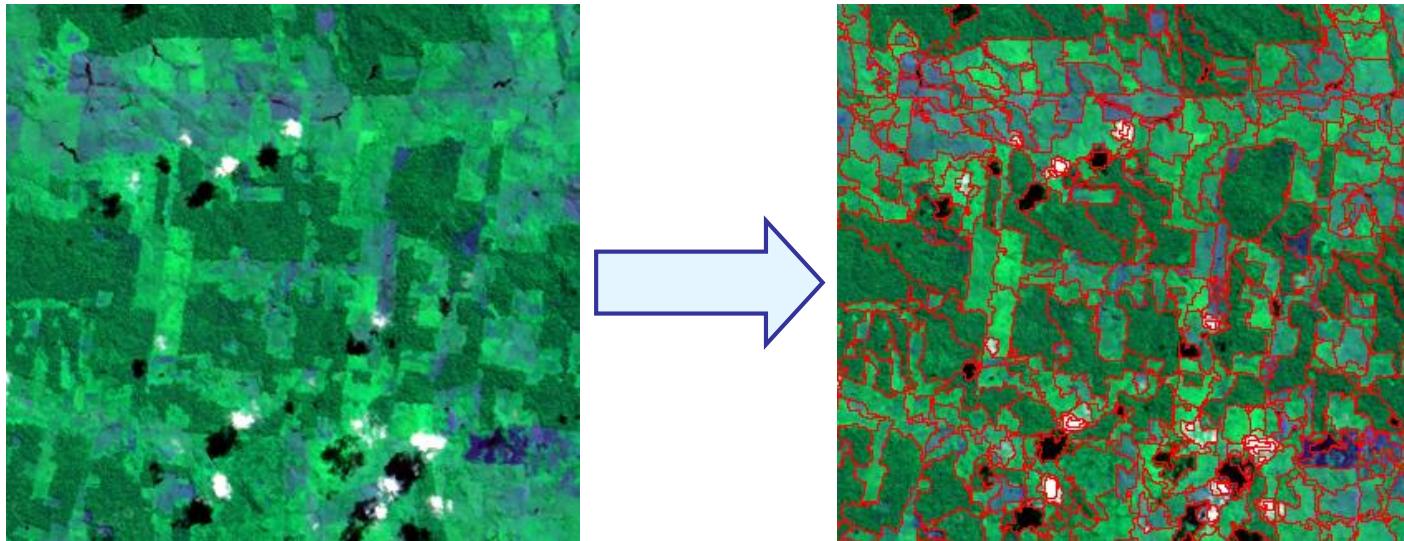
Normalized Distance Vegetation Index



$$NDVI = (nir - red) / (nir + red)$$

Segmentação Baseada em Regiões

Objetivo é identificar sub-regiões significativas de uma imagem.



$$R = R_1 \cup R_2 \cup \dots \cup R_n$$

Segmentação Baseada em Regiões

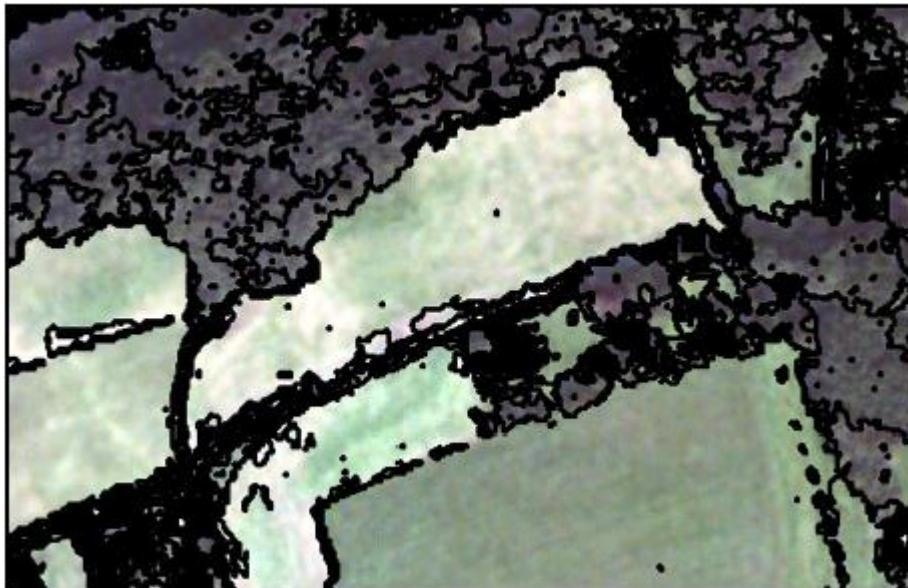
Formulação básica:

Se R representa uma imagem: a segmentação é o processo de particionar R em n sub-regiões R_1, R_2, \dots, R_n , de forma que:

- a) $\bigcup_{i=1}^n R_i = R$ { cobre toda a imagem }
- b) R_i é uma região conectada { cada sub-região contém pixels contíguos }
- c) $R_i \cap R_j = \emptyset$ para todo i e j , $i \neq j$ { sub-regiões são disjuntas }
- d) $P(R_i) = \text{TRUE}$ para $i=1,2,\dots,n$ { P define condição de pertencimento }
- e) $P(R_i \cup R_j) = \text{FALSE}$ para $i \neq j$ { P também diferencia as sub-regiões. }

Segmentação Baseada em Regiões

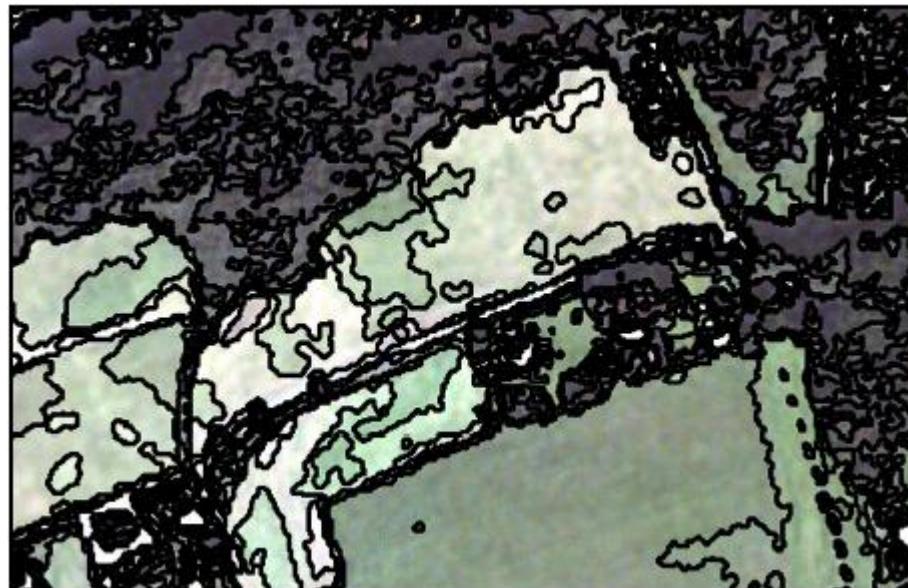
Várias técnicas: todas possuem parâmetros que devem ser definidos pelo usuário.



Data Dissecation Tools
(Clusterização – 3 parâmetros)

Segmentação Baseada em Regiões

Várias técnicas: todas possuem parâmetros que devem ser definidos pelo usuário.



InfoPACK
(Simulated Annealing – 2 parâmetros)

Segmentação Baseada em Regiões

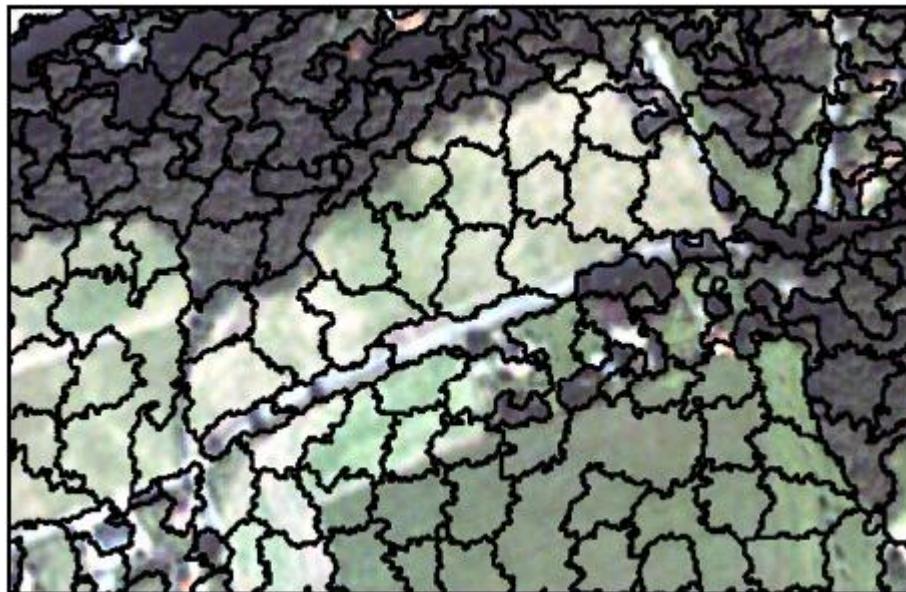
Várias técnicas: todas possuem parâmetros que devem ser definidos pelo usuário.



Minimum Entropy Approach
(Triangulação – 2 parâmetros)

Segmentação Baseada em Regiões

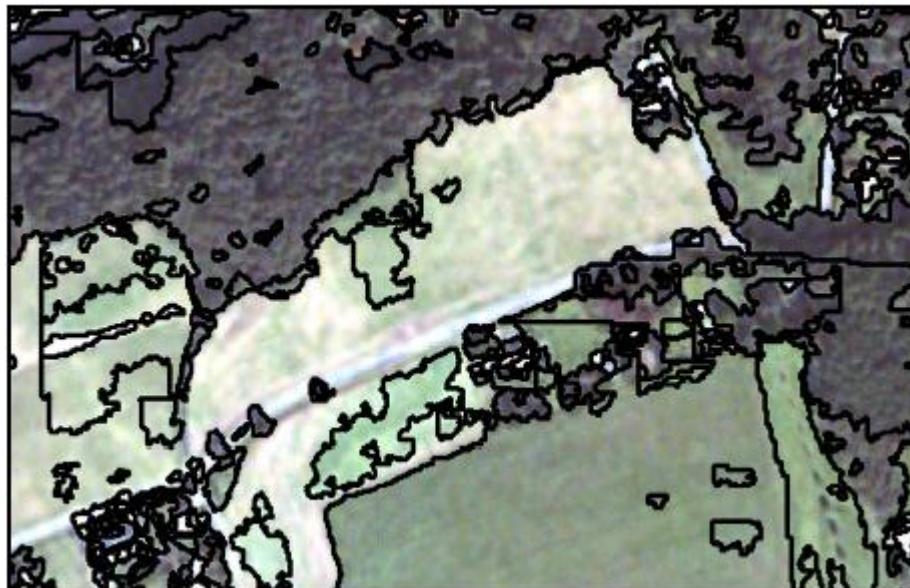
Várias técnicas: todas possuem parâmetros que devem ser definidos pelo usuário.



CAESAR
(Simulated Annealing – 6 parâmetros)

Segmentação Baseada em Regiões

Várias técnicas: todas possuem parâmetros que devem ser definidos pelo usuário.



ERDAS
(Crescimento de Regiões – 2 parâmetros)

Segmentação Baseada em Regiões

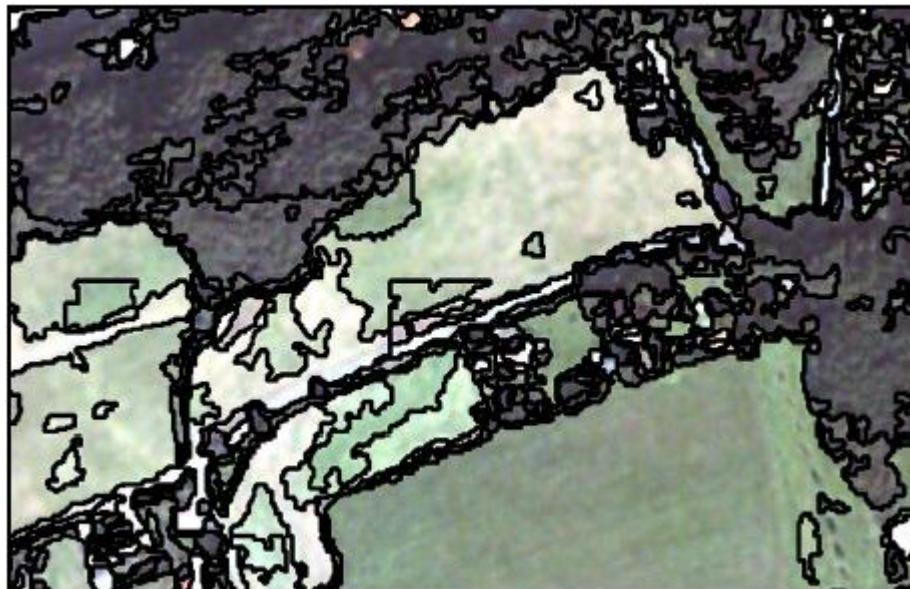
Várias técnicas: todas possuem parâmetros que devem ser definidos pelo usuário.



eCognition
(Crescimento de Regiões – 3 parâmetros)

Segmentação Baseada em Regiões

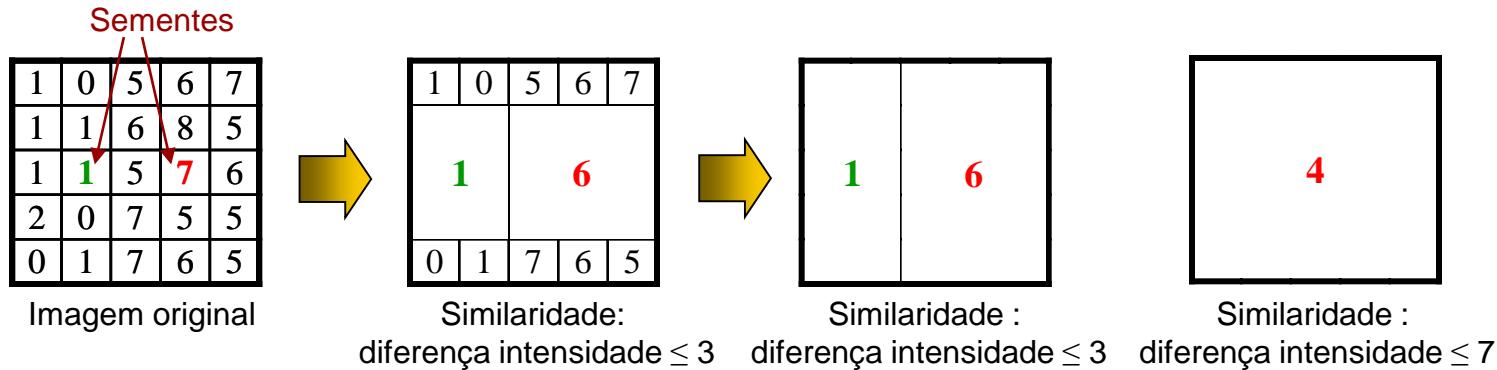
Várias técnicas: todas possuem parâmetros que devem ser definidos pelo usuário.



SPRING
(Crescimento de Regiões – 2 parâmetros)

Crescimento de Regiões

- **Definição:** procedimento que agrupa pixels ou sub-regiões em regiões maiores, baseado em algum critério.
- Envolve dois passos iterativos:
 - Seleciona um conjunto de “sementes”, e
 - Sementes/regiões crescem, agregando pixels vizinhos “similares”.



Crescimento de Regiões

■ Principais questões:

- Seleção do conjunto de sementes:
 - usa informações a priori sobre o problema específico; ou
 - calcula atributos para cada pixel e encontra grupos (*clusters*) cujos centróides serão usados como sementes; ou
 - determina as sementes aleatoriamente.
- Escolha do critério de similaridade:
 - depende dos dados;
 - uma diversidade de atributos (cor, textura, forma, etc.) podem ser levados em consideração.
- Regra de parada:
 - geralmente, quando mais nenhum pixel satisfaz a condição de inclusão.

Crescimento de Regiões

■ Exemplo:

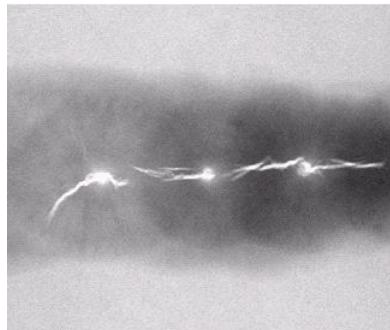
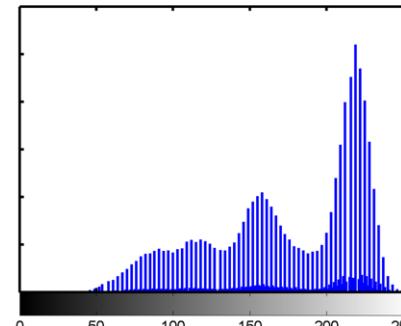
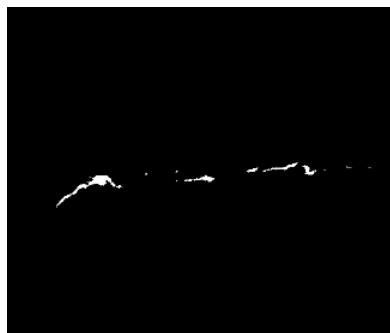


Imagen mostrando solda defeituosa



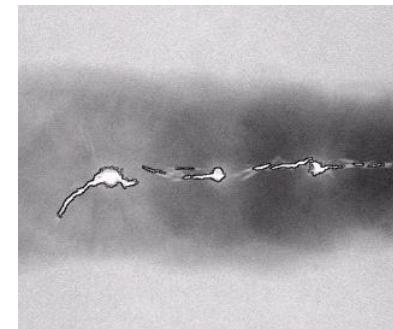
Histograma da imagem de entrada



Sementes: pixels com valores de intensidade máximos



Critério: diferença de intensidade com as sementes menor que 65



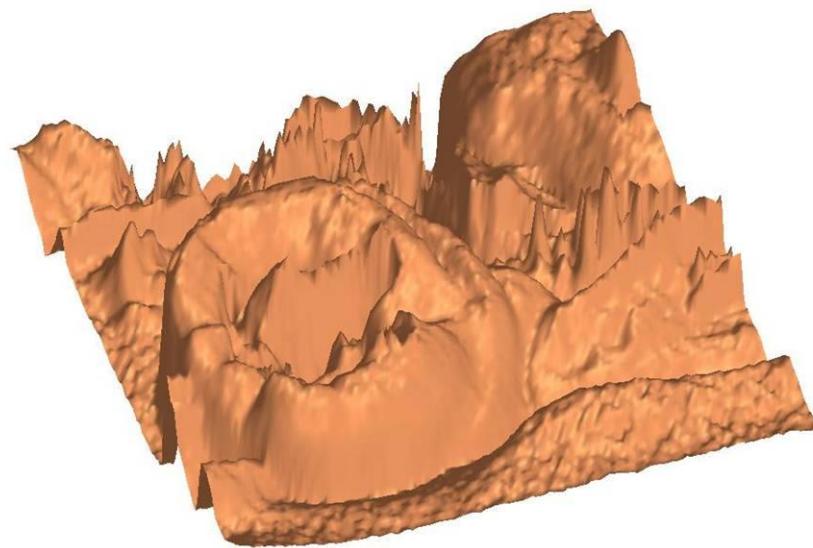
Bordas das soldas defeituosas (pretas)

Algoritmo Watershed

- Imagem vista em 3D (terceira dimensão é a intensidade).



imagem original image



vista “topográfica” da imagem

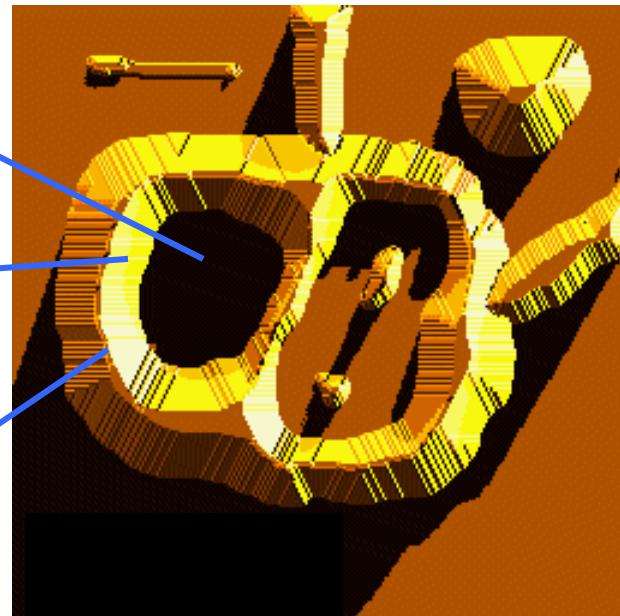
Algoritmo Watershed

■ Três tipos de pontos:

mínimo: pontos pertencentes a um mínimo local.

bacia hidrográfica de um mínimo: pontos em que, uma gota de água, se colocada sobre qualquer desses pontos, cairá com certeza em algum mínimo.

linhas de bacia hidrográfica: pontos em que a água seria igualmente suscetível de cair em mais de um mínimo.



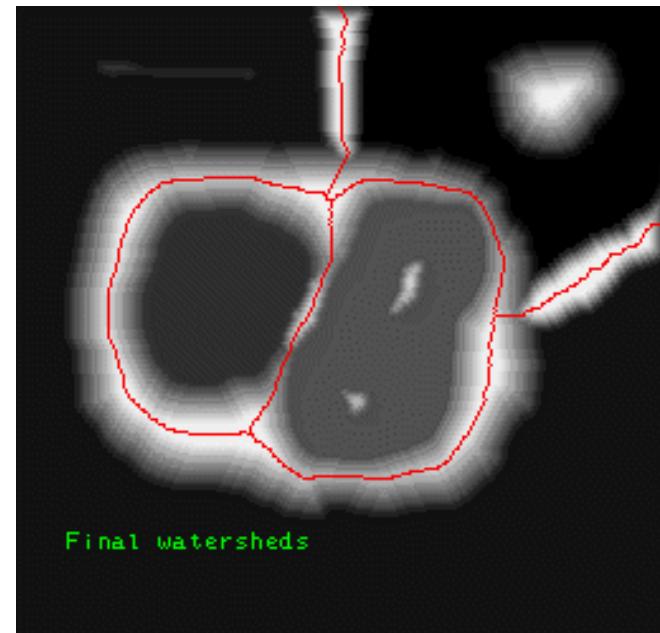
Vista “topográfica”

Algoritmo Watershed

- Um buraco é perfurado em cada mínimo local e toda a topografia é inundada de baixo, deixando a água subir através dos buracos a uma taxa uniforme.



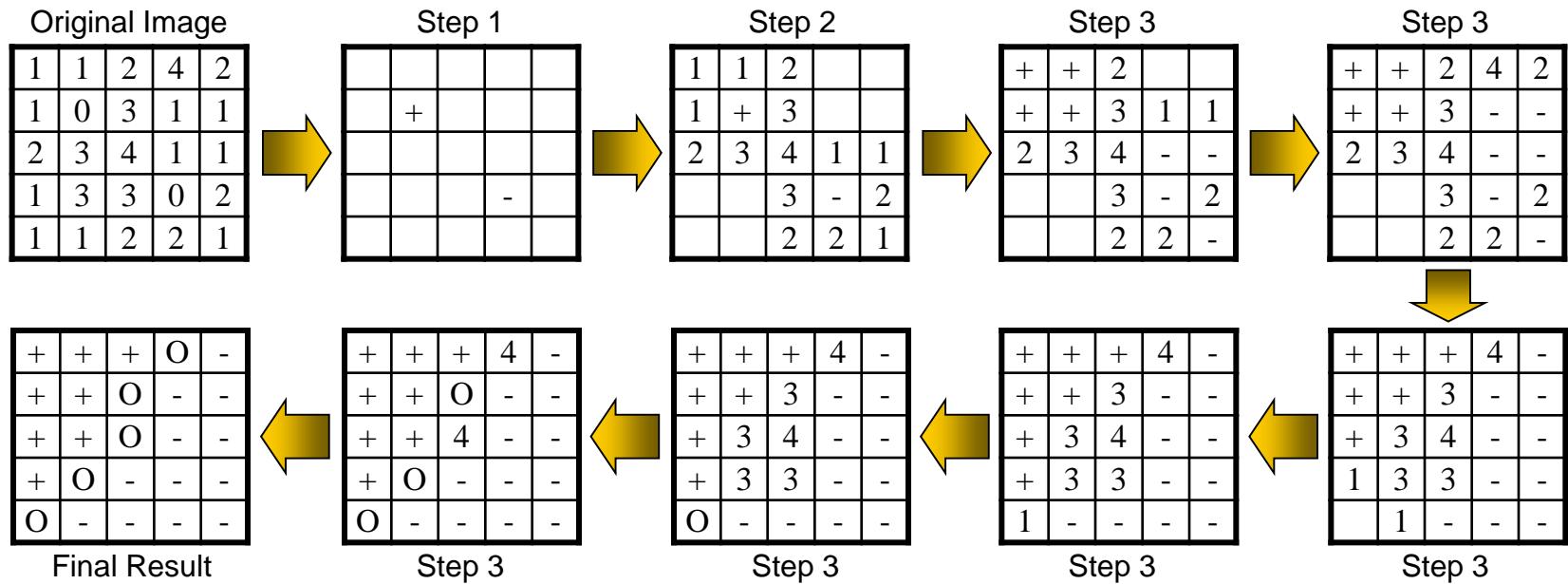
Initial image



Final watershed

processo de inundação

segmentação final



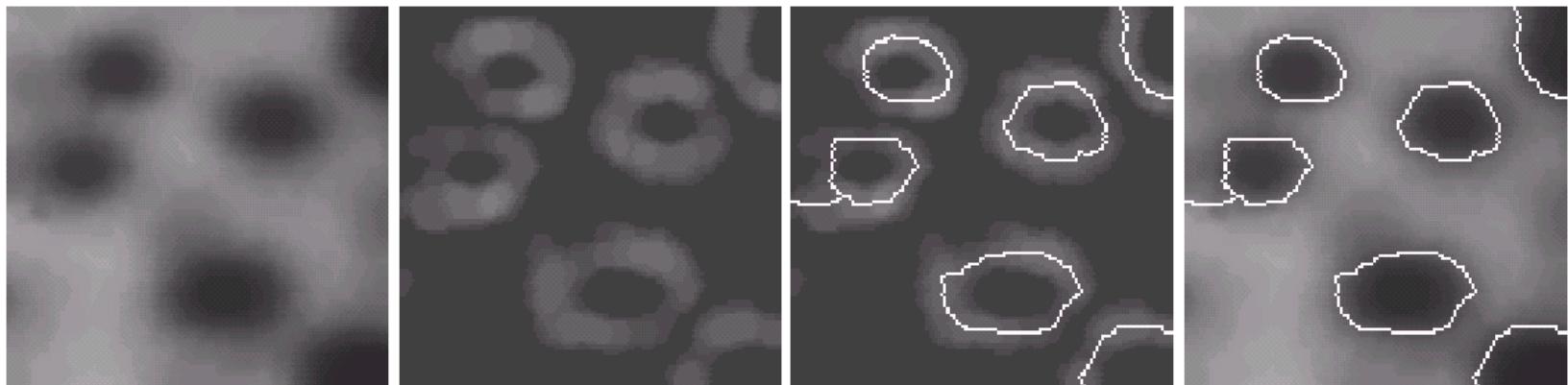
Algoritmo de inundação de Meyer (Watershed):

1. Um conjunto de marcadores, pixels onde a inundação deve começar, são escolhidos. Cada um recebe um rótulo diferente.
2. Os vizinhos de cada marcador são inseridos em uma fila de prioridade com um nível de prioridade correspondente à intensidade do pixel (baixo valor = alta prioridade).
3. O pixel com o nível de prioridade mais alto é extraído da fila de prioridade:
 - Se todos os vizinhos que já foram rotulados tenham o mesmo rótulo, marque o pixel com esse rótulo.
 - Todos os vizinhos não rotulados que ainda não estão na fila de prioridade são inseridos na fila de prioridade.
4. Repita o passo 3 até a fila de prioridade estar vazia.

Os pixels não rotulados são as linhas das bacias hidrográficas.

Algoritmo Watershed

- Para obter objetos homogêneos, o algoritmo Watershed é aplicado à magnitude do gradiente



imagem

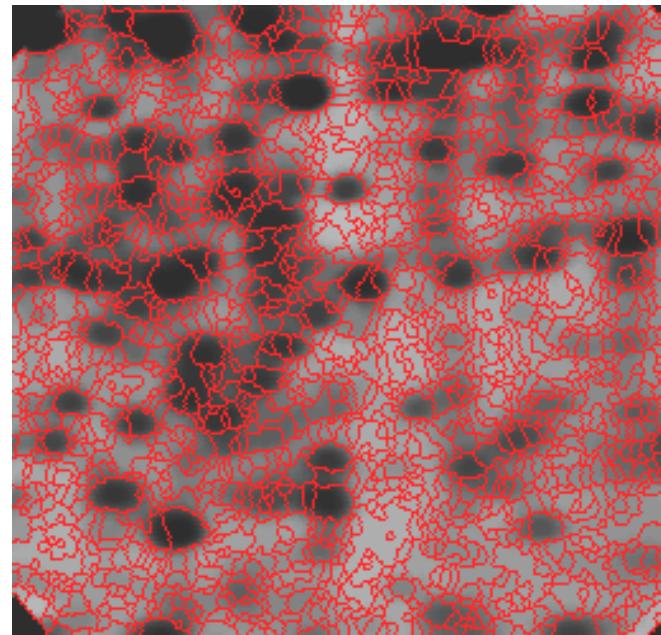
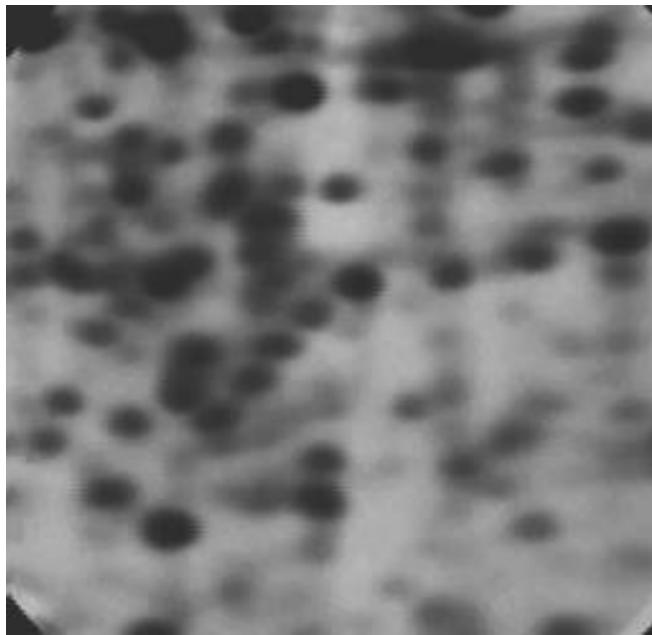
gradiente

bordas dos
segmentos sobre o
gradiente

bordas dos
segmentos sobre a
imagem original

Algoritmo Watershed

- Devido ao ruído e a irregularidades locais, a aplicação direta do algoritmo anterior leva à super-segmentação (*over segmentation*).



- Solução: limitar o número de regiões permitidas (sementes).

Algoritmo Watershed



marcadores (sementes)

segmentação final

Algoritmo do Spring

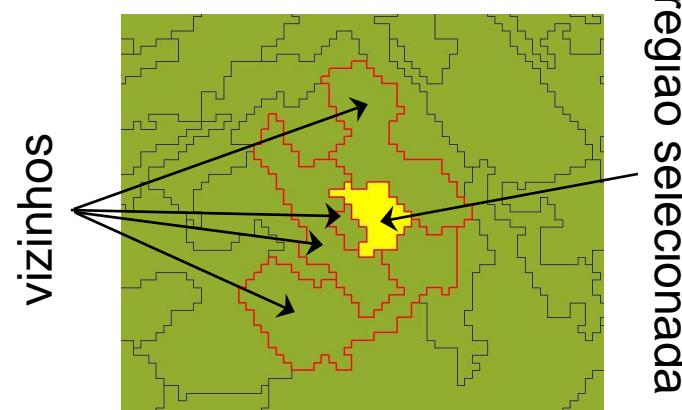
- Algoritmo de segmentação de crescimento de regiões implementado no *software* Spring (INPE).
- Inicialmente, cada pixel é considerado um segmento.
- Em cada etapa, os segmentos são fundidos com exatamente um vizinho: o vizinho mais "semelhante".

Algoritmo do Spring

- Similaridade entre as regiões é baseada na distância Euclidiana entre seus valores médios de intensidade (considerando todas as bandas):

$$D(R_i, R_k) = \|M_i - M_k\|$$

- Dois parâmetros:
 - Limiar de distância (T)
 - Área minima dos segmentos

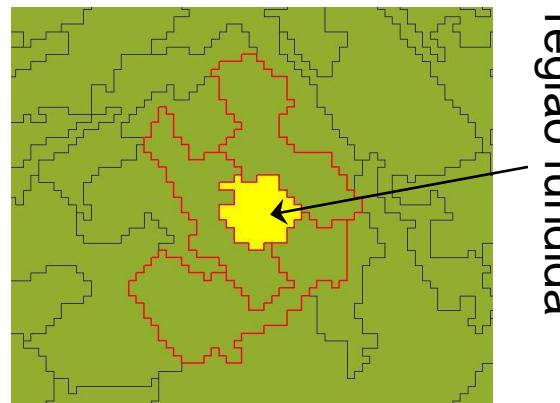


Algoritmo do Spring

- Similaridade entre as regiões é baseada na distância Euclidiana entre seus valores médios de intensidade (considerando todas as bandas):

$$D(R_i, R_k) = \|M_i - M_k\|$$

- Dois parâmetros:
 - Limiar de distância (T)
 - Área minima dos segmentos



Algoritmo do Spring

1. Uma lista de regiões $\{R_i; i = 1, \dots, n\}$ é criada (n é o número de pixels na imagem).
2. Para cada região R_i , o vetor do valor médio e as regiões vizinhas são armazenadas.
3. Para cada região R_i , as regiões vizinhas $N(R_i)$ são examinadas e:
 - a região vizinha mais semelhante $R_k \in N(R_i)$ é escolhida. Se $D(R_i, R_k) < T$, então R_k é chamado de "o melhor vizinho" de R_i .
 - Se o melhor vizinho de R_k existe e é R_i , então ambas as regiões são fundidas.
4. Quando uma região é fundida com outra, ela é retirada da lista.
5. O valor médio das regiões fundidas é atualizado.
6. Repetir a partir do passo 3 até que não existam regiões passíveis de fusão.
7. Pequenas regiões são fundidas com regiões adjacentes maiores, de acordo com um limiar de área definido pelo usuário.

Baatz & Shäpe Algorithm

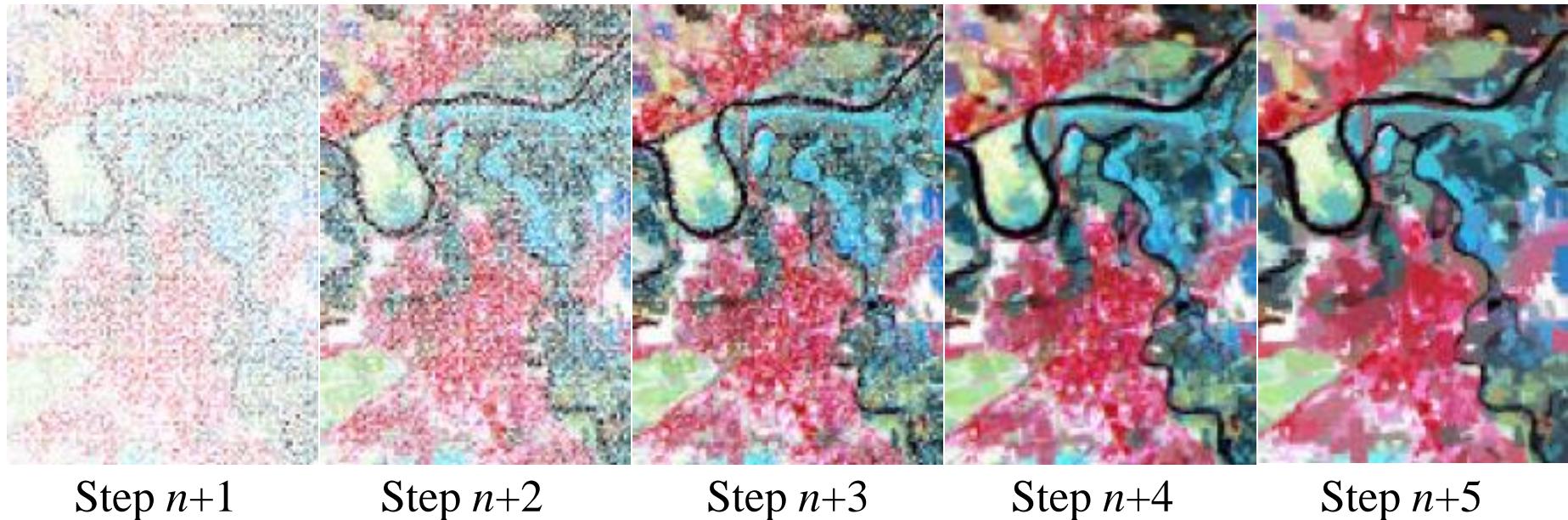
- Originalmente implementado no *software* eCognition (Definiens).
- Algoritmo multi-escala: possui parâmetro específico que controla o tamanho das regiões resultantes.
- Regiões/segmentos são caracterizados pela sua heterogeneidade interna.

$$h_{total} = w \cdot h_{color} + (1 - w) \cdot h_{shape}$$

Baatz & Shäpe Algorithm

- Inicialmente, cada pixel é considerado um segmento.
- Em cada etapa, os segmentos são fundidos com exatamente um vizinho: o que promove o menor aumento na heterogeneidade global.
- A seleção de segmentos para crescimento é "quase" aleatória: distribui as fusões subsequentes o máximo possível dentro da imagem.

Baatz & Shäpe Algorithm



Baatz & Shäpe Algorithm

- A medida de heterogeneidade tem um componente espectral (cor) e outro espacial (forma).

$$h_{total} = w \cdot h_{color} + (1 - w) \cdot h_{shape}$$

- A fusão de um segmento (*Seg1*) com um vizinho (*Seg2*) considera o aumento da heterogeneidade no segment resultante (*Seg3*).

Baatz & Shäpe Algorithm

Medida de aumento da heterogeneidade espectral:

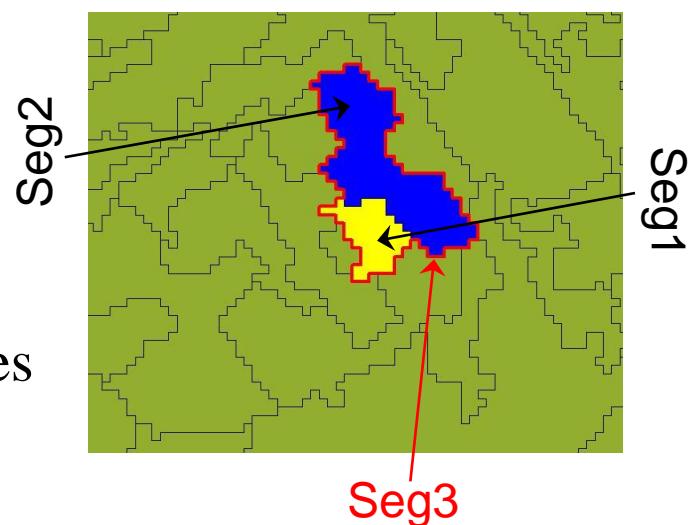
$$f_{color} = \sum_c w_c \left(n_{Seg3} \cdot \sigma_c^{Seg3} - \left(n_{Seg1} \cdot \sigma_c^{Seg1} + n_{Seg2} \cdot \sigma_c^{Seg2} \right) \right)$$

C = banda espectral

w_c = peso da banda C

n = área do segment (pixels)

σ_c = desvio padrão das intensidades
dos pixels



Baatz & Shäpe Algorithm

Medida do aumento da heterogeneidade de forma:

$$f_{shape} = w_{cmpct} \cdot f_{cmpct} + (1 - w_{cmpct}) \cdot f_{smooth}$$

f_{cmpct} = componente de compacidade

f_{smooth} = componente de suavidade

w_{cmpct} = peso compacidade/suavidade

Baatz & Shäpe Algorithm

Componente de compacidade na heterogeneidade:

$$f_{cmpct} = n_{Seg3} \cdot \frac{l_{Seg3}}{\sqrt{n_{Seg3}}} - \left(n_{Seg1} \cdot \frac{l_{Seg1}}{\sqrt{n_{Seg1}}} + n_{Seg2} \cdot \frac{l_{Seg2}}{\sqrt{n_{Seg2}}} \right)$$

Componente de suavidade na heterogeneidade:

$$f_{smooth} = n_{Seg3} \cdot \frac{l_{Seg3}}{b_{Seg3}} - \left(n_{Seg1} \cdot \frac{l_{Seg1}}{b_{Seg1}} + n_{Seg2} \cdot \frac{l_{Seg2}}{b_{Seg2}} \right)$$

n = área do segmento

l = perímetro do segmento

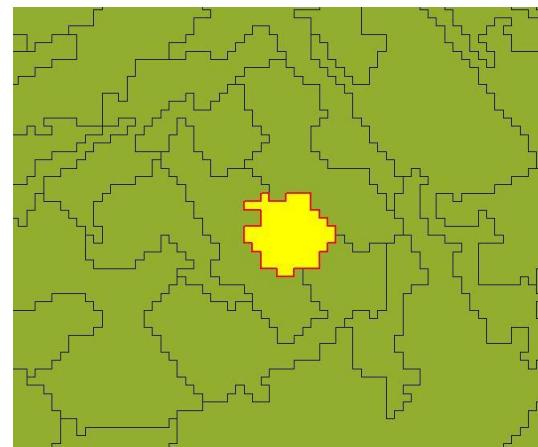
b = perímetro da caixa envoltória do segmento

Baatz & Shäpe Algorithm

Aumento combinado de heterogeneidade (fator de fusão):

$$f = w_{color} \cdot f_{color} + (1 - w_{color}) \cdot f_{shape}$$

w_{color} = peso cor/forma



Baatz & Shäpe Algorithm

Aumento combinado de heterogeneidade (fator de fusão):

$$f = w_{color} \cdot f_{color} + (1 - w_{color}) \cdot f_{shape}$$

w_{color} = peso cor/forma

Para uma fusão ocorrer, o fator de fusão deve ser menor que o quadrado do *parâmetro de escala (sp)*

Baatz & Shäpe Algorithm

Combined increase in heterogeneity (**fusion factor**):

$$f = w_{color} \cdot f_{color} + (1 - w_{color}) \cdot f_{shape}$$

w_{color} = weight of color against shape

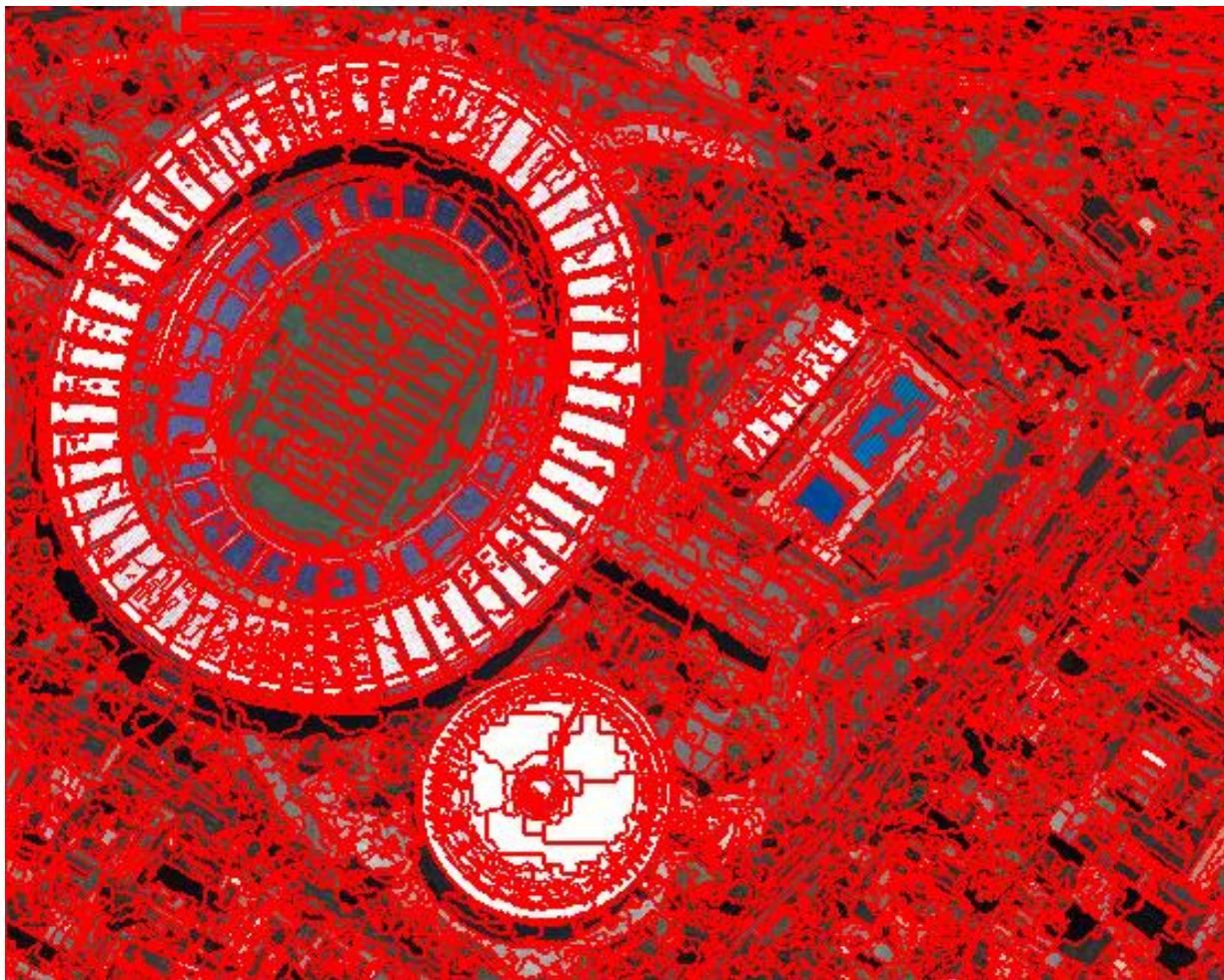
For a merge to occur, fusion factor must be less than the square of the **scale parameter (sp)**.

Baatz & Shäpe Algorithm



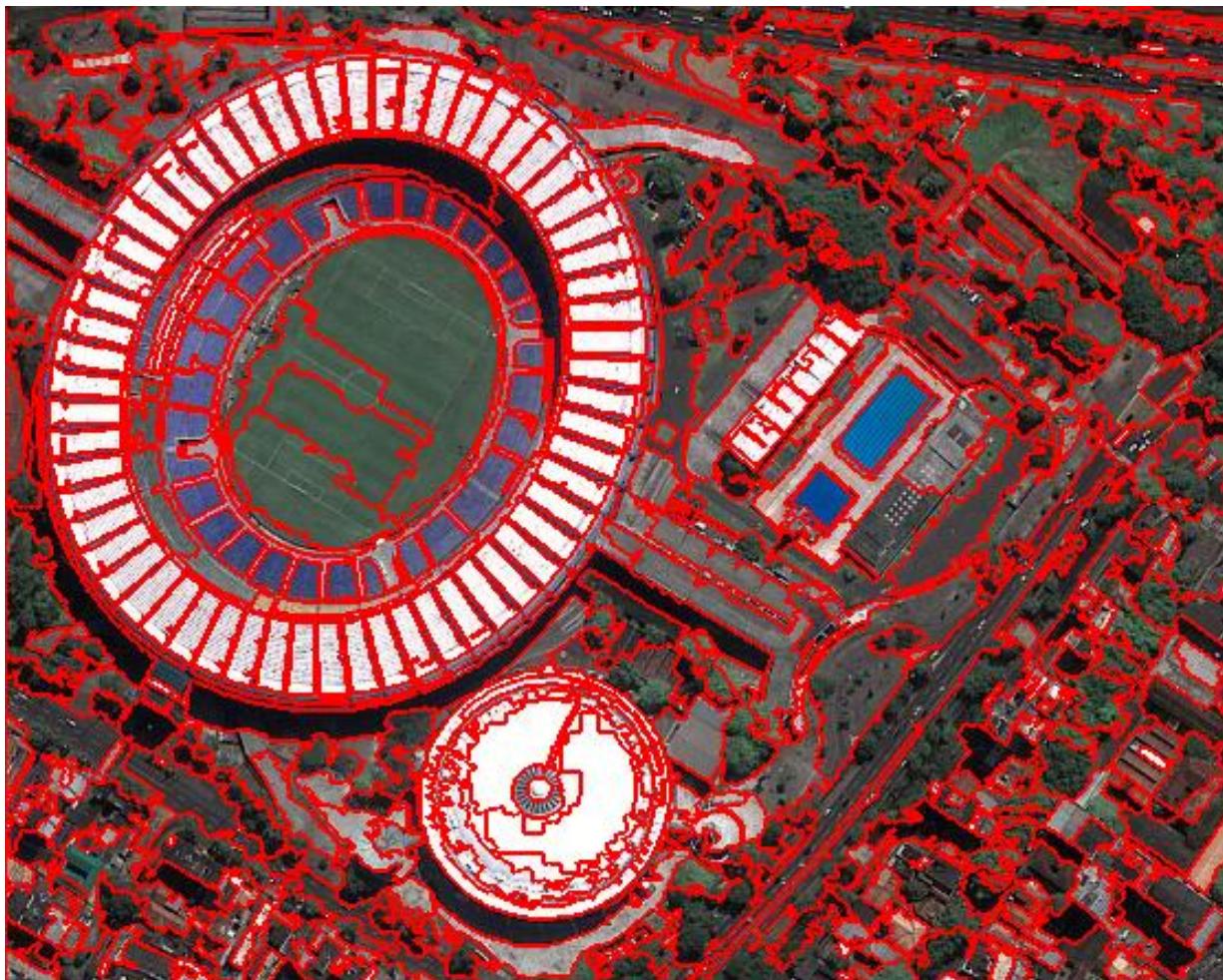
Imagen original

Baatz & Shäpe Algorithm



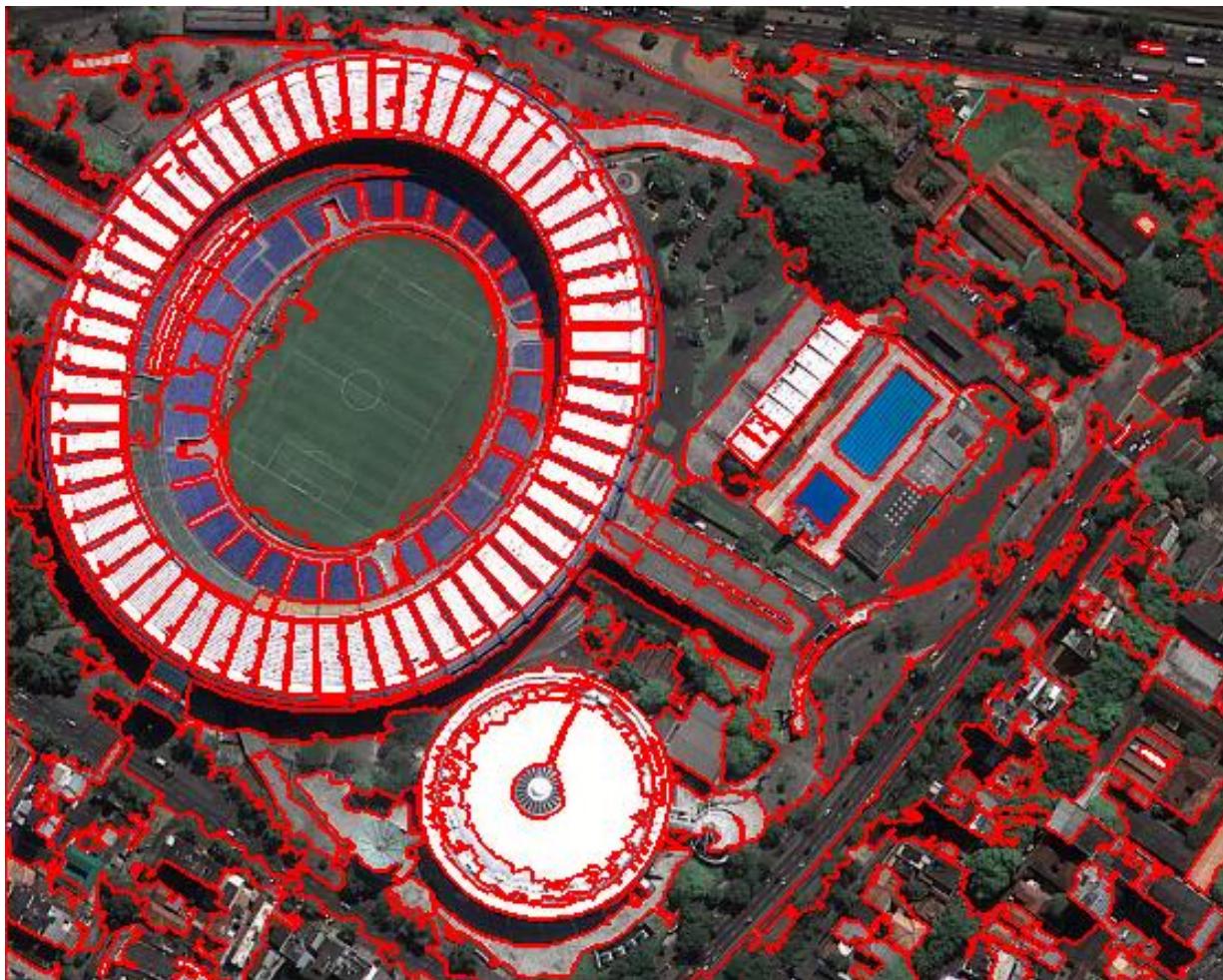
$$sp = 10 \quad w_{color} = 0.9 \quad w_{cmpct} = 0.5$$

Baatz & Shäpe Algorithm



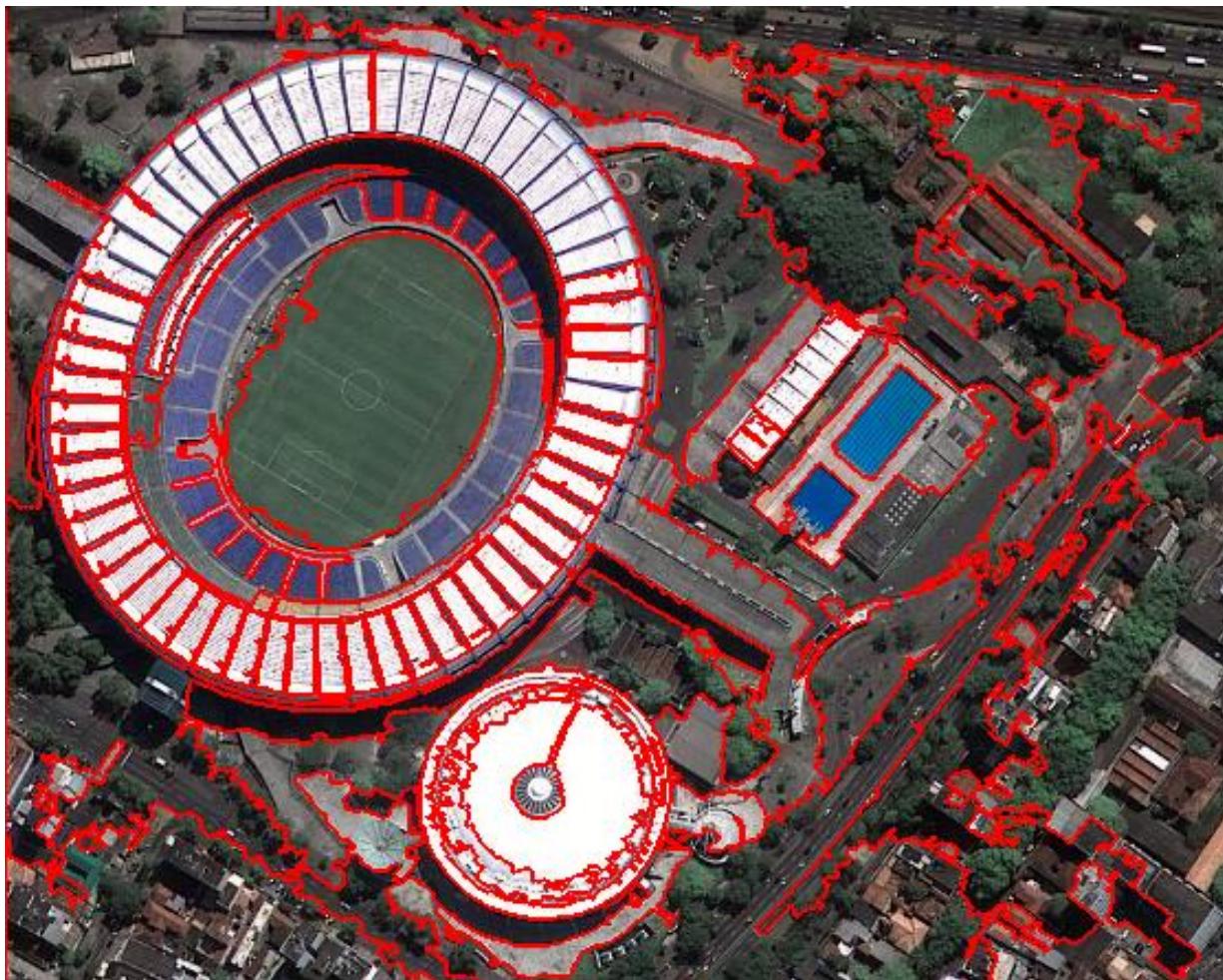
$$sp = 50 \quad w_{color} = 0.9 \quad w_{cmpct} = 0.5$$

Baatz & Shäpe Algorithm



$$sp = 100 \quad w_{color} = 0.9 \quad w_{cmpct} = 0.5$$

Baatz & Shäpe Algorithm



$$sp = 200 \quad w_{color} = 0.9 \quad w_{cmpct} = 0.5$$

Baatz & Shäpe Algorithm



$$sp = 200 \quad w_{color} = 0.5 \quad w_{cmpct} = 0.8$$