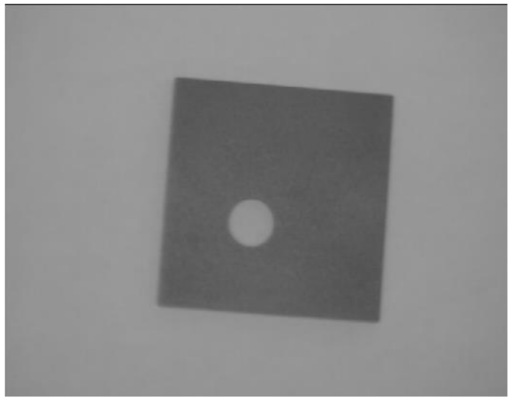
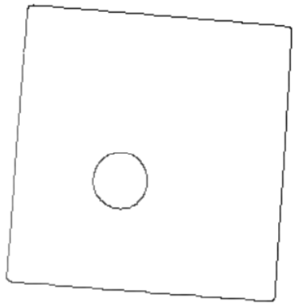
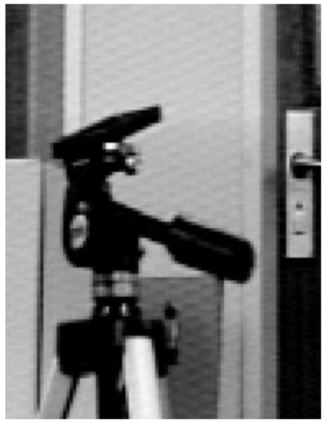
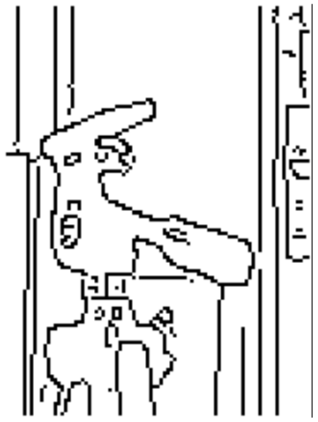
University of Bologna

Edge Detection

Luigi Di Stefano (luigi.distefano@unibo.it)

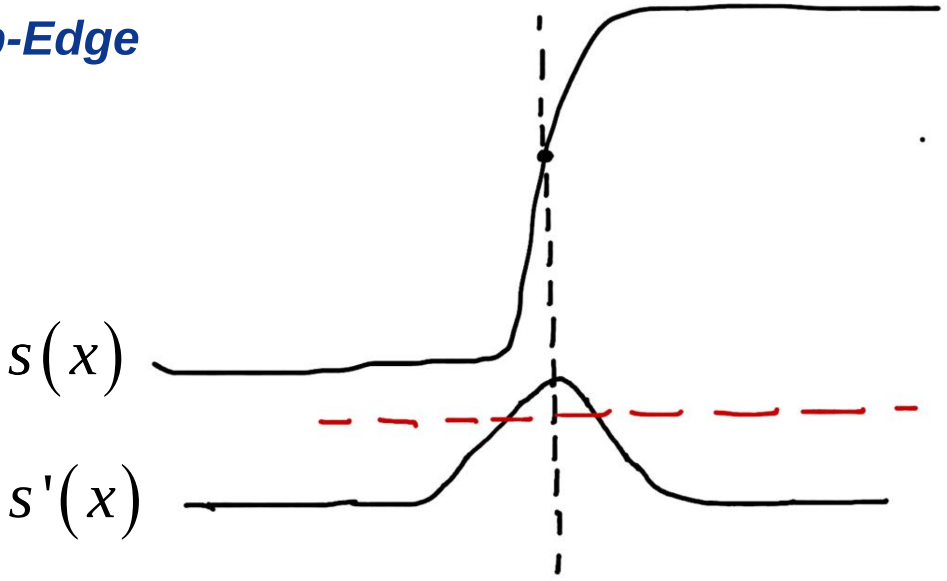
Introduction

- *Edge* or *contour* points are local features of the image that capture effectively important information related to its semantic content.
- Therefore, edges are exploited in countless image analysis tasks, such as e.g. foreground-background segmentation, template matching, stereo matching, visual tracking. In machine vision, edge detection is key to *measurement tools*.
- Edges are pixels that can be thought of as lying exactly in between image regions of different intensity, or, in other words, to separate different image regions.

1D Step-Edge

- ***1D Step-Edge***



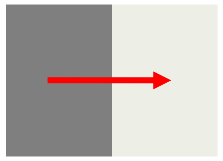
In the transition region between the two constant levels the absolute value of the first derivative gets high (and reaches an extremum at the inflection point).

- **Accordingly, the simplest edge-detection operator relies on thresholding the absolute value of the derivative of the signal:**

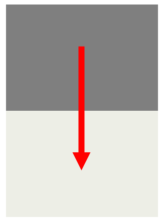
$$\begin{tikzcd} s(x) \arrow[r] & \boxed{\left|\frac{d}{dx}\right|} \arrow[r, "|s'(x)|"] & \boxed{T_h} \arrow[r] & e(x) \end{tikzcd}$$

2D Step-Edge

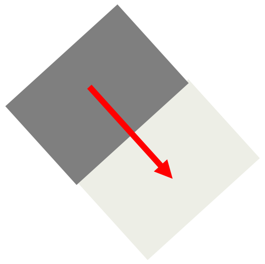
- A 2D *Step-Edge* is characterized not only by its strength but also its direction:



Vertical Edge



Horizontal Edge



Diagonal Edge

- Hence, we cannot use a *directional* derivative but instead need an operator allowing to sense the edge whatever its direction is. Such operator is the gradient:

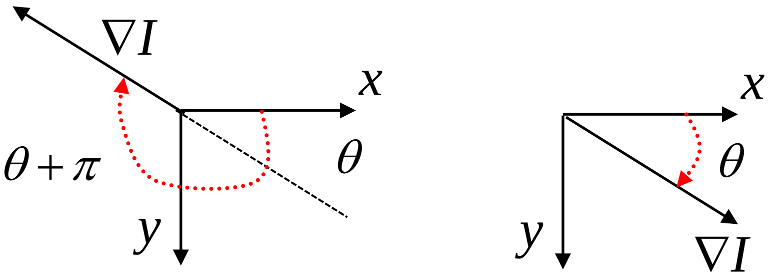
$$\nabla I(x,y) = \frac{\partial I(x,y)}{\partial x}\mathbf{i} + \frac{\partial I(x,y)}{\partial y}\mathbf{j}$$

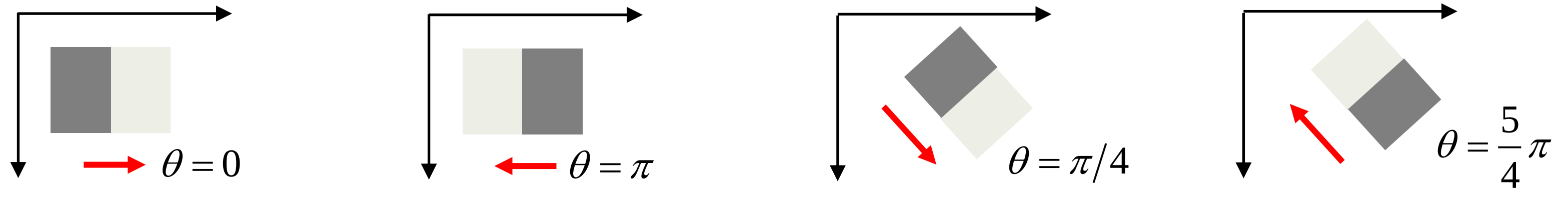
Gradient's direction gives the direction along which the function exhibits its maximum variation, gradient's magnitude being the absolute value of the *directional* derivative along such direction (i.e. the absolute value of the maximum *directional* derivative). Indeed, a generic *directional* derivative can be computed by the dot product between the gradient and the unit vector along the direction.

Edge Detection by Gradient Thresholding

$$\begin{tikzcd}
 & \boxed{I_x = \frac{\partial I(x,y)}{\partial x}} \arrow[dr] & & \\
I(x,y) \arrow[ur] \arrow[dr] & & \boxed{\left|\nabla I(x,y)\right| = \sqrt{\left(I_x\right)^2 + \left(I_y\right)^2}} \arrow[r] \arrow[d, dashed, "\theta"] & \boxed{T_h} \arrow[d] \\
 & \boxed{I_y = \frac{\partial I(x,y)}{\partial y}} \arrow[ur] & \left\{ \begin{array}{l} \vartheta \in \left[-\frac{\pi}{2}, \frac{\pi}{2}\right] = \operatorname{atan}\left(\frac{I_y}{I_x}\right) \\ \vartheta \in [0, 2\pi] = \operatorname{atan2}\left(I_x, I_y\right) \end{array} \right. & e(x,y)
\end{tikzcd}$$

Same direction but different sign (i.e. contrast polarity)





Discrete Approximation of the Gradient (1)

- **We can use either *backward* or *forward* differences:**

$$\frac{\partial I(x,y)}{\partial x} \cong I_x(i,j) = I(i,j) - I(i,j-1), \qquad \frac{\partial I(x,y)}{\partial y} \cong I_y(i,j) = I(i,j) - I(i-1,j)$$

$$\frac{\partial I(x,y)}{\partial x} \cong I_x(i,j) = I(i,j+1) - I(i,j), \qquad \frac{\partial I(x,y)}{\partial y} \cong I_y(i,j) = I(i+1,j) - I(i,j)$$

which may be thought of as *correlation* by:

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

$(0,0)$

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

$(0,0)$

- **Or also central differences:**

$$I_x(i,j) = I(i,j+1) - I(i,j-1), \qquad I_y(i,j) = I(i+1,j) - I(i-1,j)$$

the corresponding correlation kernels being:

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

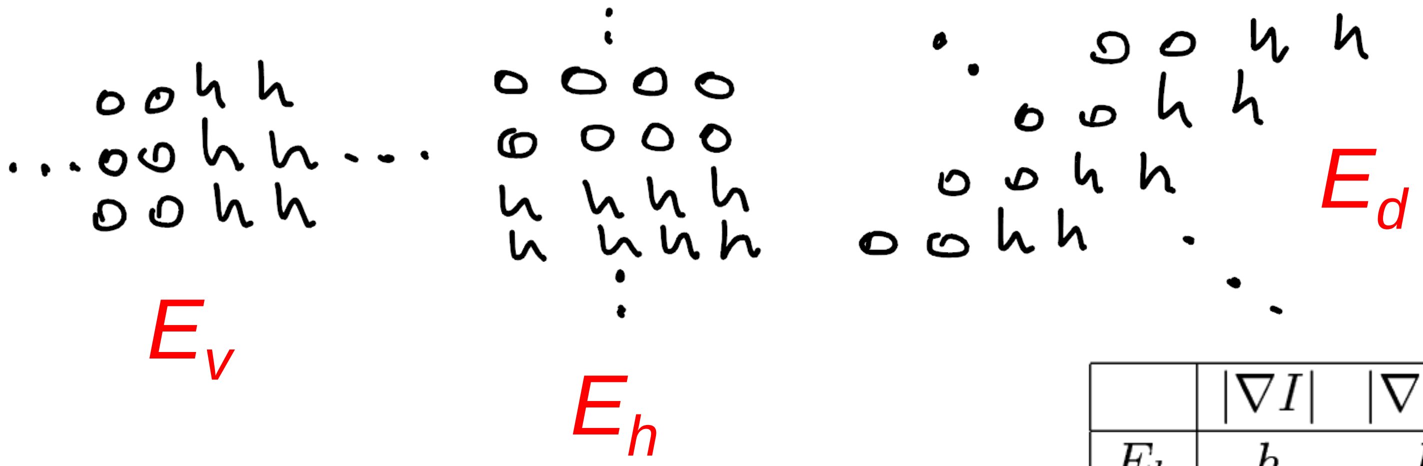
$(0,0)$

Discrete Approximation of the Gradient (2)

- **We can estimate the magnitude by different approximations:**

$$|\nabla I| = \sqrt{(I_x)^2 + (I_y)^2} \qquad |\nabla I|_+ = |I_x| + |I_y| \qquad |\nabla I|_{max} = max\left(|I_x|, |I_y|\right)$$

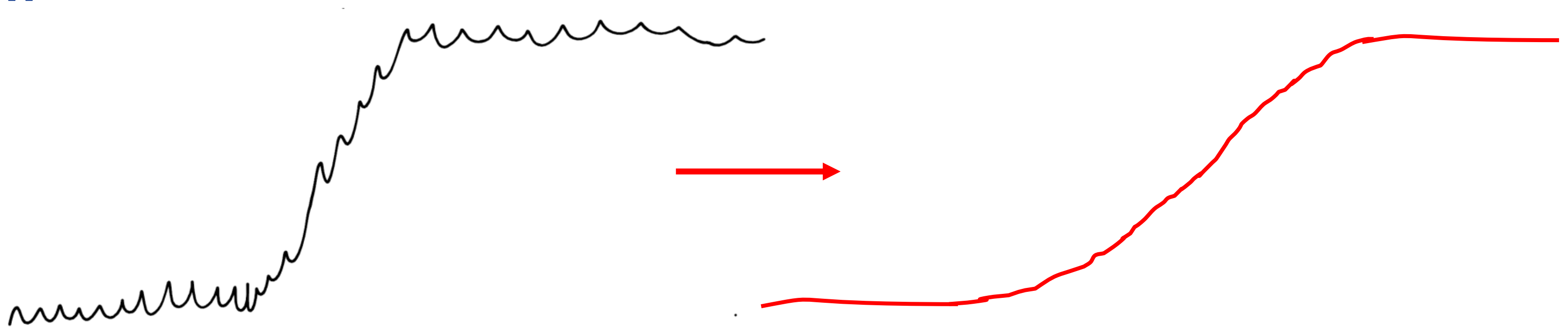
- **The third approximation is faster and more invariant with respect to edge direction:**



	$\lvert\nabla I\rvert$	$\lvert\nabla I\rvert_+$	$\lvert\nabla I\rvert_{max}$
E_h	h	h	h
E_v	h	h	h
E_d	$h\sqrt{2}$	$2h$	h

Dealing with Noise

- **Due to noise, in real images an edge will likely look as depicted in the picture below**



which makes it hard to detect the main step edge out of the many spurious signal changes due to noise.

- **To work with real images, an edge detector should therefore be robust to noise, so as to highlight the meaningful edges only and filter out effectively the spurious transitions caused by noise.**
- **This is usually achieved by smoothing the signal before computing the derivatives required to highlight edges. Unfortunately, smoothing yields also the side-effect of blurring true edges, making it more difficult to detect and localize them accurately.**

Smooth Derivatives (1)

- **Smoothing and differentiation can be carried out jointly within a single step. This is achieved by computing *differences of averages* (rather than averaging the image and then computing differences). To try avoiding smoothing across edges the two operations are carried out along orthogonal directions.**

- **For example, should we wish to smooth out noise by averaging over 3 pixels:**

$$\begin{aligned} I_{3y}(i,j) &= \frac{1}{3}[I(i-1,j)+I(i,j)+I(i+1,j)] \\ I_{3x}(i,j) &= \frac{1}{3}[I(i,j-1)+I(i,j)+I(i,j+1)] \end{aligned}$$

$$\begin{aligned} \tilde{I}_x(i,j) &= I_{3y}(i,j+1)-I_{3y}(i,j) = \frac{1}{3}[I(i-1,j+1)+I(i,j+1)+I(i+1,j+1) \\ &- I(i-1,j)-I(i,j)-I(i+1,j)] \\ &\Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 1 \\ -1 & 1 \\ -1 & 1 \end{bmatrix} \end{aligned}$$

$$\begin{aligned} \tilde{I}_y(i,j) &= I_{3x}(i+1,j)-I_{3x}(i,j) = \frac{1}{3}[I(i+1,j-1)+I(i+1,j)+I(i+1,j+1) \\ &- I(i,j-1)-I(i,j)-I(i,j+1)] \\ &\Rightarrow \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 1 & 1 & 1 \end{bmatrix} \end{aligned}$$

Smooth Derivatives (2)

- **Given the same smoothing, one might wish to approximate partial derivatives by central differences (*Prewitt operator*):**

More isotropic response (i.e. less attenuation of diagonal edges) $\longleftarrow$

$$\tilde{I}_x(i,j) = I_{3y}(i,j+1) - I_{3y}(i,j-1) \quad \Rightarrow \quad \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\tilde{I}_y(i,j) = I_{3x}(i+1,j) - I_{3y}(i-1,j) \quad \Rightarrow \quad \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- **Likewise, the central pixel can be weighted more to further improve isotropy:**

$$\begin{aligned} I_{4y}(i,j) &= \frac{1}{4}[I(i-1,j) + 2I(i,j) + I(i+1,j)] \\ I_{4x}(i,j) &= \frac{1}{4}[I(i,j-1) + 2I(i,j) + I(i,j+1)] \end{aligned}$$

Sobel operator

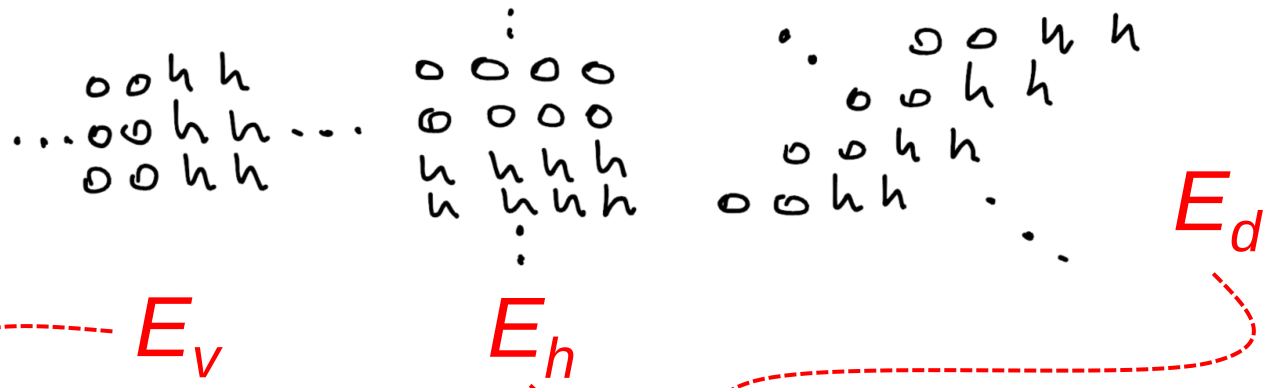
$$\tilde{I}_x(i,j) = I_{4y}(i,j+1) - I_{4y}(i,j-1) \quad \Rightarrow \quad \frac{1}{4}\begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

$$\tilde{I}_y(i,j) = I_{4x}(i+1,j) - I_{4y}(i-1,j) \quad \Rightarrow \quad \frac{1}{4}\begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

Smooth Derivatives (3)

- **Though at the expense of a significantly higher computational complexity, full isotropy is provided by the *Frei-Chen operator*:**

$$\tilde{I}_x = \begin{bmatrix} -1 & 0 & 1 \\ -\sqrt{2} & 0 & \sqrt{2} \\ -1 & 0 & 1 \end{bmatrix} \qquad \tilde{I}_y = \begin{bmatrix} -1 & -\sqrt{2} & -1 \\ 0 & 0 & 0 \\ 1 & \sqrt{2} & 1 \end{bmatrix} \qquad |\nabla I| = \sqrt{(I_x)^2 + (I_y)^2}$$

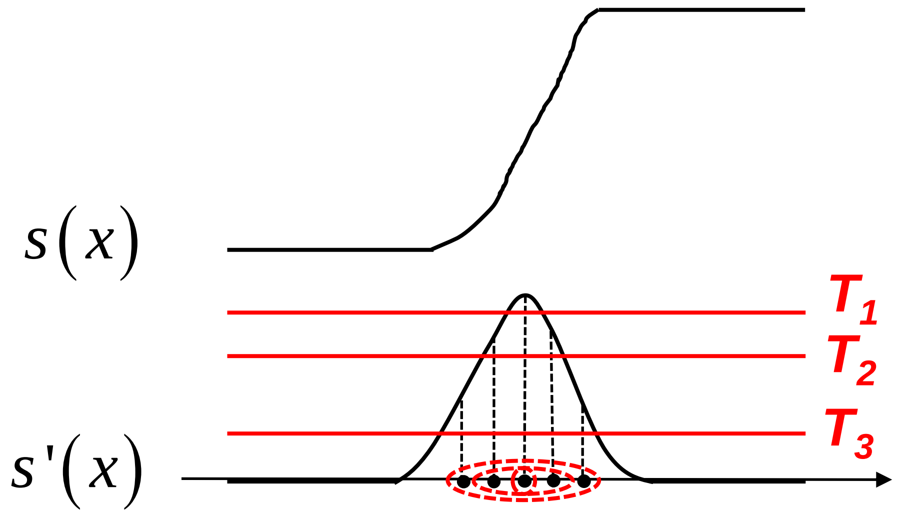


$$h\left(2+\sqrt{2}\right)$$

$$h\left(1+\sqrt{2}\right)\sqrt{2} = h\left(2+\sqrt{2}\right)$$

Finding Maxima to Localize Edges

- **Detecting edges by gradient thresholding is inherently inaccurate as regards localization:**



It turns out difficult in practise to choose the right threshold whenever the image contains meaningful edges characterized by different contrast (i.e. “stronger” as well as “weaker”). Trying to detect weak edges implies poor localization of stonger ones.

- **Analysis of the above picture suggests that a better approach to detect edges may consist in finding the local maxima of the absolute value of the derivative of the signal.**

Non-Maxima Suppression (NMS)

- **When dealing with images (2D signals) one should look for maxima of the absolute value of the derivative (i.e. the gradient magnitude) along the gradient direction (i.e. orthogonally to the edge direction). This process is called NMS.**

$$\longrightarrow \quad \begin{array}{ccccc} & & \vdots & & \\ & 9 & 40 & 8 & \\ & 11 & 42 & 12 & \\ \cdots & 10 & 40 & 10 & \cdots \\ & 8 & 44 & 10 & \\ & & \vdots & & \end{array} \qquad \begin{array}{ccccc} & & \vdots & & \\ & 0 & 1 & 0 & \\ & 0 & 1 & 0 & \\ \cdots & 0 & 1 & 0 & \cdots \\ & 0 & 1 & 0 & \\ & & \vdots & & \end{array} \qquad \begin{array}{ccccc} & & \vdots & & \\ & 0 & 0 & 0 & \\ & 1 & 1 & 1 & \\ \cdots & 0 & 0 & 0 & \cdots \\ & 0 & 1 & 0 & \\ & & \vdots & & \end{array}$$

$$|\nabla I| = |I_x|$$

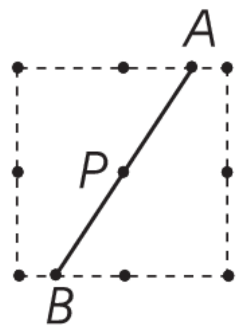
NMS along *x* *(correct)*

NMS along *y* *(wrong !)*

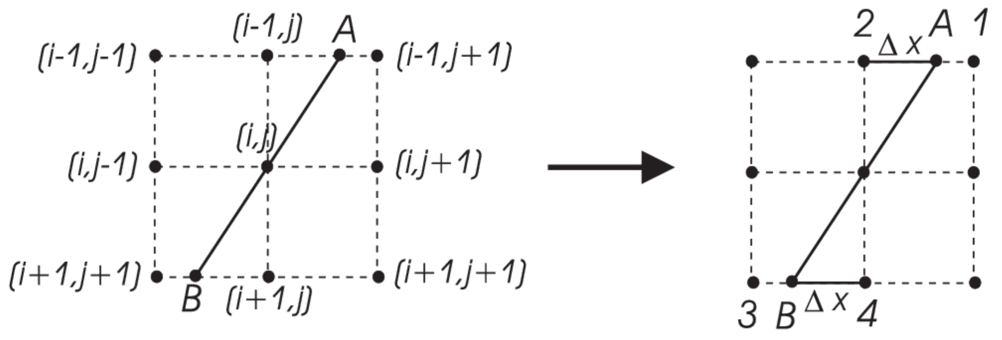
- **However, we don't know in advance the correct direction to carry out NMS. Indeed, such a direction has to be estimated locally, of course based on gradient's direction.**

NMS by Linear Interpolation (1)

- **To perform NMS at pixel P precisely, the magnitude of the gradient has to be estimated at points which do not belong to the pixel grid (e.g. A,B in the picture below):**



Such values can be estimated by linear interpolation of those computed at the closest points belonging to the grid (possibly after projection along the gradient at P) .



$$G_1 = |\nabla I(1)| \qquad G_2 = |\nabla I(2)| \qquad G_3 = |\nabla I(3)|$$

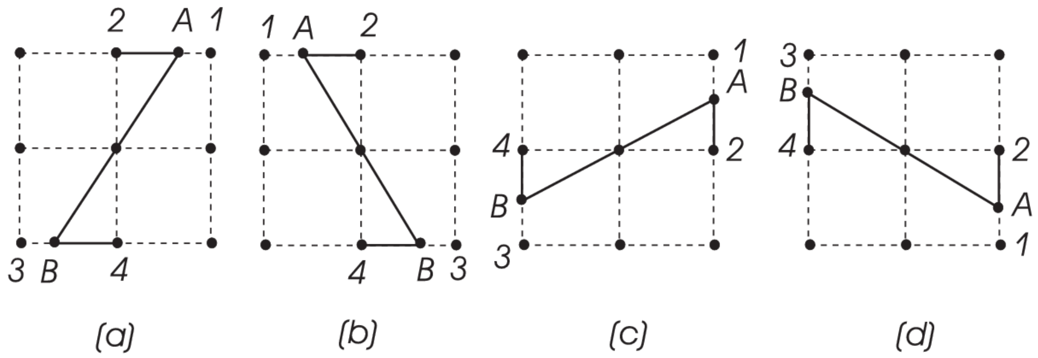
$$G_4 = |\nabla I(4)| \qquad G_A = |\nabla I(A)| \qquad G_B = |\nabla I(B)|$$

$$\longrightarrow$$

$$\begin{aligned} G_A &= G_2 + (G_1 - G_2)\,\Delta x \\ G_B &= G_4 + (G_3 - G_4)\,\Delta x \end{aligned}$$

NMS by Linear Interpolation (2)

- **Points previously denoted as 1,2,3,4 are indeed different ones depending on gradient's direction:**



$$(a),(b) \ : \ |\vartheta| \geq \frac{\pi}{4} \quad \left(|\tilde{I}_y| \geq |\tilde{I}_x|\right)$$

$$(c),(d) \ : \ |\vartheta| < \frac{\pi}{4} \quad \left(|\tilde{I}_y| < |\tilde{I}_x|\right)$$

$$(a),(b) \ \Rightarrow \begin{cases} (a) & : \ \vartheta \leq 0 \quad \left(sign(\tilde{I}_y) \neq sign(\tilde{I}_x)\right) \\ (b) & : \ \vartheta > 0 \quad \left(sign(\tilde{I}_y) = sign(\tilde{I}_x)\right) \end{cases}$$

$$(c),(d) \ \Rightarrow \begin{cases} (c) & : \ \vartheta \leq 0 \quad \left(sign(\tilde{I}_y) \neq sign(\tilde{I}_x)\right) \\ (d) & : \ \vartheta > 0 \quad \left(sign(\tilde{I}_y) = sign(\tilde{I}_x)\right) \end{cases}$$

$$(a),(b) \ \Rightarrow \ \Delta : \Delta x = \frac{1}{|\tan\vartheta|} \quad \left(\frac{|\tilde{I}_x|}{|\tilde{I}_y|}\right)$$

$$(c),(d) \ \Rightarrow \ \Delta : \Delta y = |\tan\vartheta| \quad \left(\frac{|\tilde{I}_y|}{|\tilde{I}_x|}\right)$$

Edges Detection by Smooth Derivatives and NMS

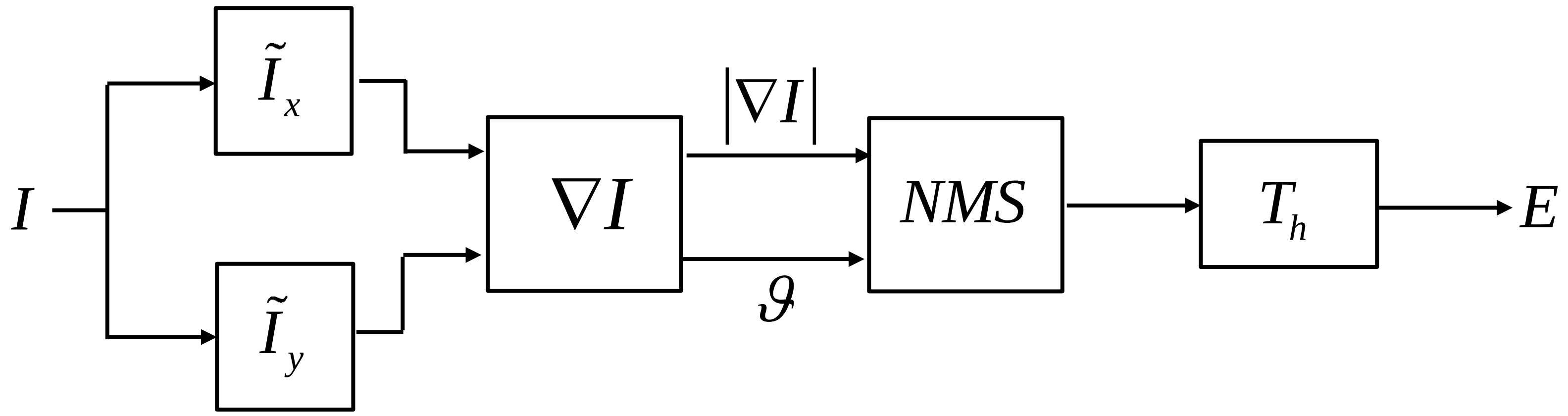
- **The overall flow-chart of an edge detector based on smooth derivatives and NMS is sketched below:**

$$\begin{tikzcd}
& \boxed{\tilde{I}_x} \arrow[dr] & & & \\
I \arrow[ur] \arrow[dr] & & \boxed{\nabla I} \arrow[r, shift left=1.5, "|\nabla I|"] \arrow[r, shift right=1.5, "\vartheta"'] & \boxed{NMS} \arrow[r] & {} \\
& \boxed{\tilde{I}_y} \arrow[ur] & & &
\end{tikzcd}$$

A final thresholding step typically helps pruning out unwanted edges due to either noise or less important details.

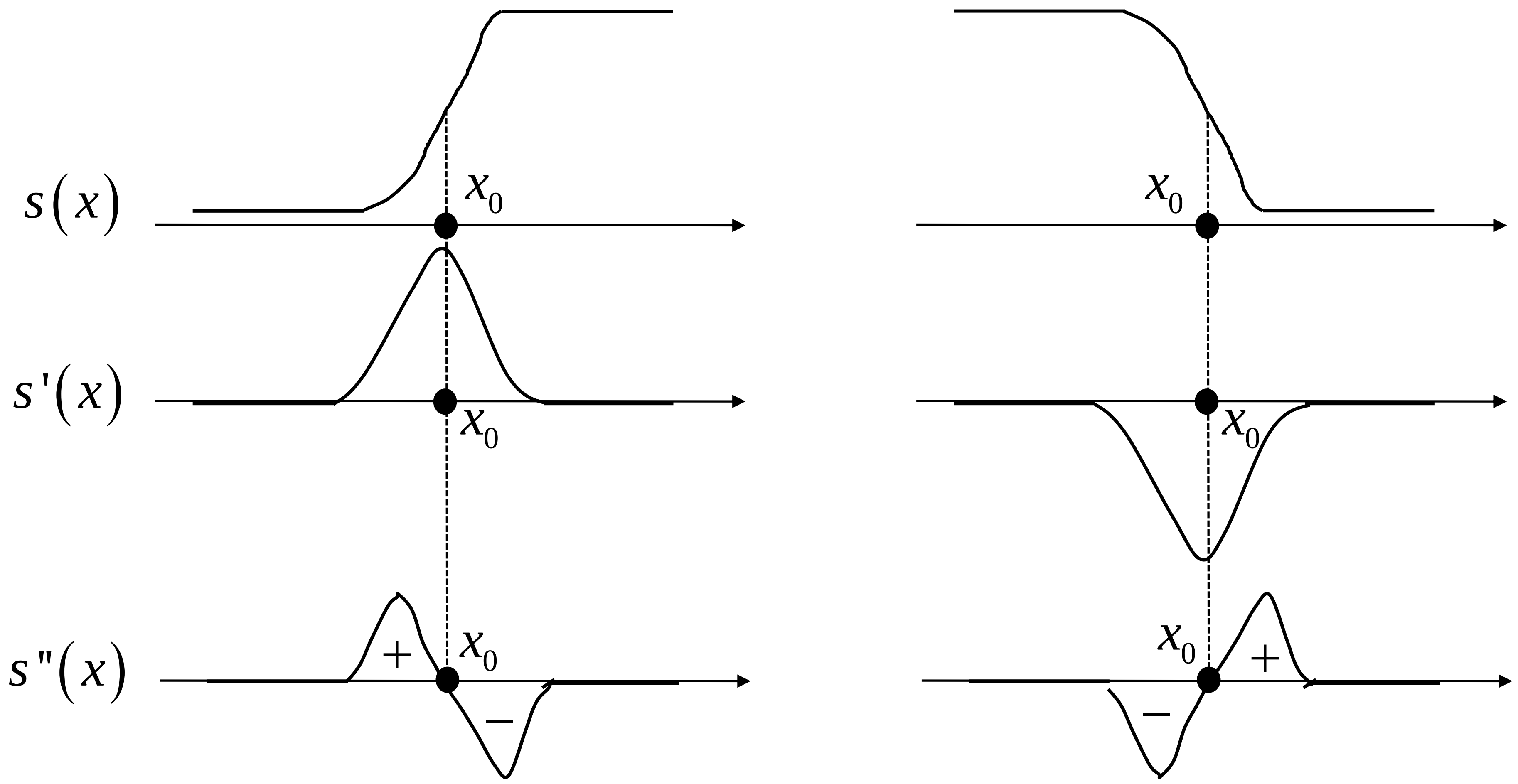
Edges Detection by Smooth Derivatives and NMS

- **The overall flow-chart of an edge detector based on smooth derivatives and NMS is sketched below:**



A final thresholding step typically helps pruning out unwanted edges due to either noise or less important details.

Zero-crossing of the Second Derivative



Edges may also be located by looking for zero-crossing of the second derivative of the signal.

Zero-crossing along the Gradient

- **Again, in the case of images (2D signals) we should look for zero-crossing of the second derivative *along gradient's direction*:**

$$\frac{\partial^2 I}{\partial n^2}, \qquad \vec{n} = \frac{\nabla I}{|\nabla I|} \qquad \longrightarrow \qquad \frac{\partial^2 I}{\partial n^2} = \frac{\partial |\nabla I|}{\partial n} = \nabla\left(|\nabla I|\right)\vec{n}$$

$$\nabla\left(|\nabla I|\right) = \nabla\left(I_x^2 + I_y^2\right)^{\frac{1}{2}} = \frac{\left(I_x I_{xx} + I_y I_{yx}\right)\vec{x} + \left(I_y I_{yy} + I_x I_{yx}\right)\vec{y}}{\left(I_x^2 + I_y^2\right)^{\frac{1}{2}}}$$

$$\nabla\left(|\nabla I|\right)\vec{n} = \frac{I_x^2 I_{xx} + 2 I_x I_y I_{xy} + I_y^2 I_{yy}}{I_x^2 + I_y^2}$$

Requires a significant computational effort !

The Laplacian

- **In their seminal work on edge detection, Marr & Hildreth proposed to rely on the Laplacian as second order differential operator:**

$$\nabla^2 I(x,y) = \frac{\partial^2 I(x,y)}{\partial x^2} + \frac{\partial^2 I(x,y)}{\partial y^2} = I_{xx} + I_{yy}$$

- **Using forward and backward differences to approximate, respectively, first and second order derivatives:**

$$I_{xx} \cong I_x(i,j) - I_x(i,j-1) = I(i,j-1) - 2I(i,j) + I(i,j+1)$$

$$I_{yy} \cong I_y(i,j) - I_y(i-1,j) = I(i-1,j) - 2I(i,j) + I(i+1,j)$$

$$\nabla^2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

Laplacian of Gaussian (LOG)

- **It can be shown that the zero-crossing of the Laplacian typically lay close to those of the second derivative along the gradient, being the former differential operator much faster to compute (i.e. just a convolution by a 3x3 kernel) than the latter.**

- **As already discussed, a robust edge detector should include a smoothing step to filter out noise (especially in case second rather than first order derivatives are deployed). In their edge detector, Marr&Hildreth proposed to use a Gaussian filter as smoothing operator. Hence, their edge detector is referred to as LOG (Laplacian of Gaussian).**

- **Edge detection by the LOG can the be summarized conceptually as follows:**

  1. **Gaussian smoothing:** $\tilde{I}(x,y) = I(x,y) * G(x,y)$
  2. **Second order differentiation by the Laplacian:** $\nabla^2 \tilde{I}(x,y)$
  3. **Extraction of the zero-crossing of** $\nabla^2 \tilde{I}(x,y)$

Laplacian of Gaussian (LOG)

- Unlike those based on smooth derivatives, the LOG edge detector allows the degree of smoothing to be controlled (i.e. by changing the σ parameter of the Gaussian filter). This, in turn, allows the edge detector to be tuned according to the degree of noise in the image (i.e. higher noise -> larger σ).

- Likewise, σ may be used to control the scale at which the image is analyzed, with larger σ typically chosen to extract the edges related to main scene structures and smaller σ to capture also small size details.

- Zero-crossing are usually sought for by scanning the image by both rows and columns to identify changes of the sign of the LOG.

- Once a sign change is found, the actual edge may be localized:

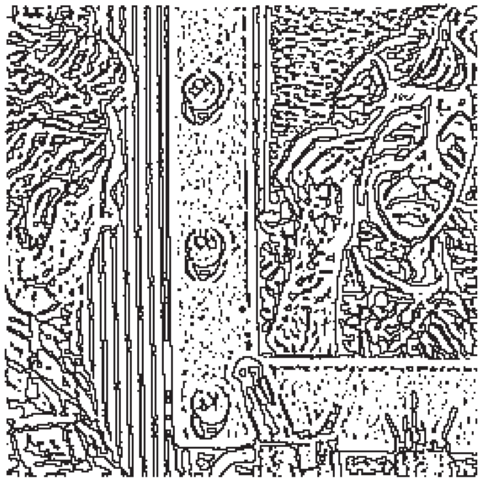
  1. At the pixel where the LOG is positive (darker side of the edge).

  2. At the pixel where the LOG is negative (brighter side of the edge).

  3. At the pixel where the absolute value of the LOG is smaller (the best choice, as the edge turns out closer to the "true" zero-crossing).

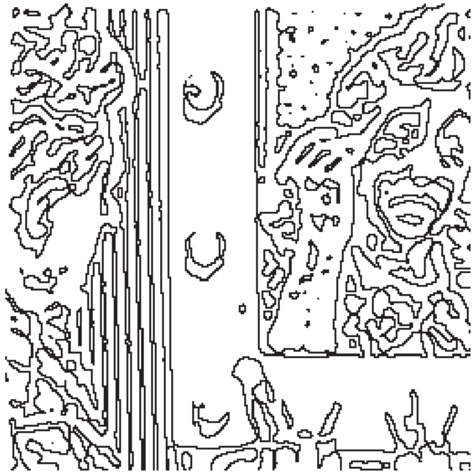
- Again, to help discarding spurious edges, a final thresholding step may be enforced (usually based on the slope of the LOG at the found zero-crossing).

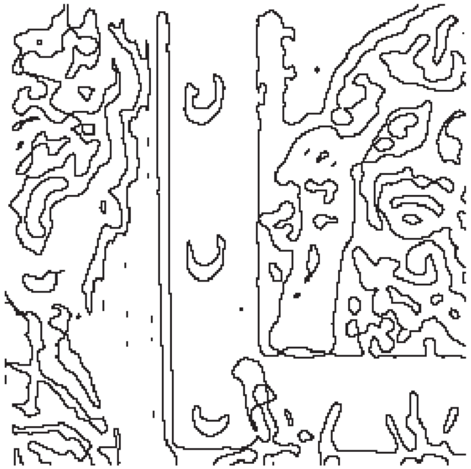
Exemplar Results by the LOG (1)



$ZCs,\ \sigma = 1$

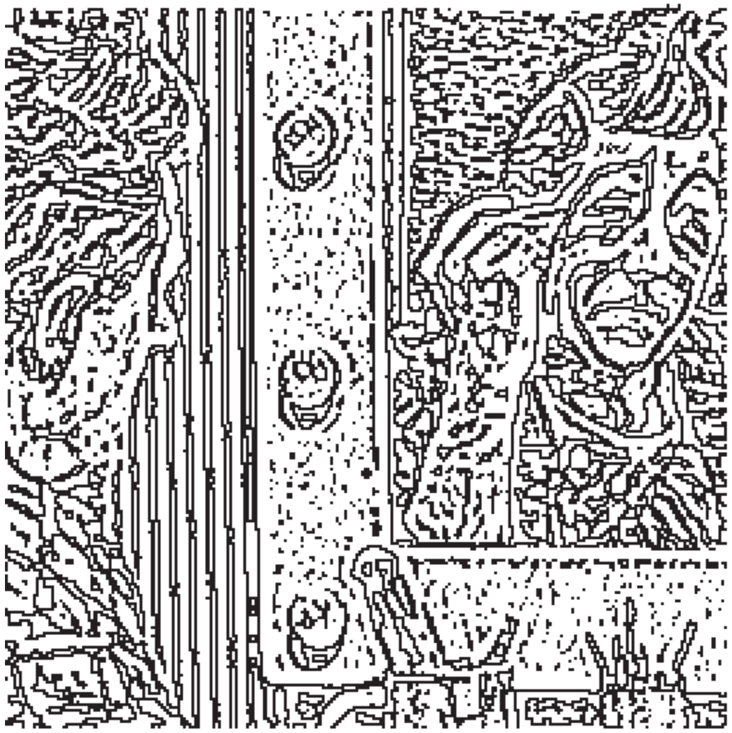
$ZCs,\ \sigma = 2$



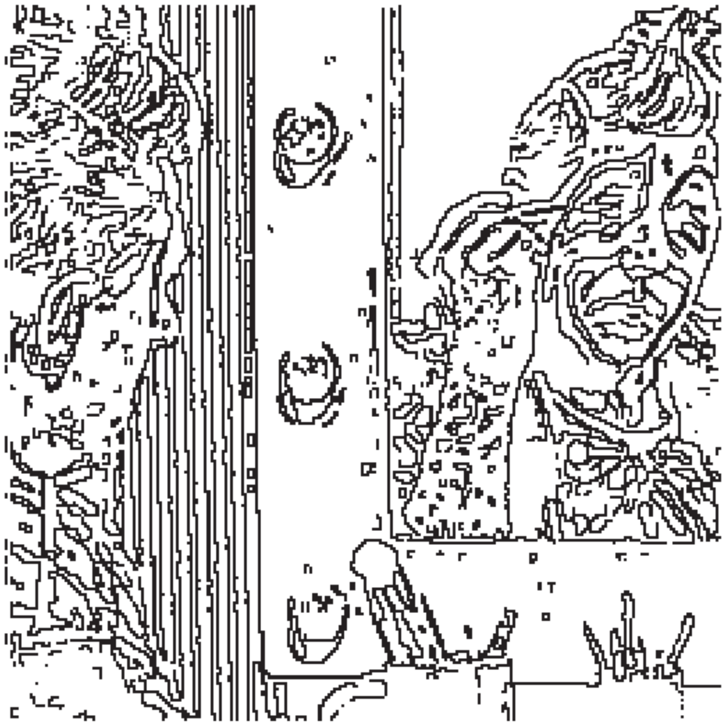


$ZCs,\ \sigma = 3$

Exemplar Results by the LOG (2)



$ZCs, \ \sigma = 1$



$T_h\left(ZCs\right), \ \sigma = 1$

Thresholding based on the slope of the LOG (≥40)

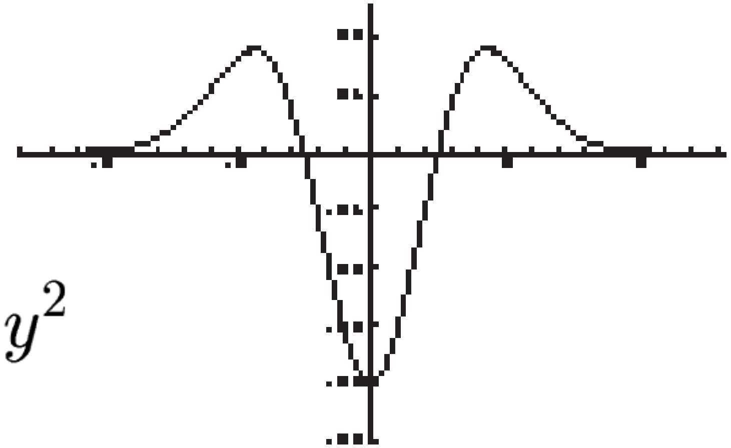
Computation of the LOG (1)

$$\nabla^2 \tilde{I}(x,y) = \nabla^2 \left(I(x,y) * G(x,y)\right) = I(x,y) * \nabla^2 G(x,y)$$

$$\nabla^2 G(x,y) = \frac{\partial^2 G(x,y)}{\partial x^2} + \frac{\partial^2 G(x,y)}{\partial y^2}$$

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

Mexican Hat filter



Computation of the LOG (2)

- **2D convolution by the Mexican Hat can be expensive in terms of computation, especially when the size of the filter is large. As it is the case of the Gaussian, the size of the filter, *d*, must increase with σ. According to several studies:**

$$3\omega \leq d \leq 4\omega, \quad \omega = 2\sqrt{2} \cdot \sigma$$

$$\longrightarrow$$

$\sigma = 0.5$	$d = 4.24$	$\Rightarrow$	$5x5$
$\sigma = 1$	$d = 8.48$	$\Rightarrow$	$9x9$
$\sigma = 2$	$d = 16.97$	$\Rightarrow$	$17x17$
$\sigma = 3$	$d = 25.45$	$\Rightarrow$	$26x26$
$\sigma = 4$	$d = 33.94$	$\Rightarrow$	$34x34$
$\sigma = 5$	$d = 42.42$	$\Rightarrow$	$43x43$

- **However, thanks to separabillty of the Gaussian, computing the LOG boils down to four 1D convolutions, which is substantially faster (*4d* rather that *d*^{2} ops per pixel):**

$$I(x,y) * \nabla^2 G(x,y) \;=\; I(x,y) * \left(G''(x)G(y) + G''(y)G(x)\right)$$

$$=\; I(x,y) * \left(G''(x)G(y)\right) + I(x,y) * \left(G''(y)G(x)\right)$$

$$=\; \left(I(x,y) * G''(x)\right) * G(y) + \left(I(x,y) * G''(y)\right) * G(x)$$

Canny's Edge Detector (1)

- **Canny proposed to set forth quantitative criteria to measure the performance of an edge detector and then to find the optimal filter with respect to such criteria. Accordingly, he proposed the following three criteria:**

  1. ***Good Detection*: the filter should correctly extract edges in noisy images.**

  2. ***Good Localization*: the distance between the found edge and the "true" edge should be minimum.**

  3. ***One Response to One Edge*: the filter should detect one single edge pixel at each "true" edge.**

- **Addressing the 1D case and modeling an edge as a noisy step, he shows that the optimal edge detection operation consists in finding local extrema of the convolution of the signal by a first order Gaussian derivative (i.e. *G'(x)*) . This theoretical result can also be regarded as a proof of the optimality of the Gaussian as smoothing filter to detect noisy edges.**

- **As usual, to end up with a practical 2D edge detector to be applied to images, we should look for local extrema of the directional derivative along the gradient. As such, a straightforward Canny edge detector can be achieved by Gaussian smoothing followed by gradient computation and NMS along the gradient direction.**

Canny’s Edge Detector (2)

- **As already pointed out, 2D convolution by a Gaussian can be slow and we can leverage on separability of the Gaussian function to speed-up the calculation:**

$$\tilde{I}_x(x,y) = \frac{\partial}{\partial x}\left(I(x,y) * G(x,y)\right) = I(x,y) * \frac{\partial G(x,y)}{\partial x}$$

$$\tilde{I}_y(x,y) = \frac{\partial}{\partial y}\left(I(x,y) * G(x,y)\right) = I(x,y) * \frac{\partial G(x,y)}{\partial y}$$

$$G(x,y) = G(x)G(y)$$

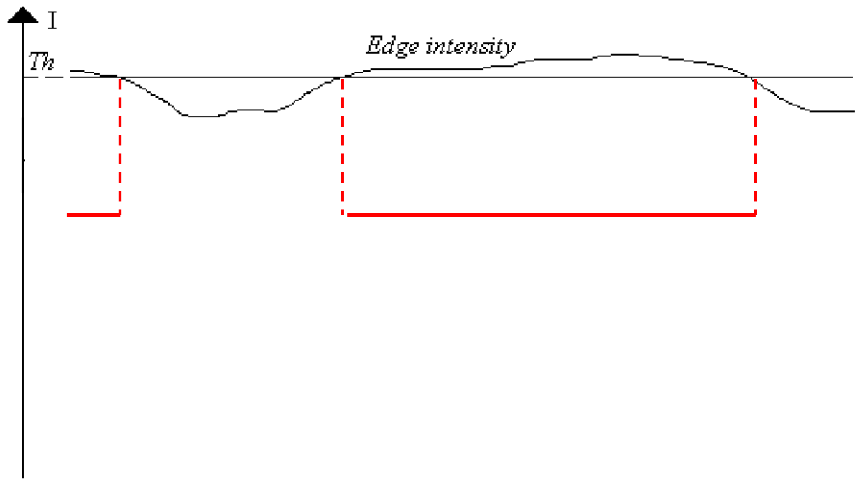
$$\Big\downarrow$$

$$\tilde{I}_x(x,y) = I(x,y) * \left(G'(x)G(y)\right) = \left(I(x,y) * G'(x)\right) * G(y)$$

$$\tilde{I}_y(x,y) = I(x,y) * \left(G'(y)G(x)\right) = \left(I(x,y) * G'(y)\right) * G(x)$$

Canny's Edge Detector (3)

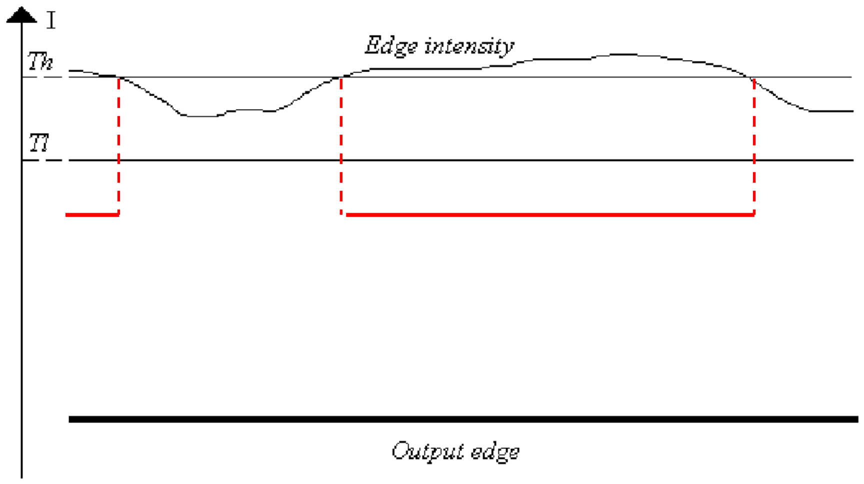
- **As already discussed, NMS is often followed by thresholding of gradient magnitude to help distinguish between true "semantic" edges and unwanted ones. However, *edge streaking* may occur when magnitude varies along object contours.**



- **To address the above issue, Canny proposed a "hysteresis" thresholding approach relying on a higher (T_h) and a lower (T_l) threshold. A pixel is taken as an edge if either the gradient magnitude is higher than T_h OR higher than T_l AND the pixel is a neighbor of an already detected edge.**

Canny’s Edge Detector (3)

- **As already discussed, NMS is often followed by thresholding of gradient magnitude to help distinguish between true “semantic” edges and unwanted ones. However, *edge streaking* may occur when magnitude varies along object contours.**

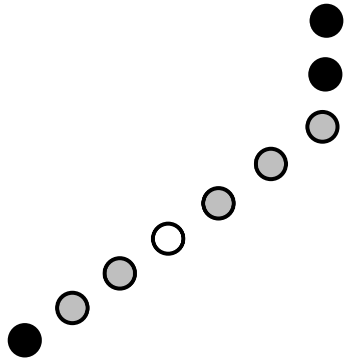


- **To address the above issue, Canny proposed a “hysteresis” thresholding approach relying on a higher (T_h) and a lower (T_l) threshold. A pixel is taken as an edge if either the gradient magnitude is higher than T_h OR higher than T_l AND the pixel is a neighbor of an already detected edge.**

Canny's Edge Detector (4)

- Hysteresis thresholding is usually carried out by tracking edge pixels along contours, which also brings in the "side-effect" of Canny's providing as output chains of connected edge pixels rather than edge maps.

Edge candidates provided by NMS



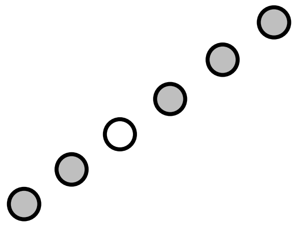
○ : $|\nabla I| > T_h$

◉ : $|\nabla I| > T_l$

● : $|\nabla I| < T_l$

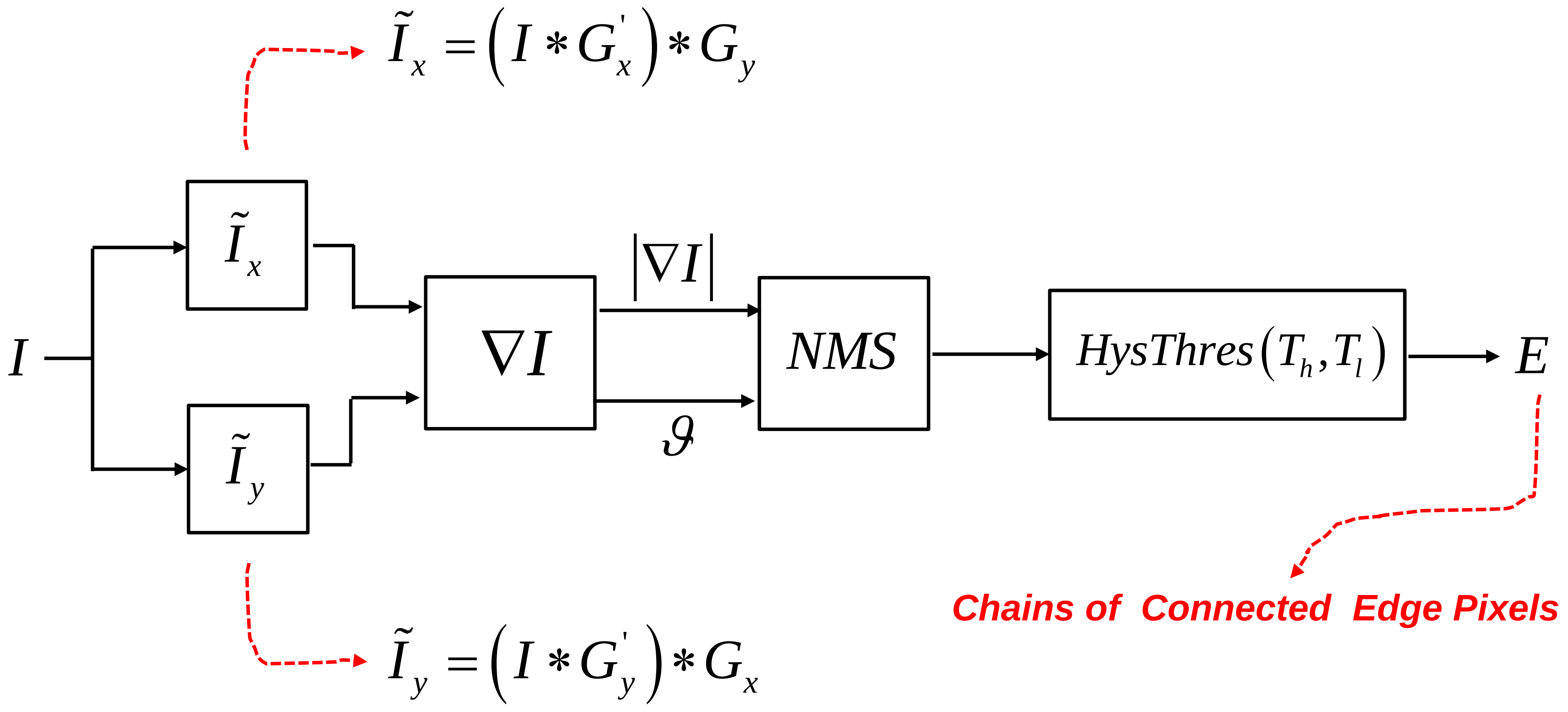
Step-1: pick up all strong edges

Step-2: for each strong edge track weak edges along contours



Canny's Edge Detector (5)

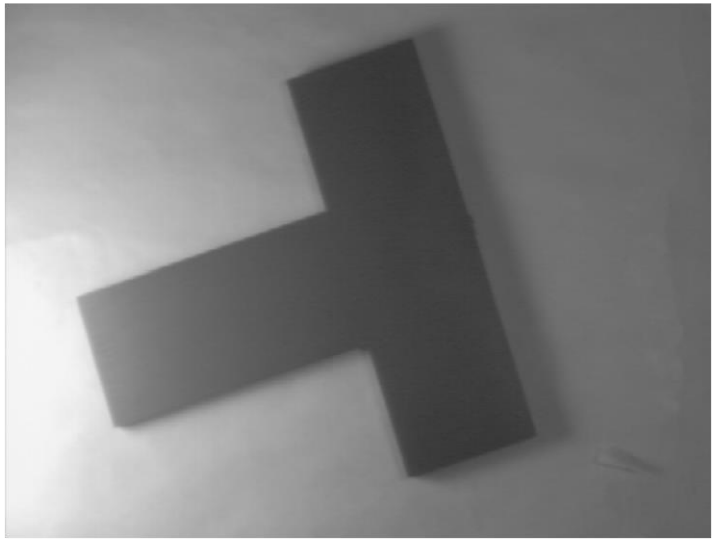
- **The overall flow-chart of the Canny edge detector is provided below:**



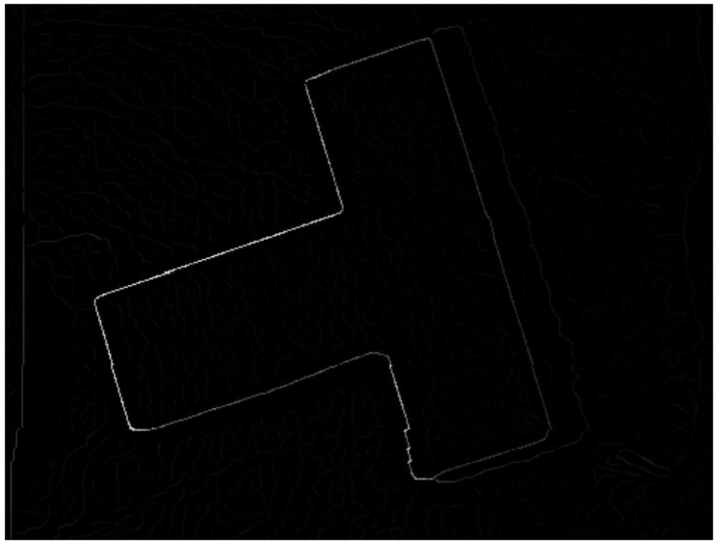
$$\tilde{I}_x = \left(I * G_x^{'}\right) * G_y$$

$$\tilde{I}_y = \left(I * G_y^{'}\right) * G_x$$

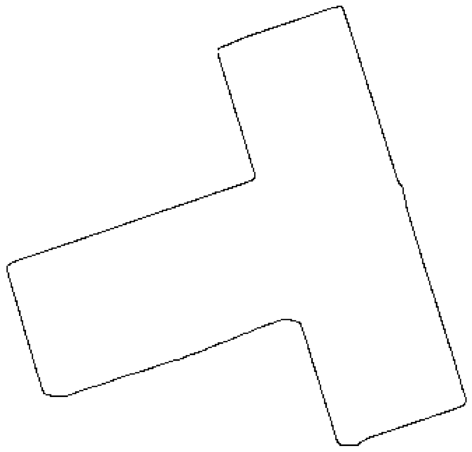
Exemplar Results by Canny's



Input Image



NMS output (with gradient magnitude represented by gray-scales)



Final Output

Main References

1) A. Rosenfeld, A. Kak, “Digital Picture Processing – Vol. 2, Academic Press, 1982.

2) R. Fisher, S. Perkins, A. Walker, E. Wolfart, “Hypermedia Image Processing Reference”, Wiley, 1996 (http://homepages.inf.ed.ac.uk/rbf/HIPR2/)

3) R. Schalkoff, “Digital Image Processing And Computer Vision, Wiley, 1989.

4) D. Marr, E. Hildreth, “Theory of Edge Detection”, Proc. Royal Society of London, 1980.

5) J. Canny, “A computational approach to edge detection” , IEEE Trans. On PAMI, 1986.