TÉCNICO LISBOA

TÉCNICO LISBOA

Principles of Biosignals and Biomedical Imaging

3rd year, P_3 (ECTS: 3.0), LEBiom

2022/2023

João Sanches
jmrs@tecnico.ulisboa.pt
Department of Bioengineering
Instituto Superior Técnico
University of Lisbon

Outline

Edge detection

- Sobel
- Canny

Segmentation

- Intensity
- Texture

TÉCNICO LISBOA

Edge detection

Gradient field

Approximation for the **first** derivatives

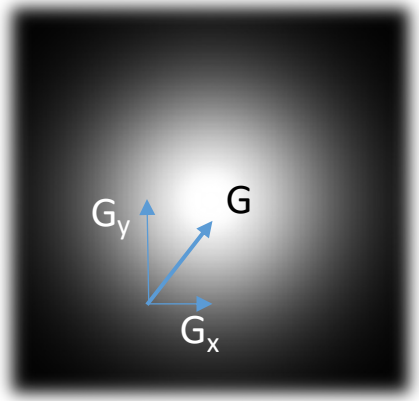
$$G_x(i,j) = I(i,j) - I(i-1,j)$$

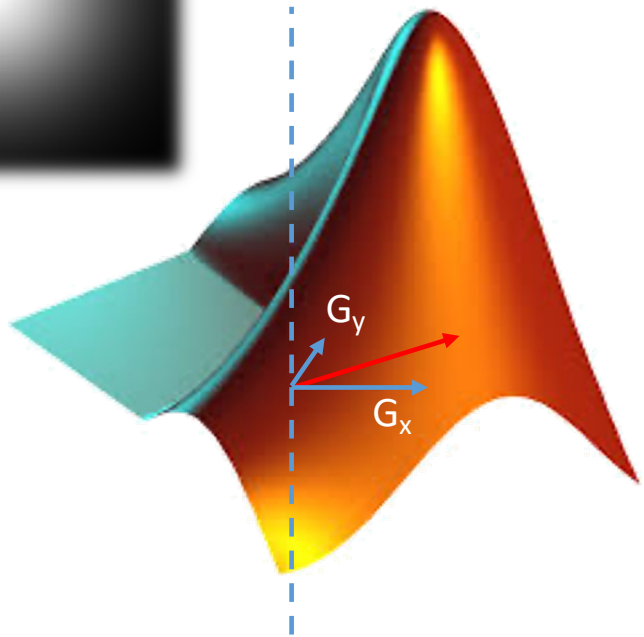
$$G_y(i,j) = I(i,j) - I(i,j-1)$$

Gradient vector

$$G(i,j) = \left(G_x(i,j), G_y(i,j)\right)$$

Contours occur in intensity transitions where the gradient magnitude is large





Gradient magnitude and angle

$$G_x(i,j) = I(i+1,j) - I(i-1,j)$$

$$G_y(i,j) = I(i,j+1) - I(i,j-1)$$

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

$G_x = w * I$

Horizontal differences

$G_y = w' * I$

Vertical differences

$$G(i,j) = \sqrt{G_x(i,j)^2 + G_y(i,j)^2}$$

$$\theta(i,j) = atan\left(\frac{G_{y(i,j)}}{G_{x(i,j)}}\right)$$

Phantom

Gradient magnitude

Horizontal gradient

Vertical gradient

Laplace Operator

Approximation for the **Second** derivatives

$$G_x^2(i,j) = G_x(i,j) - G_x(i-1,j)$$
$$= [I(i,j) - I(i-1,j)] - [I(i-1,j) - I(i-2,j)]$$
$$= I(i,j) - 2I(i-1,j) + I(i-2,j)$$
$$\sim \boldsymbol{I(i+1,j) - 2I(i,j) + I(i-1,j)}$$

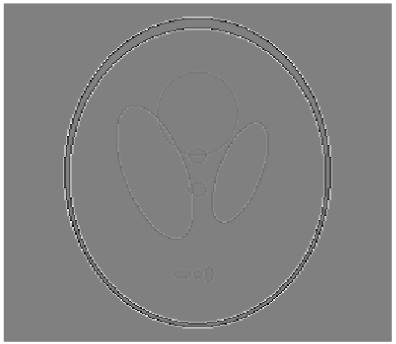
$$G_x^2 = I * [1 \quad -2 \quad 1]$$

$$G_y^2(i,j) = G_y(i,j) - G_y(i,j-1)$$
$$= [I(i,j) - I(i,j-1)] - [I(i,j-1) - I(i,j-1)]$$
$$= I(i,j) - 2I(i,j-1) + I(i,j-2)$$
$$\sim \boldsymbol{I(i,j+1) - 2I(i,j) + I(i,j-1)}$$

$$G_y^2 = I * [1 \quad -2 \quad 1]^T$$

$$\nabla^2 = \nabla.\nabla = \left(\frac{\partial^2}{\partial x^2}\right) + \left(\frac{\partial^2}{\partial y^2}\right)$$

$$\approx \begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix} + \begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$



Sobel operator

$$W_x = \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \qquad W_y = \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

$$G_x = W_x * I$$

Filtered Horizontal differences

$$G_y = W_y * I$$

Filtered Vertical differences

$$G(i,j) = \sqrt{G_x(i,j)^2 + G_y(i,j)^2}$$

$$\theta(i,j) = atan\left(\frac{G_{y(i,j)}}{G_{x(i,j)}}\right)$$

Phantom

Gradient magnitude

Horizontal gradient

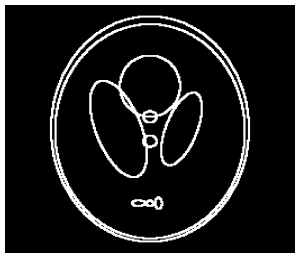
Vertical gradient

Edge detection

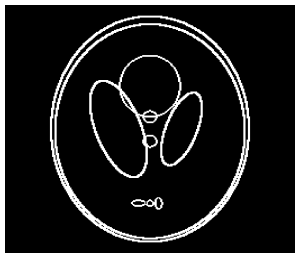
$$G(i,j) = \sqrt{G_x(i,j)^2 + G_y(i,j)^2}$$

$$C(i,j) = \begin{cases} 0 & if\ G(i,j) < thrs \\ 1 & otherwise \end{cases}$$

thrs=0.01

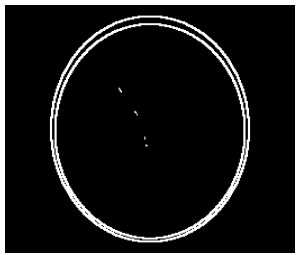


thrs=0.20



thrs=0.50

thrs=1.00



Canny edge detector

1. Noise reduction
2. Gradient calculation
3. Non-maximum suppression
4. Double thresholding
5. Edge Tracking by Hysteresis

Canny Edge Detection Step by Step in Python
by Sofiane Sahir

https://docs.opencv.org/4.x/da/d22/tutorial_py_canny.html
https://indiantechwarrior.com/hypothesis-testing/

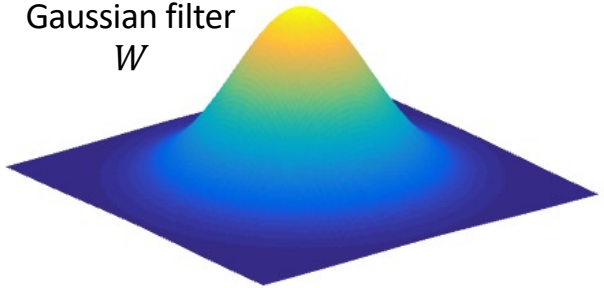
https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

Canny edge detector

1. Noise reduction

$$X = I * W\text{c}$$

Gaussian filter
W



https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

Canny edge detector

2. Gradient computation

$$G_x = W_x * I \qquad G_y = W_y * I$$

$$G(i,j) = \sqrt{G_x(i,j)^2 + G_y(i,j)^2}$$

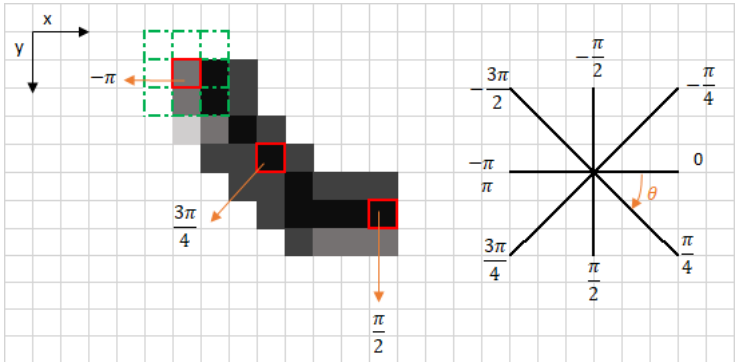
$$\theta(i,j) = atan\left(\frac{G_{y(i,j)}}{G_{x(i,j)}}\right)$$

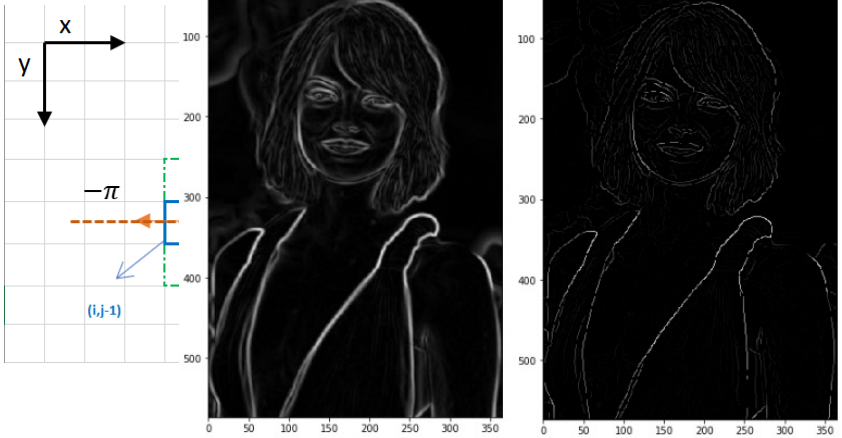
$I(i,j)$ $|G(i,j)|$

https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

Canny edge detector

3. Non-maximum suppression





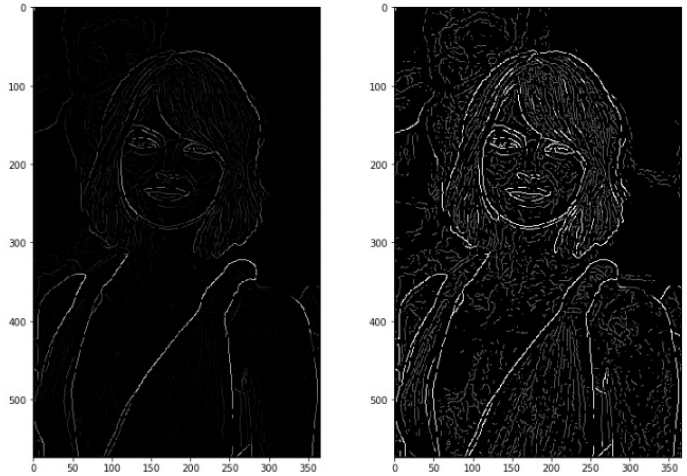
The purpose of the step is to check if the pixels on the same gradient direction are more or less intense than the ones being processed. If one those two pixels are more intense than the one being processed, then only the more intense one is kept. **The intensity value of the current pixel *(i, j)* is set to 0 if it is not the the highest intense**.

Canny edge detector

$$C(i,j) = \begin{cases} 1 & I(i,j) > thrs_1 \\ 0.5 & if\ thrs_2 < I(i,j) < thrs_1 \\ 0 & if\ I(i,j) < thrs_2 \end{cases}$$

4. Double thresholding

- The double threshold step aims at identifying 3 kinds of pixels: **strong, weak, and non-relevant**:
- Strong pixels are pixels that have an intensity so high that we are sure they contribute to the final edge.
- Weak pixels are pixels that have an intensity value that is not enough to be considered as strong ones, but yet not small enough to be considered as non-relevant for the edge detection.
- Other pixels are considered as non-relevant for the edge.

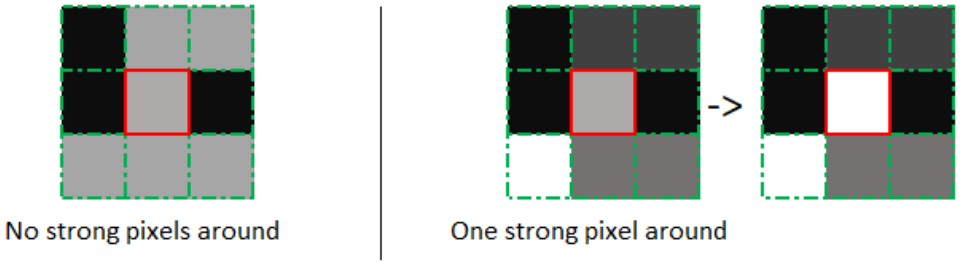


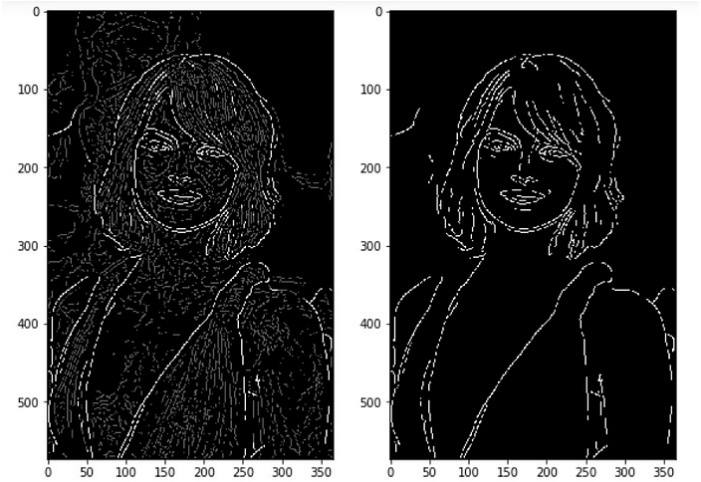
https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

Canny edge detector

5. Edge Tracking by Hysteresis

Based on the threshold results, the hysteresis consists of transforming weak pixels into strong ones, if and only if at least one of the pixels around the one being processed is a strong one, as described below





https://towardsdatascience.com/canny-edge-detection-step-by-step-in-python-computer-vision-b49c3a2d8123

TÉCNICO LISBOA

Break

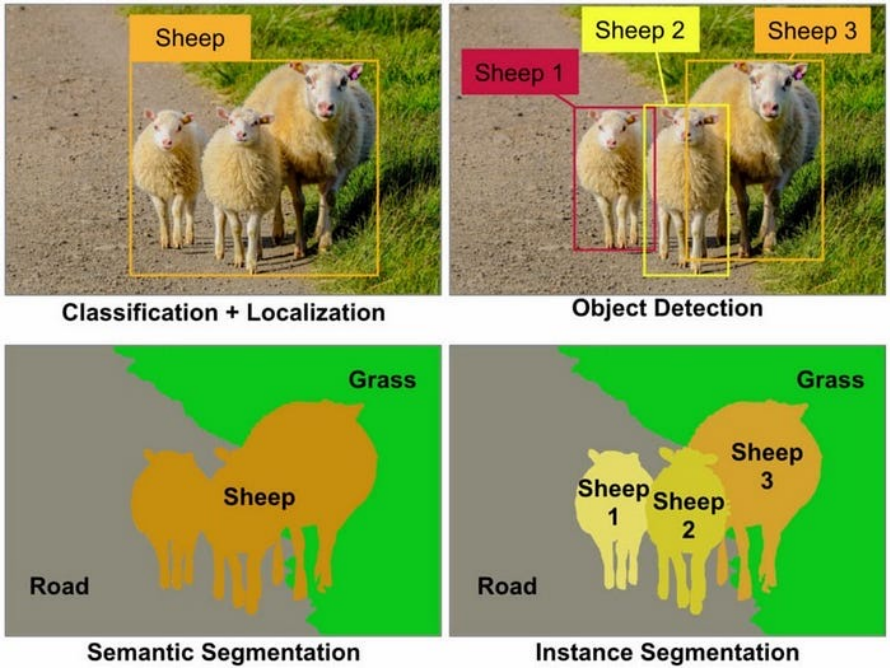
TÉCNICO LISBOA

Segmentation

(Chapter 14)

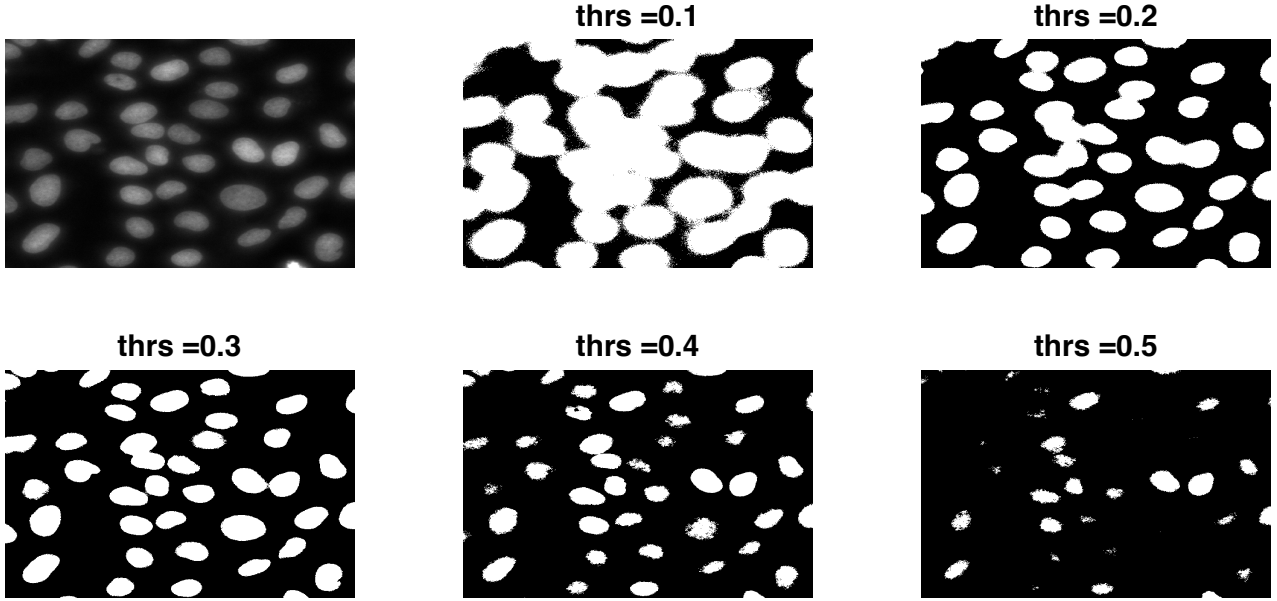
Segmentation

- Semantic segmentation (e.g. object vs background)
- Instance Segmentation



Segmentation by thresholding

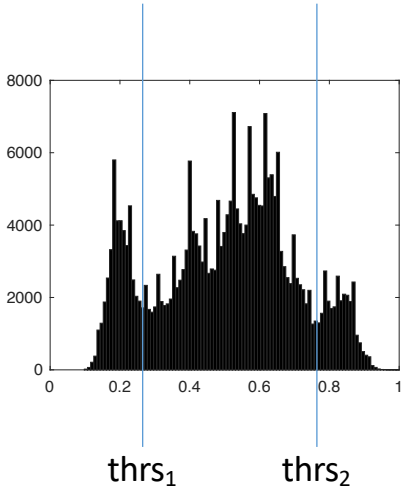
$$S(i,j) = \begin{cases} 0 & if\ x(i,j) < thrs \\ 1 & otherwise \end{cases}$$



Multi-threshold segmentation

$$S(i,j) = \begin{cases} 1 & if\ thrs_1 < x(i,j) < thrs_2 \\ 0 & otherwise \end{cases}$$

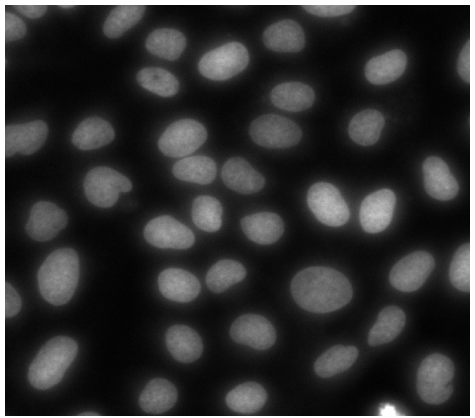
I(I,j)

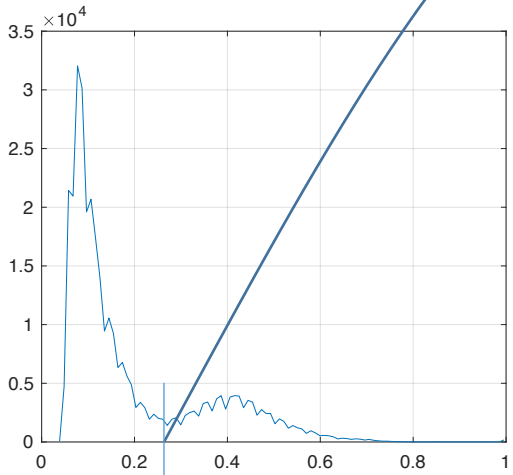


S(I,j)

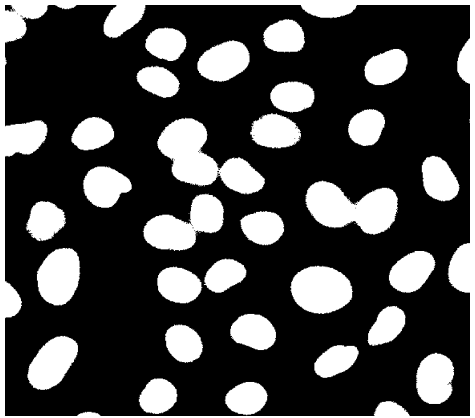
S(I,j)I(I,j)

Otsu (Automatic threshold)

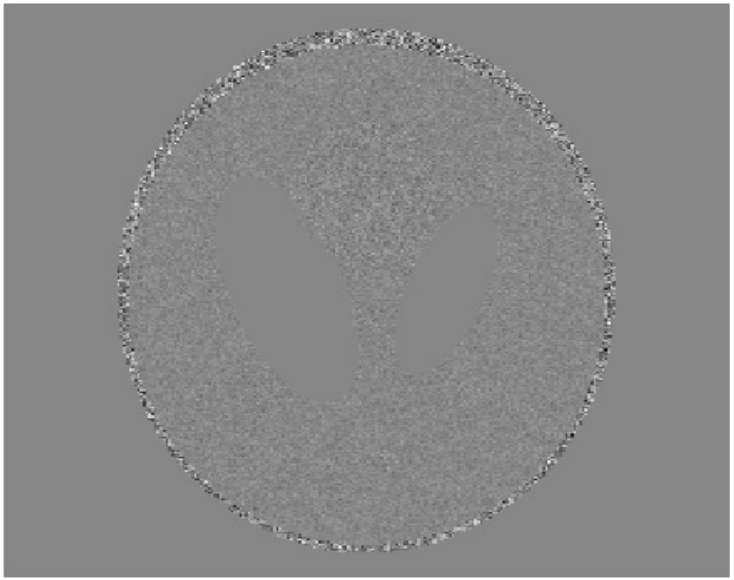


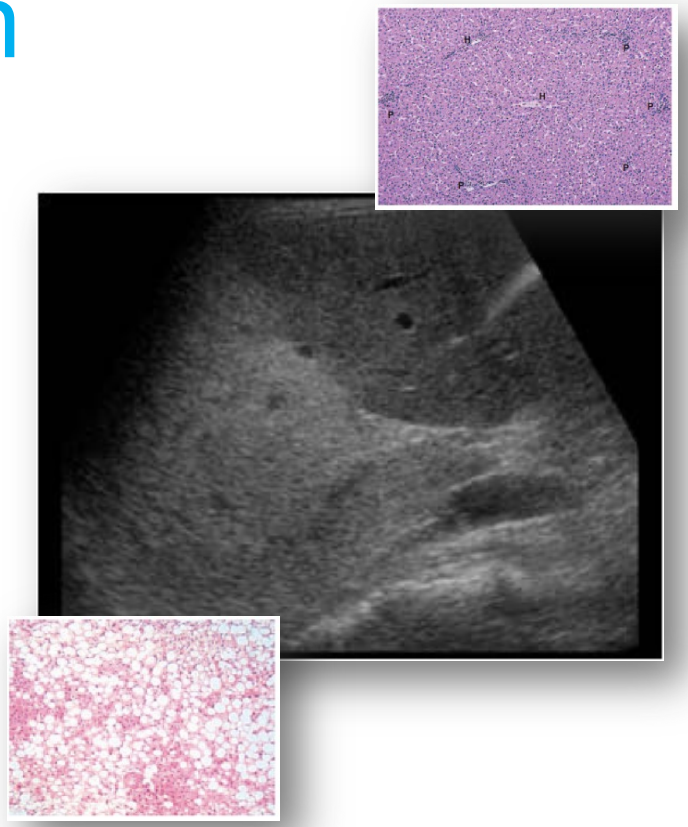


0.27

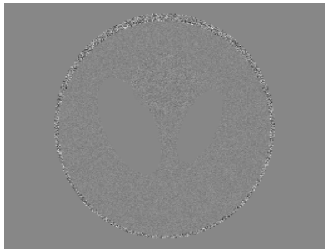


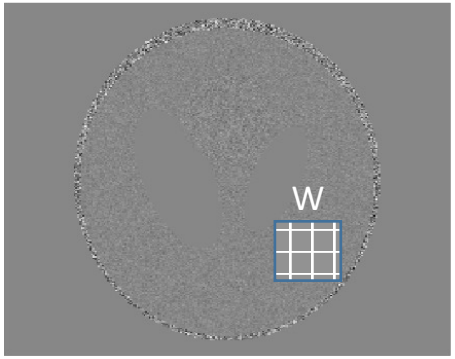
Texture segmentation





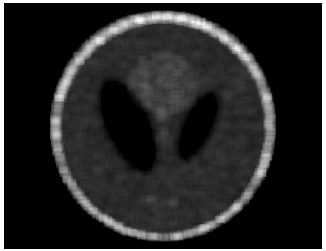
Texture segmentation



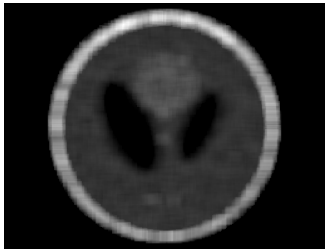


$$\mu(i,j) = \frac{1}{N_W} \sum_{i,j \in W} x(i,j)$$

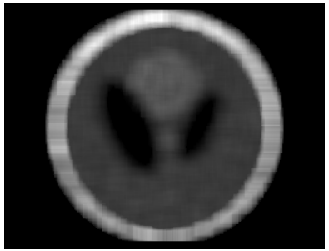
W:3



W:5



W:7



$$\sigma(i,j) = \sqrt{\frac{1}{N_W} \sum_{i,j \in W} \left(x(i,j) - \mu(i,j)\right)^2}$$

TÉCNICO LISBOA