Edge detection

Javier González Jiménez

Reference Books:

- *Computer Vision: Algorithms and Applications*. Richard Szeliski. Springer. 2010.

http://szeliski.org/Book

Content

1. Introducción

Edges provide very valuable information about the object contours: useful for segmentation, object recognition, stereo, etc.

750x531x8bit x3
= 1.2Mbytes

1. Introduction

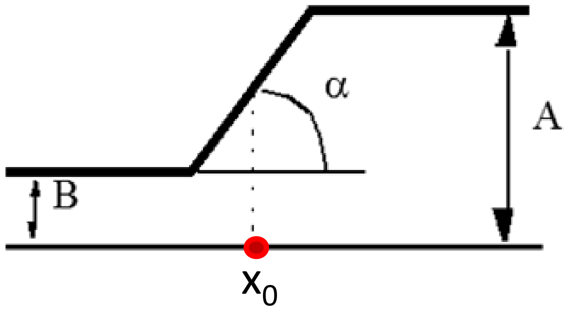
Most of the info of an image is embeded in its edges, and only requires a small proportion of the data

750x531x1bit
= 70 kbytes

1. Introduction

Edges: transitions between two image regions that have very different gray levels (intensities)

Unidimensional, continuous model of an ideal edge:

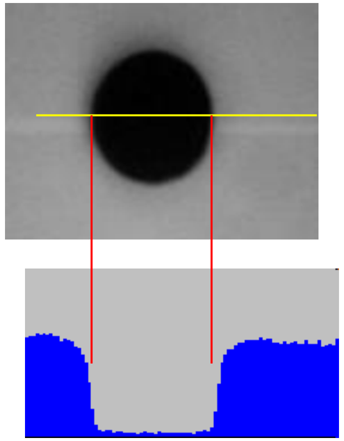


Parameters

- intensity increment H = A - B.
- angle of the slope: “α”.
- coordinate "x_0" of the midpoint

1. Introduction

In real images, edges do not follow this model exactly since:



- image are discrete
- are corrupted by noise

The nature of edges may be diverse:

- occlusion borders
- different orientation of surfaces
- different reflectance properties
- different texture
- illumination effects: shadows, reflections, etc.

1. Introduction

Three types of errors involved:

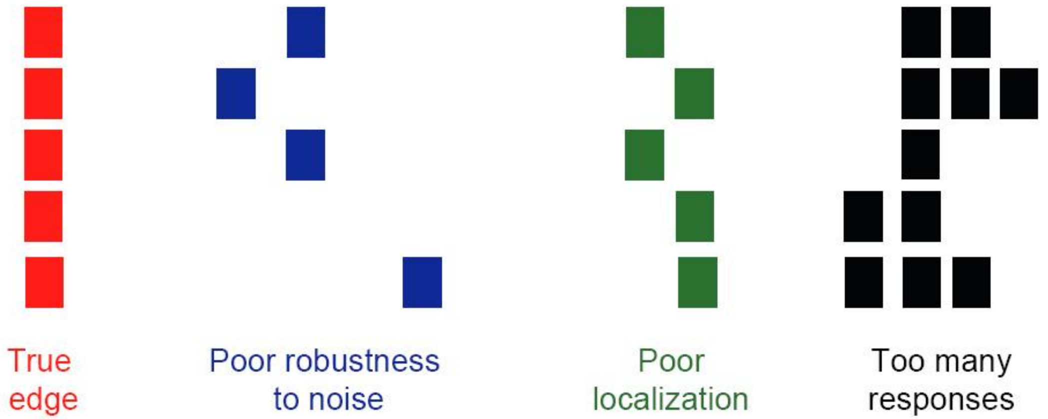
- **Detection error.**

  A good detector exhibits low ratio of false negative and false positive.

  – **False negatives**: Existing edges are not detected (go unnoticed for the operator)
  – **False positives**: Some of the edges detected are no real (induced by noise)

- **Localization error.** Good localizing when the response is at the real, exact position

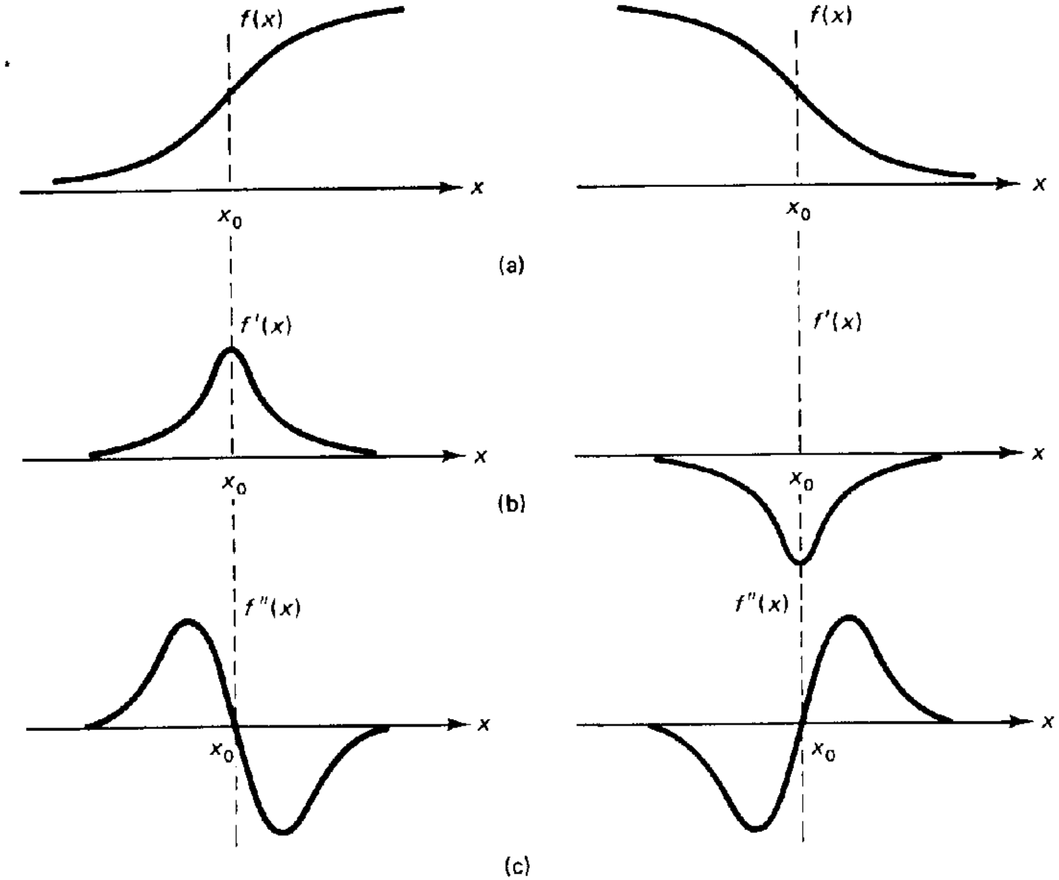
- **Multiple Response.** Several responses for the same edge



Source: L. Fei-Fei

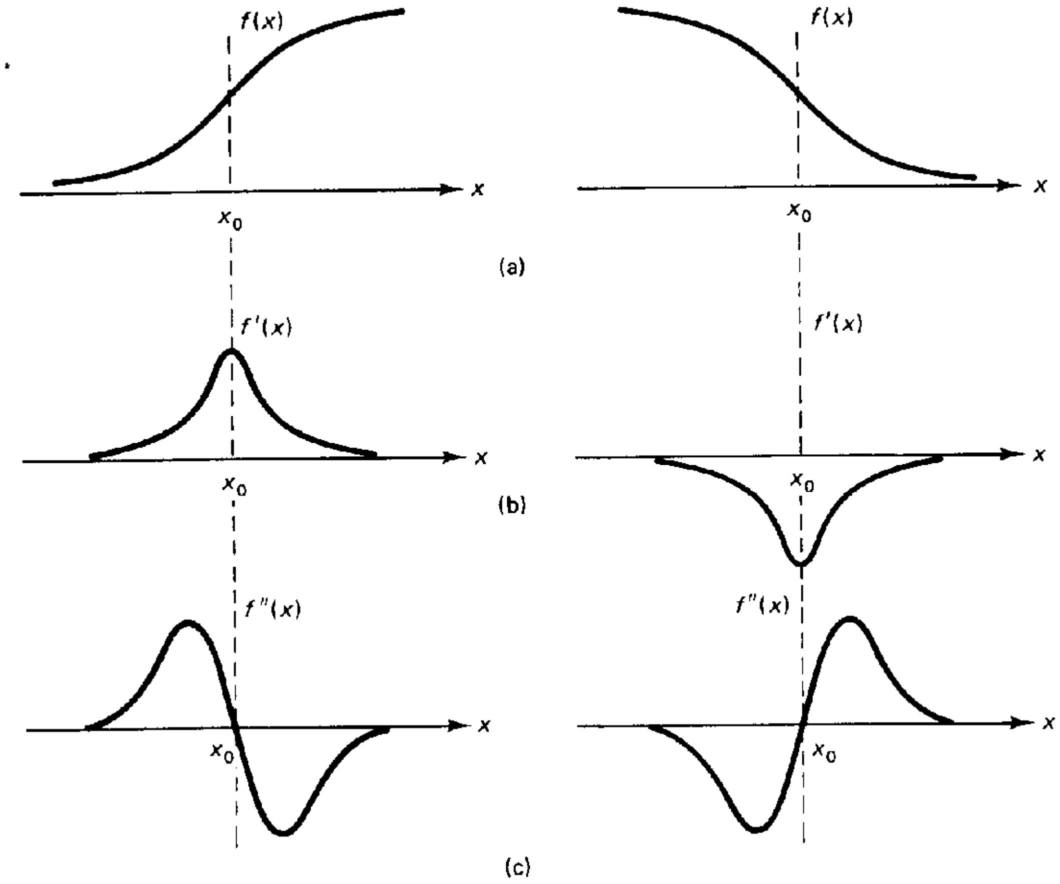
2. Operators based on first derivative (Gradient)

For a one-dimensional continuous function f(x)



2. Operators based on first derivative (Gradient)

Intensity transitions are detected through derivatives: For a 1D continuous function f(x)



Fastest variation at x_0

x_0 at the inflection point of f(x)

- x_0 at the maximum of f'(x)
- The importance of the change is given by $|f'(x_0)|$

x_0 at zero-crossing of f''(x)

For a **two-dimensional** continuous function f(x,y)

Its derivative is a vector (**gradient**), defined as:

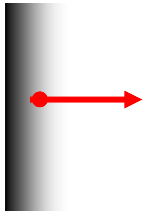
$$\nabla f(x,y) = \begin{bmatrix} \frac{\partial}{\partial x} f(x,y) \\ \frac{\partial}{\partial y} f(x,y) \end{bmatrix} = \begin{bmatrix} f_x(x,y) \\ f_y(x,y) \end{bmatrix}$$

- points at the direction of maximum (positive) variation of f(x,y):

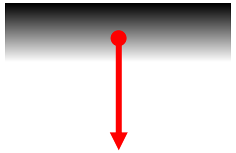
$$\alpha(x,y) = \operatorname{atan}\left(\frac{f_y(x,y)}{f_x(x,y)}\right)$$

- module proportional to the strength of this variation :

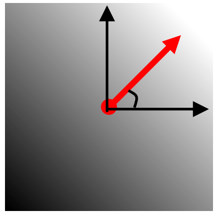
$$|\nabla f(x,y)| = \sqrt{(f_x(x,y))^2 + (f_y(x,y))^2} \approx |f_x(x,y)| + |f_y(x,y)|$$



$$\nabla f = \left[\frac{\partial f}{\partial x}, 0\right]$$



$$\nabla f = \left[0, \frac{\partial f}{\partial y}\right]$$



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

Discrete approximations of a gradient operator

Based on the differences between gray levels

For example:

- Backward difference of pixels along a row

$$f_x(x,y) \approx G_R(i,j) = \left[F(i,j) - F(i-1,j)\right]/T$$

0	0	0
0	1	-1
0	0	0

- Symmetric difference of pixels along a row

$$f_x(x,y) \approx G_F(i,j) = \left[F(i+1,j) - F(i-1,j)\right]/2T$$

0	0	0
1	0	-1
0	0	0

- Implemented through the convolution of the image with templates H_R (row) y H_C (column)

$$G_R(i,j) = F(i,j) \otimes H_R(i,j) \qquad G_C(i,j) = F(i,j) \otimes H_C(i,j)$$

Discrete approximations of a gradient operator

Roberts

$$\begin{array}{|c|c|c|}\hline 0 & 0 & 0 \\ \hline 0 & 0 & 1 \\ \hline 0 & -1 & 0 \\ \hline \end{array} \qquad \begin{array}{|c|c|c|}\hline -1 & 0 & 0 \\ \hline 0 & 1 & 0 \\ \hline 0 & 0 & 0 \\ \hline \end{array}$$

Prewitt

$$\frac{1}{3}\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \frac{1}{3}\begin{array}{|c|c|c|}\hline -1 & -1 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$$

Sobel

$$\frac{1}{4}\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \frac{1}{4}\begin{array}{|c|c|c|}\hline -1 & -2 & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & 2 & 1 \\ \hline \end{array}$$

Frei-Chen

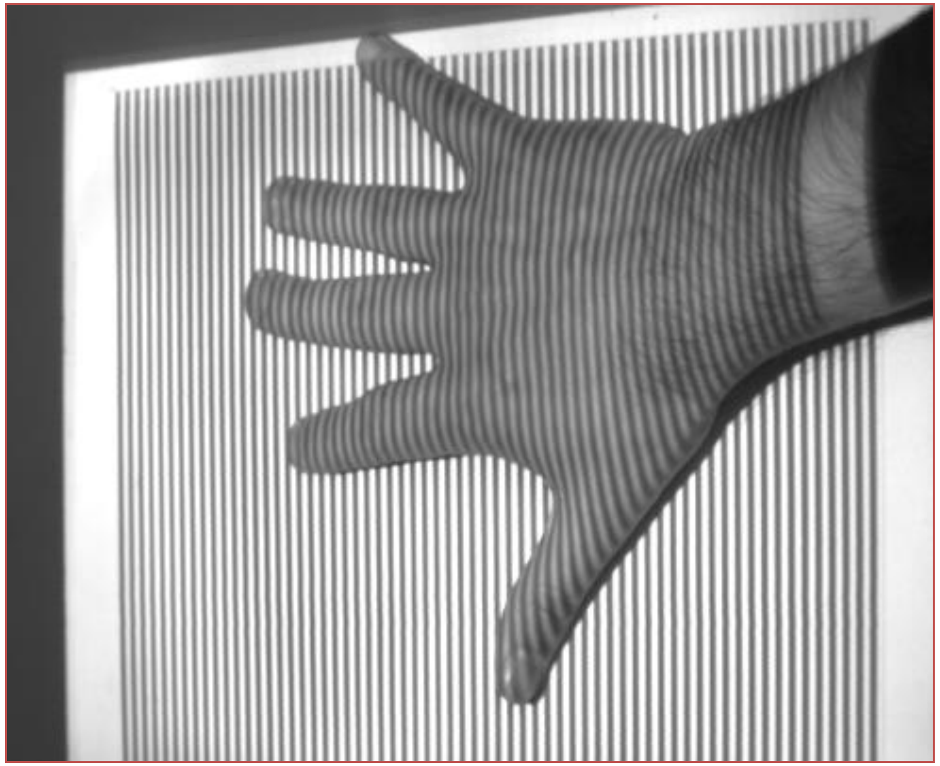
$$\frac{1}{2+\sqrt{2}}\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline \sqrt{2} & 0 & -\sqrt{2} \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \frac{1}{2+\sqrt{2}}\begin{array}{|c|c|c|}\hline -1 & -\sqrt{2} & -1 \\ \hline 0 & 1 & 0 \\ \hline 1 & \sqrt{2} & 1 \\ \hline \end{array}$$

$$\frac{1}{2+K}\begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline K & 0 & -K \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \frac{1}{2+K}\begin{array}{|c|c|c|}\hline -1 & -K & -1 \\ \hline 0 & 0 & 0 \\ \hline 1 & K & 1 \\ \hline \end{array}$$

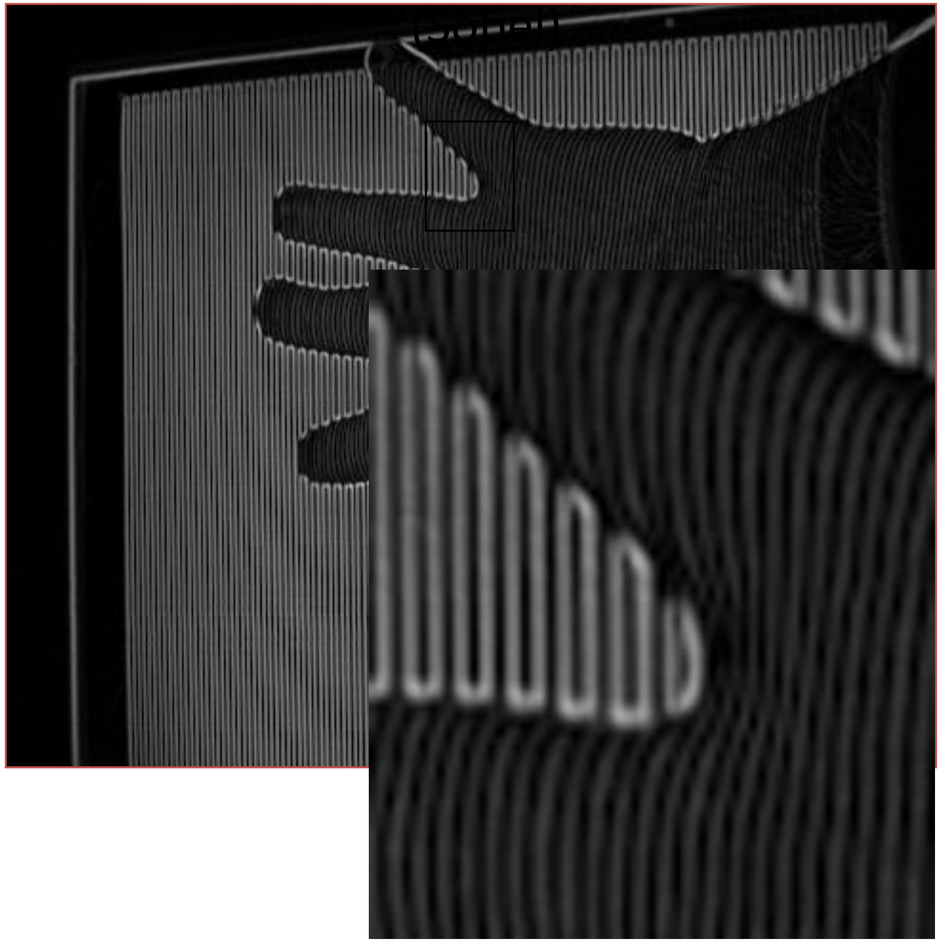
$H_F(i,j)$ $\qquad$ $H_C(i,j)$

Example

Original image



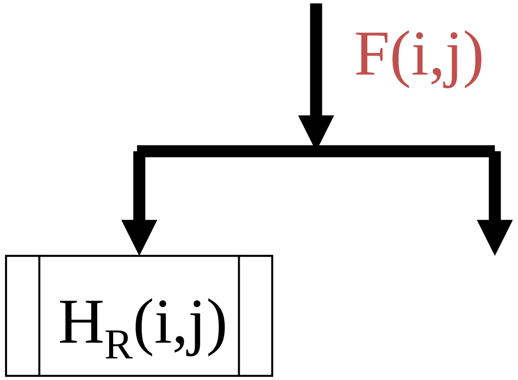
(Module) Gradient image



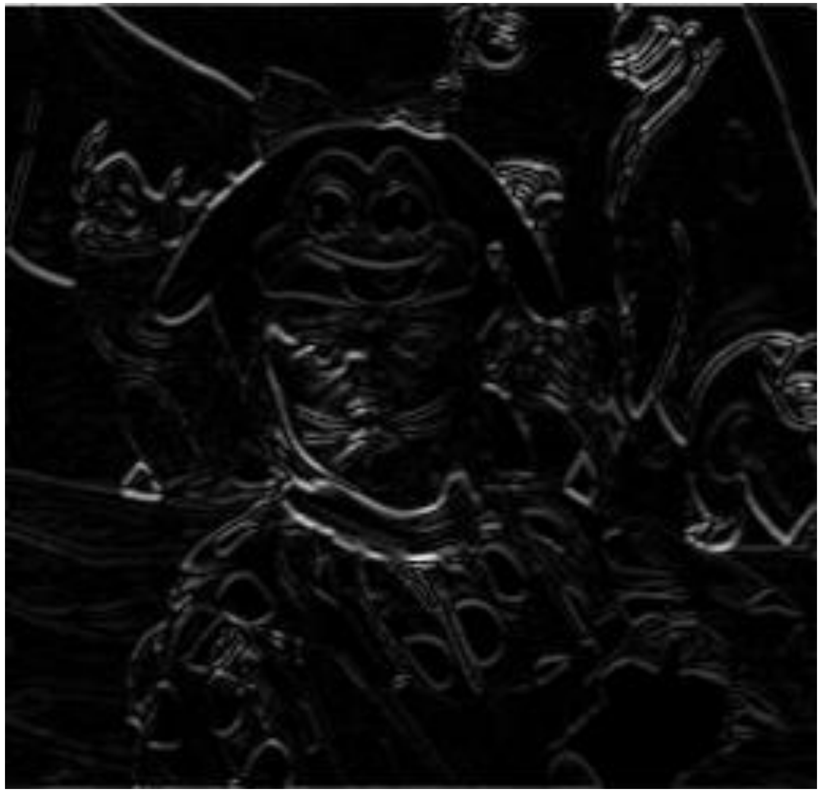
Implementation

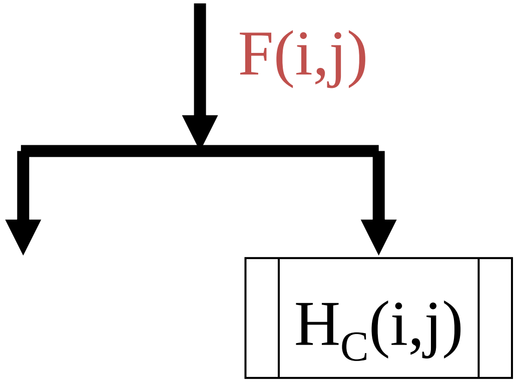
F(i,j)

Implementation

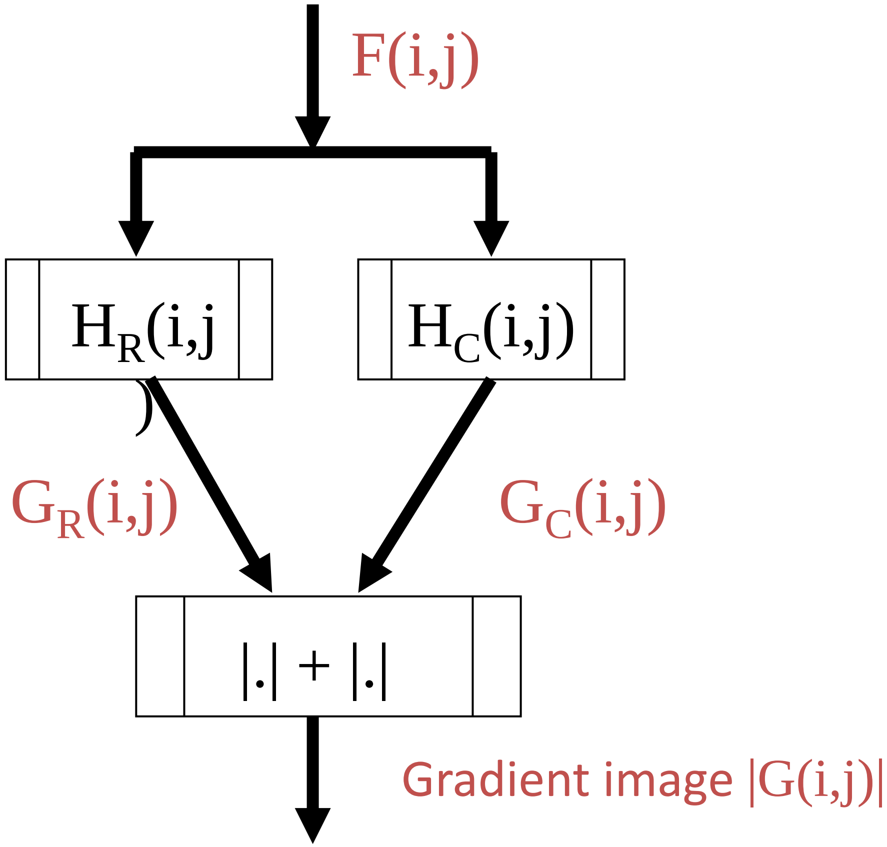


Implementation



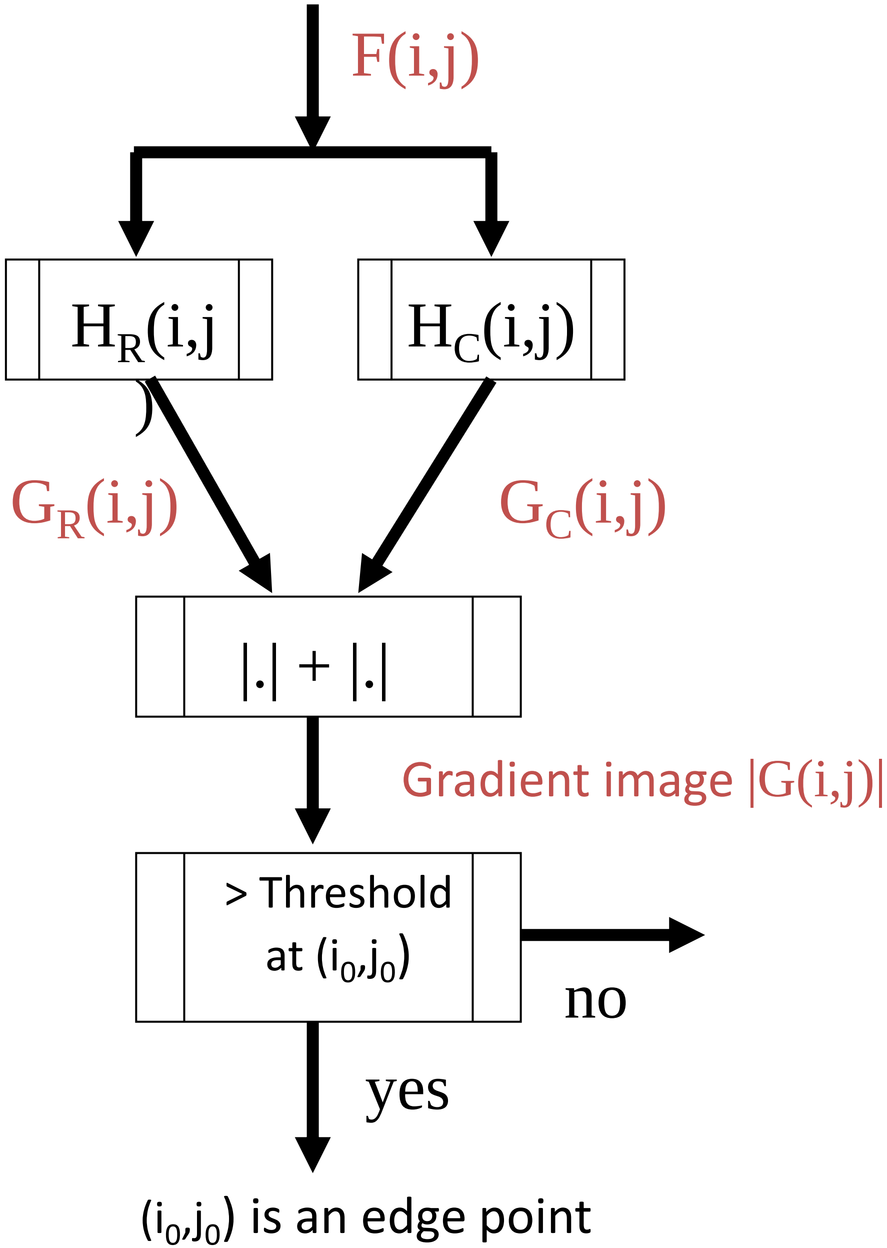


Implementation



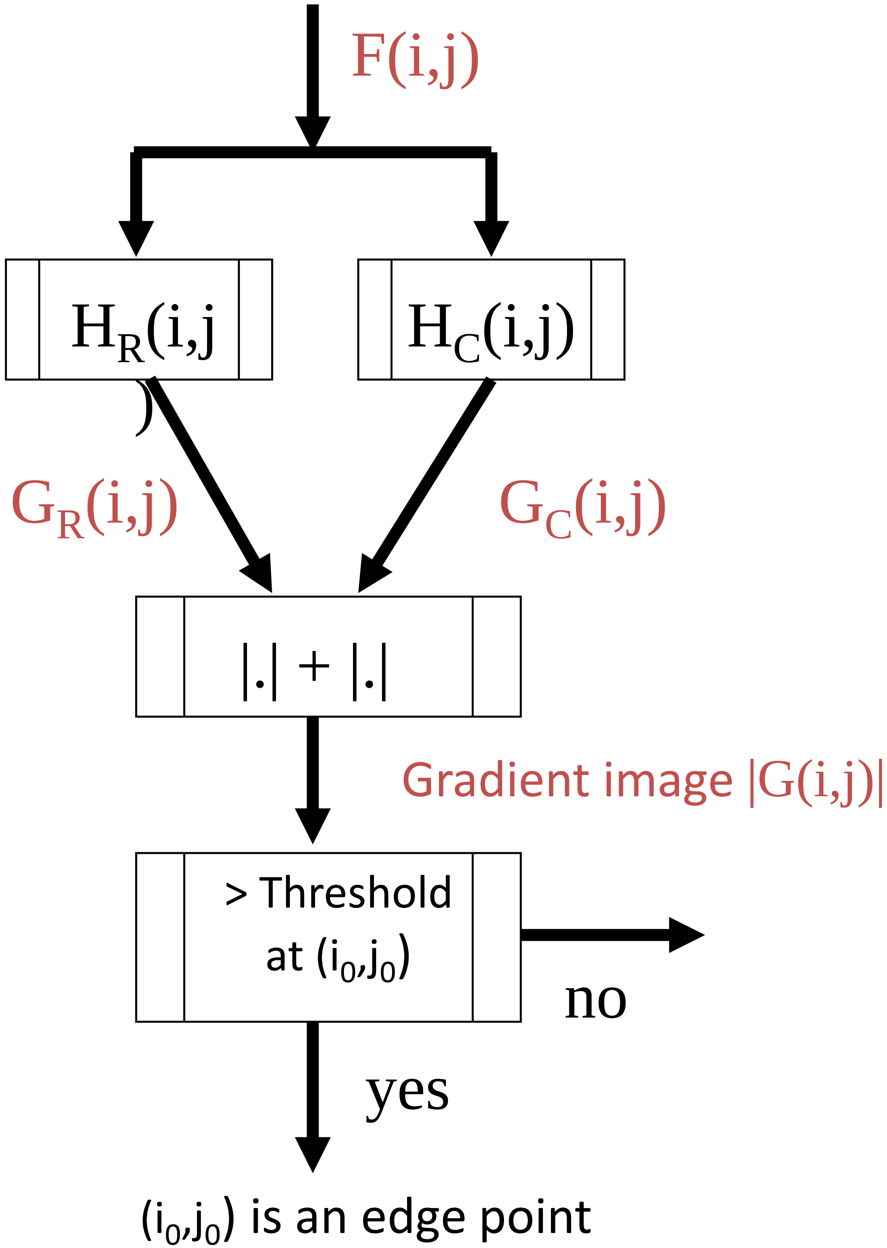
Implementation

Threshold at 120



Implementation

Threshold at 80



Size of the template …

Affects the quality of detection and localization

Small template → - more precise localization → good localization
- more affected by noise → likely produces false positive

Large template → - less precise localization
- more robust to noise → good detector
- higher computational cost O(NxN)

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

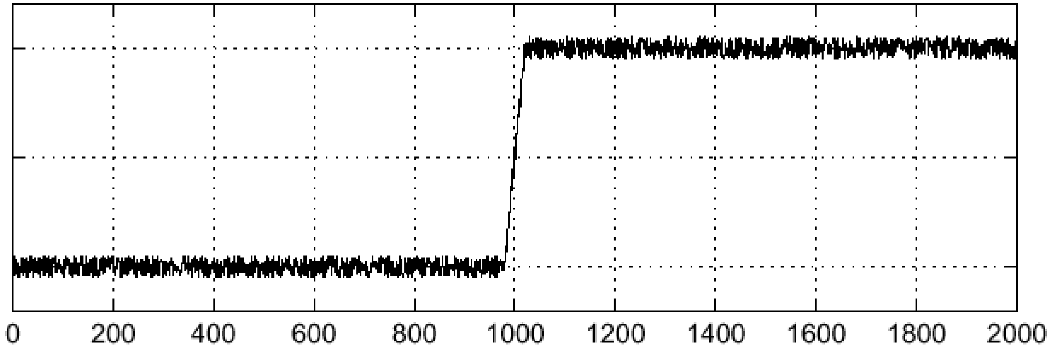
Sobel 3x3

1	2	0	-2	-1
2	3	0	-3	-2
3	5	0	-5	-3
2	3	0	-3	-2
1	2	0	-2	-1

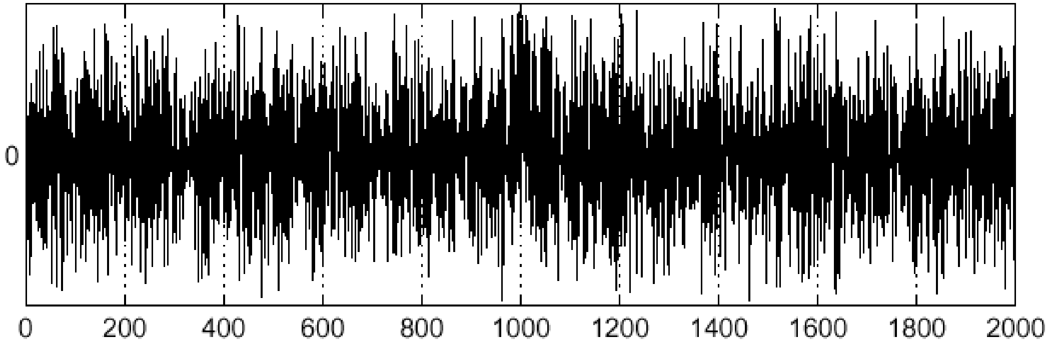
Sobel 5x5

Noise does matter

$f(x)$



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



Where is the edge?

The response of the derivative to noise is as bigger as the step itself

Source: S. Seitz

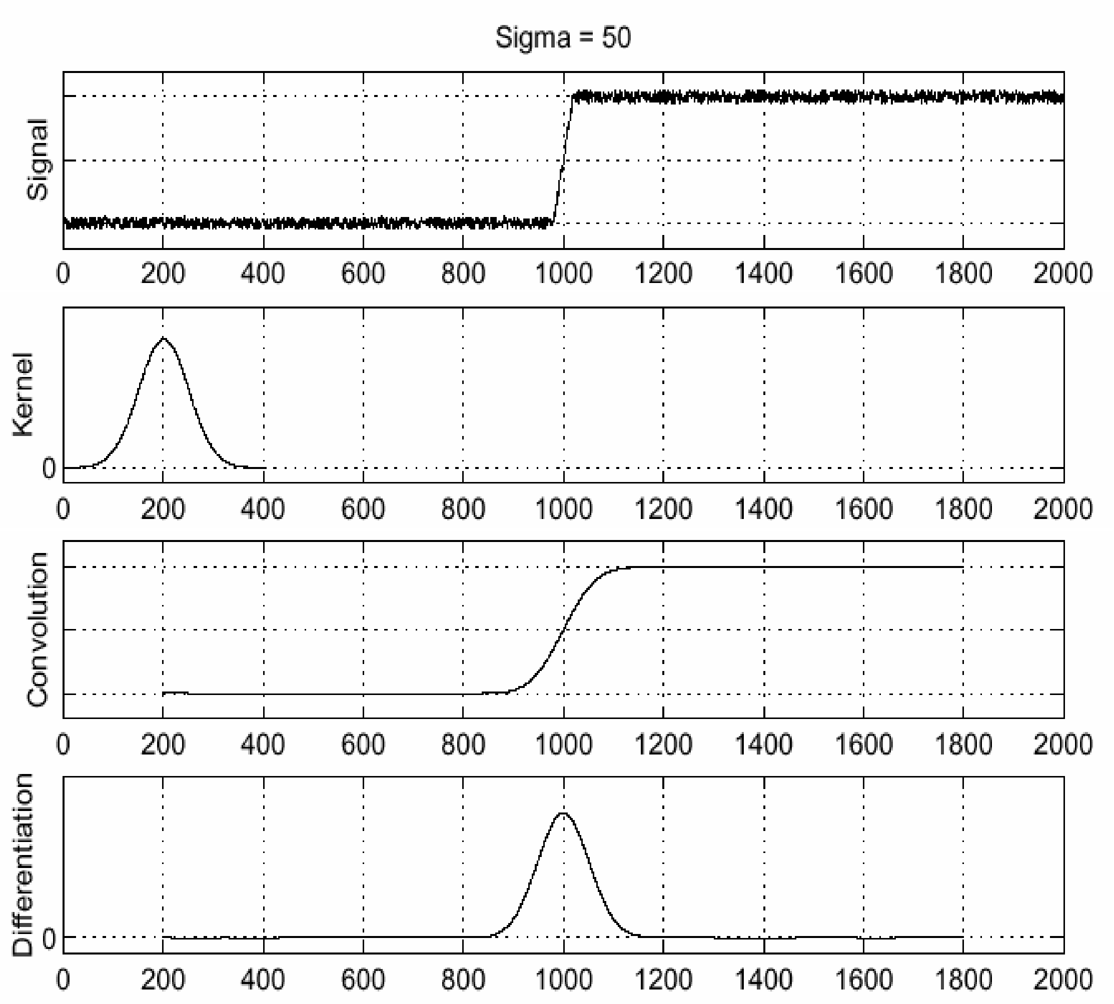
Solution: smoothing with a Gaussian

f

g

$f * g$

$\frac{d}{dx}(f * g)$



Noisy signal to process (to detect edges on it)

Gaussian kernel

Smoothed signal (one convolution)

Result of edge detection on the smoothed signal

To localize edges we have to detect maxima of $\frac{d}{dx}(f * g)$

Source: S. Seitz

Smoothing and Gradient combined

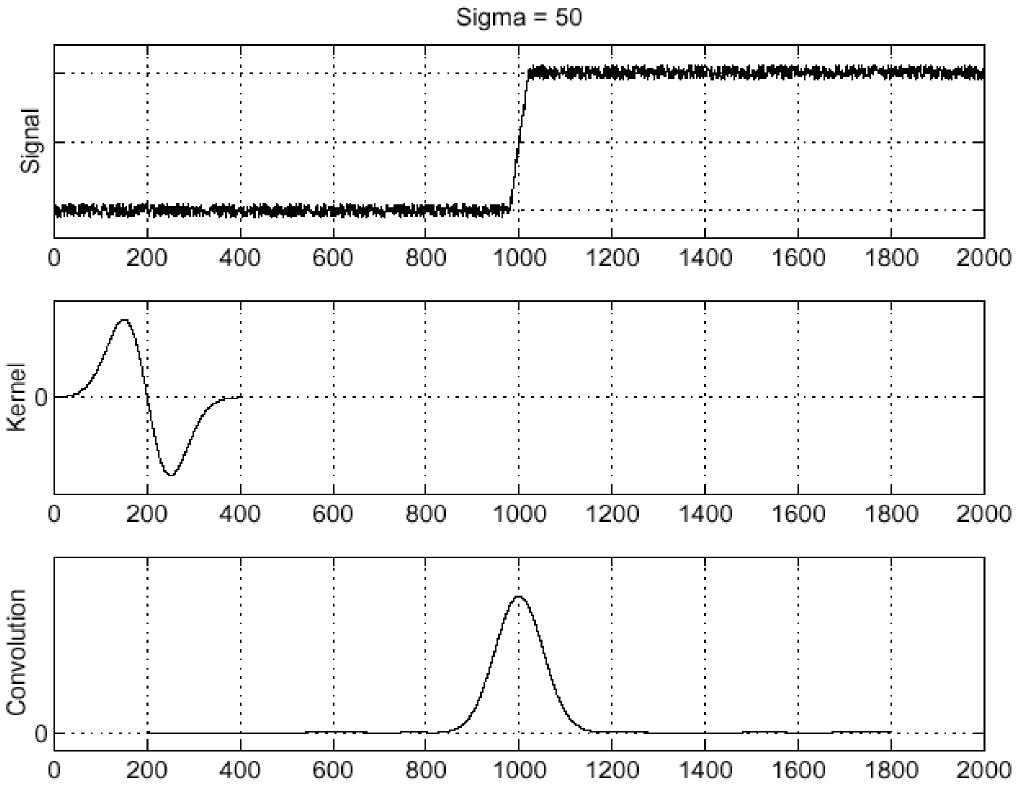
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$ Convolution property

More efficient if we convolve the image with the derivative of the gaussian (we save one operation)

f

$\frac{d}{dx} g$

$f * \frac{d}{dx} g$



This is precomputed

One convolution operation

Smoothing and Gradient combined (2D). The DroG operator

DroG (“Derivative of Gaussian”):

- blends smoothing and gradient: $\nabla\underbrace{[f(x,y) \otimes g_\sigma(x,y)]}_{\text{Smoothed image}}$
- the standard deviation σ controls the degree of smoothness

$$g_\sigma(x,y) = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{(x^2+y^2)}{\sigma^2}} = \frac{1}{2\pi\sigma^2} e^{-\frac{1}{2}\frac{r^2}{\sigma^2}} = g_\sigma(r^2)$$

Radial function.
Not depend on x,y

$$\nabla[f(x,y) \otimes g_\sigma(x,y)] = f(x,y) \otimes \nabla[g_\sigma(x,y)] = f(x,y) \otimes \mathrm{DroG}(x,y)$$

separability

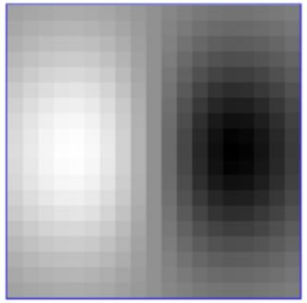
$g(x)' = -xg(x)/\sigma^2$

$$\mathrm{DroG}(x,y) = \nabla[g_\sigma(x,y)] = \begin{bmatrix} \frac{\partial}{\partial x}[g_\sigma(x)g_\sigma(y)] \\ \frac{\partial}{\partial y}[g_\sigma(x)g_\sigma(y)] \end{bmatrix} = \begin{bmatrix} \frac{-xg_\sigma(x)g_\sigma(y)}{\sigma^2} \\ \frac{-yg_\sigma(x)g_\sigma(y)}{\sigma^2} \end{bmatrix}$$

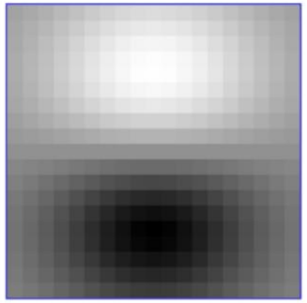
DroG Operator

Implementation:

- to build a discrete DroG operator, “x, y” are taken in $\mathbb{N}x\mathbb{N}$
- the DroG template is created just once

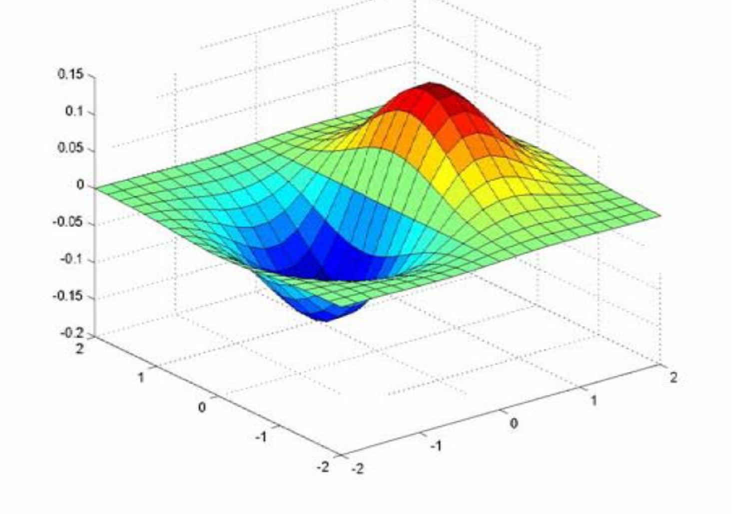


DroG along x



DroG along y

$$\mathrm{DroG}(x,y) = \nabla[g_\sigma(x,y)] = \begin{bmatrix} \dfrac{\partial}{\partial x}[g_\sigma(x)g_\sigma(y)] \\ \dfrac{\partial}{\partial y}[g_\sigma(x)g_\sigma(y)] \end{bmatrix} = \begin{bmatrix} \dfrac{-xg_\sigma(x)g_\sigma(y)}{\sigma^2} \\ \dfrac{-yg_\sigma(x)g_\sigma(y)}{\sigma^2} \end{bmatrix}$$




```
% Separabilidad del operador DroG
% La mascara de derivada en x es: -x*g(x)*g(y)/sigma^2
tamano=5; %Tamaño de la mascara 2xtamano+1
x=[-tamano:tamano];
g_x = fspecial('gaussian',[1,length(x)],2)
g_y = fspecial('gaussian',[length(x),1],2)
drog_x=g_y*(-x.*g_x) %Propiedad de separabilidad
surf(drog_x)
```

DroG Operator

Original image

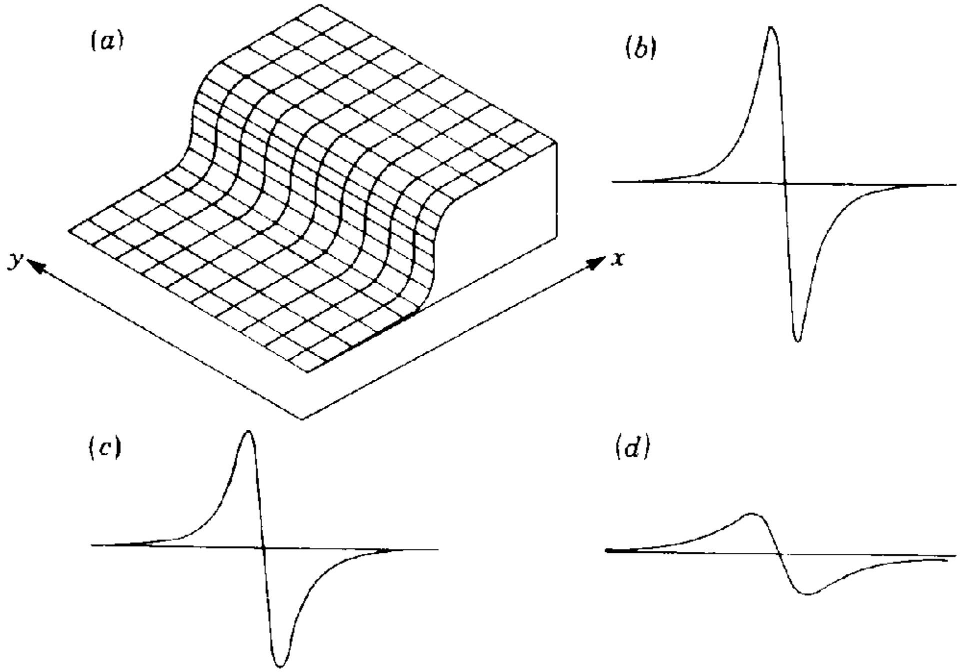
DroG result
(small σ)

DroG result
(medium σ)

DroG result
(big σ)

3. Operators based on second derivative (Laplacian)

- A second derivative yields a zero-crossing at points where the gradient presents a maximum
- Depending on the edge orientation, this zero-crossing may go almost unnoticed



Response depends on the derivative orientation

Solution:

Combine (second) derivatives in perpendicular directions → **Laplacian operator**

Laplacian

- linear operator
- invariant to image orientation
- precise edge localization

Definition: $\nabla^2 f(i, j) = \frac{\partial^2}{\partial x^2} f(i, j) + \frac{\partial^2}{\partial y^2} f(i, j)$ It's not a vector, but a scalar!

Laplacian is the **trace of the *Hessian matrix***, which fully characterizes the second derivative of a function.

$$\Delta f(x, y) = \begin{bmatrix} \frac{\partial f^2}{\partial x^2} & \frac{\partial f}{\partial x}\frac{\partial f}{\partial y} \\ \frac{\partial f}{\partial y}\frac{\partial f}{\partial x} & \frac{\partial f^2}{\partial y^2} \end{bmatrix}$$

Hessian matrix

Implementation:

First derivative $\frac{\partial f(x, y)}{\partial x} = f_x(x, y) \approx G_F(i, j) = f(i+1, j) - f(i, j)$

$$\frac{\partial f^2}{\partial x^2} = f_{xx}(x, y) \approx G_F(i, j) - G_F(i-1, j) = f(i+1, j) - 2f(i, j) + f(i-1, j)$$

0	0	0
1	-2	1
0	0	0

$$\frac{\partial f^2}{\partial y^2} = f_{yy}(x, y) \approx G_C(i, j) - G_C(i-1, j) = f(i, j+1) - 2f(i, j) + f(i, j-1)$$

0	1	0
0	-2	0
0	1	0

Laplacian

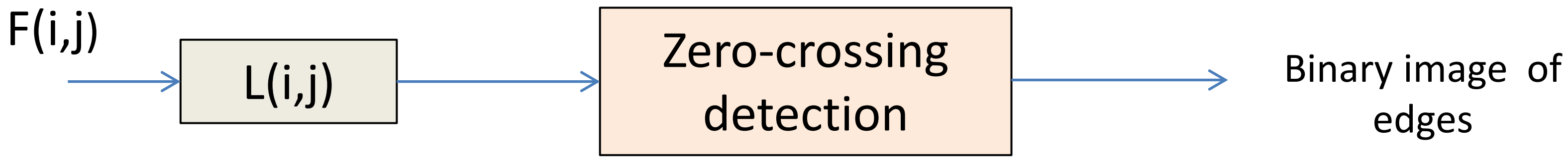
Implemented as a convolution: $L[F(i,j)] = F(i,j) \otimes L(i,j)$

$$\underset{G}{\begin{bmatrix} 0 & 0 & 0 \\ 1 & -2 & 1 \\ 0 & 0 & 0 \end{bmatrix}} + \underset{H}{\begin{bmatrix} 0 & 1 & 0 \\ 0 & -2 & 0 \\ 0 & 1 & 0 \end{bmatrix}} = \underset{G+H=L}{\begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}}$$

Distributive property of Convolution

$$\underbrace{f * (g + h)}_{\text{One Convolution}} = \underbrace{(f * g) + (f * h)}_{\text{Two Convolutions}}$$

An algorithm for zero-crossing detection needed!



Properties

- zero-crossing produced is near a closed contour
- provides edges of 1-pixel width
- poor edge detection because of its sensitivity to noise

Smoothing and Laplacian combined. LoG operator

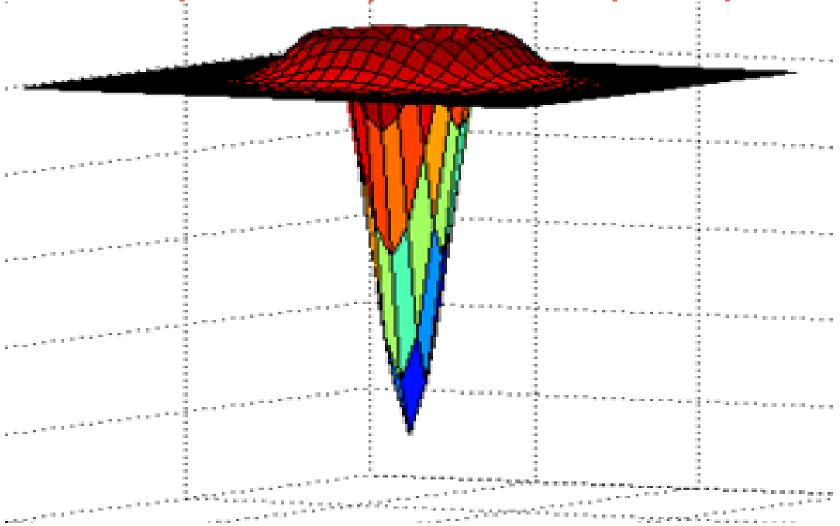
- Sensible to noise → smoothing needed

- The LoG operator: smoothing and Laplacian (or viceversa, it's conmutative)

$$\nabla^2[f(x,y)\otimes g_\sigma(x,y)] = f(x,y)\otimes\nabla^2[g_\sigma(x,y)] = f(x,y)\otimes \mathrm{LoG}_\sigma(x,y)$$

$$\mathrm{LoG}_\sigma(x,y) = \frac{1}{\pi\sigma^4}\left[\frac{x^2+y^2}{2\sigma^2}-1\right]e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{1}{\pi\sigma^4}\left[\frac{r^2}{2\sigma^2}-1\right]e^{-\frac{r^2}{2\sigma^2}} = \mathrm{LoG}_\sigma(r^2)$$

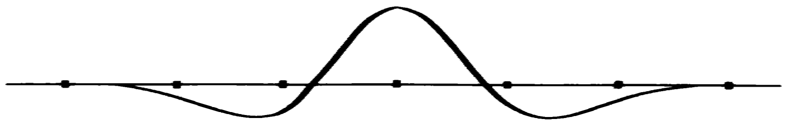
Radial symmetry → isotropic operator



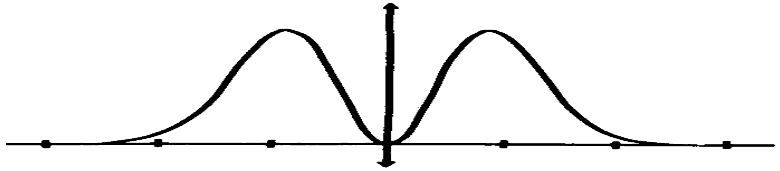
applies equally well in all directions in an image

LoG Operator

- High-pass filter (Laplacian) + Low-pass filter (Gaussiana) = Band-pass filter (width controlled by σ)



LoG in the spatial domain



LoG in the frequency domain

- LoG is not separable but can be implemented as **DoG** ("Difference of Gaussians"): O(N²) → O(4N)

$$DoG_{\sigma_1\sigma_2}(x,y) = g_{\sigma_1}(x,y) - g_{\sigma_2}(x,y) = g_{\sigma_1}(x)g_{\sigma_1}(y) - g_{\sigma_2}(x)g_{\sigma_2}(y)$$

Sum of Separable Operators

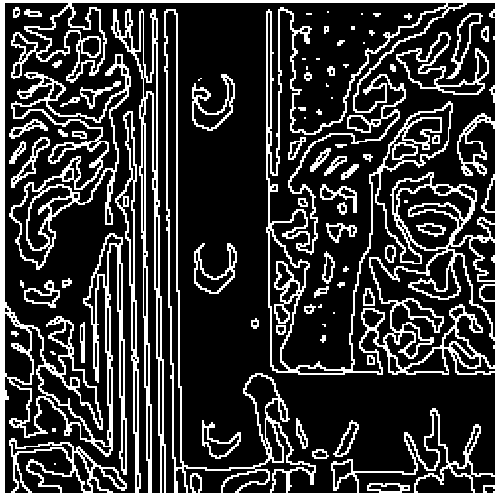
The ratio $\sigma_1/\sigma_2=1.6$ gives the best approximation of LoG

LoG Operator

Drawbacks:

- costly computationally
- not edge orientation is provide
- the output contains negative and non-integer values, so for display purposes the image is normalized to the range 0 – 255
- zero-crossing needed
- tends to round object corners (depending on σ)





LoG Operator

Implementation:

- the image is convolved with two Gaussians (of different σ) and the subtracted to each other (O(2N+2N), N being the width of the kernel)
- followed by a zero crossing algorithm to detect edges

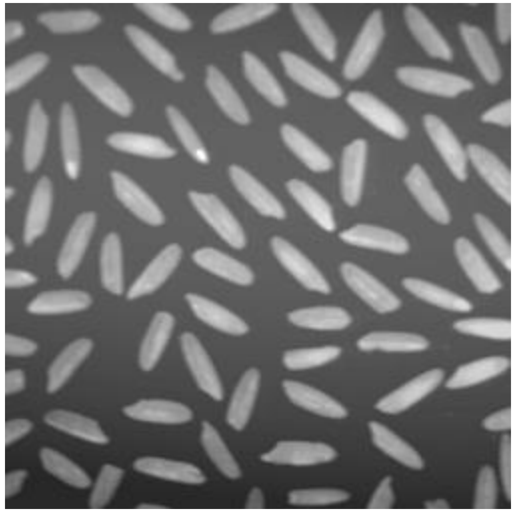
A simple zero-crossing algorithm

Select a small positive number (threshold) t

A pixel is labelled as edge if in the LoG image

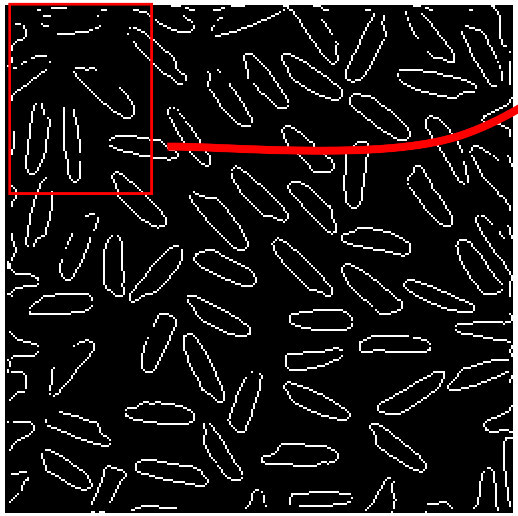
- its value is smaller than $-t$ and at least one of its neighbours is bigger that t, or
- Its value is bigger than t and at least one of its neighbours is smaller that $-t$.

LoG Operator



MATLAB

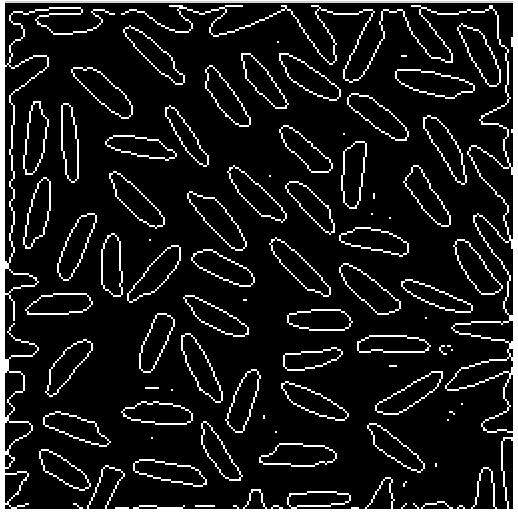
```
I = imread('rice.tif');
%BW=edge(I,'log',thresh,sigma)
log = edge(I,'log',0.001,2);
figure, imshow(log)
```



Open contours

σ=2
th=0.007

σ=2
th=0.001

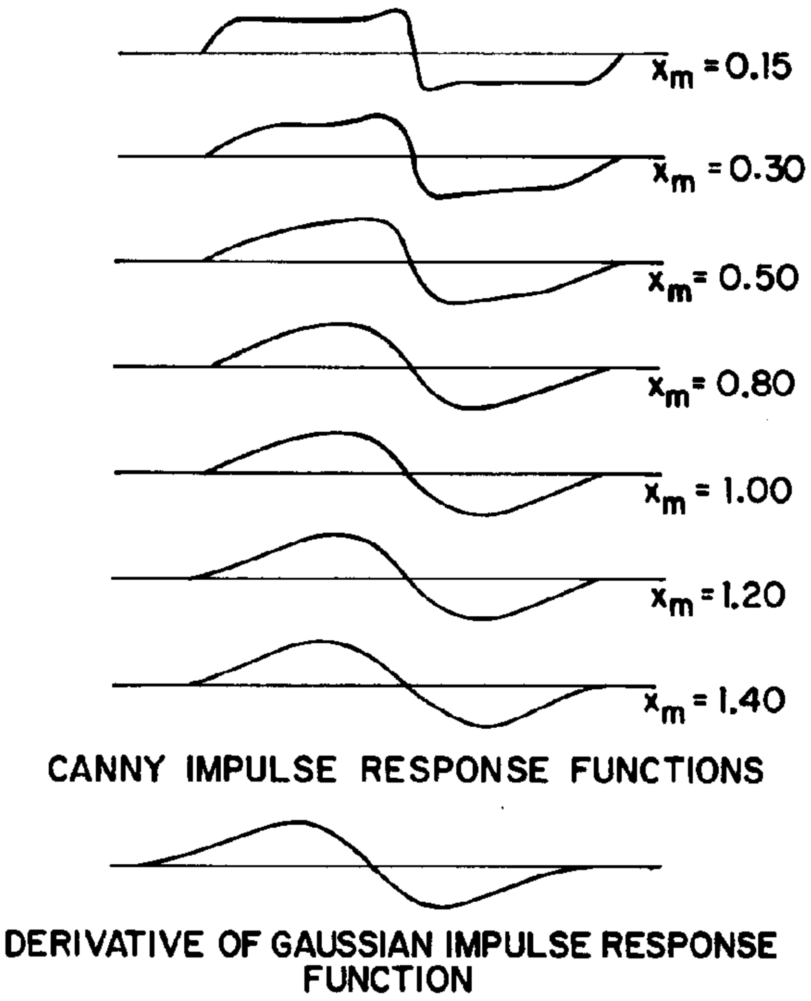


It's an interesting method for **region segmentation** since closed contours provide meaningful regions (of similar pixel intensities)

4. Canny algorithm

- Based on the edge optimal operator for a step 1D signal affected with Gaussian noise
- Optimal meeting 3 criteria
  => good detector
  => good localizator
  => single response

The result is a filter very similar to the DroG in 1D



4. Canny algorithm

Procedure for images:

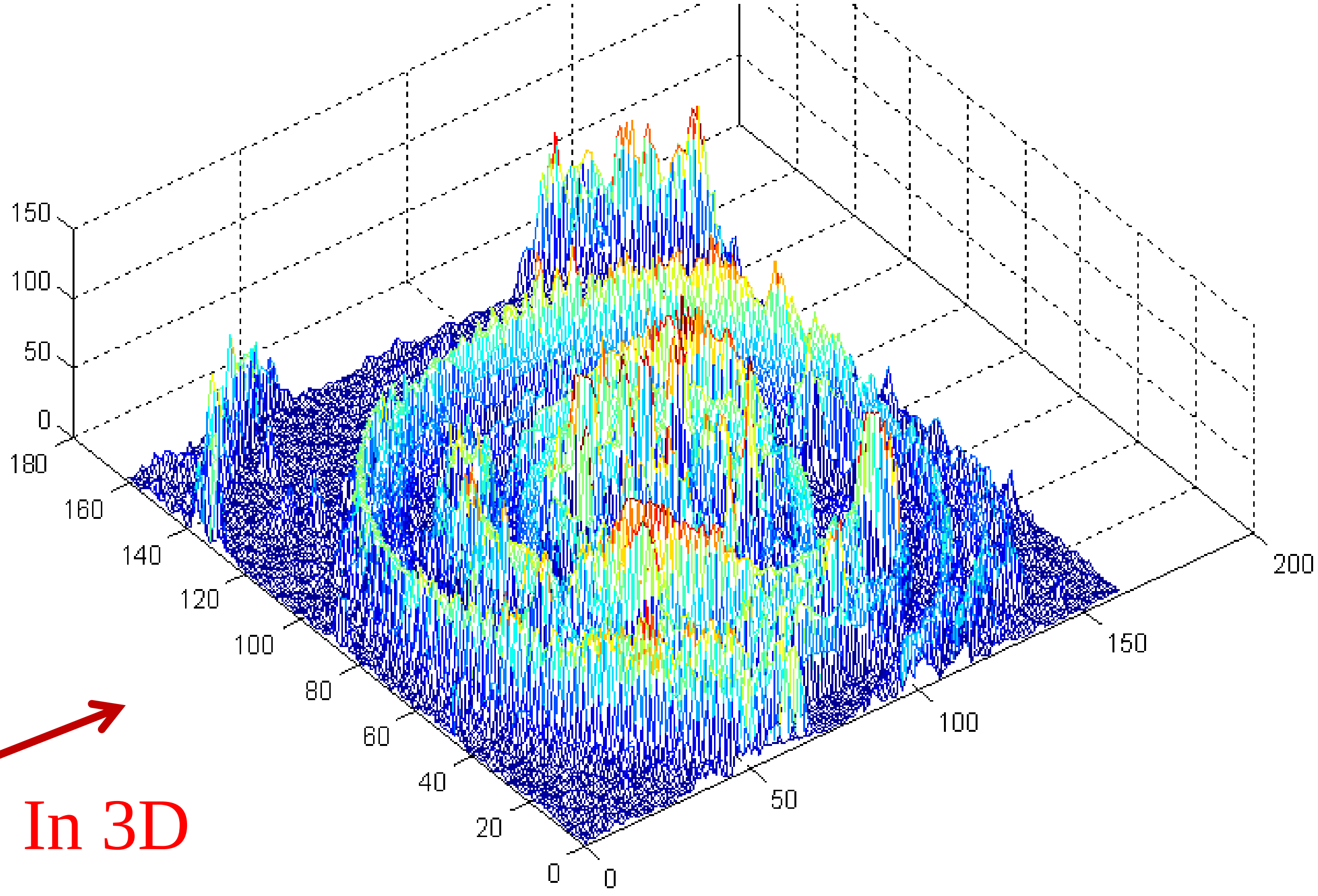
1. Compute the magnitude G[i,j] and the orientation θ[i,j] of the gradient by applying the DroG operator
2. In the direction θ[i,j] of the gradient vector, apply nonmaxima suppression
3. Threshold the gradient image with a hysteresis to remove less strong edgels (**ed**g**e**-pix**el**) while keeping the contour close

4. Canny algorithm

DroG

DroG

In 3D

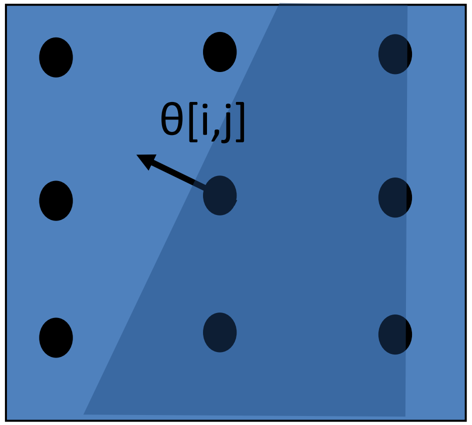


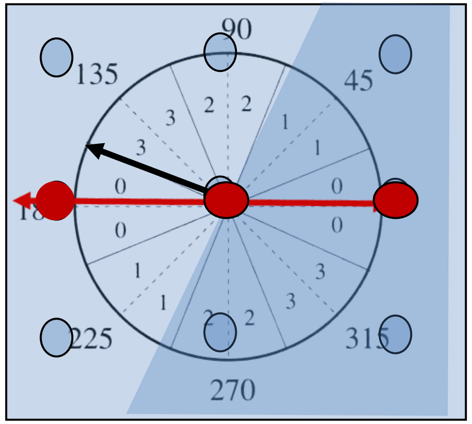
We want to follow the summits of these mountains

4. Canny algorithm

Non-maxima suppression

- We consider only the 4 main directions (angular sectors): [0,45], [45,90], [90,135], [135,180].
- The gradient angle θ[i,j] is approximated to the sector where it lays
- We check G[i,j] at the three points along the selected direction (in red below) and pick the one where G[i,j] is maximum.





We pick the pixel with the biggest where G[i,j]

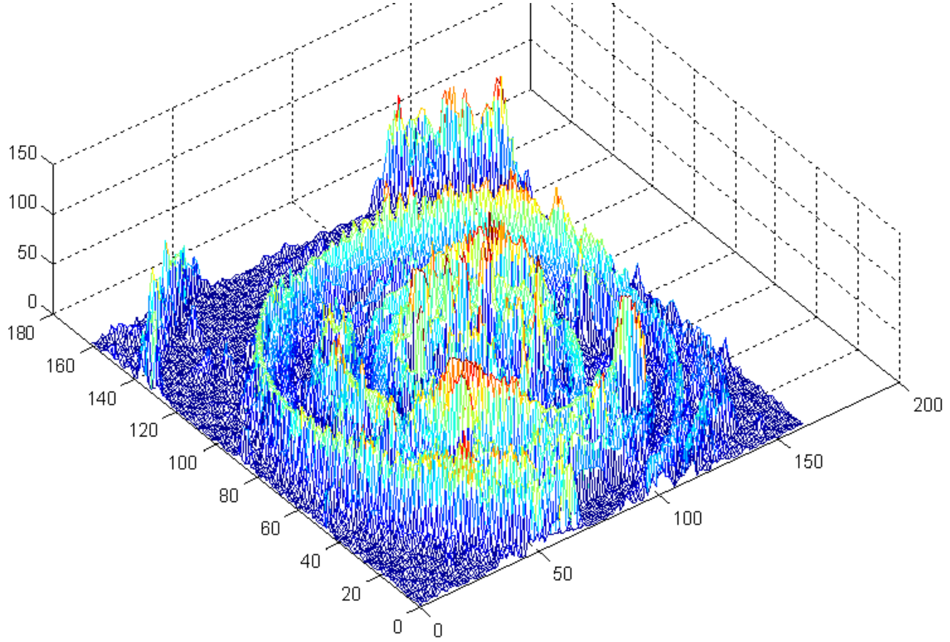
We get the maximum and one single response at each edge

4. Canny algorithm

Why hysteresis: We want to keep the strongest edgels but prevent the disconnection of the contour

With a **single threshold**:

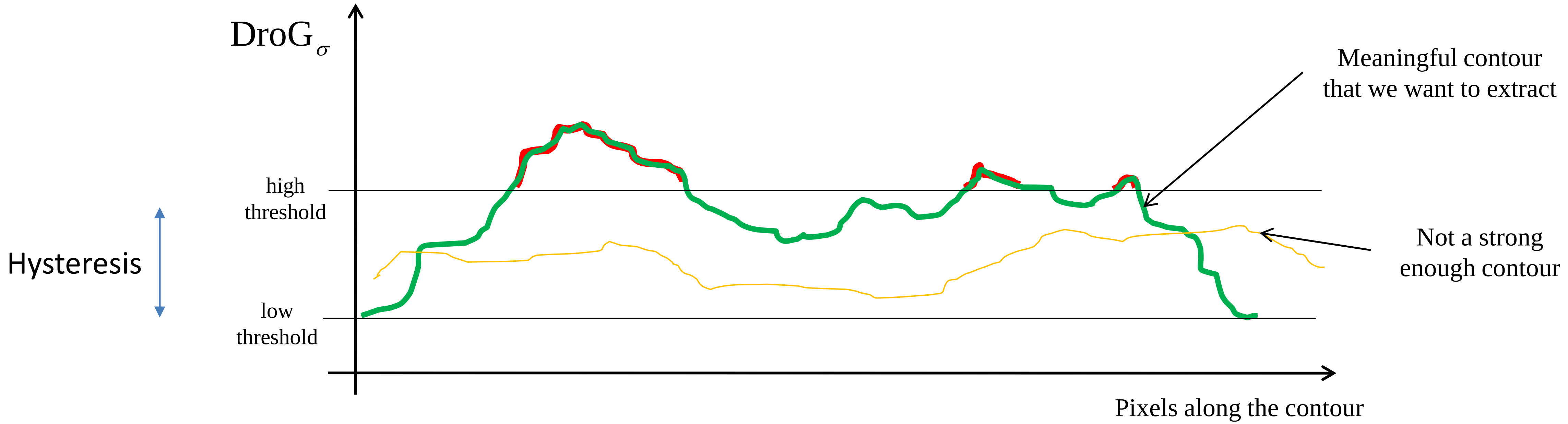
- **Large threshold** → few strong edgels, but disconnected contour
- **Small threshold** → too many edgels (including weak ones)



4. Canny algorithm

Hysteresis

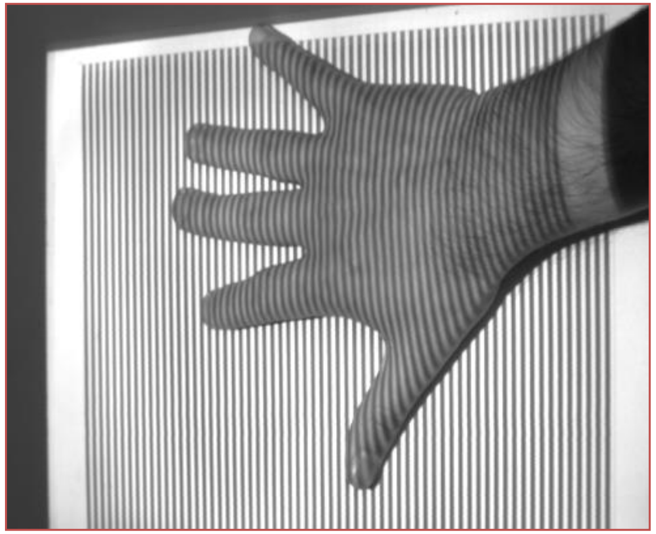
- A high threshold to incorporate strong edgels to begin a contour
- A lower threshold to cut-off a contour



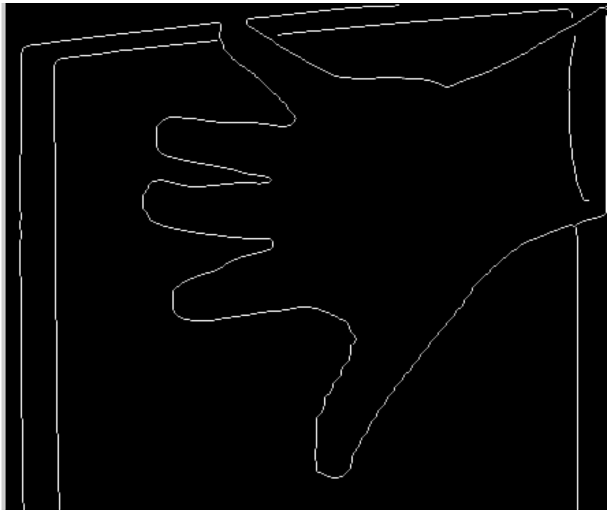
4. Canny algorithm

- The algorithm can be repeated with different level of smoothing (changing the sigma of the DroG operator).
- Different sigma produces edges at different spatial scales

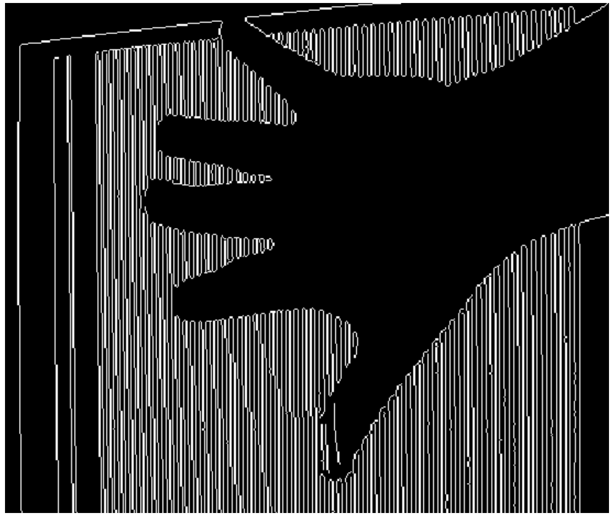
Original Image



$\sigma=2$



$\sigma=1$



```
I=imread('mano.bmp');
imshow(I)
canny_s1=edge(I,'canny',0.3,1);%BW = edge(I,'canny',thresh,sigma)
figure,imshow(canny_s1)
canny_s2=edge(I,'canny',0.3,2);
figure, imshow(canny_s2)
```

Summary

- Edges are **transitions** between image regions that have different intensities
- An **color image** has 3 edge images! (We can combine them)
- Edges are detected through **derivatives** (smoothing required!):
  - 1st derivative: DroG operator
  - 2nd derivative: LoG/DoG + Zero crossing
- **Canny operator**:
  - Computational expensive (yet in real time!)
  - Very good detection/localization + control of the multiple response
- In all these options, the **Standard Deviation (σ) of the Gaussian** plays an important role → Size of the operator (trade-off between localization and detection)