CHAPITRE 5 : IMAGE SEGMENTATION

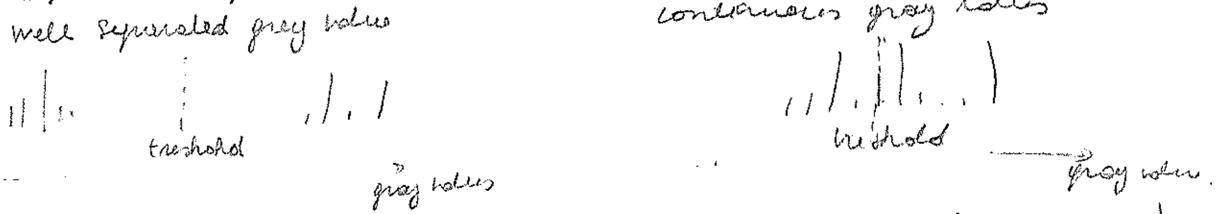
DEFINITION

Segmentation is the process by which an original image I is partitioned into N regions R_i, $1 \leq i \leq N$, which are homogeneous in some sense (e.g. brightness, color, etc)
so that : $\bigcup_{i=1}^{N} R_i = I$ / $R_i \cap R_j = \emptyset$, $1 \leq i, j \leq N$, $i \neq j$ / $H(R_i) = TRUE$, $1 \leq i \leq N$

$$H(R_i \cup R_j) = FALSE, \ 1 \leq i, j \leq N, \ i \neq j$$

THRESHOLDING

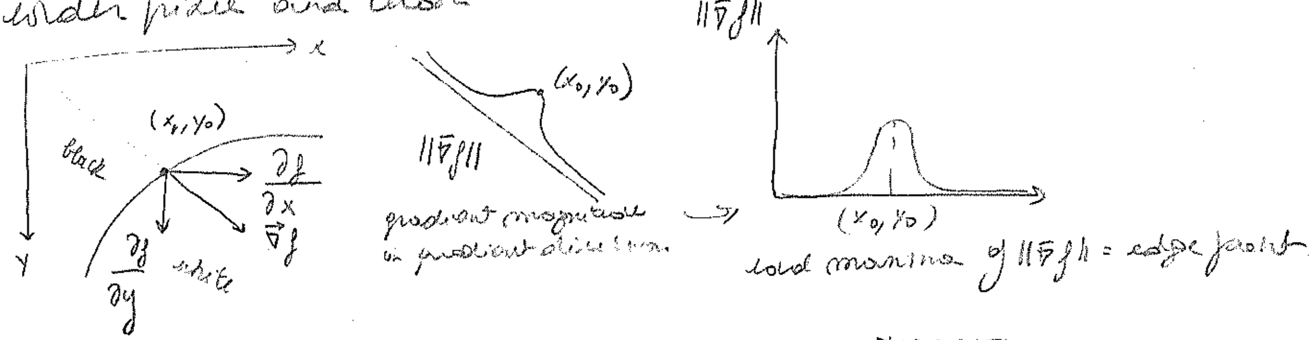
The simplest way to do image segmentation is thresholding. The segmentation of objects/regions on the basis of pixel grey value ranges only works well when the pixels belonging to these objects have clearly distinct, well separated grey value ranges. If this is not the case, some pixels of the objects will be wrongly classified (some pixels belonging to object 1 might be attributed to object 2 or vice versa)



Thresholding fails when there is a gradient from down to top (see slide 7)
there is no spatial structure taken into account

EDGE DETECTION (pg 128 du livre de réf et sq TB!)

Edge detection is important in a variety of applications, not only in segmentation (e.g. applications involving measurements of physical characteristics)
The simplest approach is to use the magnitude of a gradient (differential) operator in combination with thresholding. If the border is quite distinct, the magnitude of the gradient operator will have high values for the border pixels and these can be selected via thresholding.



	CONTINUOUS	DISCRETE
gradient component	$\frac{\partial f(x,y)}{\partial x}$	$d\,grad_i\, f(i,j) = f(i,j) - f(i-1,j)$
	$\frac{\partial f(x,y)}{\partial y}$	$d\,grad_j\, f(i,j) = f(i,j) - f(i,j-1)$
gradient magnitude	$\sqrt{\left(\frac{\partial f(x,y)}{\partial x}\right)^2 + \left(\frac{\partial f(x,y)}{\partial y}\right)^2}$	$\sqrt{d\,grad_i^2\, f(i,j) + d\,grad_j^2\, f(i,j)} \cong \lvert f(i,j) - f(i-1,j)\rvert + \lvert f(i,j) - f(i,j-1)\rvert$ (Roberts gradient operator)
gradient orientation	$\phi_n = \operatorname{atan} \frac{\frac{\partial f(x,y)}{\partial y}}{\frac{\partial f(x,y)}{\partial x}}$	$\phi_n = \operatorname{atan} \frac{d\,grad_j\, f(i,j)}{d\,grad_i\, f(i,j)}$

The robert's cross gradient operator approximates the gradient by the approximation of $\sqrt{a^2+b^2} \approx |a|+|b|$, decomposing it into two 2x2 masks:

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

1	0
0	-1

0	1
-1	0

and take the absolute value of the response to these masks and sum the results.

computes the differences in diagonal direction

Other discrete differential operators exists as well such as:

Prewitt operator

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

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

→ computes the average in this 2 horizontal direction, then computes the difference to find the horizontal edge corresponding to a vertical gradient (as they are ⊥)

Sobel operator

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

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

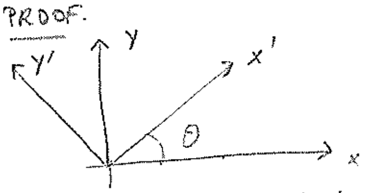
→ these are very sensitive to noise because of their fixed values for any image

Example: see slide 16-18

In slide 18, we see that thresholding techniques on gradient magnitude loose fine details and keeps only strong value i.e strong edges (3rd image). A low threshold give a too spurious response (4th image). The solution is to apply thresholding hysteresis which consists in taking only the edge which are connected with firm edges. The gradient details must be rotationally invariant, as the gradient is an intrinsic property of the image. This can be proven as:

$\begin{cases}\cos(\theta+\varphi) = \frac{x_0}{a} \text{ with } a = \|\vec{P}\| \\ \cos\varphi = \frac{x_0'}{a} \quad \sin\varphi = \frac{y_0'}{a}\end{cases}$

PROOF.



$$\begin{cases} x = x'\cos\theta - y'\sin\theta \\ y = x'\sin\theta + y'\cos\theta \end{cases}$$

because from (1): $\cos\theta\cos\varphi - \sin\theta\sin\varphi = \frac{x_0}{a}$

$\cos\theta\frac{x_0'}{a} - \sin\theta\frac{y_0'}{a} = \frac{x_0}{a}$ et idem pour y

Considering the gradient $s(x,y)$ and its partial derivatives

$$\frac{\partial s}{\partial x'} = \frac{\partial s}{\partial x}\frac{\partial x}{\partial x'} + \frac{\partial s}{\partial y}\frac{\partial y}{\partial x'} = \frac{\partial s}{\partial x}\cos\theta + \frac{\partial s}{\partial y}\sin\theta$$

$$\frac{\partial s}{\partial y'} = \frac{\partial s}{\partial x}\frac{\partial x}{\partial y'} + \frac{\partial s}{\partial y}\frac{\partial y}{\partial y'} = -\frac{\partial s}{\partial x}\sin\theta + \frac{\partial s}{\partial y}\cos\theta$$

The gradient magnitude is isotropic as:

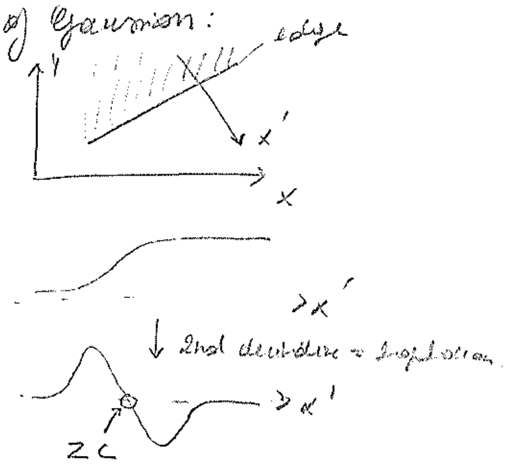
$$\left(\frac{\partial s}{\partial x'}\right)^2 + \left(\frac{\partial s}{\partial y'}\right)^2 = \left(\frac{\partial s}{\partial x}\right)^2\cos^2\theta + \left(\frac{\partial s}{\partial y}\right)^2\sin^2\theta + \left(\frac{\partial s}{\partial x}\right)^2\sin^2\theta + \left(\frac{\partial s}{\partial y}\right)^2\cos^2\theta$$

$$= \left(\left(\frac{\partial s}{\partial x}\right)^2 + \left(\frac{\partial s}{\partial y}\right)^2\right)(\cos^2\theta + \sin^2\theta)$$

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

To achieve rotational invariance, multiple masks should be applied as they augment directional selectivity. Indeed, filtering with X masks will generate X different directional gradient images. → cf slide 22-23

Another method for edge detection is zero crossing (ZC) detection of the laplacian of Gaussian:



CONTINUOUS

$$\frac{\partial f(x,y)}{\partial x}$$

$$\frac{\partial^2 f(x,y)}{\partial x^2} = \frac{\partial}{\partial x}\left(\frac{\partial f(x,y)}{\partial x}\right)$$

$$\frac{\partial^2 f(x,y)}{\partial y^2}$$

$$\frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2}$$

DISCRETE

$$d\,grad_i\, f(i,j) = f(i+1,j) - f(i,j)$$

$$d\,grad_i\, f(i+1,j) - d\,grad_i\, f(i,j)$$

$$\{f(i+1,j) - f(i,j)\} - \{f(i,j) - f(i-1,j)\}$$

$$\to f(i+1,j) - 2f(i,j) + f(i-1,j)$$

$$\{f(i,j+1) - f(i,j)\} - \{f(i,j) - f(i,j-1)\}$$

$$\to f(i,j+1) - 2f(i,j) - f(i,j-1)$$

$$f(i+1,j) - 4f(i,j) + f(i-1,j) + f(i,j+1) + f(i,j-1) \quad (2)$$

The Laplacian is proportional to the difference of the gray value at (i,j) and the average gray level in a neighbourhood of (i,j). Indeed, (2) can be written as: $-\{5f(i,j) - (f(i,j) + f(i-1,j) + f(i,j+1) + f(i,j-1) + f(i+1,j))\}$

And the formed mask is:

0	1	0
1	-4	1
0	1	0

with a sum of coefficient equal to 0 (DC component removed → as this is a high pass filter)

The Laplacian is also rotational invariant as: (referring to some scheme)

$$\frac{\partial^2 s}{\partial x'^2} = \frac{\partial^2 s}{\partial x^2}\cos^2\theta + \frac{\partial^2 s}{\partial x \partial y}\cos\theta\sin\theta + \frac{\partial^2 s}{\partial y \partial x}\sin\theta\cos\theta + \frac{\partial^2 s}{\partial y^2}\sin^2\theta$$

$$\frac{\partial^2 s}{\partial y'^2} = \frac{\partial^2 s}{\partial x^2}\sin^2\theta - \frac{\partial^2 s}{\partial x \partial y}\sin\theta\cos\theta - \frac{\partial^2 s}{\partial y \partial x}\cos\theta\sin\theta + \frac{\partial^2 s}{\partial y^2}\cos^2\theta$$

$$\frac{\partial^2 s}{\partial x'^2} + \frac{\partial^2 s}{\partial y'^2} = \left(\frac{\partial^2 s}{\partial x^2} + \frac{\partial^2 s}{\partial y^2}\right)\underbrace{(\cos^2\theta + \sin^2\theta)}_{=1} + \left(\frac{\partial^2 s}{\partial x \partial y} + \frac{\partial^2 s}{\partial y \partial x}\right)\underbrace{(\cos\theta\sin\theta - \sin\theta\cos\theta)}_{=0}$$

$$= \frac{\partial^2 s}{\partial x^2} + \frac{\partial^2 s}{\partial y^2}$$

The laplacian is the simplest isotropic linear operator. However, after discretization, the sensitivity of the Laplacian varies with the orientation (why?)
This can be solved eg by applying multiple masks, for example:

orientation 0 and $\frac{\pi}{2}$: $f(i+1,j) - 2f(i,j) + f(i-1,j)$
$f(i,j+1) - 2f(i,j) + f(i,j-1)$ — computed here above

orientation $-\frac{\pi}{4}$ and $\frac{\pi}{4}$: $\frac{1}{2}\left(f(i+1,j-1) - 2f(i,j) + f(i-1,j+1)\right)$
$\frac{1}{2}\left(f(i-1,j-1) - 2f(i,j) + f(i+1,j+1)\right)$

la distance en diagonale est $\sqrt{2}$ → on normalise < la divise pour ça et elle passe plus par la dérivée seconde → on divise par $\frac{1}{2}$.

mask 2 = superposition

	-1	0	1
-1	$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$
0	$\frac{1}{6}$	-1	$\frac{1}{6}$
1	$\frac{1}{12}$	$\frac{1}{6}$	$\frac{1}{12}$

↓ ↑ 1/12

0	1	0
1	-4	1
0	1	0

+

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

=

1/2	1	1/2
1	-6	1
1/2	1	1/2

$\times 2$ →

1	2	1
2	-12	2
1	2	1

Σ coeff = 0 car c'est HPF
→ annule la DC.

The discrete 3x3 Laplacian filter has the disadvantage that it will amplify the noise in the image tremendously. Indeed, in the continuous spectral domain, frequencies are multiplied with
- $-\omega^2$ for a 1D second derivative
- $-(u^2+v^2)$ for the Laplacian.

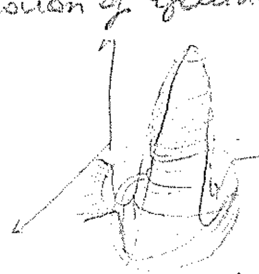
In order to avoid these noise amplification, one can combine the Laplacian with a low pass filter. A low pass filter often use in image processing is the Gaussian filter. It has the advantage that it is an isotropic (rotation invariant filter) and has a flexible parameter σ determining its frequency response. Note that the Fourier transform of a Gaussian with variance σ^2 is also a Gaussian but with variance $\frac{1}{\sigma^2}$. The combination of the Gaussian with the Laplacian gives the Laplacian of Gaussian filter (also called Mexican filter because of its particular shape in the spatial domain.)

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

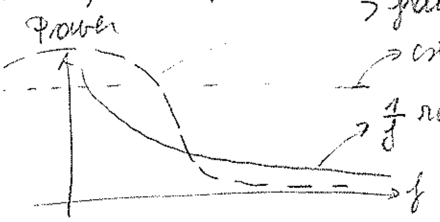
2D Gaussian: $\frac{1}{2\pi\sigma^2} e^{-\left(\frac{x^2+y^2}{2\sigma^2}\right)}$

2D Laplacian of Gaussian

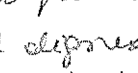
$$LoG(u,v) = (ju\cdot ju + jv\cdot jv)\, G(u,v) = -(u^2+v^2)\, e^{-\left(\frac{u^2+v^2}{2}\right)\sigma^2}$$



The gaussian filter deletes high frequency components of the noisy recorded signal. Hence, it suppresses the noise but also the fine details of the signal.



The choice of σ defines the width of the gaussian filter. For each σ, the contours detected by zero crossing of the Laplacian of a gaussian are continuous. However, some meaningless contours (phantom edges) appears for each σ and other effect can be listed:
- $\sigma <<$ → many spurious responses, thus many edges and too much noise but precise edge location
- $\sigma \sim$ → contours well defined with true edge location and less spurious edges
- $\sigma >>$ → false location of edge because the zero crossing of a gaussian with large σ overlap on 2 edges and 2 closely located edges influences each other

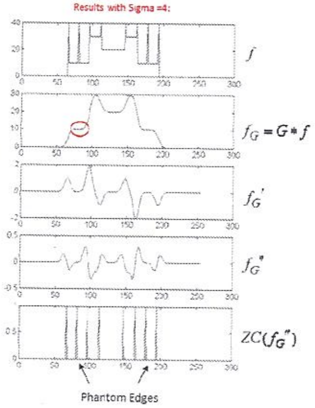


but good noise removal.

Thus σ choice is of critical importance and signal dependent. In practice, combined responses of different σ is used in practice.
An important problem faced with edge detection via zero crossing of LoG is that it can lead to the detection of false edges, called phantom edges. These edges should not be confused with the so-called spurious edges, corresponding to small intensity variations which are in fact real edges, in the image due to noise. Spurious edges disappear at a higher scale, i.e for higher σ. Oppositely, phantom edges are systematic across scales but vanish at $\sigma = 0$. This allow to discriminate between both type of edges.

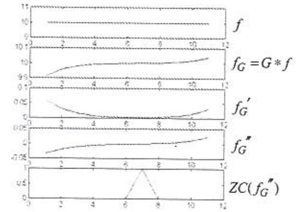
Phantom edges (2/6)

Results with Sigma =4:



Top: Original Signal + Detected Edges by Zero Crossing superimposed (Fifth signal)
Second: Original Signal smoothed by a Gaussian with $\sigma = 4$
Third: First Derivative of the second signal
Fourth: Second Derivative of the second signal
Fifth: Zero Crossings (detected edges) of the fourth signal.

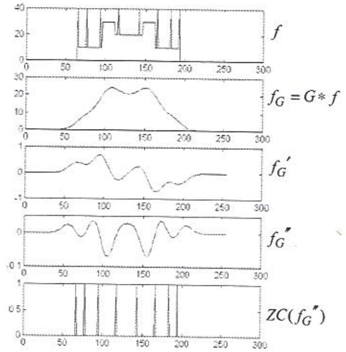
Observations:
Remark the extra phantom edges found by the zero crossing method in the middle of the outer ridge.



Zoom of the area around the middle of the left outer ridge. Remark this phantom edge corresponds to a positive minimum in the first derivative. Also negative maxima in the first derivative (this is the case for the right outer ridge) will lead to phantom edges.

Phantom edges (3/6)

Results with Sigma =10:

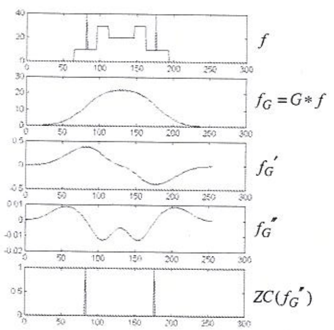


Top: Original Signal + Detected Edges by Zero Crossing superimposed (Fifth signal)
Second: Original Signal smoothed by a Gaussian with s = 10
Third: First Derivative of the second signal
Fourth: Second Derivative of the second signal
Fifth: Zero Crossings (detected edges) of the fourth signal.

Observations:
- The edges are less well localised as with s = 4.
- The phantom edges are shifted as well.

Phantom edges (4/6)

Results with Sigma =20:



Top: Original Signal + Detected Edges by Zero Crossing superimposed (Fifth signal)
Second: Original Signal smoothed by a Gaussian with s = 20
Third: First Derivative of the second signal
Fourth: Second Derivative of the second signal
Fifth: Zero Crossings (detected edges) of the fourth signal.

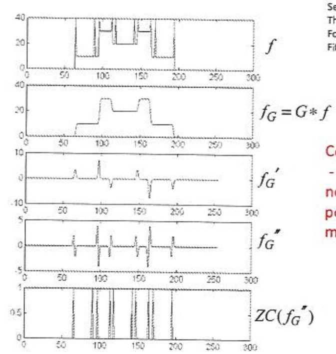
Observations:

- Only the phantom edges remain.

- This is an important rule:
- Phantom edges will remain when s goes towards infinity and will vanish for s = 0, while
- True edges are observable for s = 0 and will vanish when s goes towards infinity.

Phantom edges (5/6)

Results with Sigma =1:



Top: Original Signal + Detected Edges by Zero Crossing superimposed (Fifth signal)
Second: Original Signal smoothed by a Gaussian with s = 1
Third: First Derivative of the second signal
Fourth: Second Derivative of the second signal
Fifth: Zero Crossings (detected edges) of the fourth signal.

Conclusion:
- Zero-crossing detection is unstable in the neighbourhood of saddle points or near-saddle points (almost perfectly flat regions between a maximum and minimum) in the first derivative.

the phantom edges originate from the fact that zero-crossing detection becomes unstable in the neighbourhood of saddle points or near-saddle points, which are almost perfectly flat regions (btw a max. and a min) in the first derivative. Phantom edges can be distinguished from the real edges for 1D signals, by examining the absolute value of the first derivative (1D). Real edges correspond to a local maxima in this signal and phantom edges to a local minima. Hence, to discriminate between both, we can use the heuristic method on the slides briefly or the mathematical criterion:

If the signal is : $h(x,y) = gm(x,y) * G(x,y,\sigma)$
and its gradient is : $\bar{n} = \dfrac{\nabla h(x,y)}{\|\nabla h(x,y)\|} \Leftrightarrow \bar{n} = (\cos\theta, \sin\theta)$

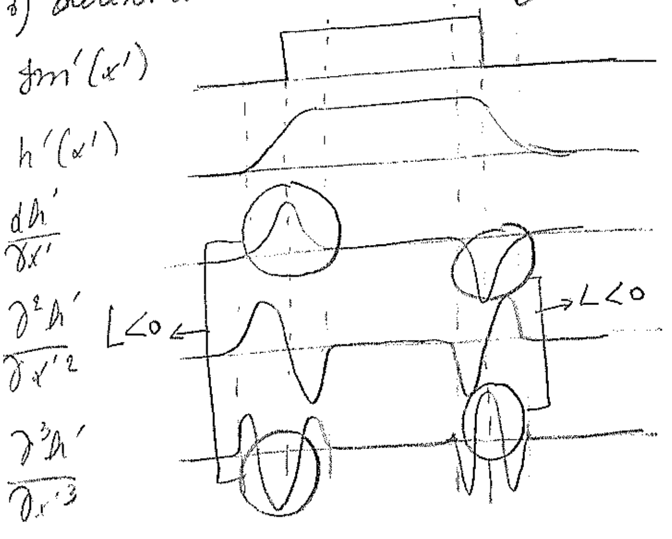
1) take the gradient direction $\bar{n}$
2) rotate the coordinate to be aligned with $\bar{n}$, i.e to be in the gradient direction.

$x = x'\cos\theta - y'\sin\theta$
$y = x'\sin\theta + y'\cos\theta$

3) take the derivative of the signal in the gradient direction to obtain the gradient magnitude

$$\frac{\partial h}{\partial x'} = \frac{\partial h}{\partial x}\frac{\partial x}{\partial x'} + \frac{\partial h}{\partial y}\frac{\partial y}{\partial x'} = \frac{\partial h}{\partial x}\cos\theta + \frac{\partial h}{\partial y}\sin\theta = \frac{\frac{\partial h}{\partial x}\frac{\partial h}{\partial x} + \frac{\partial h}{\partial y}\frac{\partial h}{\partial y}}{\sqrt{\left(\frac{\partial h}{\partial x}\right)^2 + \left(\frac{\partial h}{\partial y}\right)^2}} = \sqrt{\left(\frac{\partial h}{\partial x}\right)^2 + \left(\frac{\partial h}{\partial y}\right)^2} = \|\nabla\|$$

4) detect the edges as corresponding to the maxima of $h'(x')$ which are the maxima of the gradient magnitude in the gradient direction or to the zero-crossing of the Laplacian in gradient direction
5) calculate the third derivative of the signal at zero-crossing points.
6) determine the nature of the edge on the product $\frac{\partial h'}{\partial x'}\frac{\partial^3 h'}{\partial x'^3}$ sign.



$gm'(x') = gm(x,y)$ and $x' = x\cos\theta + y\sin\theta$

$h'(x') = gm(x,y) * G(x,y,\sigma)$ and $x' = \cos\theta x + y\sin\theta$

$\frac{\partial h'}{\partial x'} = \|\bar{\nabla} h\|$ and $\left(\frac{\partial h'}{\partial x'}\right)^2 = \|\bar{\nabla} h\|^2$

$$M := \frac{\partial}{\partial x'}\|\bar{\nabla} h\|^2 = \frac{\partial}{\partial x'}\left(\frac{\partial h'}{\partial x'}\right)^2 = 2\frac{\partial h'}{\partial x'}\frac{\partial^2 h'}{\partial x'^2}$$

↳ edge if local maxima of $\|\bar{\nabla} h\|^2 \rightarrow$ if $M = 0$

$$L := \frac{1}{2}\frac{\partial^2}{\partial x'^2}\|\bar{\nabla} h\|^2 = \frac{1}{2}\frac{\partial^2}{\partial x'^2}\left(\frac{\partial h'}{\partial x'}\right)^2 = \frac{\partial h'}{\partial x'}\frac{\partial^3 h'}{\partial x'^3} + \left(\frac{\partial^2 h'}{\partial x'^2}\right)^2$$

↳ edge if local maxima of $\|\bar{\nabla} h\|^2$ meaning if second derivative in gradient direction is 0
i.e $\frac{\partial^2 h'}{\partial x'^2} = 0 \rightarrow L = \frac{\partial h'}{\partial x'}\frac{\partial^3 h'}{\partial x'^3}$

Hence, if edge $\rightarrow \frac{\partial^2 h'}{\partial x'^2} = 0 \rightarrow L = \frac{\partial h'}{\partial x'}\frac{\partial^3 h'}{\partial x'^3}$
- if $L < 0 \rightarrow$ real edge
- if $L \geqslant 0 \rightarrow$ false edge (saddle point)

However, even with good results with LoG, people still prefer to use 1st order derivatives methods because:
it gives information about edge location + edge magnitude (only location for 2nd order)
they are faster operators
they don't give phantom edges
• 2nd order operators need ill posed operators ($\frac{\partial^3 h'}{\partial x'^3}$ do not work for non continuous signal)

"Plus" operator (à penna mi)

This operator is a combination of quadr and belongs to the family of zero crossing procedures for edge detection and is the most accurate one.

$PLUS(I(x,y)) = SDGD(I(x,y)) + Lapl(I(x,y))$ with SDGD is the Second Derivative in Gradient Direction:

$$SDGD = \frac{\partial^2 I}{\partial x'^2} = \frac{\partial^2 I}{\partial x^2}\cos^2\theta + \frac{\partial^2 I}{\partial x \partial y}\cos\theta\sin\theta + \frac{\partial^2 I}{\partial y \partial x}\sin\theta\cos\theta + \frac{\partial^2 I}{\partial y^2}\sin^2\theta$$

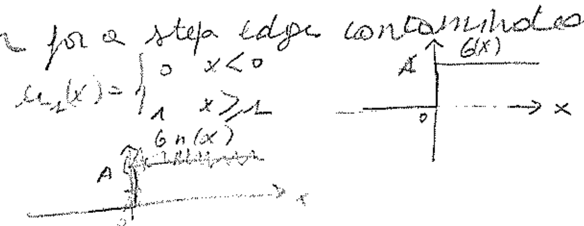
$$= I_{xx}(I_x)^2 + 2I_{xy}(I_xI_y) + I_{yy}\frac{(I_y)^2}{(I_x)^2} + I_y^2$$

Typically, the Laplacian-based zero crossing procedure overestimates the position of the edge and the SDGD-based procedure underestimates the position thus the combination of both methods tends toward the true position

Canny edge detection (A computational approach to edge detection)

Canny tried to find an optimal edge detector for a step edge contaminated with additive Gaussian noise: $G(x) = A u_{-1}(x)$ with $u_1(x) = \begin{cases} 0 & x<0 \\ 1 & x \geq 1 \end{cases}$

$G_n(x) = G(x) + n(x)$



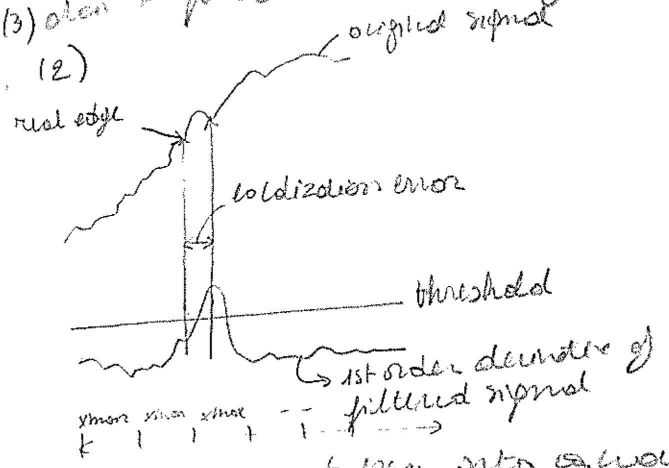
The optimal criterion to be satisfied by this 1st order derivative detectors are:

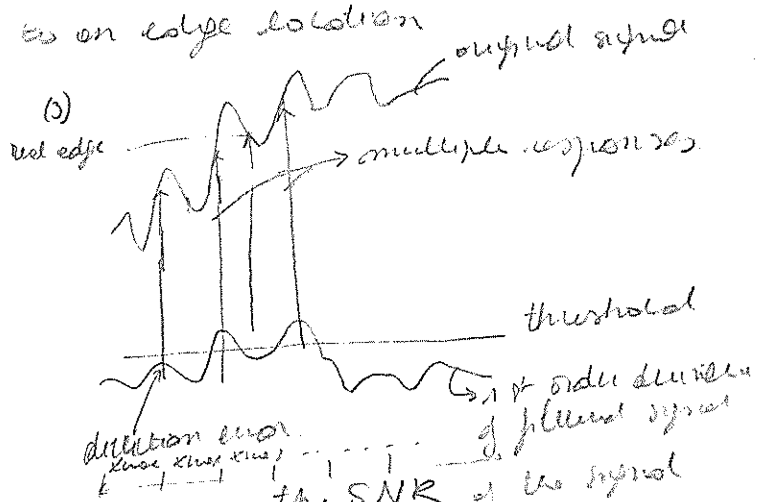
(0) linear filtering

(1) don't miss exact edge but don't respond to noise

(2) don't miss exact edge location

(3) don't produce too many responses to an edge location





(1) This criterion is taken into account by computing the SNR of the signal

$$SNR = \frac{A\left|\int_{-W}^{0} f(x)\,dx\right|}{n_0\sqrt{\int_{-W}^{W} f^2(x)\,dx}}$$ with n_0^2 = mean squared noise amplitude per unit length, f = filter

$$= \frac{A}{n_0}\Sigma(f)$$ with Σ the quotient $\frac{|\int_{-W}^{0}|}{\sqrt{\int_{-W}^{W}}}$

and for a scaled filter $f_s(x) = f\left(\frac{x}{s}\right)$, $\Sigma(f_s) = \sqrt{s}\,\Sigma(f)$ > PROOF

$\Sigma(f_s) = \frac{|\int_{-W}^{0} f(\frac{x}{s})\,dx|}{\sqrt{\int_{-W}^{W} f^2(\frac{x}{s})\,dx}}$ $u = \frac{x}{s}$ $du = \frac{dx}{s}$

$= \frac{s|\int_{-W}^{0} f(u)\,du|}{\sqrt{\int_{-W}^{W} f^2(u)\,s\,du}} = \frac{s}{\sqrt{s}}\Sigma f = \sqrt{s}\,\Sigma f$

The demonstration for this expression of the SNR comes from the Wiener-Khitchine theorem which state that the PSD of a signal is the Fourier transform of its auto-correlation: $S_x(\omega) = \mathcal{F}(\rho_{xx}(\tau))$

As the SNR is the ratio of the signal power over the noise power, computation ...

$$G(x) + n(x) \rightarrow \boxed{f} \xrightarrow{H(x)} \quad \begin{cases} G(x) \rightarrow \boxed{f} \xrightarrow{H_G(x)} \\ n(x) \rightarrow \boxed{f} \xrightarrow{H_n(x)} \end{cases}$$ by linearity of f.

signal

$$H_G(x) = G(x) * f(x) = \int_{-\infty}^{+\infty} G(x) f(x-u)\,du = \int_{-\infty}^{+\infty} G(x-u) f(u)\,du.$$

$$H_G(0) = \int_{-\infty}^{+\infty} G(-u) f(u)\,du = A \int_{-\infty}^{0} f(u)\,du \text{ by definition of } G(x)$$
$$\rightarrow S_{H_G}(\omega) = |H_G(\omega)| = A \left| \int_{-\infty}^{0} f(u)\,du \right|$$

Noise

$H_n(x) = n(x) * f(x)$ with $E(n(x)) = 0$, $E(n^2(x)) = \sigma^2 = $ cste and $n(x)$ is a stationary process, i.e. its statistical properties don't vary over time, i.e. $E(x(t_0)) = E(x(t_0+\tau))$

$$S_{H_n}(\omega) = \mathcal{F}(\rho_{H_n}\rho_{H_n}(\tau)) \text{ by W-K} \quad (1)$$
$$= S_n(\omega) \cdot |F(\omega)|^2 \text{ for LTI systems} \quad (2)$$

→ PROOF: (look $\mathcal{F}^{-1}(\ldots F(\omega)) = f(\tau) * f(-\tau)$)

As (1) & (2):

$$\rho_{H_nH_n}(\tau) = \rho_{nn}(\tau) * \mathcal{F}^{-1}\{\underbrace{|F(\omega)|^2}_{F(\omega)\cdot F(\omega)^*}\}$$
$$f(\tau) * f(-\tau) = \int_{-\infty}^{+\infty} f(u) f(u+\tau)\,d\tau$$

$$= \rho_{nn}(\tau) * \underbrace{f(\tau) * f(-\tau)}_{=: z(\tau)}$$

$$= \rho_{nn}(\tau) * z(\tau) \quad \text{and as } \rho_{nn}(\tau) = E(n^2(x)) = \sigma^2$$

$\sigma^2 \delta(\tau)$, τ

$$= \sigma^2 z(0) = \sigma^2 \int_{-\infty}^{+\infty} f(u)^2\,du \rightarrow \rho_{H_nH_n}(0) = \underset{n_0^2}{\sigma^2} \int_{-\infty}^{+\infty} f(u)^2\,du \rightarrow H_n = n_0 \sqrt{\int_{-W}^{W} f^2(x)\,dx}$$

SNR

$$SNR(f) = \mathcal{F}^{-1}\left(\frac{S_{H_G}}{S_{H_n}}\right) = \frac{A \left|\int_{-W}^{0} f(x)\,dx\right|}{n_0 \sqrt{\int_{-W}^{W} f^2(x)\,dx}}$$

(2) This criterion is taken into account by computing the LOC of the signal

$$LOC = \frac{A\,|f'(0)|}{n_0 \sqrt{\int_{-W}^{W} f'(x)^2\,dx}} \quad \text{with } n_0^2 = \text{idem}, \; f = \text{idem}$$

$$\Lambda(f_s') = \frac{A\,|f'(0)|}{n_0 \sqrt{\int_{-W}^{W} f'\left(\frac{x}{s}\right)^2 dx}}, \quad u = \frac{x}{s}, \; du = \frac{dx}{s}$$

$$= \frac{A}{n_0} \Lambda(f') \quad \text{with } \Lambda \text{ the quotient } \frac{|\;|}{N\sqrt{\;\;}^2}$$

and for a scaled filter, $\Lambda(f_s') = \frac{1}{\sqrt{s}} \Lambda(f')$ → PROOF: $\Lambda(f_s') = \frac{A\,|f'(0)|}{n_0\sqrt{s}\sqrt{\int_{-W}^{W} f'(u)^2\,du}}$

The demonstration for this is the following: the ... derivative quotient. Hence, if the let's say that the filter $H_G(x)$ is a 1st order derivative then $H_G(0)$ should be maximum in 0 edge is in 0 (for Heaviside function G) then $H_G'(0)$ should be equal to 0 in 0 but, because of noise, the response of the filter is $H_G(x_0) + H_n(x_0)$ and the zero crossing happens at x_0: (negligible since x_0 is small)

$$H_G'(x_0) + H_n'(x_0) = 0 \quad (3)$$

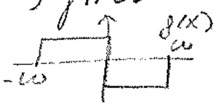
The Taylor series development of $H_G'(x_0) = \overset{0}{H_G'(0)} + x_0 H_G''(0) + \ldots$

$$\hookrightarrow H_G'(x_0) = x_0 H_G''(0) \Leftrightarrow x_0 = \frac{H_G'(x_0)}{H_G''(0)} = \frac{-H_n'(x_0)}{H_G''(0)}$$

The variance of x_0, which is a random variable, is $\frac{\text{variance } H_n'(x_0)}{\text{variance } H_G''(0)}$ → smaller variance, so the better LOC ...

HW: if the process is nonstationary, is the 1st order derivative necessary?

The criterion for optimality based on the maximization of the product SNR. LOC (scale invariant parameter) gives as a result the filter called "difference of the boxes operator"



and is extremely sensitive to noise. Hence a 3rd criterion is necessary to define an optimum filter which is the criterion (3).

3) This criterion is taken into account by the probability of false edge detection

number of noise maxima in region $2W$: $N_n = \frac{2W}{x_{max}(f)}$ $\Big\}$ $N = \frac{2}{k}$ fix nb of noise max

average distance $x_{max}(f) = 2x_{zc}(f) = \pi \sqrt{\frac{\int_{-W}^{+W} f'(x)^2 dx}{\int_{-W}^{+W} f''(x)^2 dx}} = kW$

distance btw maxima in noise response of f | distance btw zero crossing in f'

$P_{multiple\ response\ error} = 1 - \phi\left(\frac{A|f'(0)|}{n_0 \Sigma}\right)$ with $n_0 \Sigma = N\sqrt{\int_{-W}^{+W} f''(x)^2 dx}$

→ prob. of finding another maximum near the center of the edge. → normal distribution fct

$P_{false\ marking\ an\ edge} = 1 - \phi\left(\frac{A}{n_0}\Sigma\right)$

$P_m = P_f \longrightarrow \frac{|f'(0)|}{T_S} = \Sigma \simeq \frac{|f'(0)|}{T_S} = n\Sigma$ (second criterion)

→ detection of error or multiple response error equally likely

Given these 3 criterions, the optimal problem can be solved using the boundary conditions: lagrangian, giving as a result the filter satisfying the boundary conditions:

$f(x) = a_1 e^{ax}\sin\omega x + a_2 e^{ax}\cos\omega x + a_3 e^{-ax}\sin\omega x + a_4 e^{-ax}\cos\omega x + c$

The shape depends on the multiple response constraints x_{max}.

→ slide 50,61 A LIRE

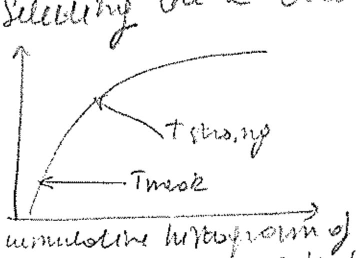
By comparison, the 1st derivative of Gaussian is worse than the optimal Canny edge detector by 20% for $\Sigma\Lambda$ (SNR. LOC) and 10% for n. As the difference is difficult to see for real image, the FDG operator is often used because simpler to implement in 2D. Other novelties introduced by Canny is the fact that Canny works with directional operators in 2D, i.e it computes the FDG for a nb of discrete direction, and it introduces the concept of thresholding with hysteresis. This means that, defining 2 thresholds, if there is a weak (over a threshold value T_1) connecting 2 strong edges (over a threshold T_2), the edge is kept. Hence:

$T > T_2$ keep, $T_2 > T > T_1$ → keep if connecting strong edges, $T < T_1$ → throw

Hence the steps for Canny edge detection are:

1) Computing the gradient magnitude with masks based on the 1st derivative of the Gaussian

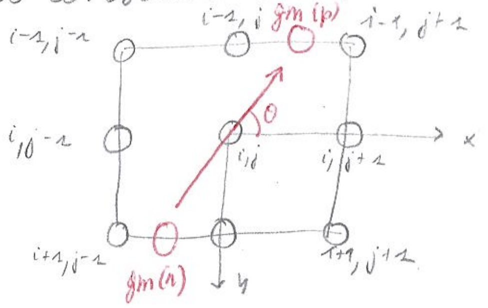
2) Selecting the 2 thresholds (T_{strong}) and (T_{weak})



cumulative histogram of the gradient magnitude

Tstrong is chosen such that a certain percentage (e.g 70%) of the pixels will not be responded as strong edges. Hence, one has to find the value in the cumulative histogram of the gradient magnitude image corresponding to the percentage of the pixels. Tweak is set at a certain percentage Y (e.g 40%) in the CCH

(3) Computing the gradient magnitude along the gradient direction via an interpolation strategy in a 3×3 neighbourhood and determining the pixels, which are maxima in the gradient magnitude along the gradient direction.
Indeed, for a given pixel and given discrete gradient direction, as there might be no pixel value in the direction, interpolation is necessary to construct its value:



O pixel needed to be computed by interpolation

↗ gradient direction

By linear interpolation,

$gm(p) = (1 - ctg(\theta))\, gm(i-1, j) + ctg(\theta) \cdot gm(i-1, j+1)$

$gm(n) = (1 - ctg(\theta))\, gm(i+1, j) + ctg(\theta)\, gm(i+1, j-1)$

if $gm(p) < gm(i, j)$ and $gm(n) < gm(i, j)$, the pixel (i, j) is a local maximum along the gradient direction and is stored as an edge pixel. The value $gm(i, j)$ gives the strength of the edge at (i, j)

(4) Selecting all the weak edges with a gradient magnitude > Tweak in the different gradient orientation

(5) Idem with strong edges

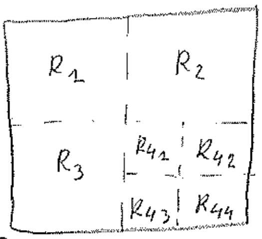
(6) Thresholding with hysteresis

(7) Thin the edges so that they become one pixel wide (not the case initially because gradient direction changes every pixel because of noise)

REGION GROWING

SPLIT & MERGE

Considering an image, this technique splits into 4 quadrants any region R when $P(R_i) = FALSE$ with P a predicate. Then it merges any adjacent regions R_j and R_k if $P(R_j \cup R_k) = TRUE$. Stop when no further splitting or merging is possible.



↳ see slide 4

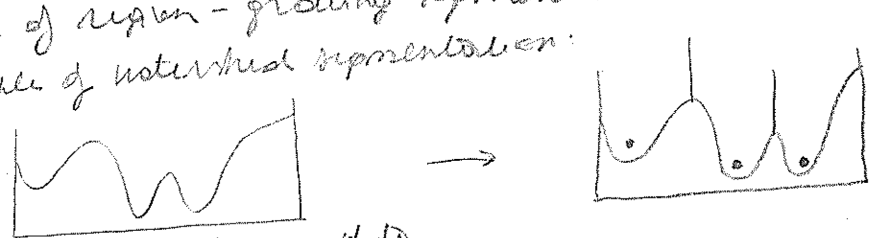
The problem is that edges are not well respected and jagged region borders appear as adjacent blocks with similar intensity are disjoint due to the quadtree position of the corresponding tree nodes. Different methods can be used in the splitting step, depending on the image contents. Starting from an over-segmented image (i.e. the one that is produced after the splitting step), the probability of loosing important information is minimized. The final segmentation can be optimized for a particular application, by selecting an appropriate merging criterion. In a semi-interactive segmentation scheme, the user may participate in the merging process by indicating which micro-segments are allowed to be merged (slide 6)

WATERSHED TRANSFORM

Image data is interpreted as a topographic surface where the image gray-levels represent altitudes. The goal is to construct:
- watershed lines that divide individual catchment basins
- catchment basins which corresponds to high watersheds and low gradient (↳ steep edges) region interiors. Catchment basins are homogeneous in the sense that all pixels belonging to the same catchment basin are connected with the basin's region of minimum altitude (gray level) by a simple path of pixels that have monotonically decreasing altitude (gray level) along the path. Such catchment basins then represent the regions of the segmented image.

The goal of region-growing segmentation is to create homogeneous regions.

1D example of watershed segmentation:

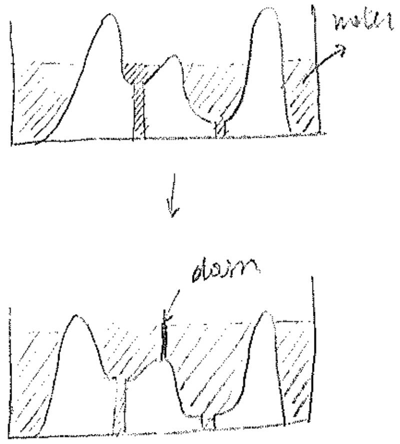


| watersheds = local maxima
• catchment basins = local minima

↳ gray profile of image data

There are 2 basic approaches to watershed image segmentation:
- downstream path approach: construction from local maxima to local minima by finding a path from each pixel of the image to a local minimum of image surface altitude. A catchment basin is then defined as the set of pixels for which their respective downstream paths all end up in the same altitude minimum. Downstream paths are easy to determine for continuous altitude surfaces by calculating the local gradients but no rules exist to define the downstream path uniquely for digital surfaces.

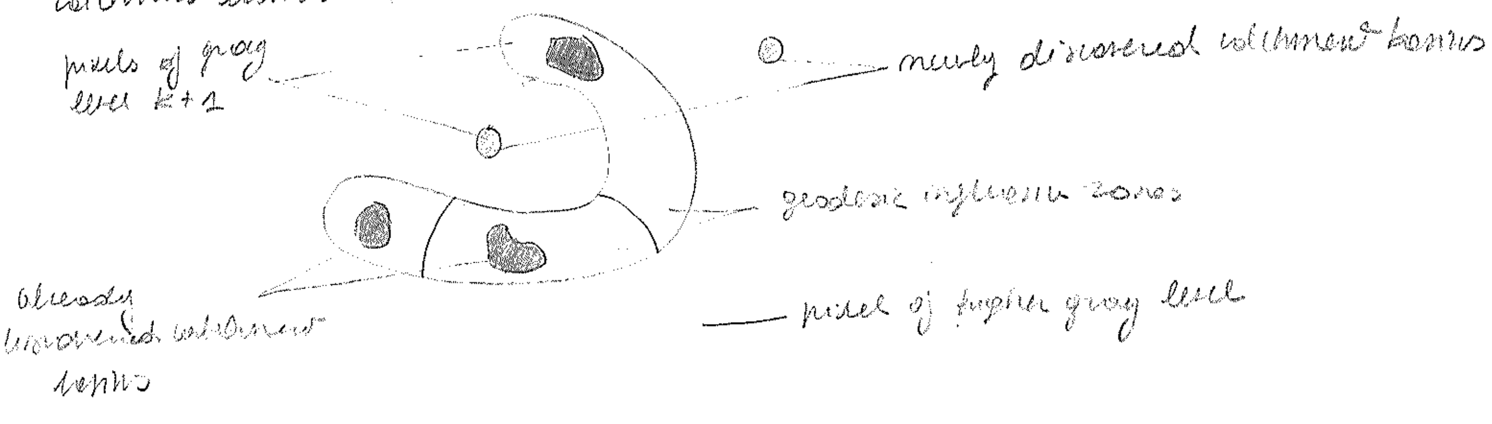
- upstream approach: more robust, construct from local minima to local maxima by defining segment as pixels belonging to the same "valley":



To do this method, it is imagined that there is hole in each local minimum and that the topographic surface is immersed in water. As a result, the water starts filling all catchment basins, minima of which are under the water level. If 2 catchment basins would merge as a result of further immersion, a dam is build all the way to the highest surface altitude and the dam represents the watershed line.

An efficient algorithm for bottom-up filling approach exists. The algorithm is based on sorting the pixels in increasing order of their gray values, followed by a "flooding" step, consisting of a fast breadth-first scanning of all pixels in the order of their gray-levels:

- suppose flooding has been completed up to a level k (gray-level altitude)
- then every pixel having gray level less than or equal to k has already be assigned a unique catchment basin label.
- next, pixels having gray-level k+1 may belong to a catchment basin labeled L if at least one of its neighbors already carries this label.
- pixels that represent potential catchment basin members are put in a FIFO queue and await further processing
- geodesic influence zones are computed for already determined catchment basins. A geodesic influence zone of a catchment basin L_i is the locus of non-labeled image pixels of gray-level k+1 that are contiguous with the catchment basin labeled L_i for which their distance to L_i is smaller than their distance to any other catchment basin L_k.
- All pixels with gray-level k+1 that belong to the geodesic influence zone of a catchment basin labeled L_i are also labeled with the label L_i, thus causing the catchment basin to grow
- The pixels from the queue are processed sequentially and all pixels from the queue that cannot be assigned an existing label represent newly discovered catchment basins and are marked with new and unique labels



the borders of the regions will only coincide with the edges when the
transform is applied to the modulus of the gradient of the image. This ↳ slide 23
is equivalent to applying edge detection to the original image. In that
case, the catchment basins should theoretically correspond to the homogeneous
gray level regions of the image. However, in practice, this transform produces
an important over-segmentation due to noise or local irregularities in
the gradient image. To overcome this, marker based segmentation is
performed: they allow to determine the number of segments to be kept.

To define a marker:

- define an amplitude threshold T under which the signal is considered to be noise such that, on the new image, the average amplitude $\gg T$.
- define a minimum size of the local minimum areas such that, to be considered as a catchment basin, the area must be $\geq S_k$ with pixels original value $< T$.

N.B. applying a watershed on a denoised image thanks to a Gaussian filter reduces ↓ nb of segment but, because of σ of gaussian, edges are not properly localized → solution is to compute nonlinear diffusion i.e. ↓σ of gaussian when edges are encountered.
↳ cf slide 23, 24, 25, 26 ⚠

In practice, multiresolution segmentation techniques are used as they allow a global understanding and detailed understanding of the situation. Every resolution gives different information needed for full understanding. They consider the image at different resolution levels, which are used to guide the decision process for the constitution of the segments. Images at each level of the hierarchy correspond to the successive levels of the Gaussian pyramid of the original image. Coarse levels in the hierarchy yield only a few regions and are used to guide the segmentation at finer levels, in which new regions are progressively introduced.

Slide 22 (AIMIRISEG). We can see from the middle pillar last row that we [have] between two segments and they are not well aligned. This is useful to construct the segment hierarchy in the scale space. For example, from last pillar on the right, we get 1 segment and its position has to be refilled with the 1st left pillar and so on and so on. We can thus define how many segments are created on a pillar with their exact position.

N.B: to find the gradient of the image:

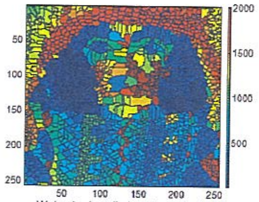
- compute Sobel / Prewitt on the filtered image with the Gaussian filter to get $\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}$ ⇔ $(\nabla G_\sigma) * I$ (i.e. first derivative of the Gaussian), as indeed Sobel & Prewitt gradients are too noise sensitive and that are ill-posed, meaning that the discrete gradients (there are multiple choices on the derivative to use (left or right))

The watershed lines will correspond to the maxima of the gradient magnitude which corresponds the maxima in the edges.

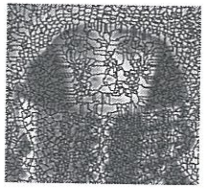
CH6

Watershed: Results with Matlab (1)

Original image

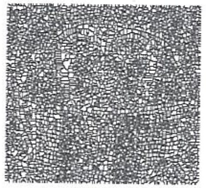


Watershed applied on the original image: about 2000 regions

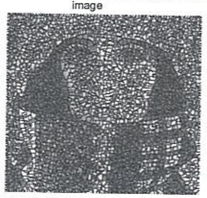


Original image plus watershed contours. Note that the contours don't follow the edges in the image

Modulus of the gradient image



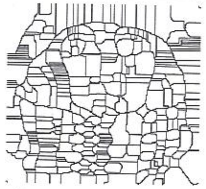
Watershed applied on the modulus of the gradient image



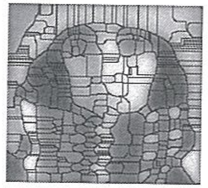
Original image plus watershed contours. Note that the contours follow the edges.

Watershed: Results with Matlab (2)

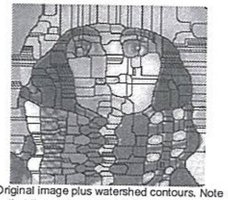
Modulus of the gradient of the smoothed image



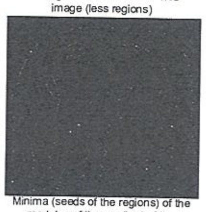
Watershed applied on the modulus of the gradient of the smoothed image (less regions)



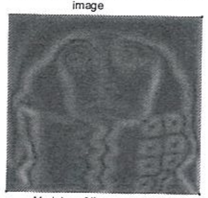
Smoothed image plus watershed contours. Note that the contours follow the edges in the image



Original image plus watershed contours. Note that the contours are dislocated from the edges of the original image

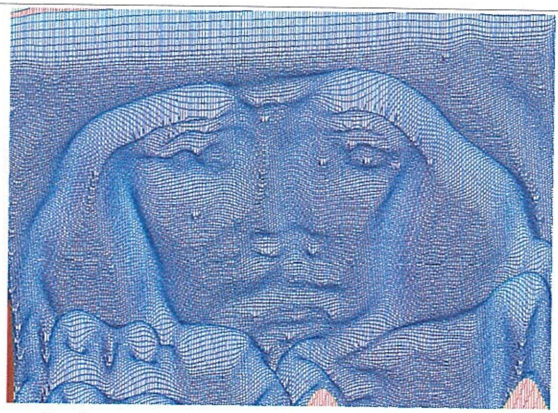


Minima (seeds of the regions) of the modulus of the gradient of the smoothed image



Modulus of the gradient of the smoothed image plus minima (seeds of the regions) in black

Watershed: Results with Matlab (3)



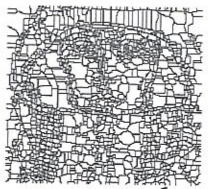
Elevation map of the modulus of the gradient of the smoothed image plus minima (seeds of the regions) represented by small holes

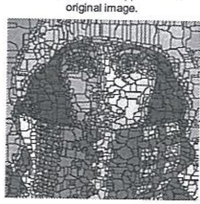
Watershed: Results with Matlab (4)

Original image

Anisotropic diffusion applied on the original image.

Modulus of the gradient of the image processed with anisotropic diffusion





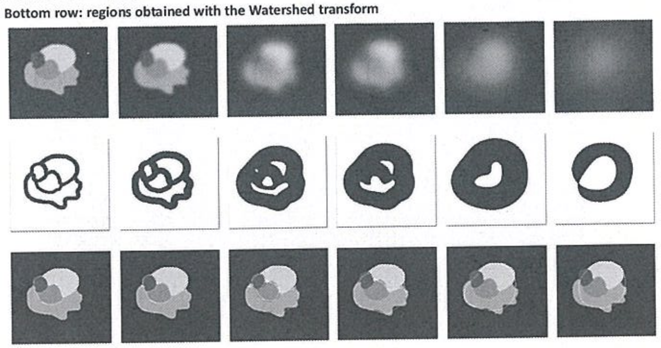
Original image plus watershed contours computed for the modulus of the gradient of the image processed with anisotropic diffusion

(less segments, located on the edges)

Watershed contours computed for the modulus of the gradient of the image processed with anisotropic diffusion

Annihilation and merging of minima in scale space

Top row: smoothing of the original image with a gaussian filter with increasing σ from left to right
Middle row: merging of the catchment basins for increasing σ
Bottom row: regions obtained with the Watershed transform



The Distance Transform

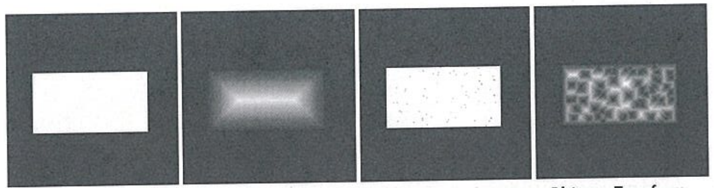
Examples

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

→

0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	2	2	2	2	1	0
0	1	2	3	3	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

Distance Transform – Chessboard distance



Original binary image | **Distance Transform - Euclidean distance** | **Original noisy image** | **Distance Transform - Euclidean distance**

Efficient computation of the distance transform via a two-pass algorithm

```
distrans (int **inim, int **outim, int bg, int low_val, int high_val)
{
  int x, y, min_dist;

  for (x = 1; x < im_height-1; x++)
   for (y = 1; y < im_width-1; y++)
    if (inim[x][y] == bg)
      outim[x][y] = 0;
    else if ((outim[x][y] = border(x, y, low_val, high_val, inim)) == 0)
    {
      min_dist = outim[x][y-1] + low_val;
      min_dist = min (min_dist, outim[x-1][y] + low_val);
      min_dist = min (min_dist, outim[x-1][y-1] + high_val);
      min_dist = min (min_dist, outim[x-1][y+1] + high_val);
      outim[x][y] = min_dist;
     }
  for (x = im_height-2; x > 0; x--)
   for (y = im_width-2; y > 0; y--)
    if (inim[x][y] == bg)
      outim[x][y] = 0;
    else
    {
      min_dist = outim[x][y];
      min_dist = min (min_dist, outim[x][y+1] + low_val);
      min_dist = min (min_dist, outim[x+1][y] + low_val);
      min_dist = min (min_dist, outim[x+1][y-1] + high_val);
      min_dist = min (min_dist, outim[x+1][y+1] + high_val);
      outim[x][y] = min_dist;
    }
}
border(int X, int Y, int low_val, int high_val, int **im)
{
  register int I, J;

  for (I=X-1; I<=X+1; I++)
    for (J=Y-1; J<=Y+1; J++)
      if (((X != I) || (Y != J))
          && (I >= 0) && (J >= 0) && (I <im_height) && (J < im_width))
               if (im[I][J] != im[X][Y])
                  if ((X == I) || (Y == J))
                    return (low_val);
                  else
                    return(high_val);
  return (0);
}
```

3	2	3
2		

Example using the chamfer distance (low_val = 2; high_val = 3)

For approximation of the Euclidean distance divide by 2

Scanning Order

b	b	b	b	b	b	b	b
b	2	2	2	2	2	2	b
b	2	4	4	4	4	3	b
b	2	4	6	6	6	3	b
b	b	2	4	6	6	3	b
b	b	2	4	6	6	3	b
b	b	b	b	b	b	b	b

		2
3	2	3

b	b	b	b	b	b	b	b
b	2	2	2	2	2	2	b
b	2	4	4	4	4	2	b
b	2	3	5	6	4	2	b
b	b	2	4	4	4	2	b
b	b	2	2	2	2	2	b
b	b	b	b	b	b	b	b

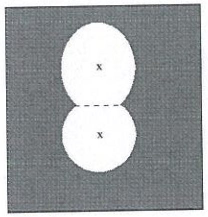
Scanning Order

Cavity Detector

Preprocessing	Discriminator: Objects/Background	Detector	Separator	Accurator
Noise Reduction →	Rough segmentation →	Root selection →	Segment Delineation →	Boundary Refinement
Gaussian Blurring Median Filtering ...	Local/Global thresholding Logical or Morphological filtering	Distance Transform of the binary image Maxima = Seeds Logical or Morphological filtering	Region growing Linking with the seeds along the steepest path in the Distance Transform	Merging of segments on greyvalue, ..., knowledge based criteria in general

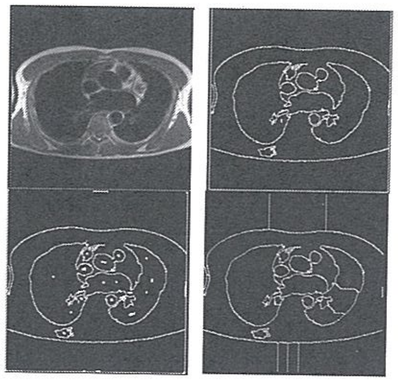
Removal of small objects, taking into account their grey value, size, shape and position in the image

The Distance Transform is sensitive to noise which produces irregularities in the borders of the regions → spurious maxima. It is logical to merge maxima for which the sum of their heights is larger than the geometrical distance in between them.



Cavity Detector: the different stages

Original image



Discriminator: Objects/Background

- Initial definition of the segments
- Logical or Morphological filtering

Detector

- Root detection
- Local maxima of the Distance Transform
- Logical or Morphological filtering

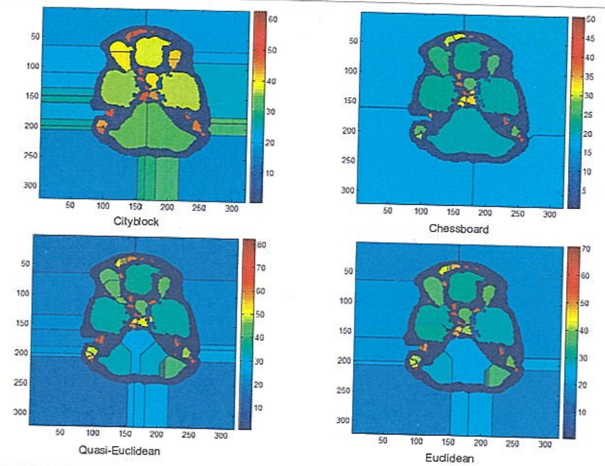
Separator

- Segments are grown along the steepest path in the Distance Transform

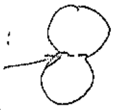
Accurator

- Border refinement

Cavity Detector: Results with Matlab: Step: Watershed (plus effect of using different kind of distance transforms)



Particle detection

The goal is to separate partially connected contours from each other:  Separation here

This can be done with the distance transform which converts a binary image consisting of foreground (feature) and background (nonfeature) elements into a gray level image, where each pixel value in the foreground indicates the distance to the nearest background element. The calculation of the exact euclidean distance transform is a computationally intensive task and, therefore, approximations are often utilized:

- $\text{chessboardDist}((x_1, y_1), (x_2, y_2)) = \max(|x_1 - x_2|, |y_1 - y_2|)$
- $\text{cityBlockDist}((x_1, y_1), (x_2, y_2)) = (|x_1 - x_2| + |y_1 - y_2|)$
- $\text{chamferDist}((x_1, y_1), (x_2, y_2)) = \begin{cases} |x_1 - x_2| + 0{,}5\,|y_1 - y_2|, & |x_1 - x_2| > |y_1 - y_2| \\ 0{,}5\,|x_1 - x_1| + |y_1 - y_2| & \text{otherwise} \end{cases}$
- $\text{quasiEuclDist}((x_1, y_1), (x_2, y_2)) = \begin{cases} |x_1 - x_2| + (\sqrt{2} - 1)\,|y_1 - y_2|, & |x_1 - x_2| > |y_1 - y_2| \\ (\sqrt{2} - 1)\,|x_1 - x_2| + |y_1 - y_2| & \text{otherwise.} \end{cases}$

However, this transform is very sensitive to noise (cf slide 33) and it is thus necessary to apply a filter on the image before. The filtering is done as a such:

(IMPRIMER SLIDE 34)

It is a 2 pass algorithm. The first pass goes from left up to right down. An element-by element product is achieved by the superposition of the image and the filter. The value at the right down of the filter: $\begin{array}{|c|ccc}\hline & & & \\ \hline ② & & & \\ \hline\end{array}$ is replaced by the minimum of the result of the product.

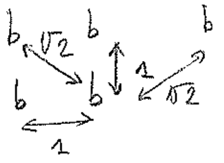
+9 (6) does [illegible]

$\begin{array}{|c|}\hline 3 \\ 2 \\ \hline\end{array}$ 2 | 3 ! , $\begin{array}{|c|}\hline 6 \\ \hline 6 \\ \hline\end{array}$ 4 | 6 | ?

Ex: $\begin{array}{|c|c|c|}\hline 6 & 6 & 6 \\ \hline 6 & & \\ \hline\end{array} * \begin{array}{|c|c|c|}\hline 3 & 2 & 3 \\ \hline 2 & & \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 3 & 2 & 3 \\ \hline 2 & 2 & \\ \hline\end{array}$

$\begin{array}{|c|c|c|}\hline 2 & 2 & 2 \\ \hline 2 & & \\ \hline\end{array} * \begin{array}{|c|c|c|}\hline 3 & 2 & 3 \\ \hline 2 & & \\ \hline\end{array} = \begin{array}{|c|c|c|}\hline 6 & 4 & 6 \\ \hline 4 & 4 & \\ \hline\end{array}$

The values of the filter comes from the distances btw the points:



and $\sqrt{2} \simeq 1{,}41 \simeq 1{,}5$

$\hookrightarrow \begin{array}{|c|c|c|}\hline 1.5 & 1 & 1.5 \\ \hline 1 & & \\ \hline\end{array} \xrightarrow[\times 2]{} \begin{array}{|c|c|c|}\hline 3 & 2 & 3 \\ \hline 2 & & \\ \hline\end{array}$

The scanning order matters!

(IMPRIMER SLIDE 36-38-39)

CLUSTERING

the goal is to classify one pixel of the image to an object (person, truck, dog ...) mathematically this is expressed as finding a set of classes Ω_s, $s \in \{1, 2, \dots k\}$ and assign each pixel (or possibly group of pixels) in the image to one of the classes Ω_s. There exist 2 major approaches: the supervised and unsupervised approach.

SUPERVISED APPROACH

Pixels and/or pixels grouped in regions (e.g. segments with arbitrary shape, rect. window) are considered as independent image objects to be classified.

There are 2 phases:

- the learning phase: the user tells the algorithm how objects should be classified. In this phase, the algorithm is trained and the class membership are learned.
- the classification phase: an arbitrary input is given and classified using the models derived in the learning phase.

The method consists in characterizing objects by feature vectors $\bar{x} = (x_1, x_2, \dots x_n)^T$. The object is which have to be assigned to a class Ω_s ($s \in \{1, 2, \dots k\}$). The object is assigned to its most probable class Ω_t, which equivalent to finding the class that maximizes $P(\Omega_t | \bar{x})$ which, by Bayes theorem is:

$$P(\Omega_t | \bar{x}) = \frac{P(\Omega_t)\, P(\bar{x}|\Omega_t)}{\sum_{s=1}^{k} P(\Omega_s)\, P(\bar{x}|\Omega_s)} =: g_t(\bar{x})$$

and, as the denominator is constant for all classes, maximizing $P(\Omega_t|\bar{x})$ is equivalent to maximizing $P(\Omega_t)\, P(\bar{x}|\Omega_t) =: d_t(\bar{x})$.

The probability distribution $P(\bar{x}|\Omega_t)$ is chosen to be modeled as a generalized normal distribution known by the TCL (i.i.d.), it is a good choice. Hence,

$$P(\bar{x}|\Omega_t) = \frac{1}{(2\pi)^{n/2} |\Sigma_t|^{1/2}} \exp\left(-\frac{1}{2} (\bar{x} - \bar{\mu}_t)^T \Sigma_t^{-1} (\bar{x} - \bar{\mu}_t)\right)$$ and,

$$d_t'(\bar{x}) = -\frac{1}{2} \log|\Sigma_t| - \frac{1}{2} (\bar{x} - \bar{\mu}_t)^T \Sigma_t^{-1} (\bar{x} - \bar{\mu}_t) + \log P(\Omega_t)$$ → quadratic distribution functions

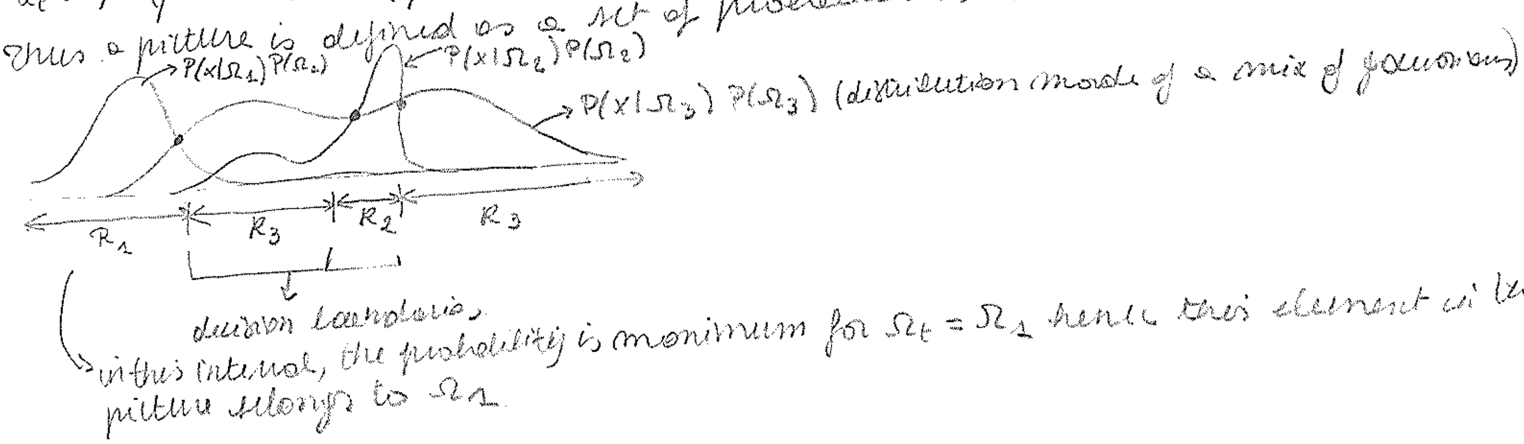
↓

if $\Sigma_t = \Sigma \ \forall t$

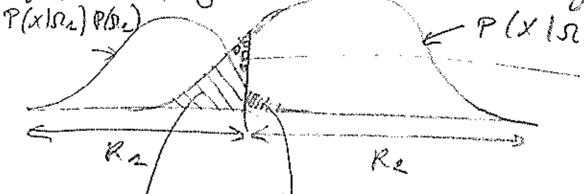
↓

$$d_t''(\bar{x}) = (\bar{\mu}_t^T \Sigma^{-1}) \bar{x} - \frac{1}{2} \bar{\mu}_t^T \Sigma^{-1} \mu_t + \log P(\Omega_t)$$

Thus a picture is defined as a set of probabilities such as:



... boundaries are chosen between any Ω_k and Ω_p determined by $P(\bar{x}|\Omega_k)P(\Omega_k) = P(\bar{x}|\Omega_p)P(\Omega_p)$ such that they minimize the probability of error, if chosen arbitrarily as:

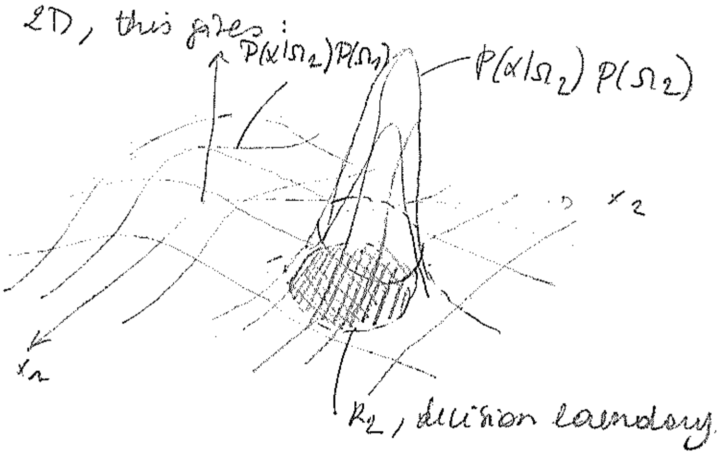


→ the probability errors will be minimum when this area is null, hence we displace the decision boundary until it is.

$\int_{R_2} P(x|\Omega_1)P(\Omega_1)\,dx$ ⇒ this area is the probability error of choosing $x \in \Omega_2$ while it belongs to Ω_1

$\int_{R_1} P(x|\Omega_2)P(\Omega_2)\,dx$ ⇒ this area is the probability error of choosing $x \in \Omega_1$ while it belongs to Ω_2

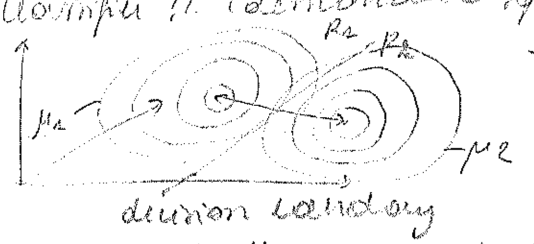
In 2D, this gives:



If we define the mahalanobis distance as the distance between an element x_i of a population and the centroid y_i of the population with a normalization by σ_i as the variance varies depending on the direction (in 3D: ≠ variance by direction), we get:

$D_M(x) = \sqrt{(x-\mu)^T \Sigma^{-1} (x-\mu)}$ or $d(\vec{x},\vec{y}) = \sqrt{\sum_{i=1}^{p} \frac{(x_i-y_i)^2}{\sigma_i^2}}$ if Σ^{-1} is identity and $\vec{x}, \vec{y}$ vectors

Then the decision boundary can be described as a minimum for a mahalanobis classifier ?? (demonstrated before):



→ the decision boundary is as such that the $D_M(x)$ is equal from one class to the other
⇒ for complex?

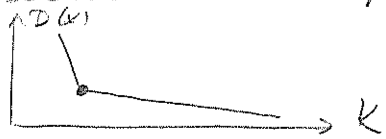
Originally, for the supervised approach, during the learning phase, manual delineation in the image to create sub-images and feature vector based on chosen features (x_{VAL} - gray value of the pixel, x_{VAR} - variance of gray values in a 5x5 domain, x_{LES} - nb of pixels having a gray value < x_{VAL} in the 5x5 domain, x_{DEF} - difference btw mean gray value of the 5x5 domain and x_{VAL}) has to be made. Nowadays, through deep learning, all of this is learned.

INSUPERVISED APPROACH

the method used is called K-means clustering and consists in defining a number of classes K and iteratively classify the pixels

- At first iteration:
  (0) initialization of the K-means chosen randomly = code vectors
  (1) computation of the distances between pixels and these means
  (2) attribution of a class to each pixel based on the lowest distance btw the pixel and the means.
  (3) compute the new means based on the pixels belonging to ~~a~~ same class.
- Other iteration: recompute steps (1) to (3) until no change is observed anymore in the mean computation.

See slides 16-21 for mathematical explanation
As it is not sure whether the algorithm will converge or not, it is necessary to recompute it several times. It is important to notice that if the goal is indeed to ↓ the euclidian distance, taking as many K as pixels (which would result in an euclidean distance of zero for every pixel) doesn't truly help as it is not the goal and moreover, there exist a value of K which doesn't improve D(x) so much anymore as the graph D(x) as a function of K follows an "elbow" and the goal is thus to find the value of K of the elbow:



Slide 22 for silhouette plot. A IMPRIMER
from the example slide 28-42 we can deduce that to solve classifications, feature (RGB, etc) are not sufficient, it is necessary to include notions of size and shape

CHAPITRE 6: MATHEMATICAL MORPHOLOGY

41 slides imprimés.

· laplacian general : $\frac{\partial^2 f}{\partial x^2}$ $\frac{\partial f}{\partial x}$ à gauche $\frac{\partial f}{\partial x}$ à droite

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

$\frac{\partial^2 f}{\partial y^2}$ $\frac{\partial f}{\partial y}$ à gauche $\frac{\partial f}{\partial y}$ à droite

· Wiener filter → voir Anyse

· low pass filter

$\frac{1}{3} \cdot \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} =$ convolution of a 3x1 mask $(1,1,1)^T$ on columns and a 1x3 mask $(1,1,1)$ on rows.

↳ separable filter.

· Prewitt :

$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix} \longrightarrow \text{gradient}$$

↳ low pass filter

· Canny edge:

$x_0 = -\frac{H_n'(x_0)}{H_G''(0)}$ and $H_n'(x_0)$ is a gaussian random quantity whose variance is the mean squared value of $H_n'(x_0)$: $E[H_n'(x_0)^2] = n_0^2 \int_{-w}^{w} f'^2(x)\,dx$

and $H_G''(0) = \left(\int_{-w}^{w} G'(-x) f'(x)\,dx\right)^2$

$$E(x_0^2) = \frac{n_0^2 \int_{-w}^{w} f'^2(x)\,dx}{\left(\int_{-w}^{w} G'(-x) f'(x)\,dx\right)^2} \rightarrow LOC = \frac{1}{E(x_0^2)}$$

· Isotropic properties of digital gradient/laplacian are preserved only for a limited nbr of rotational increments that depend on the mask used to approximate derivatives i.e:

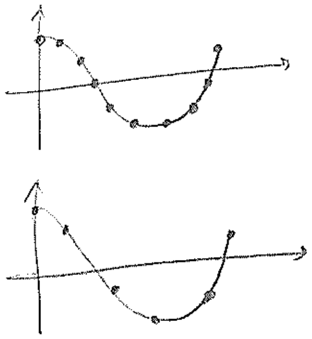
if the mask is $\begin{bmatrix} & 1 & \\ 1 & -4 & 1 \\ & 1 & \end{bmatrix}$ → only multiple of $\frac{\pi}{2}$ are invariant (direction, left and right taken into account)

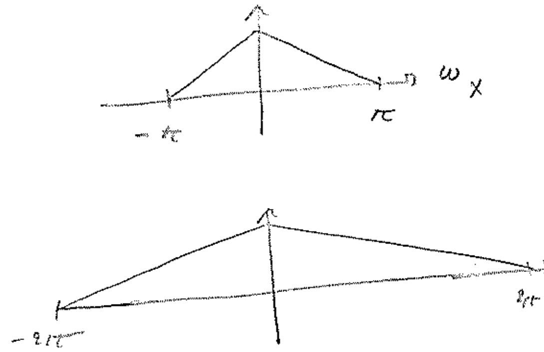
if the mask is $\begin{bmatrix} 1 & & 1 \\ & -4 & \\ 1 & & 1 \end{bmatrix}$ → only multiples of $\frac{\pi}{2}$ are invariant (direction diagonales taken into account)

if the mask is $\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$ → multiple of $\frac{\pi}{4}$ are invariant (derivatives diag + horizontal + vert taken into account)

· laplacian → fine details
gradient → edges.

Downsampling effect on BW.





$\omega_x = \frac{2\pi F}{F_x} = 2\pi F T_x$

$\omega_y = \frac{2\pi F}{F_y} = 2\pi F T_y = D\omega_x$

ω_y

so $F_y = \frac{F_x}{D}$