Unidade II – Imagem Digital (Parte 2)

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

Segmentação de Imagens

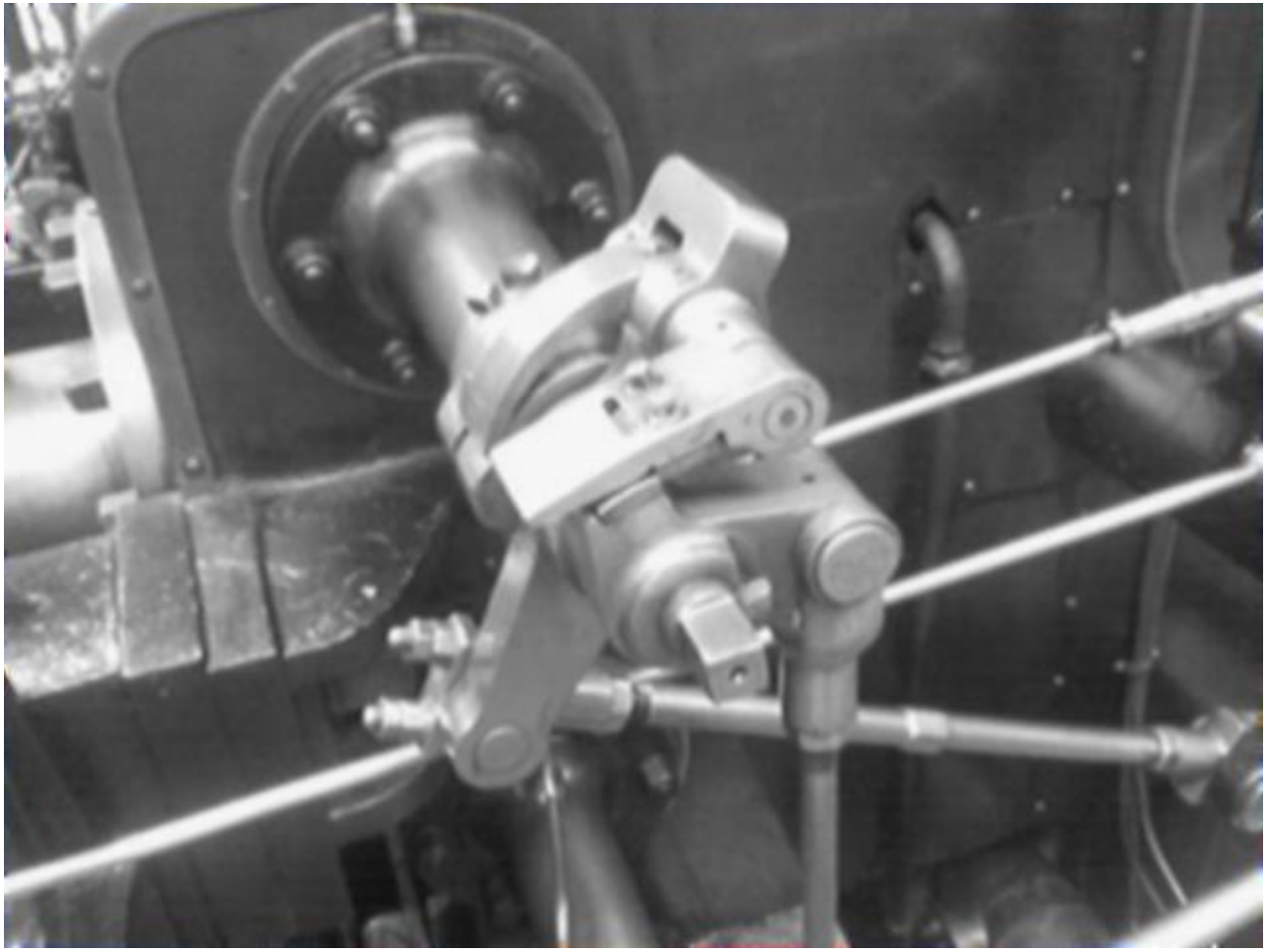
- Objetivo

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

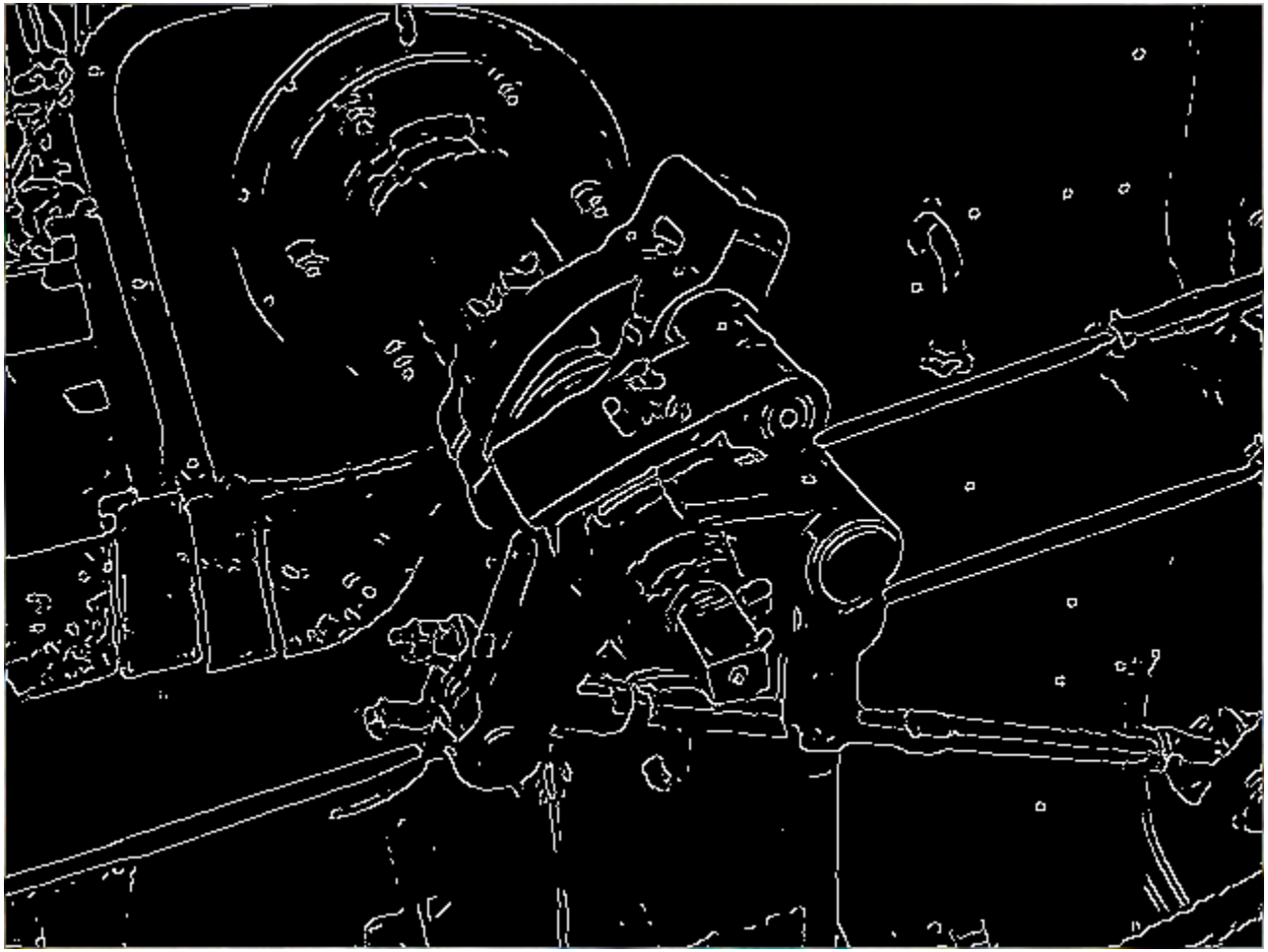
Segmentação de Imagens

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

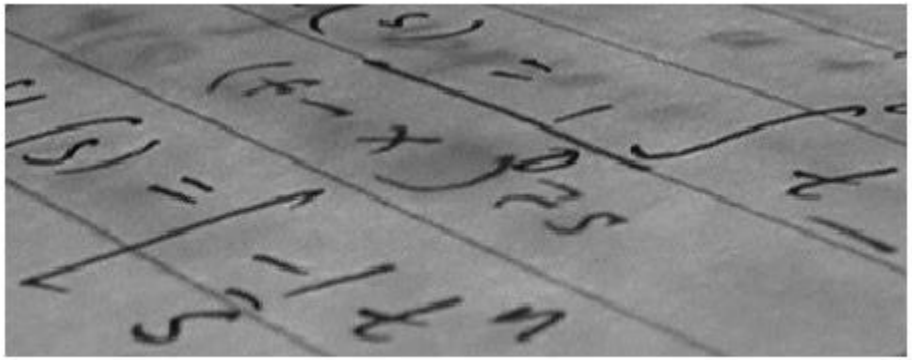
Descontinuidade (bordas)

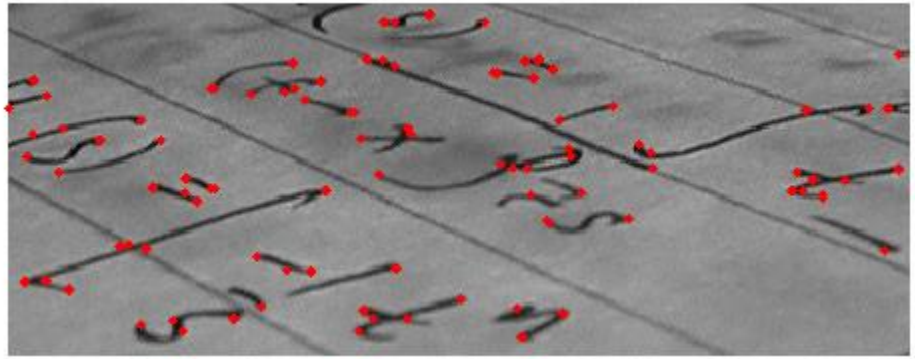


Descontinuidade (bordas)



Descontinuidade (cantos)





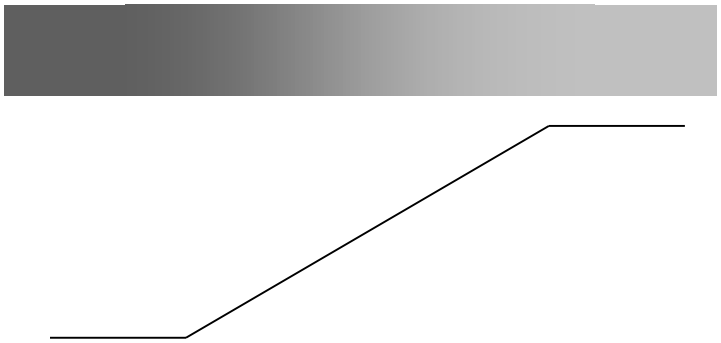
Similaridade (regiões)

Similaridade (regiões)

Detecção de Bordas

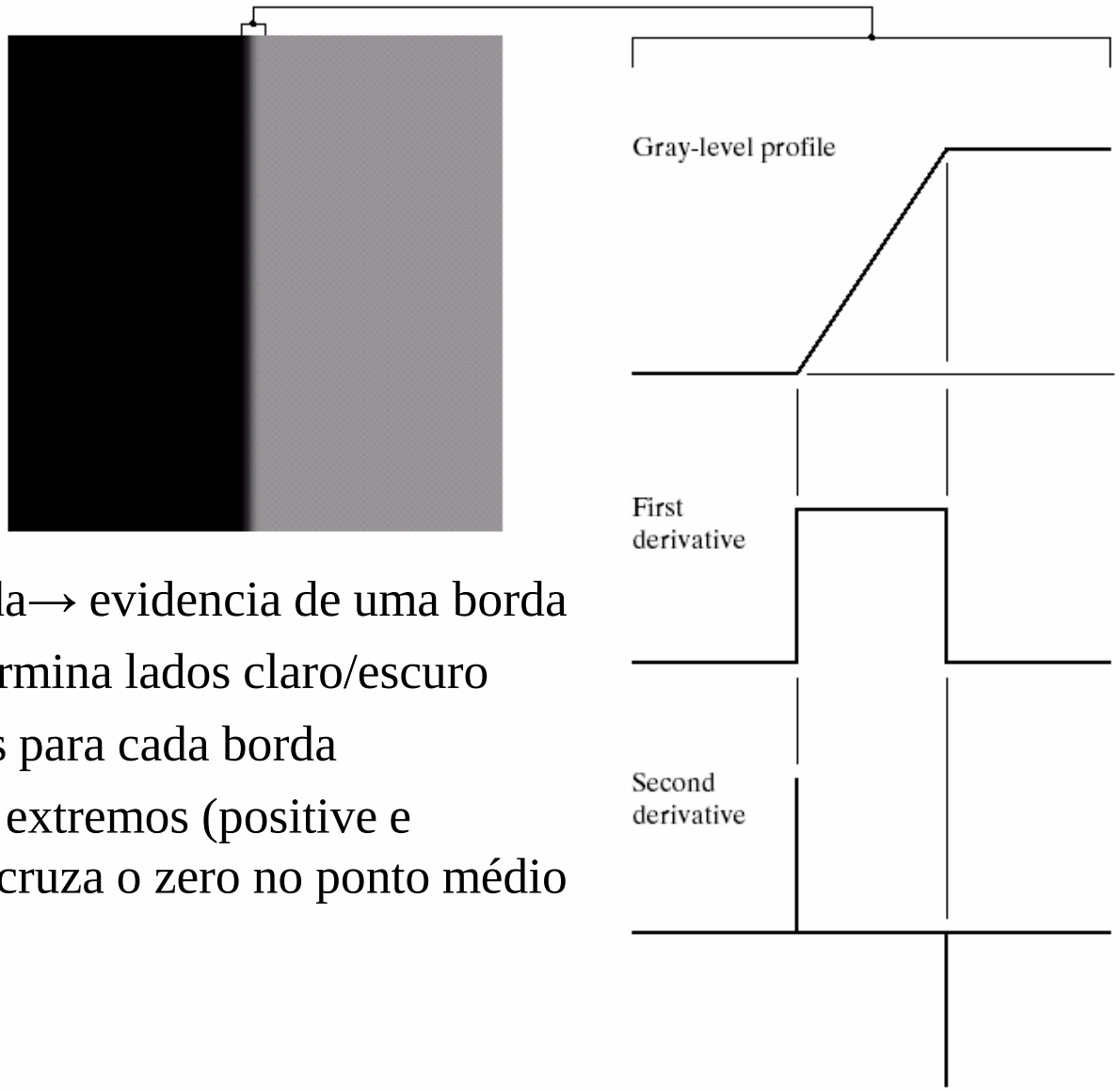
- Formulação básica:
  - *Borda ideal:* conjunto de pixels conectados, localizados em uma transição ortogonal dos valores de intensidade
  - *Borda real*: tem um perfil de "rampa"

Borda ideal



Borda real

Detecção de Bordas



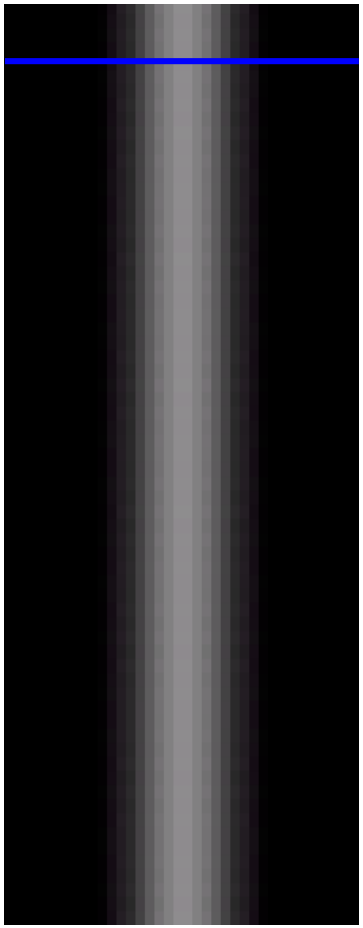
- **Conclusões:**

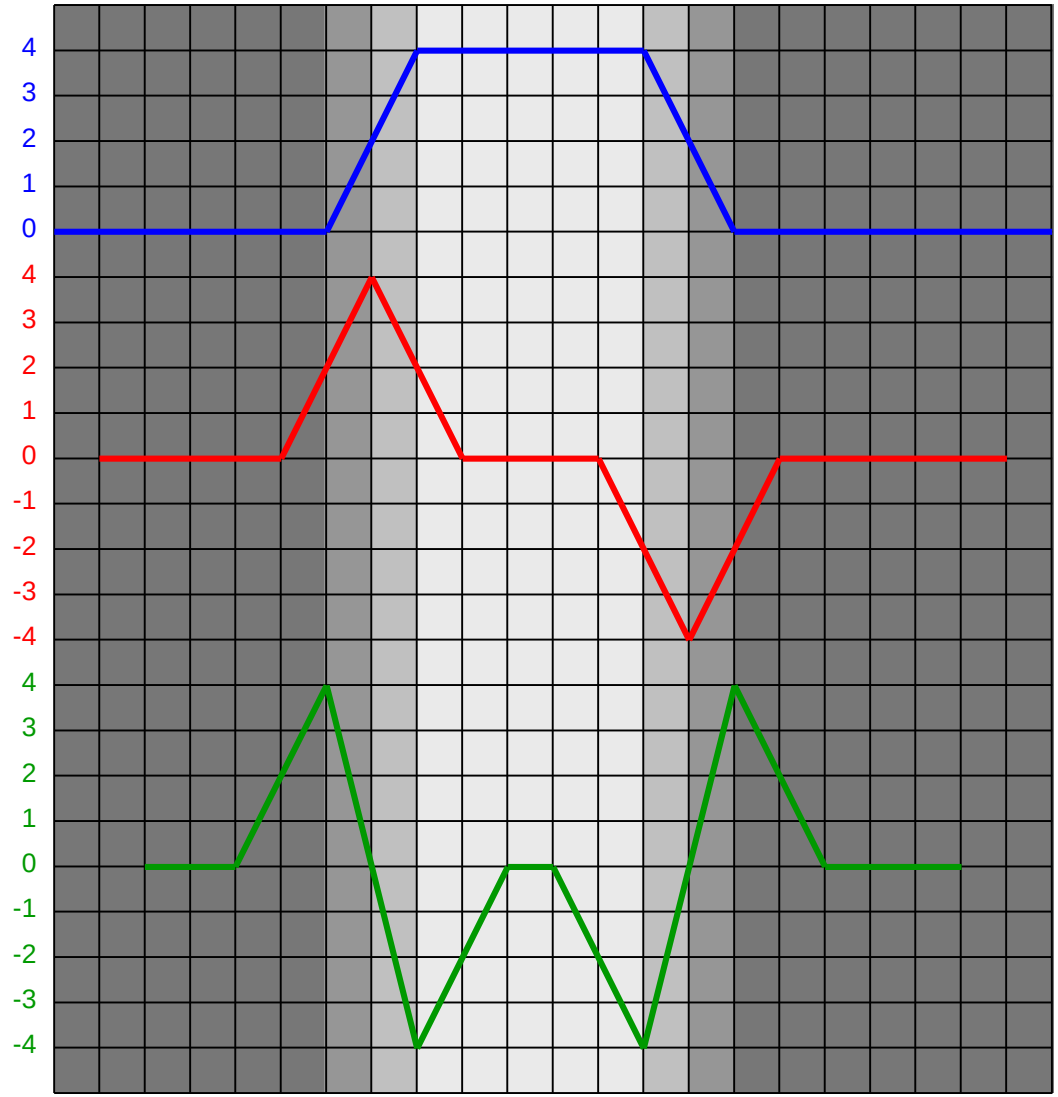
1. Magnitude da 1ª derivada→ evidencia de uma borda
2. Sinal da 2ª derivada determina lados claro/escuro
3. 2ª derivada tem 2 valores para cada borda
4. Linha ligando os valores extremos (positive e negative) da 2ª derivada cruza o zero no ponto médio

Detecção de Bordas

-1	0	1
-1	0	1
-1	0	1

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

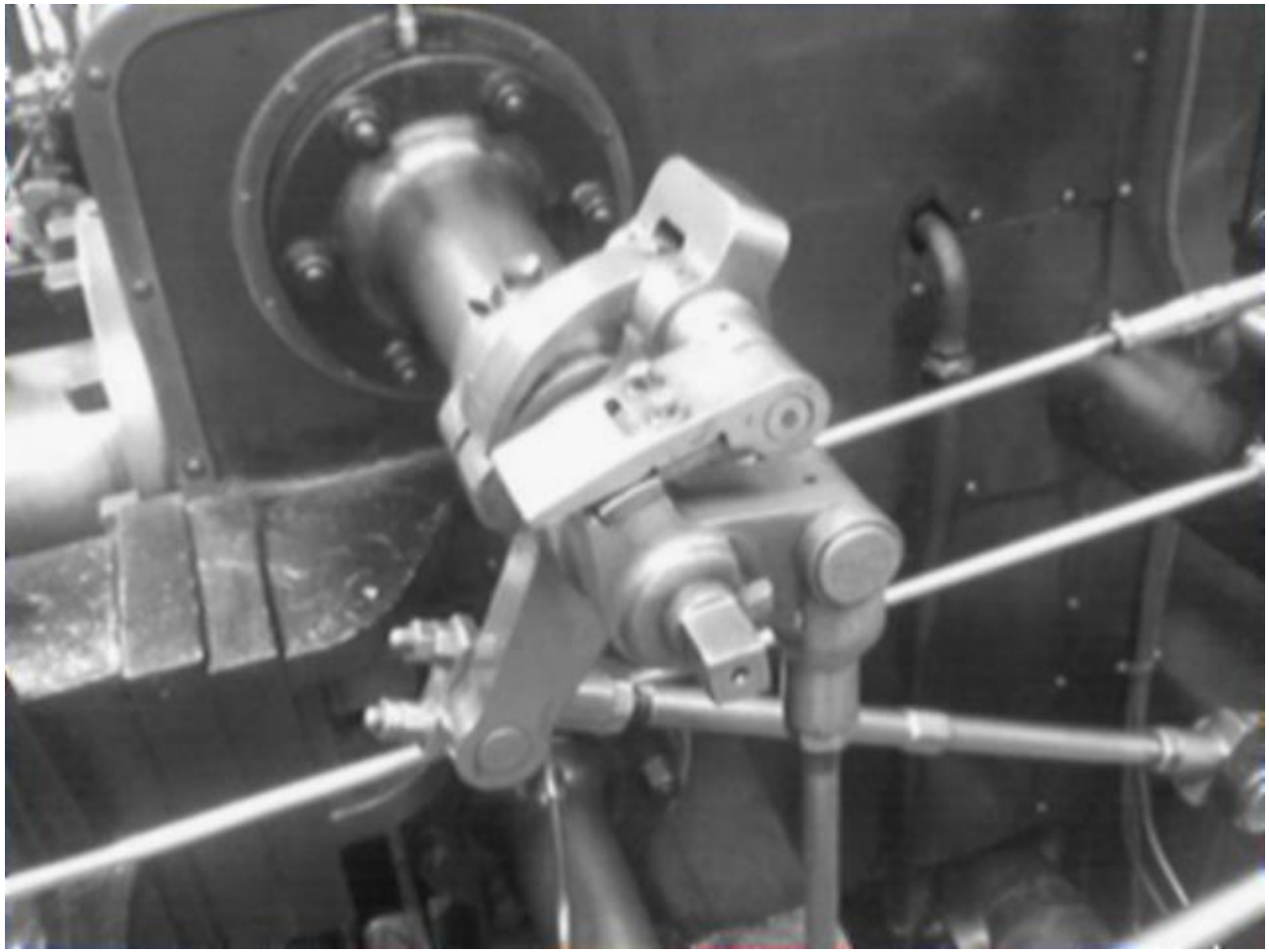
$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2} \approx |G_x| + |G_y|$$

magnitude

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

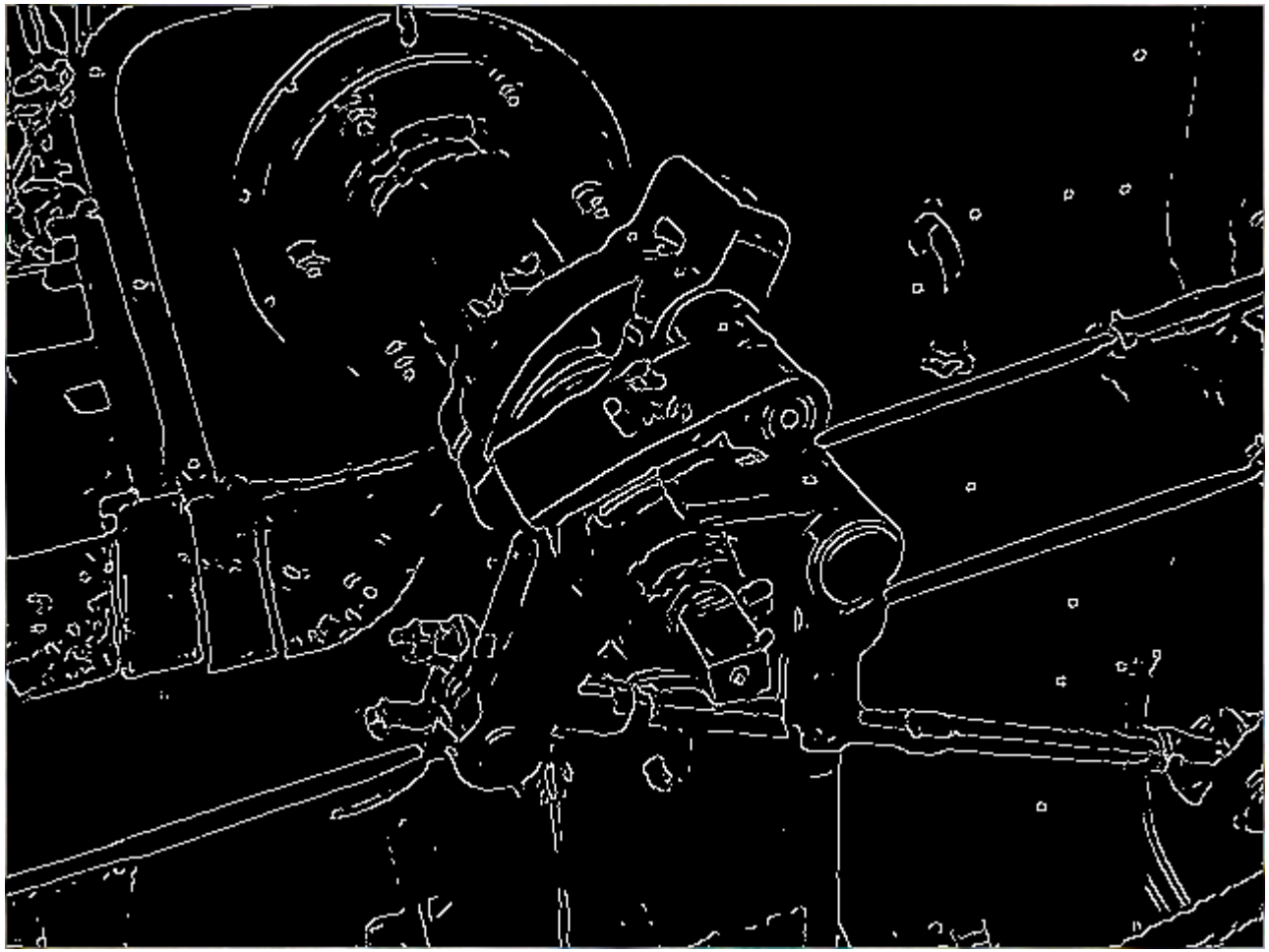
ângulo

Detecção de Bordas



Detecção de Bordas

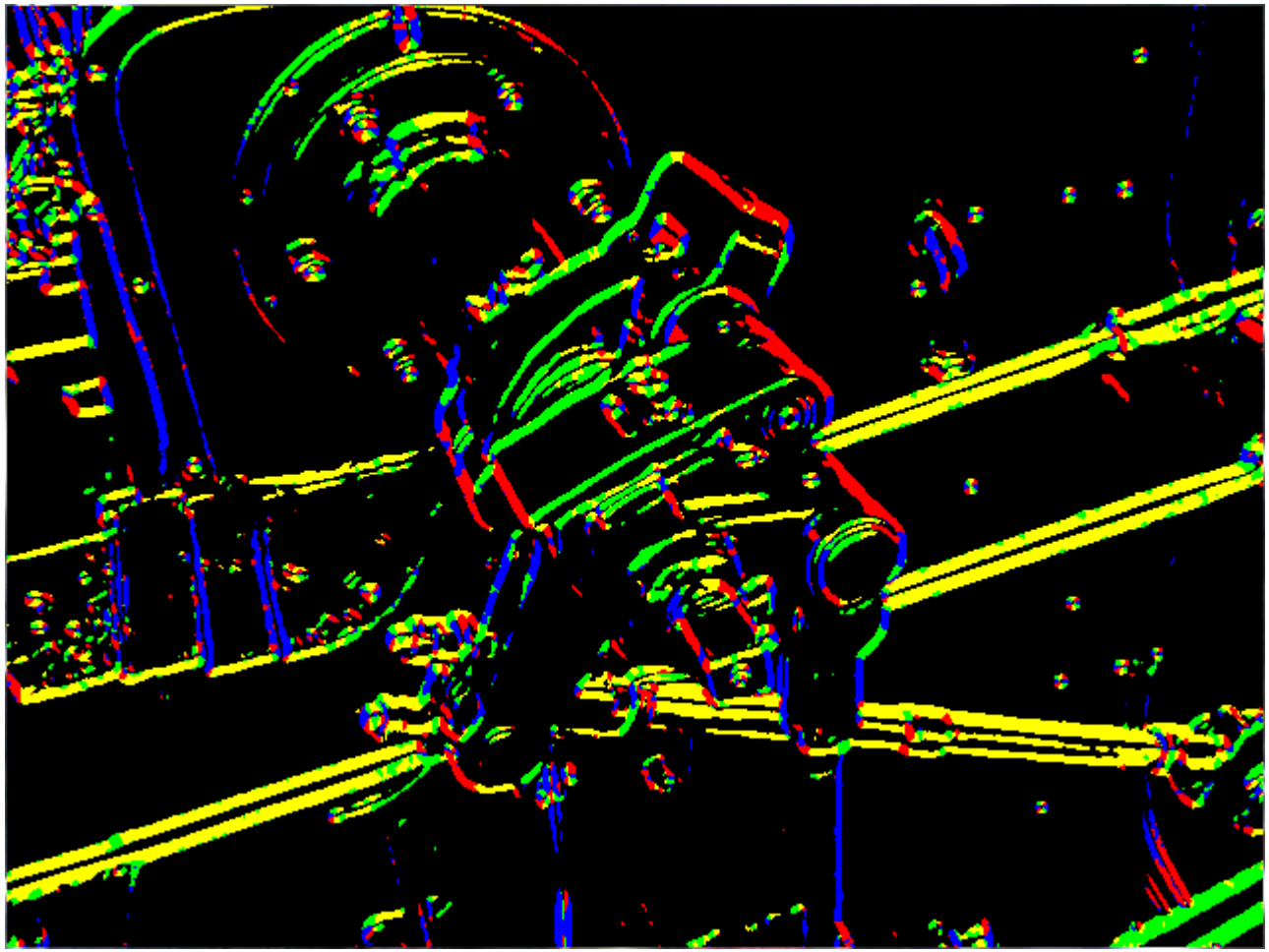
$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[G_x^2 + G_y^2\right]^{1/2} \approx |G_x| + |G_y|$$



magnitude > 80

Detecção de Bordas

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



ângulo: azul ~ 0º/180º; verm. ~ 45º/225º; amarelo ~ 90º/270º; verde ~ 135º/315º

Detecção de Bordas

- Derivada de 1a. Ordem:

  Gradiente

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

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

magnitude

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

ângulo

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

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

Máscaras de Sobel para G_y e G_x

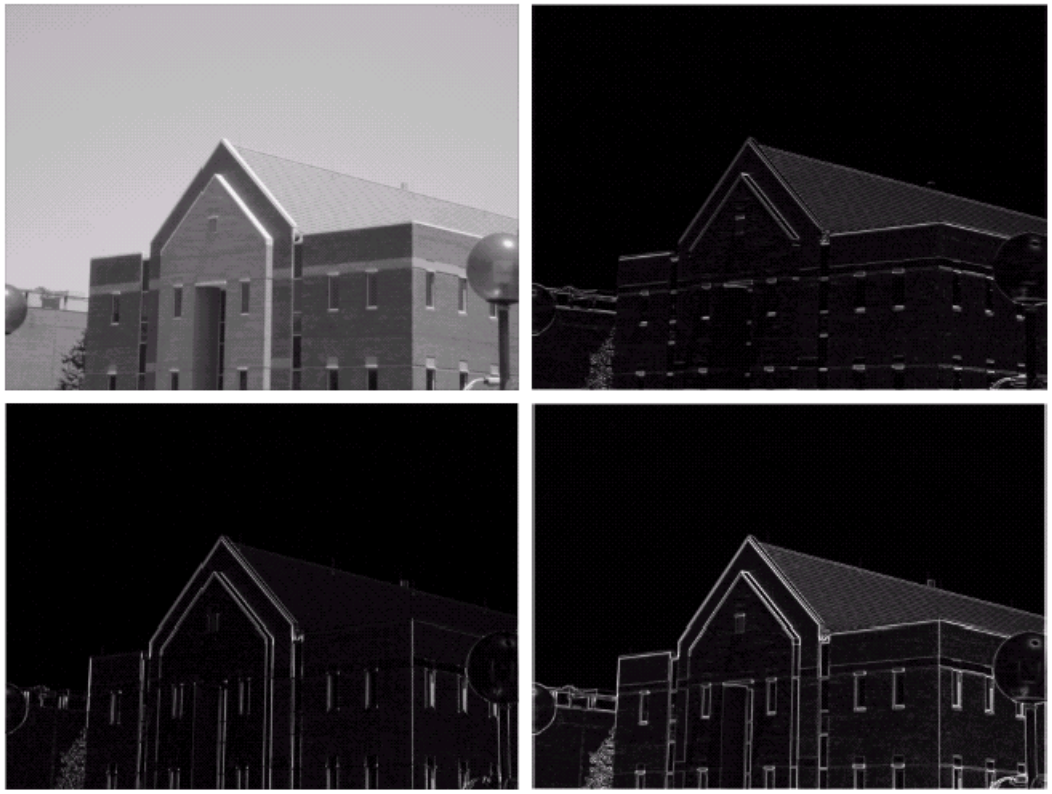
0	1	2
-1	0	1
-2	-1	0

-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**

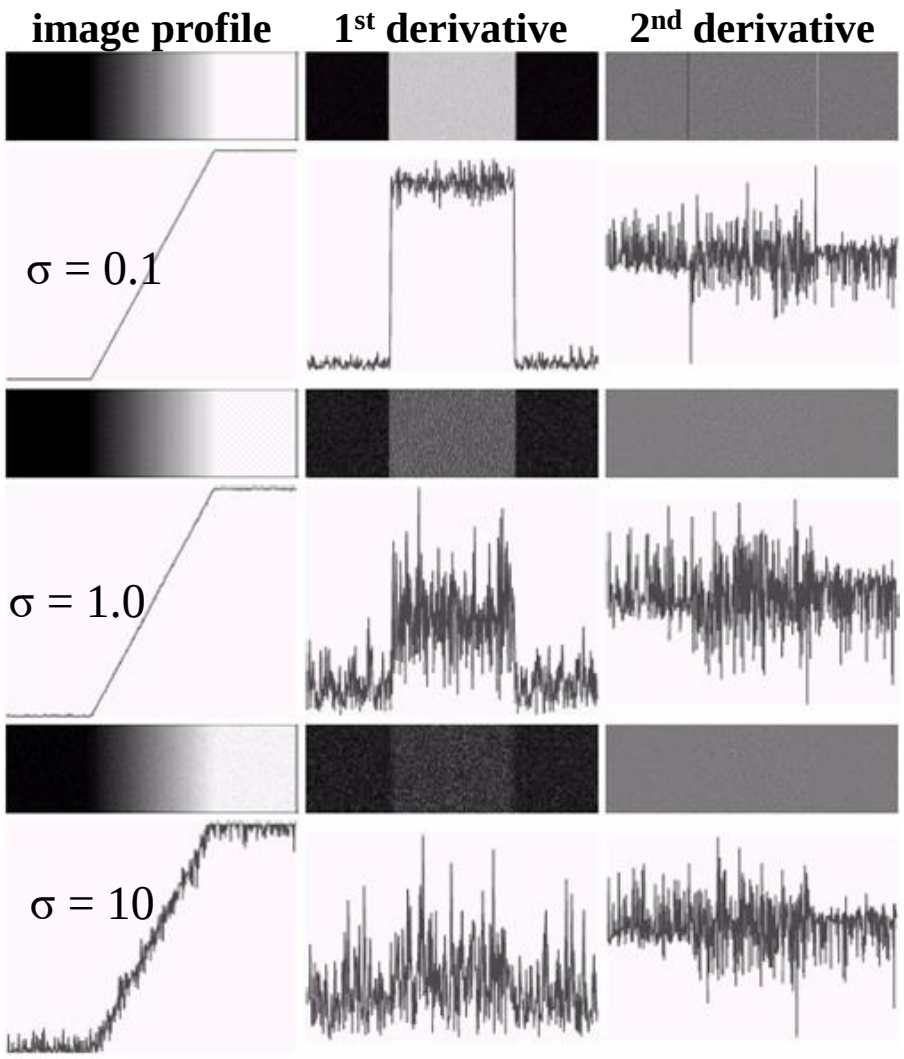
$$\nabla^2 \mathbf{f} = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

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

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

$$h_{LoG}(x,y) * f(x,y)$$

Máscara para: *Laplacian of Gaussian*

$f(x,y)$ – imagem

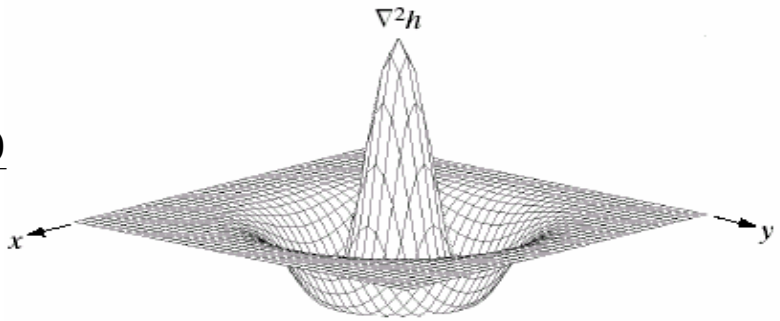
$h_L(x,y)$ – máscara laplaciana

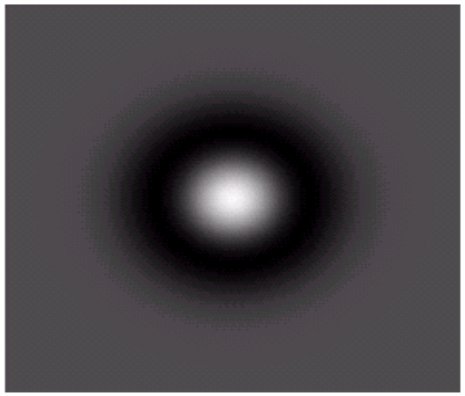
$h_G(x,y)$ – máscara gaussiana (suavização)

Detecção de Bordas

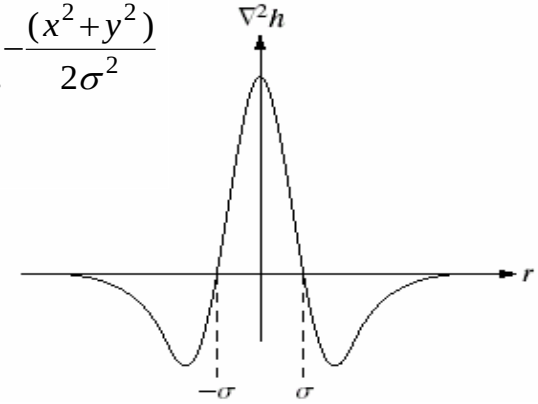
- Filtro LoG

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





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



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

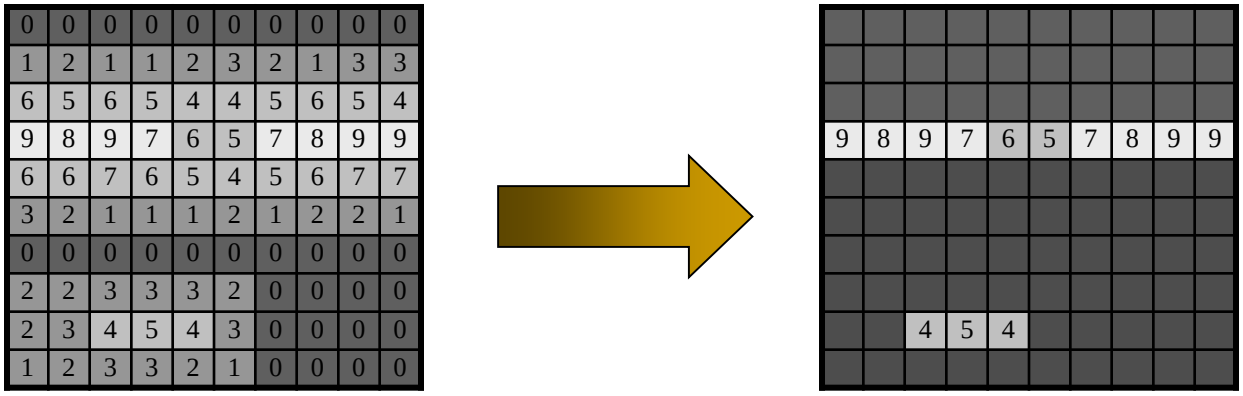
Cruzamento por zero

Ligação 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

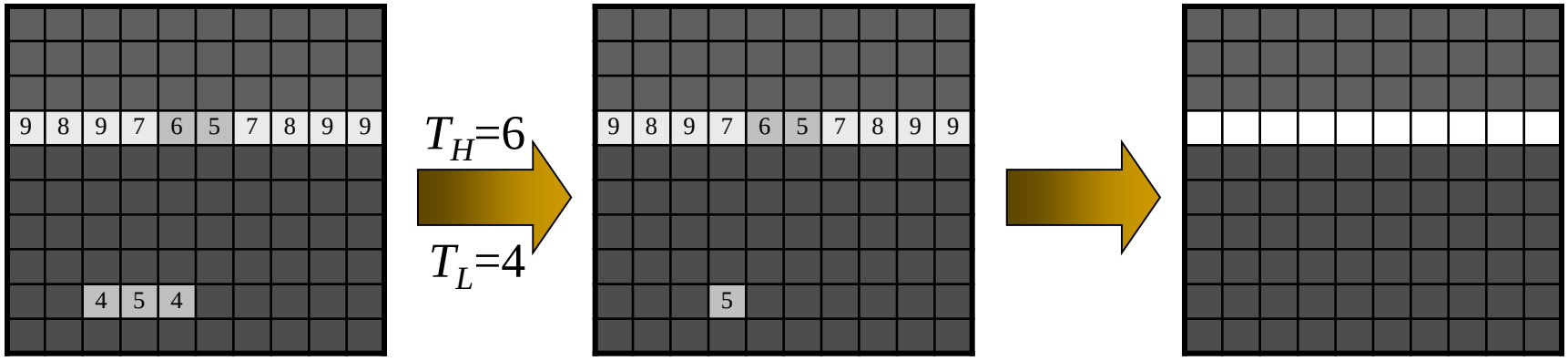
1. Smooth the image $f(x,y)$ with a Gaussian filter.
2. Compute the magnitude $\nabla f(x,y)$ and direction $\alpha(x,y)$ of the gradient.
3. Pixels having the magnitude greater than their neighbors in the gradient direction are edge pixels; other pixels are dismissed (*non maximal suppression*).



magnitude edge pixels

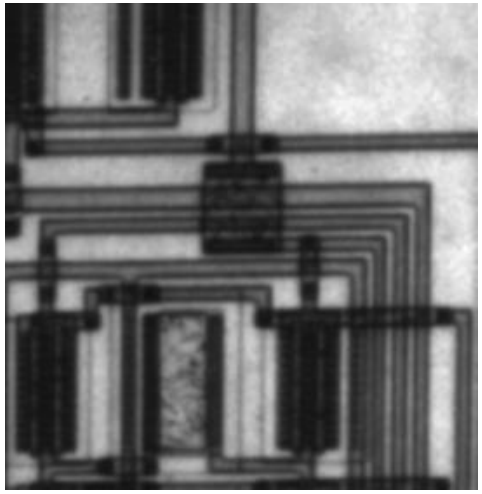
Canny Algorithm

4. Edge points with magnitude greater than a high threshold T_H are "strong" edges; the other edge points whose magnitude is greater than a lower threshold T_L $(T_L < T_H)$ are "weak" edge points.
5. Take the "strong" edge points AND the "weak" edge points connected to "strong" edge points.

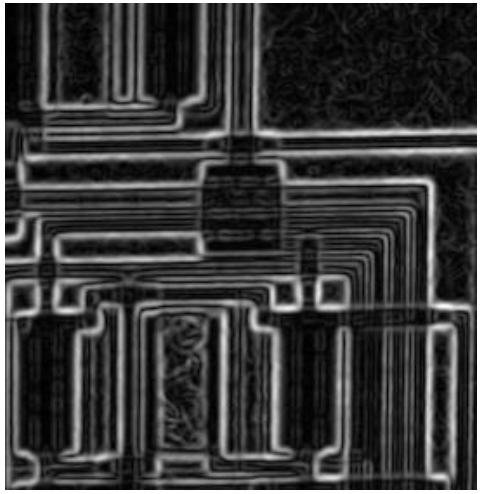


edge pixels strong and week points final edge

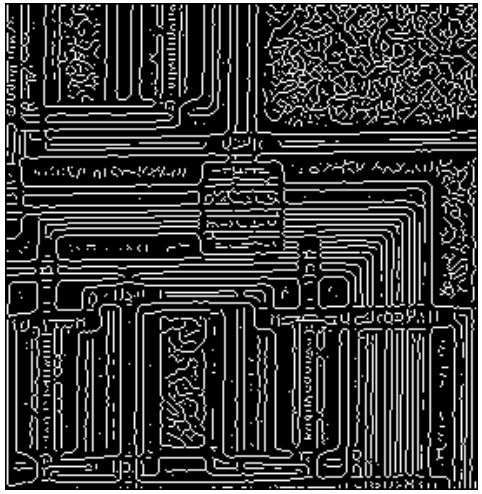
Canny Algorithm



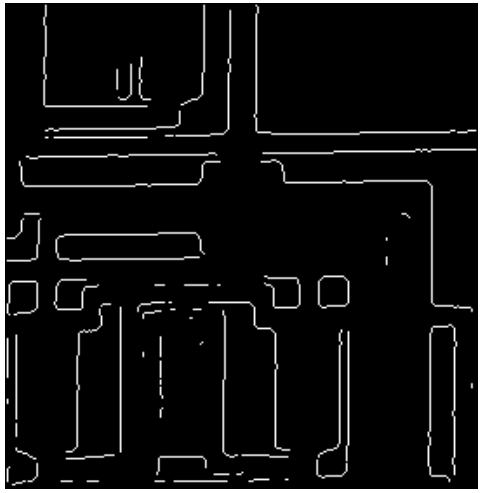
after gaussian filter



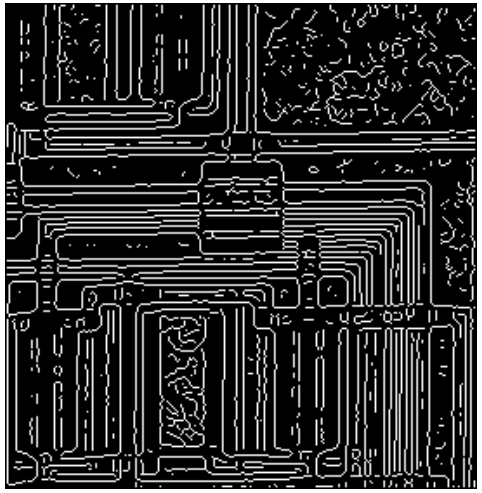
magnitude of the gradient



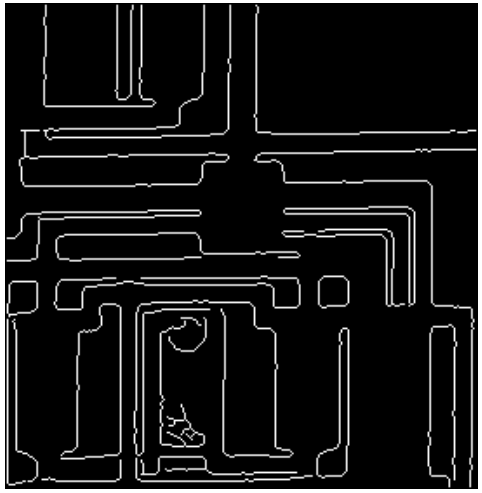
after non maximal suppression



edges > T_H

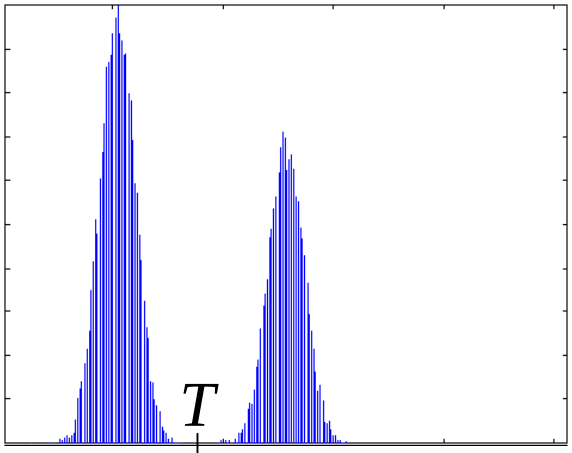


T_L <edges< T_H

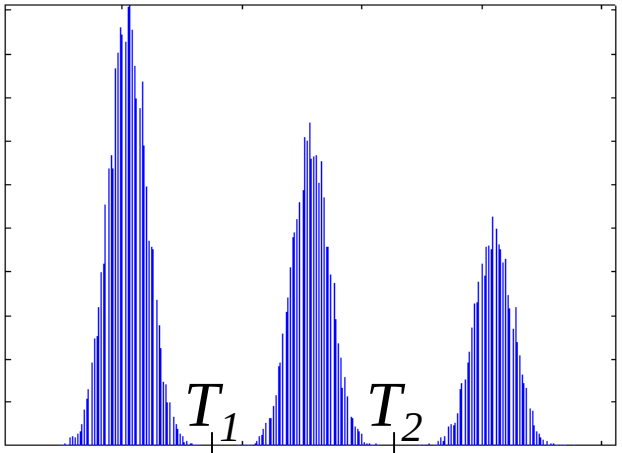


final result

Thresholding



Single threshold



Multiple thresholds

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

$T = T\,[f(x,y)]$	$T = T\,[p(x,y), f(x,y)]$	$T = T\,[x, y, p(x,y), f(x,y)]$
global	local	dynamic or adaptive

Global Thresholding

Automatic Threshold Determination: a simple algorithm

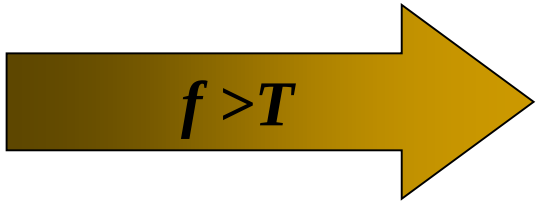
1. Select an initial threshold value T
2. Threshold the image using T:

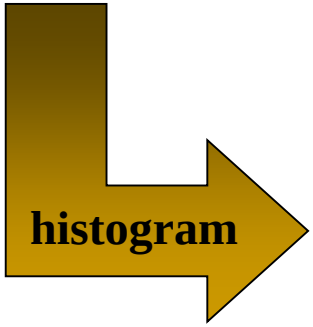
   This will generate two pixel groups: G_1 (pixels with values $< T$) and G_2 (pixels with values $\geq T$)
3. Compute the average value of each group (μ_1 and μ_2)
4. Compute a new threshold estimate according to

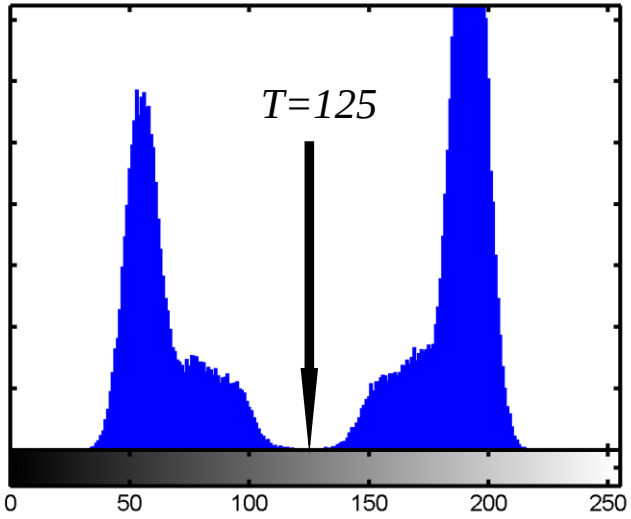
$$T = \tfrac{1}{2}(\mu_1 + \mu_2)$$

5. Repeat steps 2 to 4 till no significant change in T occurs in two consecutive iterations

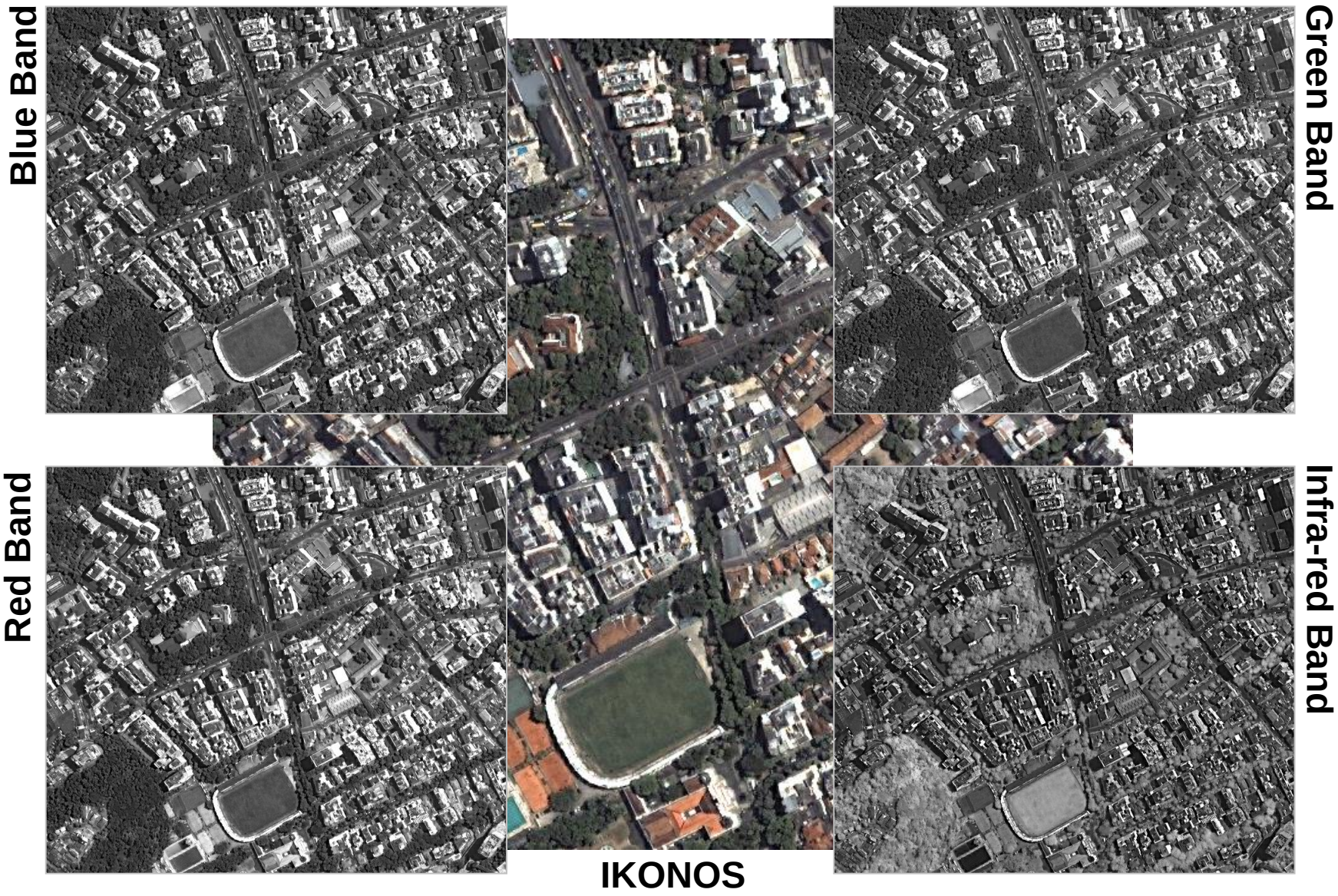
Global Thresholding







Remote Sensing Images

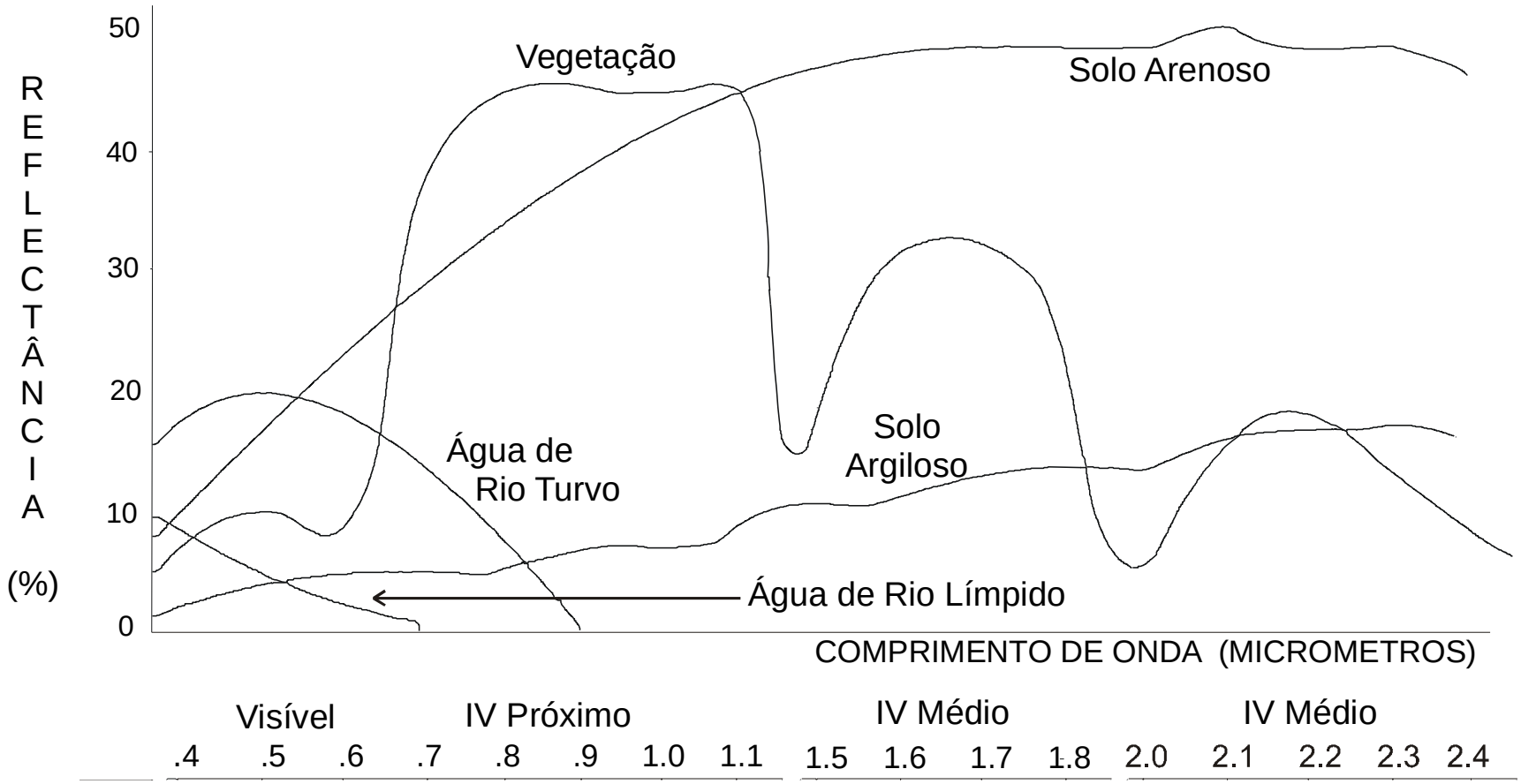


Remote Sensing Images

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

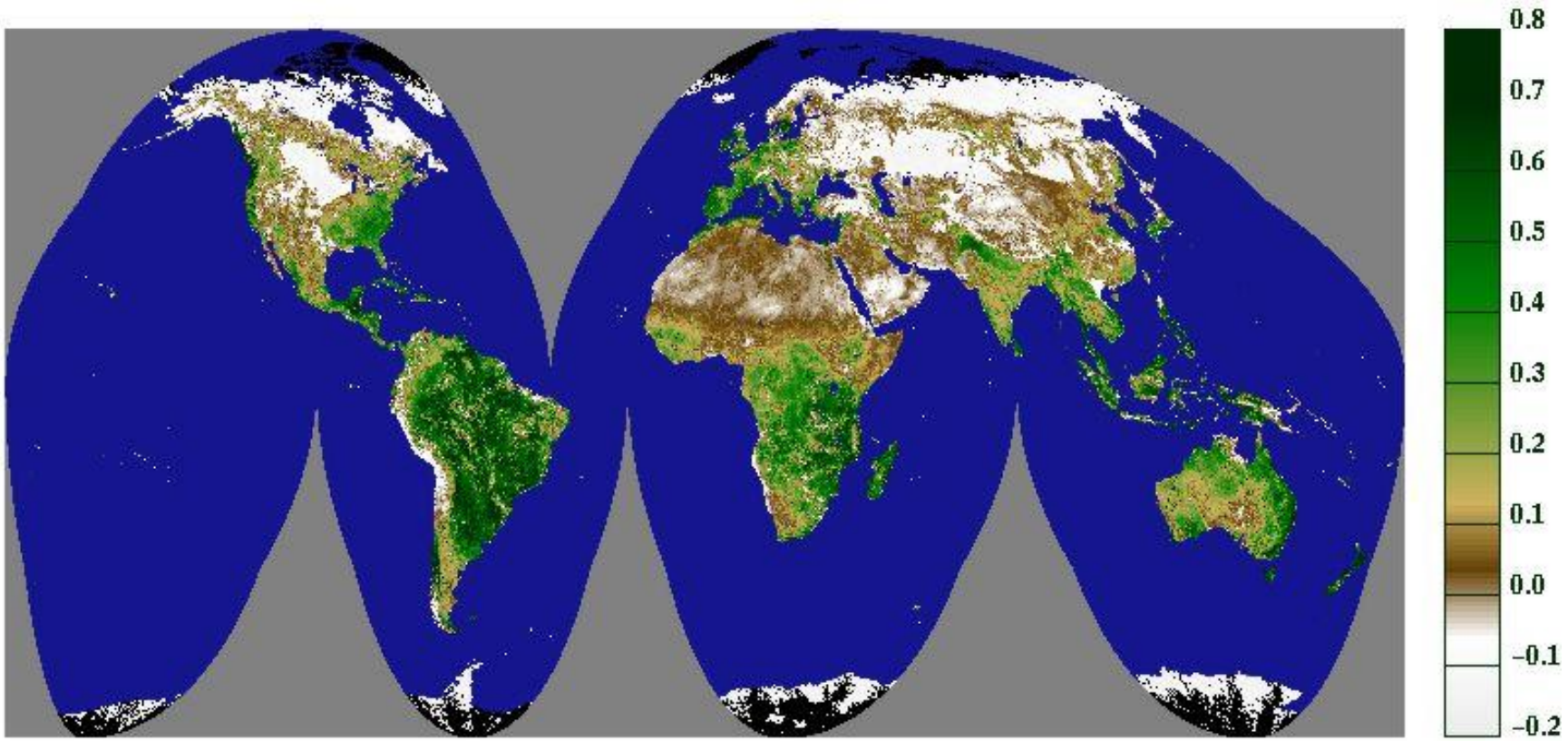
IKONOS

Remote Sensing Images



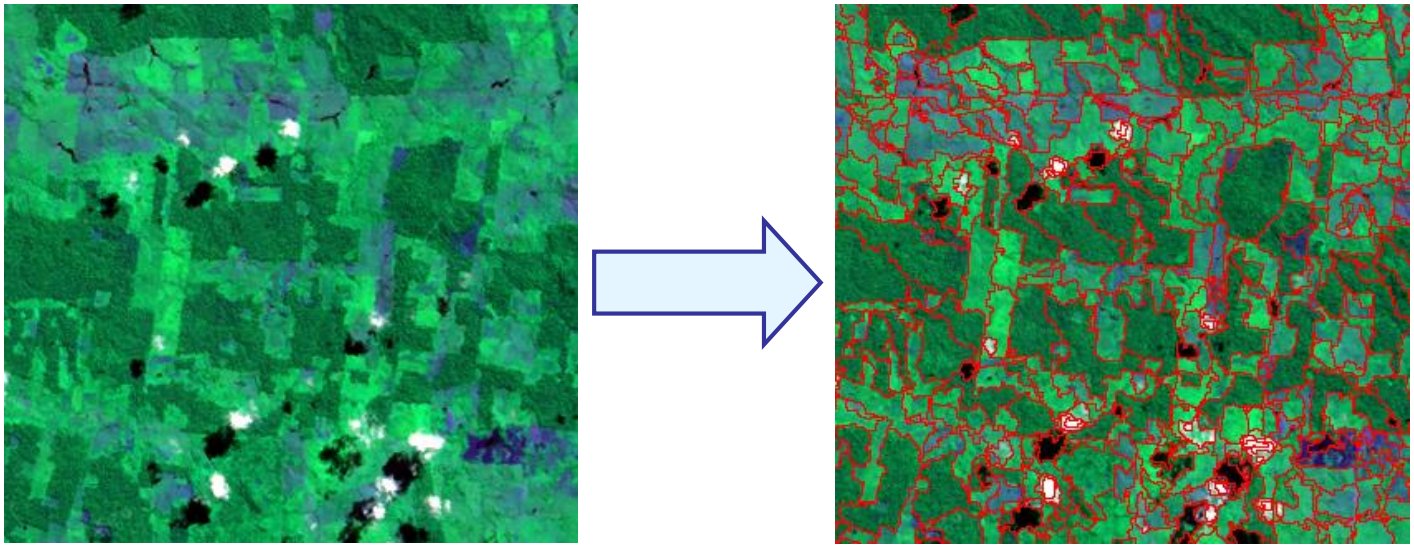
Remote Sensing Images

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



Region-Based Segmentation

Main goal is to identify meaningful *sub-regions* of an image.



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

Region-Based Segmentation

Basic Formulation

Let R represent an image. Segmentation may be viewed as the process of partitioning R in n subregions $R_1, R_2, \ldots, R_n$, such that:

a) $\bigcup_{i=1}^{n} R_i = R$ — covers all image

b) R_i is a connected region — each region is a connected component

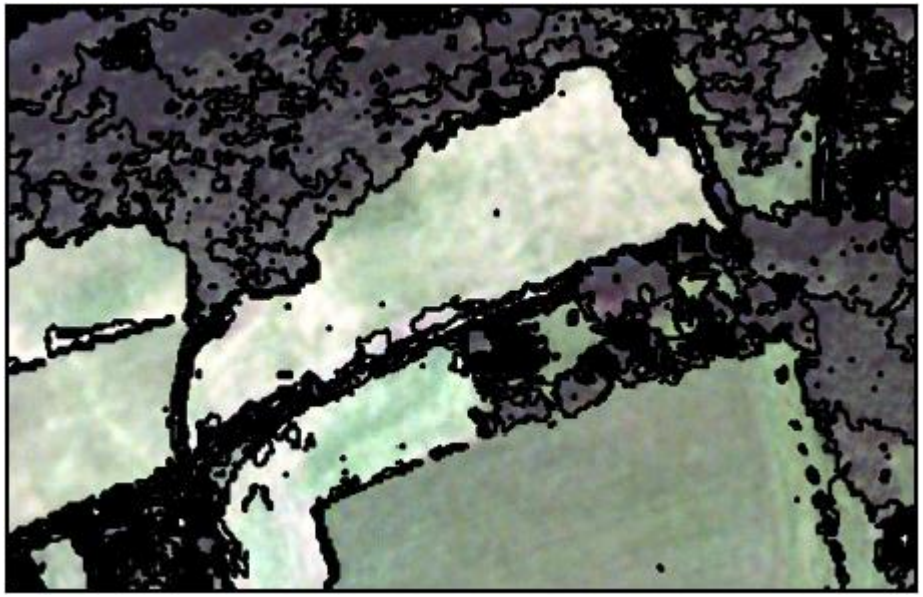
c) $R_i \cap R_j = \emptyset$ for all i and j, $i \neq j$ — regions are disjoint

d) $P(R_i)$ = TRUE for $i=1,2,\ldots,n$ — P defines the membership condition

e) $P(R_i \cup R_j)$ = FALSE for $i \neq j$ — P also differentiates the regions.

Region-Based Segmentation

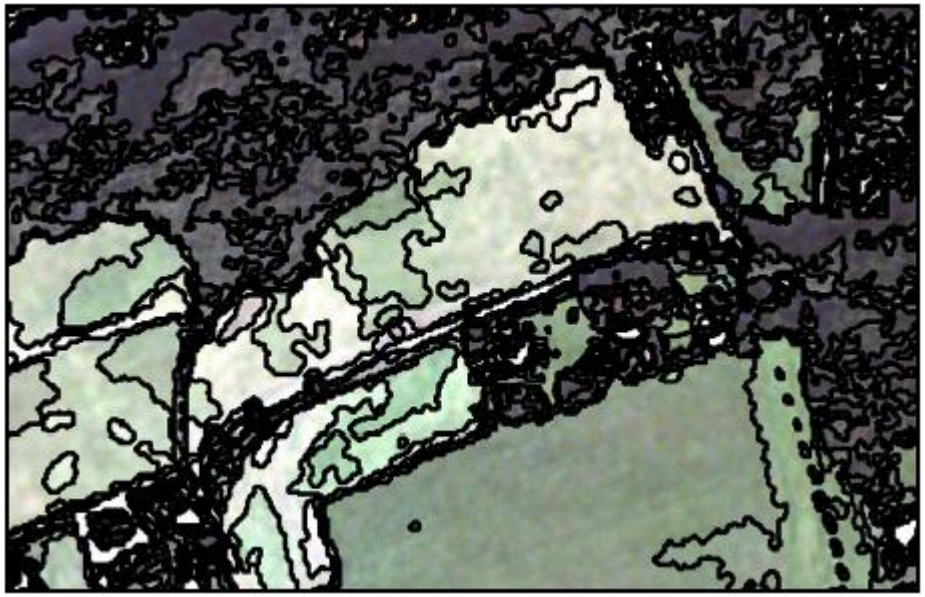
Innumerous techniques: all of them have parameters that must be defined by the user/expert.



Data Dissecation Tools
(Clusterização – 3 parâmetros)

Region-Based Segmentation

Innumerous techniques: all of them have parameters that must be defined by the user/expert.



InfoPACK
(Simulated Annealing – 2 parâmetros)

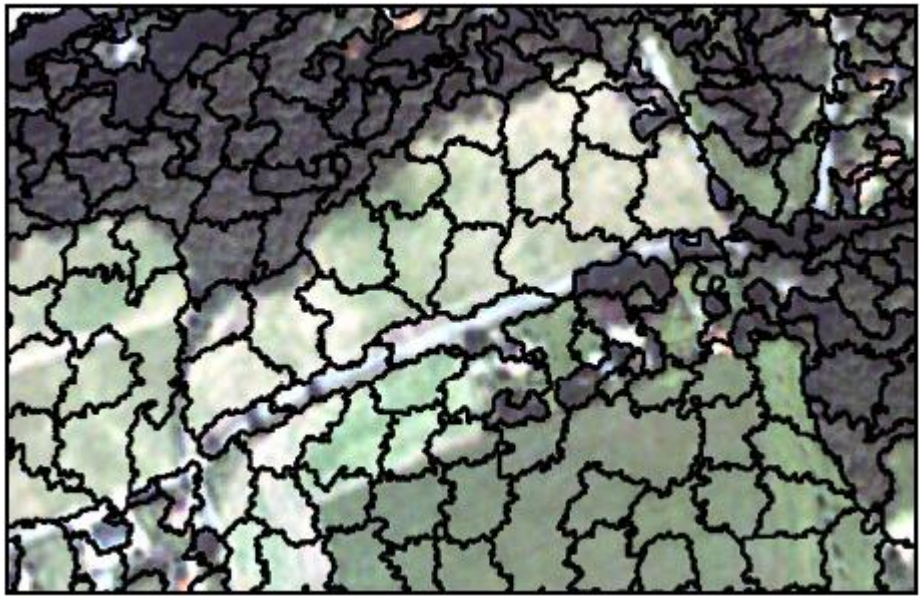
Region-Based Segmentation

Innumerous techniques: all of them have parameters that must be defined by the user/expert.

Minimum Entropy Approach
(Triangulação – 2 parâmetros)

Region-Based Segmentation

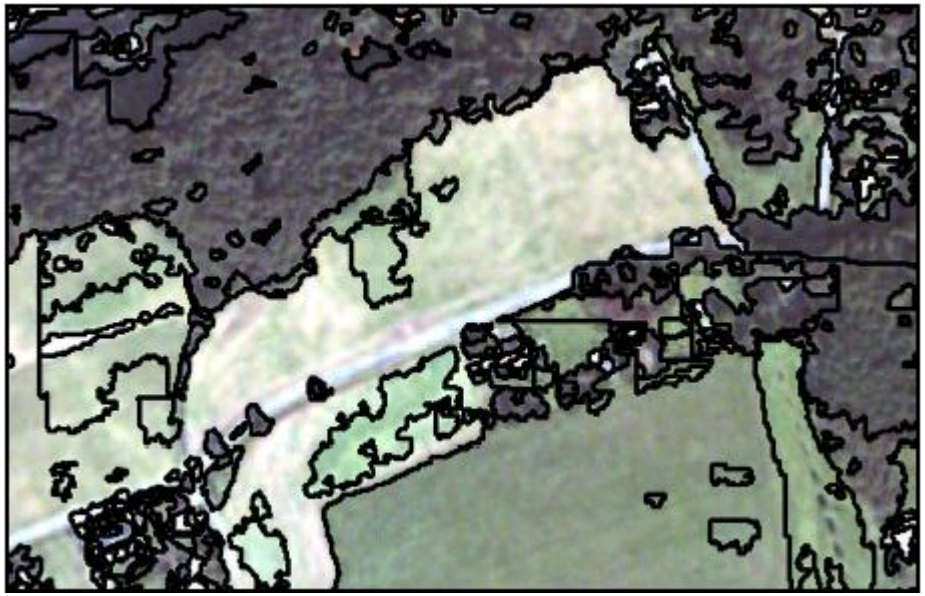
Innumerous techniques: all of them have parameters that must be defined by the user/expert.



CAESAR
(Simulated Annealing – 6 parâmetros)

Region-Based Segmentation

Innumerous techniques: all of them have parameters that must be defined by the user/expert.



ERDAS
(Crescimento de Regiões – 2 parâmetros)

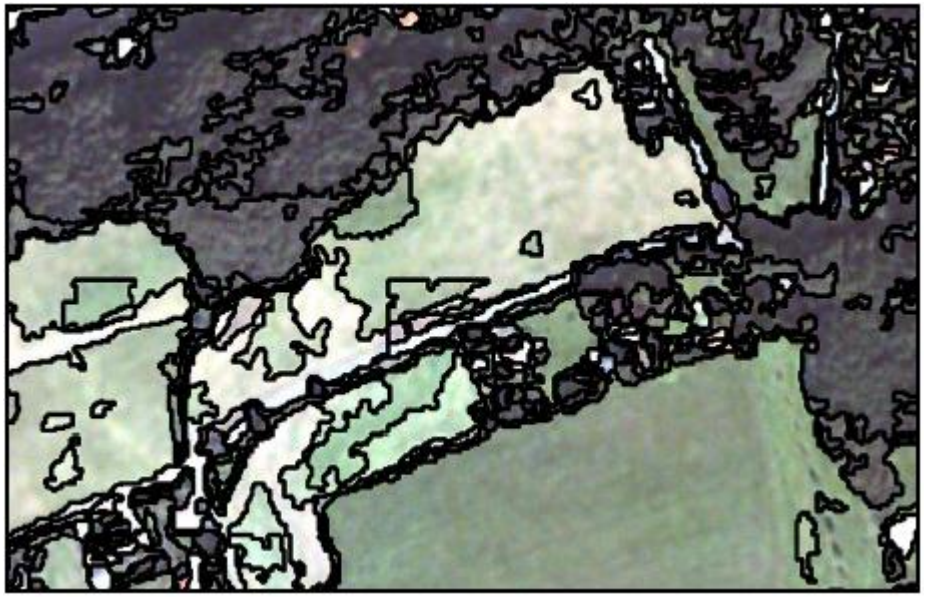
Region-Based Segmentation

Innumerous techniques: all of them have parameters that must be defined by the user/expert.

eCognition
(Crescimento de Regiões – 3 parâmetros)

Region-Based Segmentation

Innumerous techniques: all of them have parameters that must be defined by the user/expert.



SPRING
(Crescimento de Regiões – 2 parâmetros)

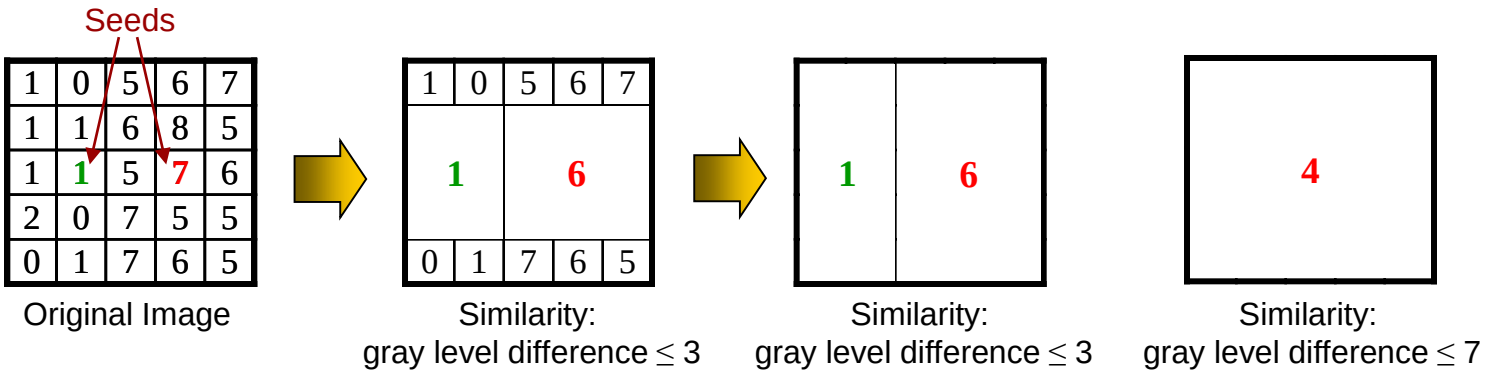
Region-Based Segmentation

Comparison/evaluation of different techniques

Segmentation program	eCognition 2.1	eCognition 3.0	Data Dissection Tools	CAESAR 3.1	InfoPACK 1.0	Image Segmentation (for Erdas Imagine)	Minimum Entropy Approach	SPRING 4.0
Number of reference areas	20	20	20	10[1]	20	20	11[1]	20
Average difference of area [%]	12,5	15,9	2100,3	75,1	11,1	107,0	13,6	8,2
Average difference of perimeter [%]	15,9	17,2	475,6	35,1	30,9	177,3	10,0	10,8
Average difference of Shape Index [%]	16,7	16,2	38,9	25,5	25,5	87,1	10,0	11,7
Average number of partial segments	1,9	1,8	134,6	10,4	17,1	5,9	9,0	6,2
Average quality, visual evaluated [0…2]	1,0	0,9	0,2	0,0	0,6	0,2	0,8	0,9

Region Growing

- **Definition:** A procedure that groups pixels or sub regions into larger regions based on predefined criteria.
- It involves two main iterative steps:
  - Select a set of starting "seeds", and
  - Grow from these by appending "similar" neighboring pixels.



Region Growing

- **Main issues:**
  - Selection of the set of seeds:
    - use a priori information about the particular problem;
    - compute features of every pixels and find clusters whose centroids will be used as seeds; or
    - determine seeds randomly.
  - The choice of the similarity criteria:
    - depends on the data;
    - a diversity of attributes (color, texture, shape, etc) may be taken into account.
  - Stopping rule:
    - usually when no more pixels satisfy the inclusion condition.

Region Growing

- **Example:**

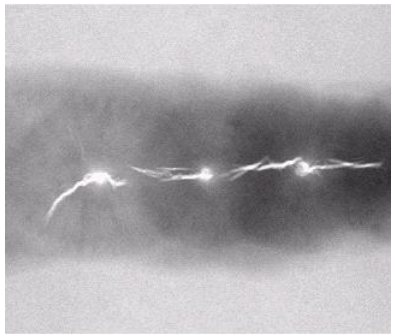
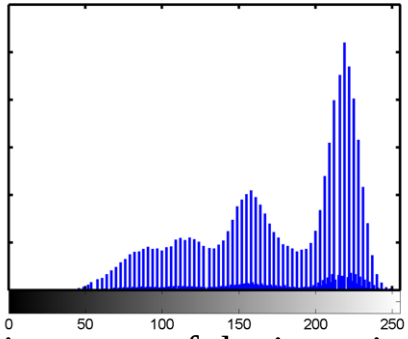
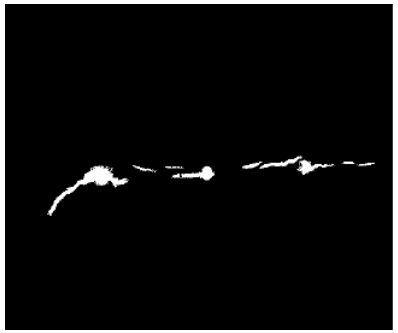


image showing defective welds

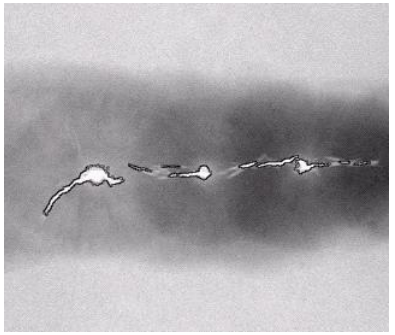


histogram of the input image

seeds are points with the maximum gray value



criteria: absolute gay-level difference from the seed less then 65

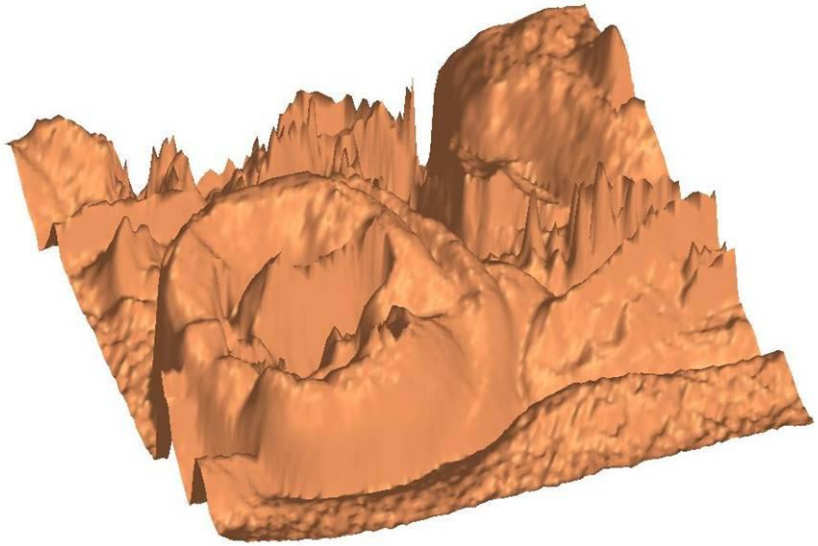


Boundaries of the defective welds (in black)

Watershed Algorithm

- The image is viewed in 3 D (third dimension is the gray level).

original image



“topographic” view of the image

Watershed Algorithm

- Three types of points:

miminum: points belonging to a regional minimum.

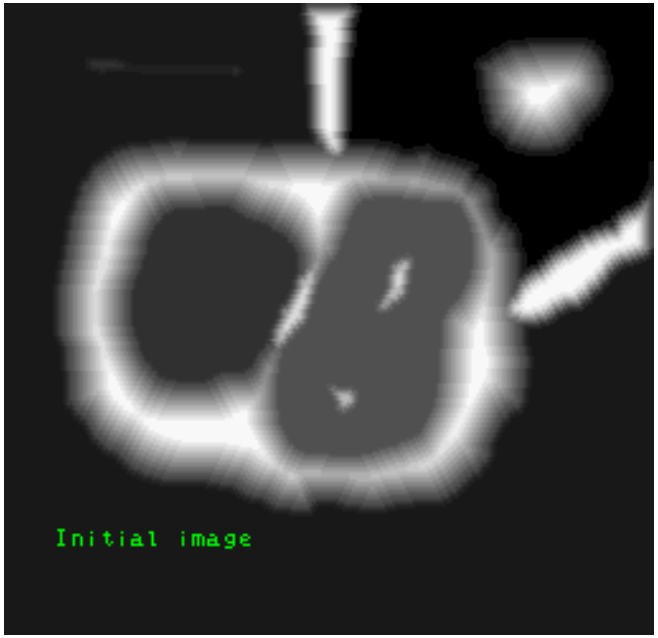
watershed of a miminum: points at which, a drop of water, if placed at the location of any of those points, would fall with certainty to a single minimum.

watershed lines: points at which water would be equally likely to fall to more than one such minimum.

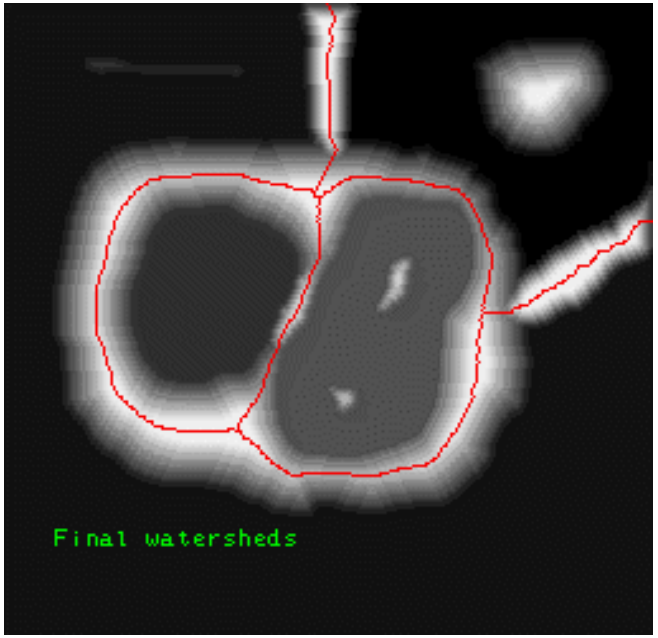
"topographic" view

Watershed Algorithm

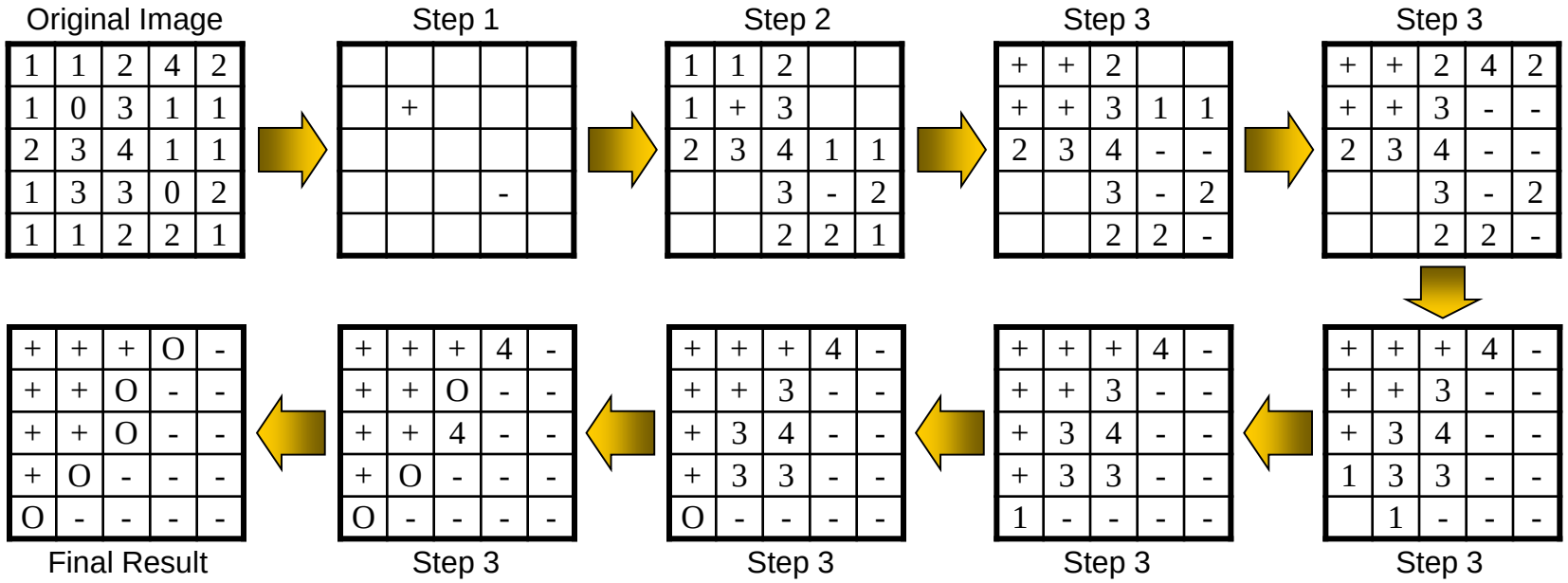
- A hole is punched in each regional minimum and the entire topography is flooded from below by letting water rise through the holes at an uniform rate.



flooding process



final segmentation



Meyer's flooding algorithm (Whatershed):

1. A set of markers, pixels where the flooding shall start, are chosen. Each is given a different label.
2. The neighbors of each marker are inserted into a priority queue with a priority level corresponding to the gray level of the pixel (low value = high priority).
3. The pixel with the highest priority level is extracted from the priority queue:
   - If all neighbors that have already been labeled have the same label, then mark the pixel with that label.
   - All non-labeled neighbors that are not yet in the priority queue are inserted into the priority queue.
4. Redo step 3 until the priority queue is empty.

The non-labeled pixels are the watershed lines.

Watershed Algorithm

- To obtain homogeneous objects the *watershed* algorithm is applied to the magnitude of the gradient.

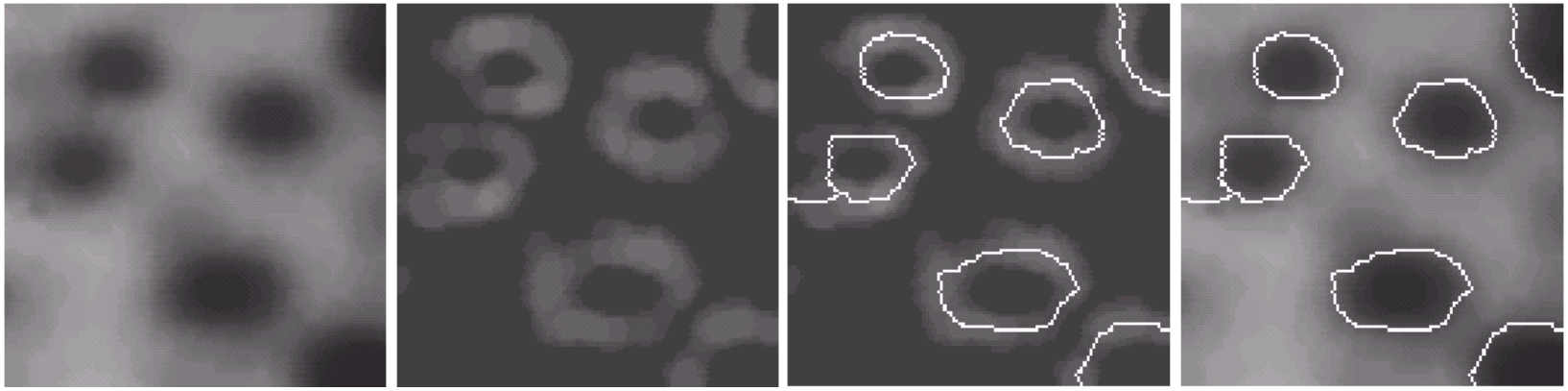
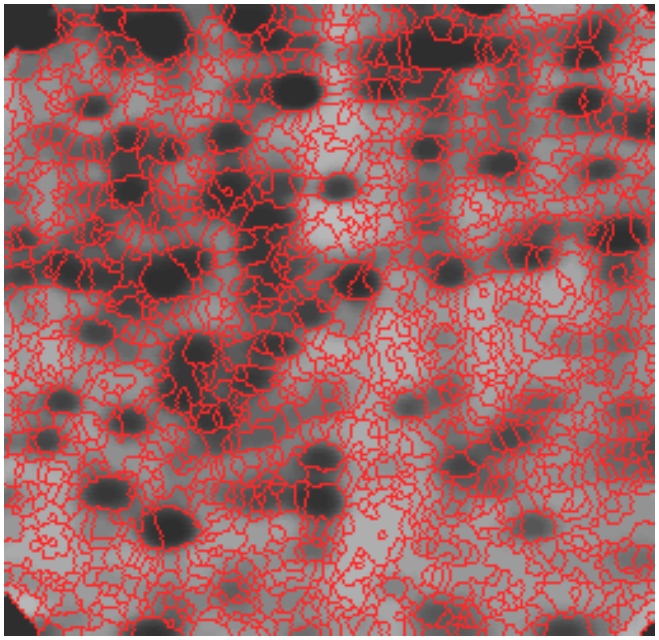


image | gradient | borders superimposed to the gradient | borders superimposed to the original image

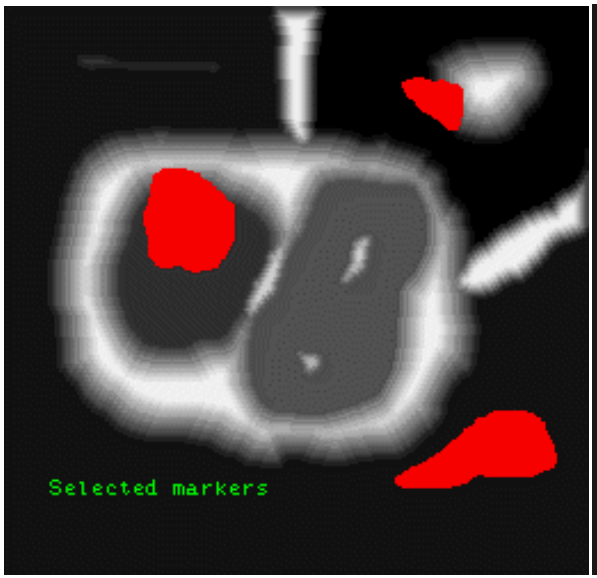
Watershed Algorithm

- Due to noise and local irregularities the direct application of the previous algorithm leads to *oversegmentation.*



- Solution: limit the number of allowable regions.

Watershed Algorithm



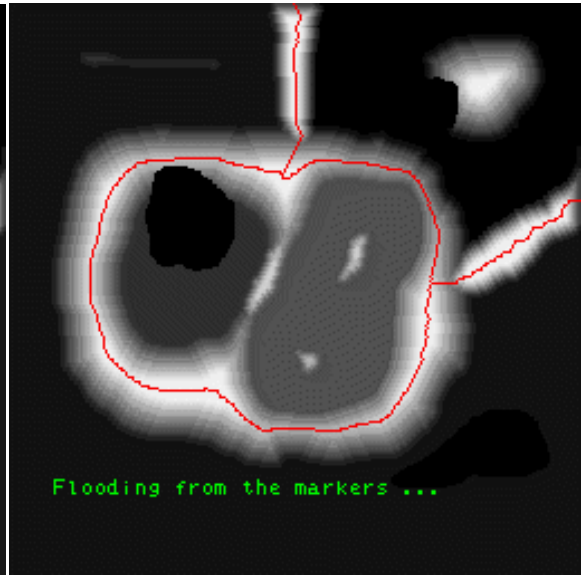


image with the selected markers

final segmentation

Spring Algorithm

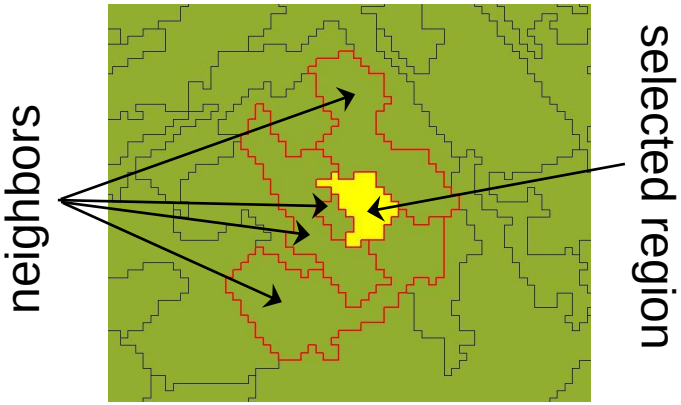
- Region growing segmentation algorithm implemented in the software Spring (INPE).
- Initially every pixel is considered a segment.
- At each step, segments are merged to exactly one neighbor: the more "similar" neighbor.

Spring Algorithm

- Similarity among regions is based on the Euclidian distance between their spectral mean values:

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

- Two parameters:
  - Distance threshold (T)
  - Minimum segment area

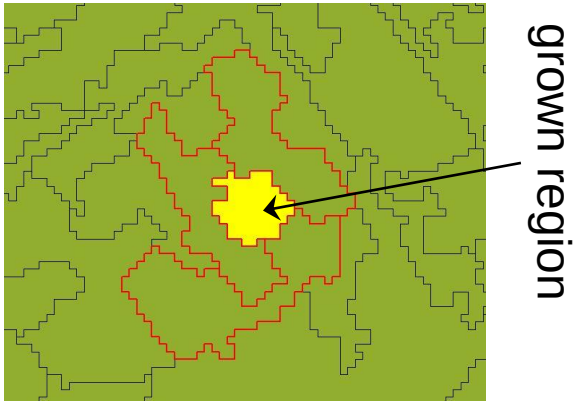


Spring Algorithm

- Similarity among regions is based on the Euclidian distance between their spectral mean values:

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

- Two parameters:
  - Distance threshold (T)
  - Minimum segment area



Spring Algorithm

1. A list of regions $\{R_i; i=1,...,n\}$ is created (n is the number of pixels in the image).
2. For each region R_i the mean value vector and neighboring regions are stored.
3. For each region R_i its neighboring regions $N(R_i)$ are examined and:
   - the most similar neighboring region $R_k \in N(R_i)$ is chosen.
     If $D(R_i, R_k) < T$ then R_k is called "the best neighbor" of R_i.
   - If the best neighbor of R_k exists and is R_i, then both regions are merged.
4. When a region is merged to another, it is taken out of the list.
5. The mean value of the merged regions are updated.
6. Repeat from step 3 until no joinable regions remain.
7. Small regions are merged with larger adjacent regions, in accordance with an area threshold value defined by the user.

Baatz & Shäpe Algorithm

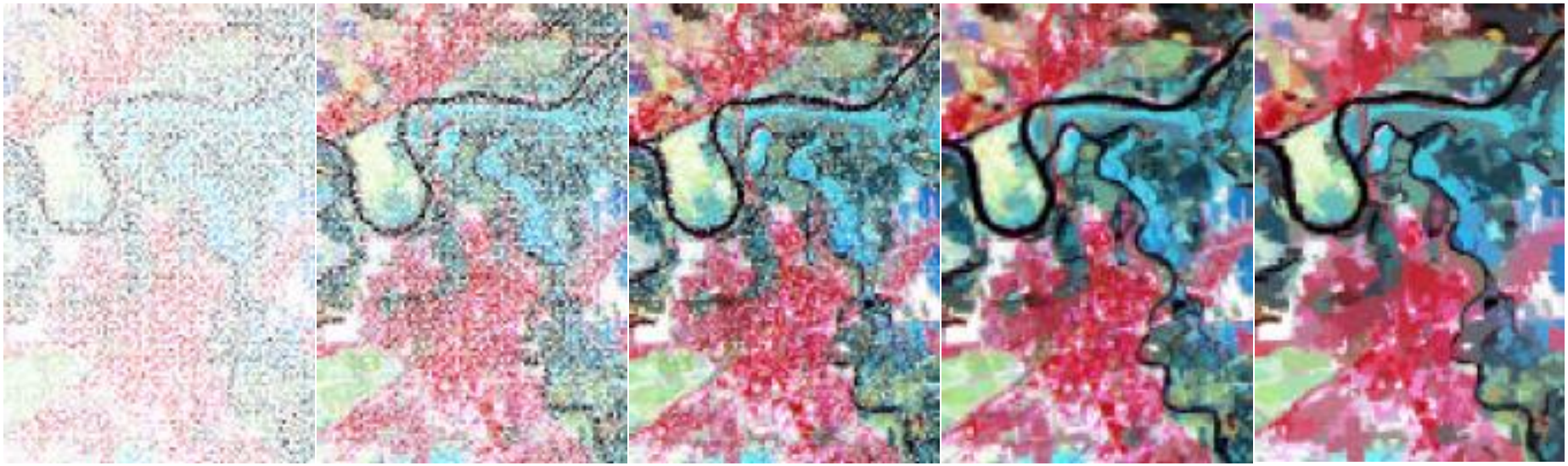
- Originaly implemented in the software eCognition (Definiens).
- Multiscale algorithm: has specific parameter to define size of resulting regions.
- Region/segments are characterized by their internal heterogeneity.

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

Baatz & Shäpe Algorithm

- Initially every pixel is considered a segment.
- At each step, segments are merged to exactly one neighbor: the one that promotes the lowest increase in global heterogeneity.
- Selection of segments to grow may be “almost” random: distributes subsequent merges as far as possible from each other over the whole scene.

Baatz & Shäpe Algorithm



Step *n*+1 Step *n*+2 Step *n*+3 Step *n*+4 Step *n*+5

Baatz & Shäpe Algorithm

- The heterogeneity measure has a spectral (color) and a spatial (shape) component.

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

- Fusion of an segment (*Seg1*) with a neighbor (*Seg2*) considers the increase in heterogeneity of the resulting segment (*Seg3*).

Baatz & Shäpe Algorithm

Measure of the increase in spectral heterogeneity:

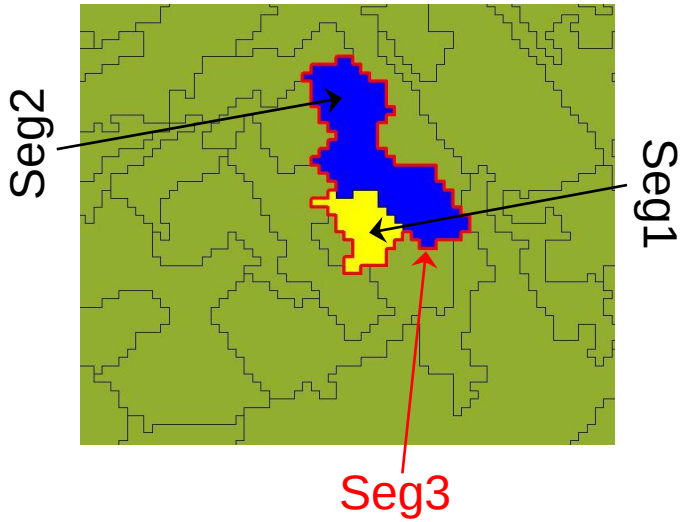
$$f_{color} = \sum_{c} w_c \left(n_{Seg3} \cdot \sigma_c^{Seg3} - \left(n_{Seg1} \cdot \sigma_c^{Seg1} + n_{Seg2} \cdot \sigma_c^{Seg2} \right) \right)$$

C = spectral band

w_c = weight of band C

n = object area (pixels)

σ_c = std deviation of pixel values



Baatz & Shäpe Algorithm

Measure of the increase in shape heterogeneity:

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

f_{cmpct} = compactness component

f_{smooth} = smoothness component

w_{cmpct} = weight of compactness against smoothness

Baatz & Shäpe Algorithm

Compactness component in shape heterogeneity:

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

Smoothness component in shape heterogeneity:

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

n = object area

l = object perimeter

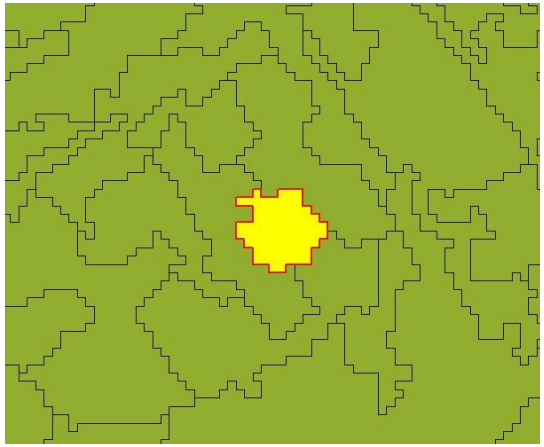
b = perimeter of the bounding box

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



Baatz & Shäpe Algorithm

Combined increase in heterogeneity (fusion factor):

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

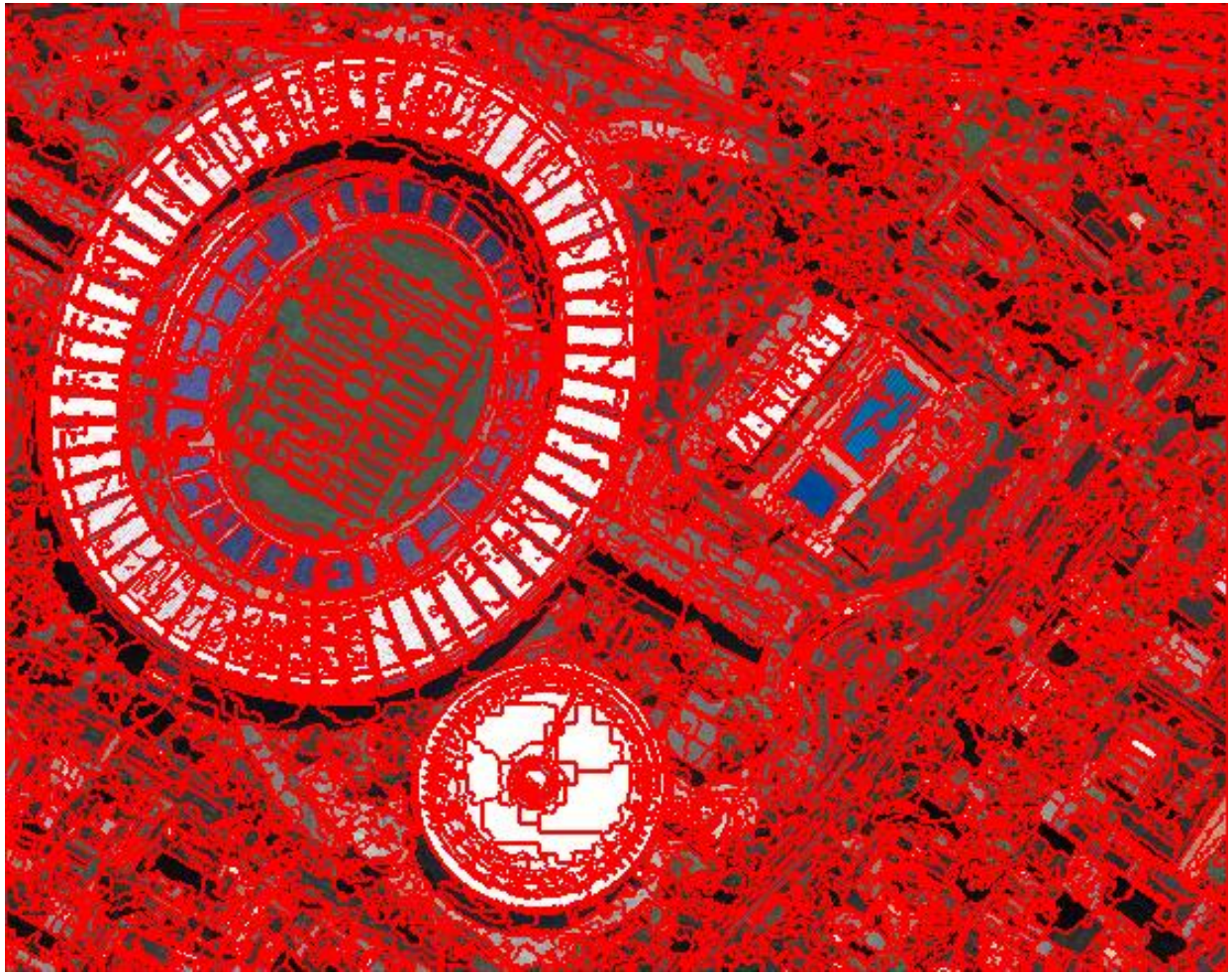
w_{color} = weight of color against shape

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

Baatz & Shäpe Algorithm

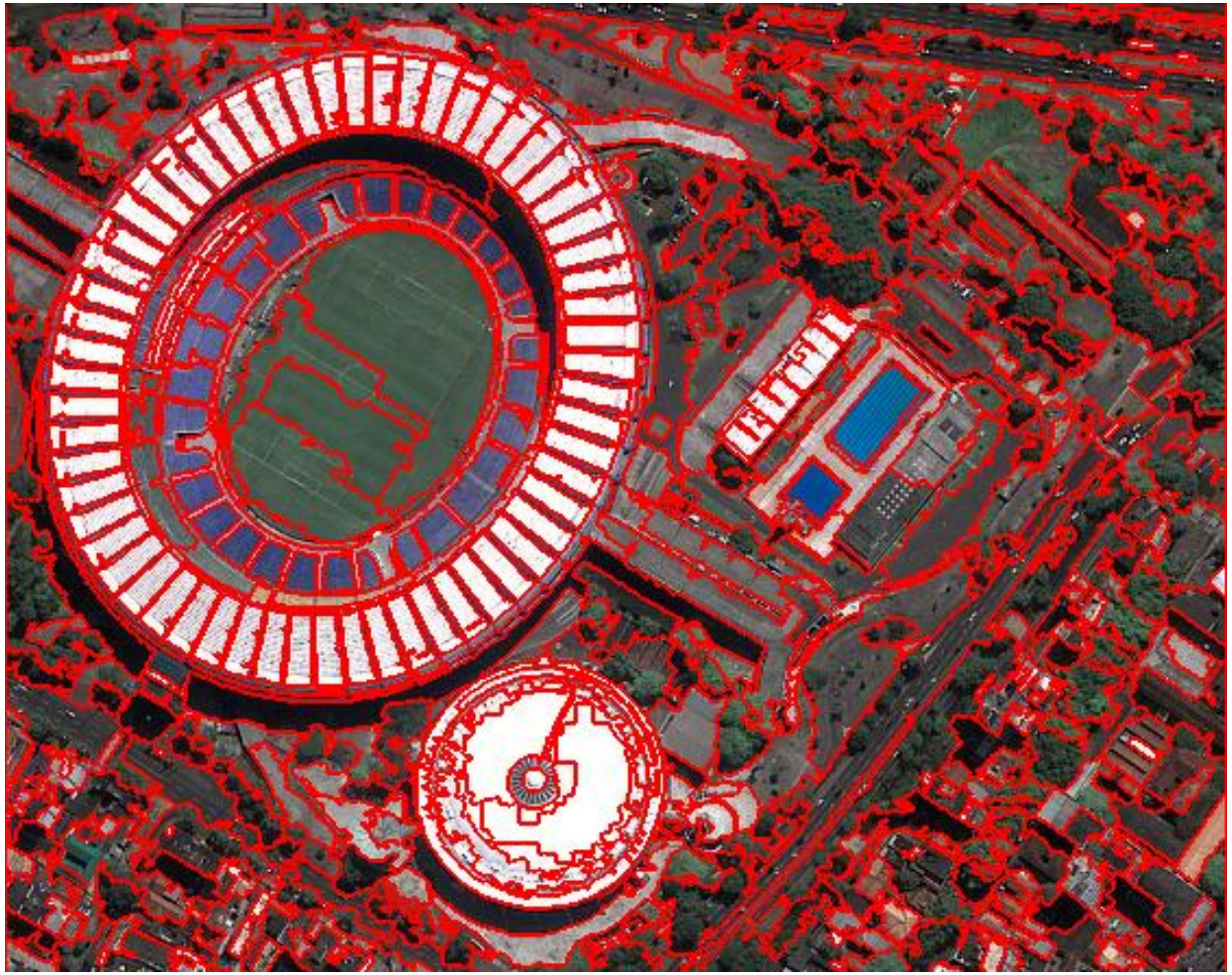
Original Image

Baatz & Shäpe Algorithm



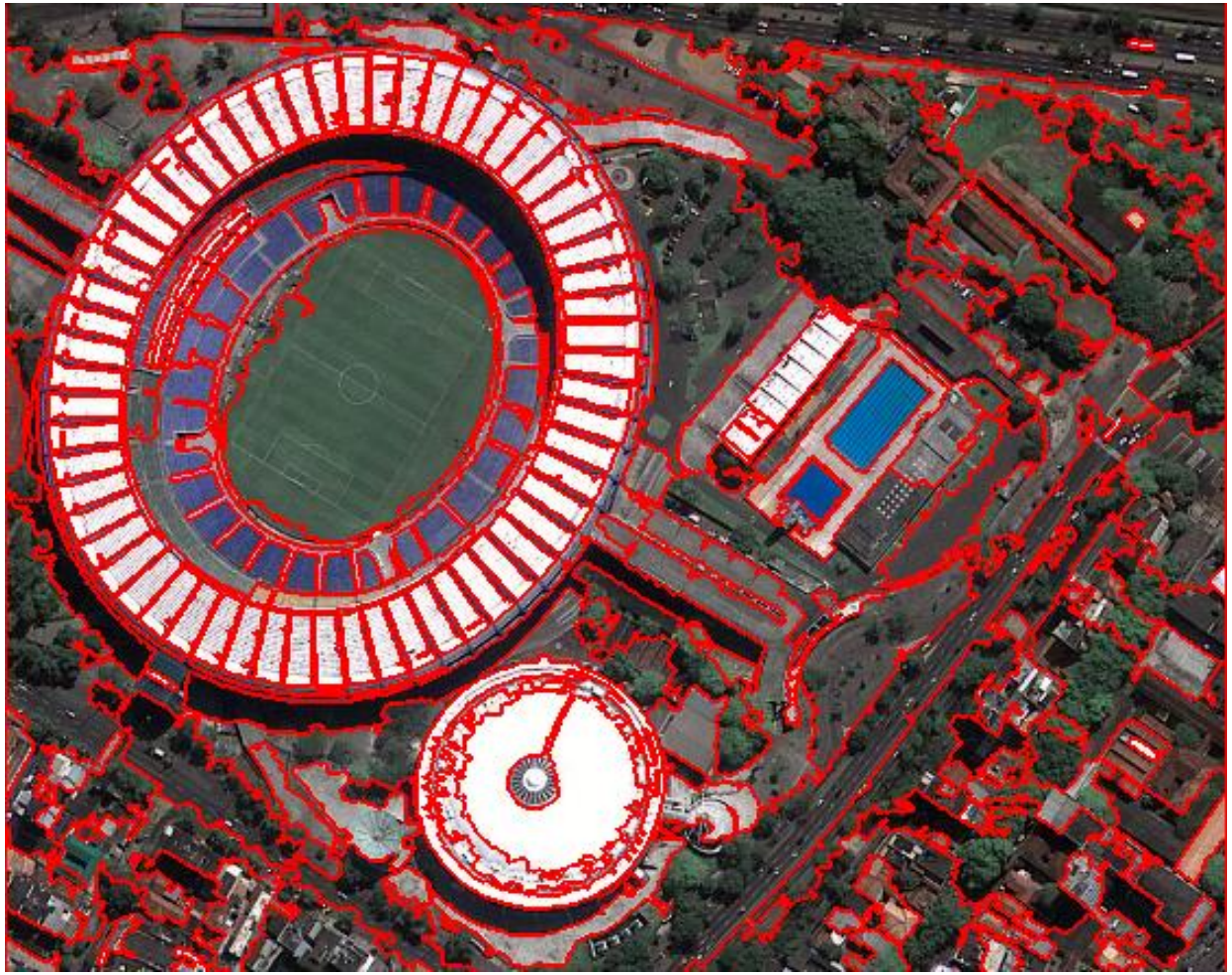
$sp = 10$ $w_{color} = 0.9$ $w_{cmpct} = 0.5$

Baatz & Shäpe Algorithm



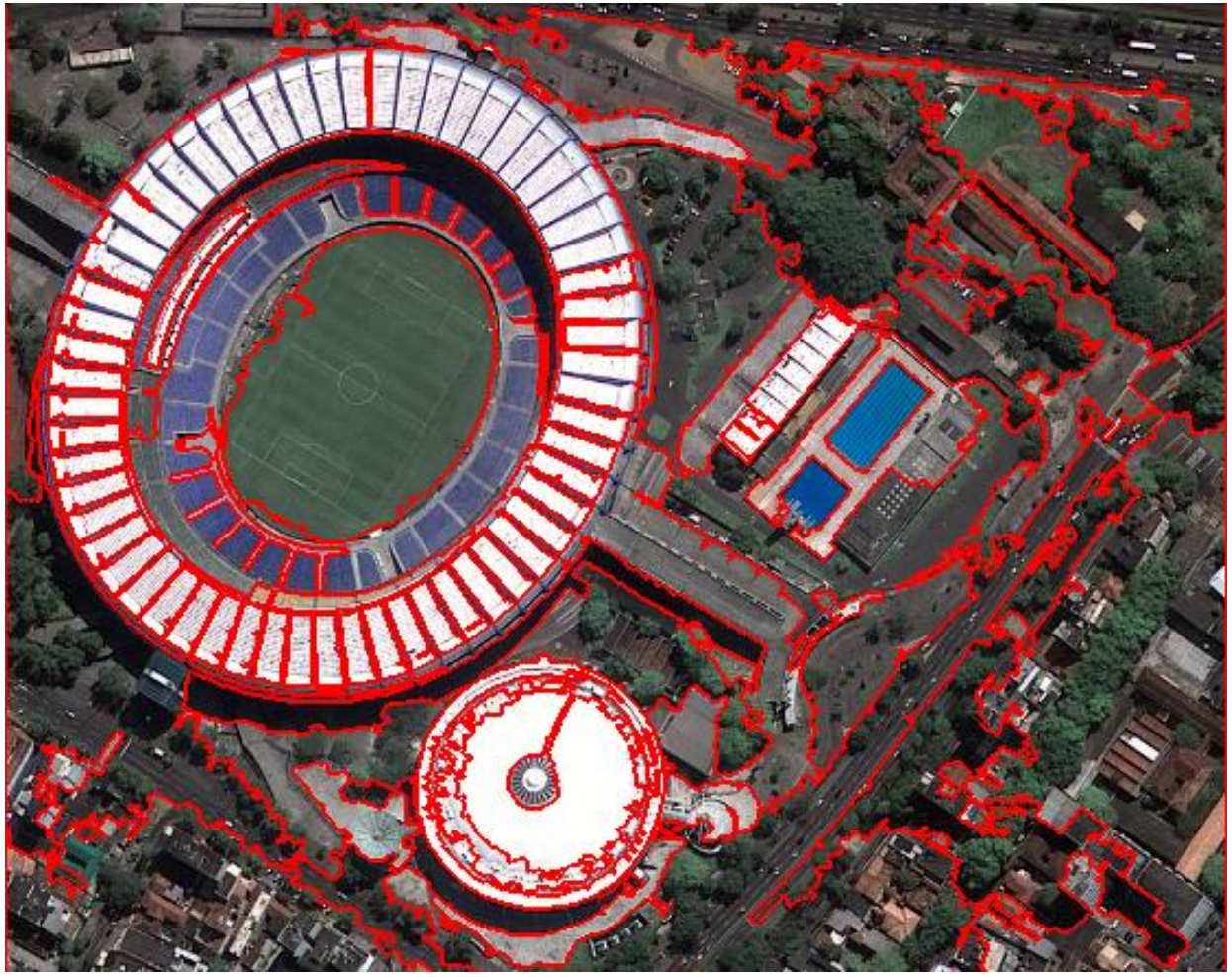
$sp = 50$ $w_{color} = 0.9$ $w_{cmpct} = 0.5$

Baatz & Shäpe Algorithm



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

Baatz & Shäpe Algorithm



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

Baatz & Shäpe Algorithm

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