Image Processing and Computer Vision Notes

by Mattia Orlandi

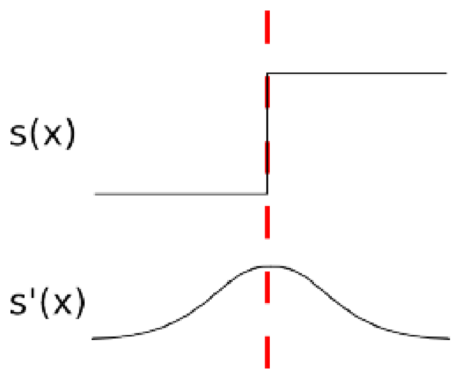
8. Edge Detection

8.1. Detecting Edges by Thresholding

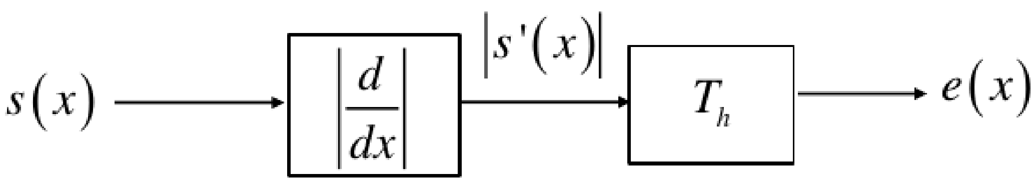
- **Edge** or *contour* points are **local features** of the image that **capture information** related to its **semantic content**.
- Edges are **pixels lying in between image regions** of **different intensities** ⇒ they **separate different semantic regions**.

1D Step-Edge

The **signal** is a **step function**, and the absolute value of its **first derivative** reaches its **maximum** at the **inflection point in the transition region**, between the two constant levels.

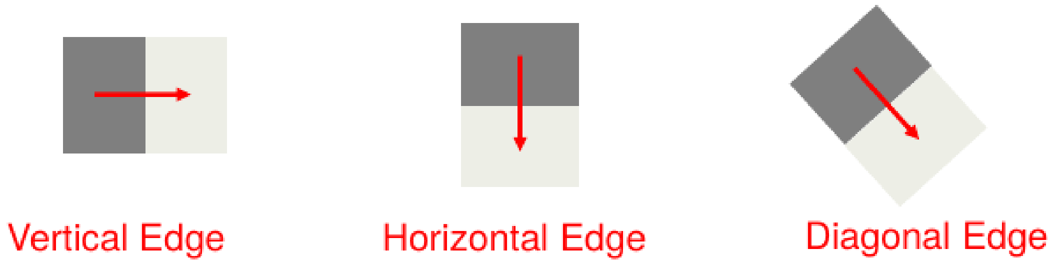


The **simplest edge-detection operator** relies on **thresholding the absolute value of the derivative** of the signal.



2D Step-Edge

- In **2D**, a **contour** is characterized not only by its **strength** but also by its **direction**.



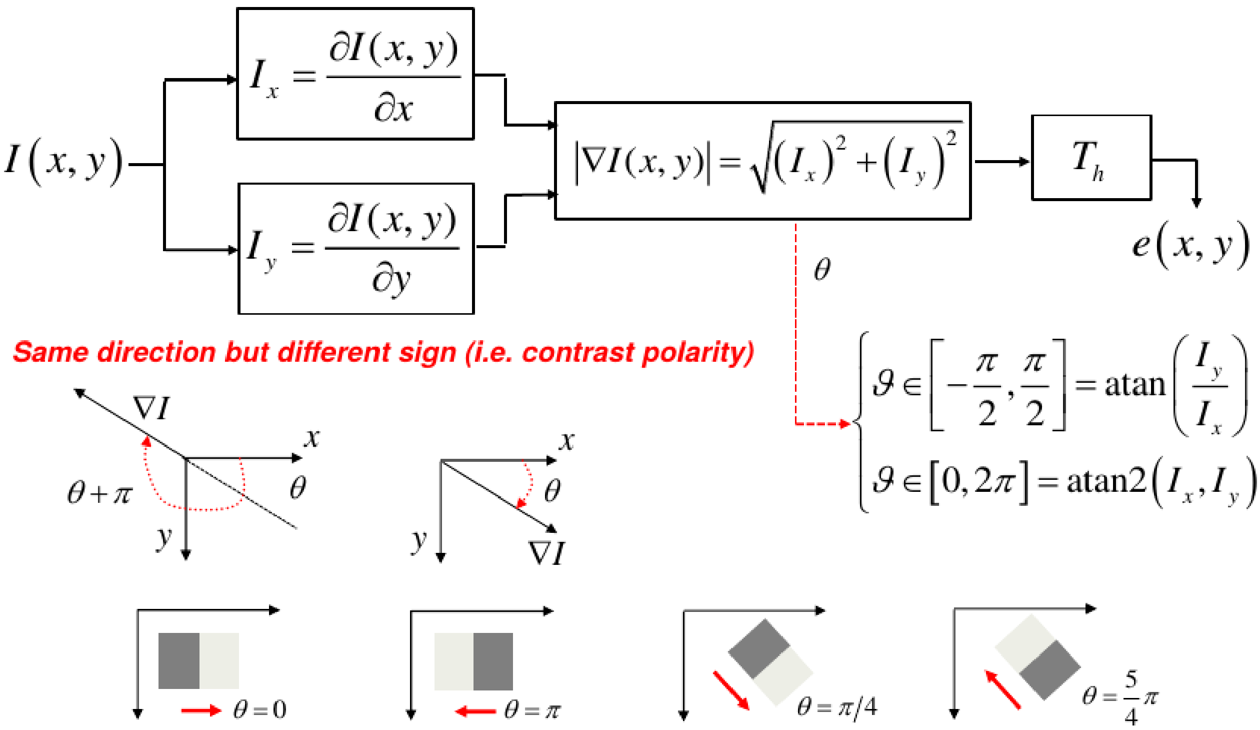
- There are **infinite direction derivatives**, thus the **gradient is used** to sense the edge:

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

- **Gradient's direction** gives the **direction along which** the **function varies the most**, and **gradient's magnitude** gives the **absolute value** of **directional derivative along such direction**.

- A **generic directional derivative** can be computed as the **dot product between gradient and unit vector** along the direction.

- The **edge detection** relies on the **thresholding of the gradient**, which consist of computing the two directional derivatives $I_x = \frac{\partial I}{\partial x}(x,y), I_y = \frac{\partial I}{\partial y}(x,y)$ along x, y, by computing gradient's magnitude as $\sqrt{I_x^2 + I_y^2}$ and by thresholding it.



Discrete Approximation of Gradient

- The image is not a continuous function, so **derivatives** are **approximated with their discrete counterparts**, which are **just differences** between nearby pixels.
- **Backward** differences are **differences between current pixel and previous one**:

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

- **Forward** differences are **differences between next pixel and current one**:

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

- Backward and forward differences can be thought of **correlations** by following kernels:

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

- **Central** differences are **differences between next pixel and previous one**:

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

  where the **corresponding correlation kernels** are:

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

- **Magnitude** can be estimated by several approximations:

$$\begin{aligned} |\nabla I| &= \sqrt{I_x^2 + I_y^2} && (L_2) \\ |\nabla I|_+ &= |I_x| + |I_y| && (L_1) \\ |\nabla I|_{max} &= \max(|I_x|, |I_y|) && (L_\infty) \end{aligned}$$

  where the third approximation is **faster and more invariant** w.r.t. edge direction.
- Because of **quantization**, rotating the same horizontal or vertical edge into a diagonal one will produce two **different derivatives** (*isotropic intrinsic problem*).

Smooth Derivatives

- Due to **noise**, **edges are not smooth** and present random peaks, which make it **hard to detect the main step** edge out of the many **spurious signal changes due to noise** $\Rightarrow$ an edge detector should be **robust w.r.t. noise**.
- The signal is **smoothed before computing the derivatives** required to highlight edges; as side-effect, **true edges are blurred** and more difficult to localize.
- **Smoothing and differentiation** can be carried out in a **single step** by computing **differences of averages** (rather than averaging the image and computing differences).

- To avoid smoothing across edges, the two operations (i.e. average and differences) are **carried out along orthogonal directions** (e.g. when computing the horizontal derivative, the smoothing is vertical and viceversa):

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

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

- To have a **more isotropic response** (i.e. less attenuation of diagonal edges), it is possible to approximate by **central differences** (**Prewitt operator**):

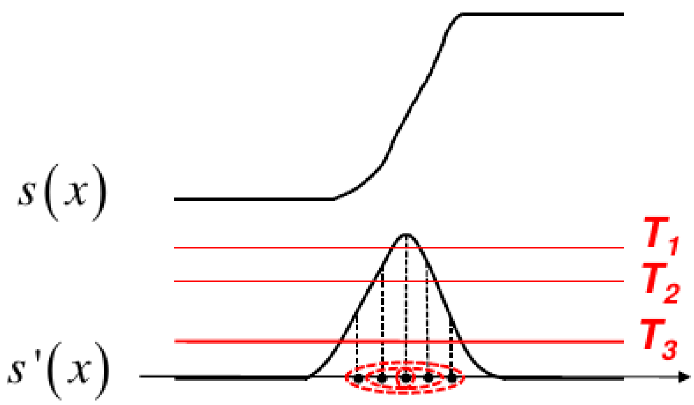
$$\tilde{I}_x(i,j) = I_{3y}(i,j+1) - I_{3y}(i,j-1) \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$
$$\tilde{I}_y(i,j) = I_{3x}(i+1,j) - I_{3x}(i-1,j) \Rightarrow \frac{1}{3}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

- The **central pixel** can be **weighted more** to further **improve isotropy** (**Sobel operator**):

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

8.2. Detecting Edges finding Maxima

- Detecting edges by **gradient thresholding** is **inherently inaccurate** as regards **localization**.
- If an image contains edges characterized by different contrast (i.e. “stronger” and “weaker”), **trying to detect weak edges** implies **poor localization of stronger ones**:

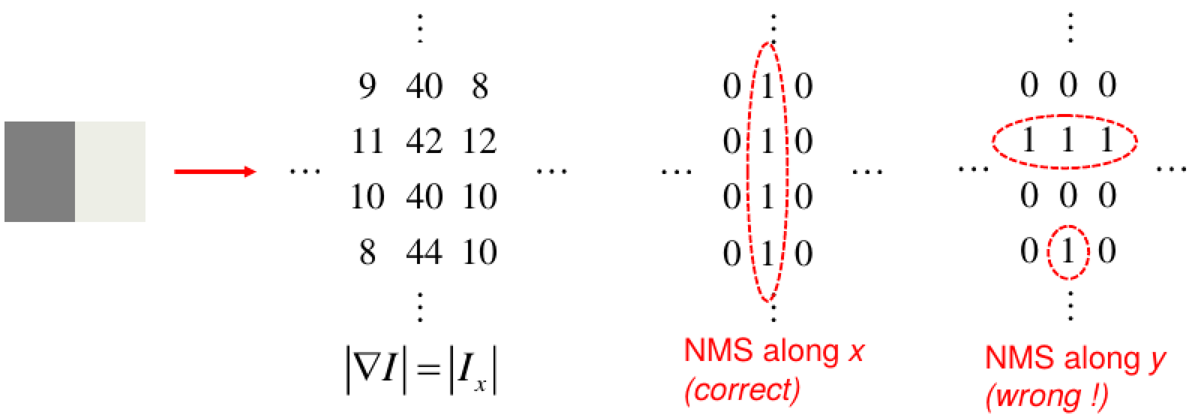


in fact, **depending on the threshold**, an **edge** could be **highlighted by more than one pixel**.

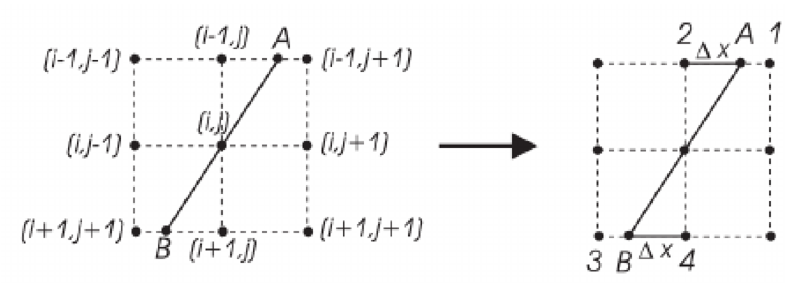
- A better approach is to detect edges by **finding the local maxima** of the absolute value of the derivative of the signal.

Non-Maxima Suppression (NMS)

- The edge is detected by **computing** the **maximum of** the **gradient magnitude along its direction**.



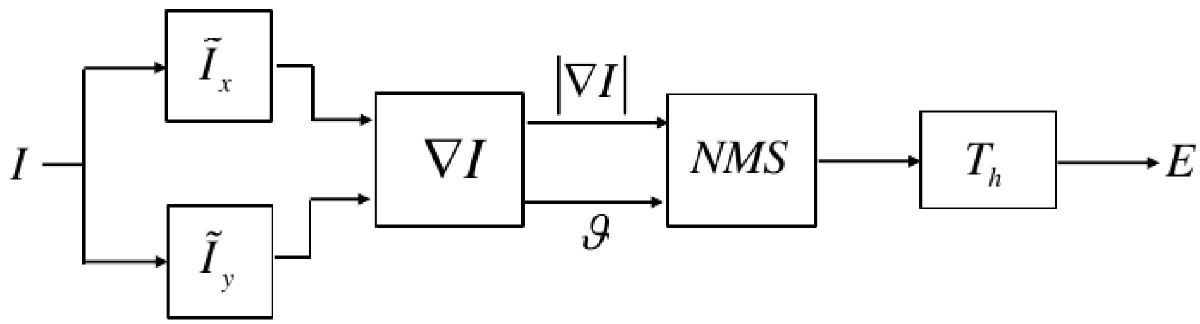
- The **correct direction** to carry out NMS is not known in advance, and has to be **estimated locally**.
- To perform NMS at pixel P precisely, the magnitude of the gradient has to be estimated at points not belonging to pixel grid (e.g. A, B):



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

$$\begin{array}{lll} G_1 = |\nabla I(1)| & G_2 = |\nabla I(2)| & G_3 = |\nabla I(3)| \\ G_4 = |\nabla I(4)| & G_A = |\nabla I(A)| & G_B = |\nabla I(B)| \end{array} \Rightarrow \begin{array}{l} G_A = G_2 + (G_1 - G_2)\Delta x \\ G_B = G_4 + (G_3 - G_4)\Delta x \end{array}$$

- The overall flow-chart of an edge detector based on smooth derivatives and NMS is the following:

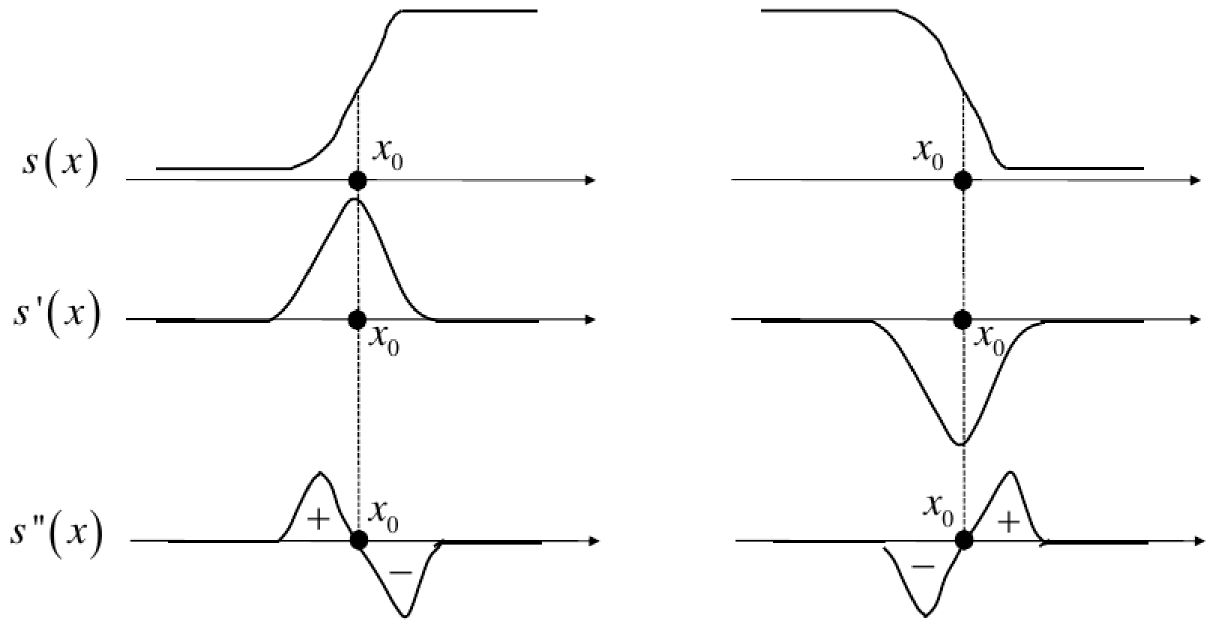


A final thresholding step typically helps pruning out unwanted edges due to either noise or less important details.

8.3. Detecting Edges with Second Order Derivatives

Zero-crossing along the Gradient

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



- In case of 2D signals, zero-crossing second derivative must be looked for **along gradient's direction**:

$$\frac{\partial^2 I}{\partial n^2},\ \vec{n} = \frac{\nabla I}{|\nabla I|} \Rightarrow \frac{\partial^2 I}{\partial n^2} = \frac{\partial |\nabla I|}{\partial n} = \nabla(|\nabla I|)\vec{n}$$

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

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

after which zero-crossing is seeked.

- **Significant computational effort**.

Laplacian

- A **more efficient** way is to rely on the **Laplacian operator**:

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

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

$$
\begin{aligned}
I_{xx} &\approx I_x(i,j) - I_x(i,j-1) = I(i,j-1) - 2I(i,j) + I(i,j+1) \\
I_{xx} &\approx I_x(i,j) - I_x(i-1,j) = I(i-1,j) - 2I(i,j) + I(i+1,j)
\end{aligned}
$$

$$\Downarrow$$

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

after which zero-crossing is seeked.

- With second order derivative, noise will spread more $\Rightarrow$ **smoothing necessary**.

8.4. Laplacian of Gaussian (LOG)

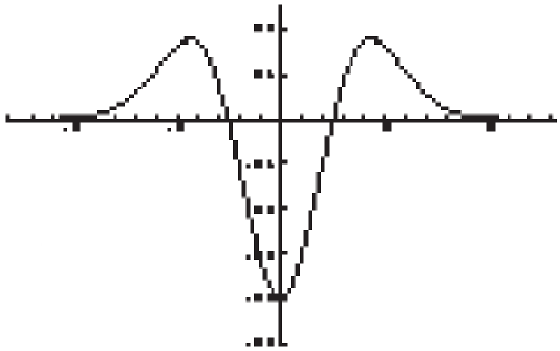
- **Zero-crossing of Laplacian lay close** to those of **second derivative** along the gradient.
- A robust edge detector should include a **smoothing step to filter noise**, so **Gaussian filter** is used $\Rightarrow$ **Laplacian of Gaussian** (LOG) operator.
- Edge detection using LOG performs following steps:
  1. **Gaussian smoothing**: $\tilde{I}(x,y) = I(x,y) * G(x,y)$
  2. Second order **differentiation by Laplacian**: $\nabla^2 \tilde{I}(x,y)$
  3. **Extraction of zero-crossing** of $\nabla^2 \tilde{I}(x,y)$
- LOG edge detector allows the **degree of smoothing to be controlled** (i.e. by changing σ parameter of Gaussian filter), thus LOG can be **tuned according to the degree of noise** (higher noise, larger σ).
- Also, σ controls the **scale** at which image is analyzed:
  - larger $\sigma \Rightarrow$ edges related to main structures;
  - smaller $\sigma \Rightarrow$ small-sized details.
- **Zero-crossing** are sought for by scanning image by both rows and columns to identify **changes of sign** of LOG.
- Once a sign change is found, the actual edge may be localized:
  1. At the **pixel where LOG is positive** (darker side of edge).
  2. At the **pixel where LOG is negative** (brighter side of edge).
  3. At the **pixel where the absolute value of LOG is smaller** (best choice, as edge turns out closer to “true” zero-crossing).
- A **final thresholding step**, based on the slope of LOG at the found zero-crossing, may be enforced to help discard spurious edges.

Computation of LOG

Convolution has the property of differentiation, thus Gaussian and Laplacian can be computed in one step:

$$
\begin{aligned}
&\nabla^2 \tilde{I}(x,y) = \nabla^2 (I(x,y) * G(x,y)) = I(x,y) * \nabla^2 G(x,y), \\
&\nabla^2 G(x,y) = \frac{\partial^2 G}{\partial x^2}(x,y) + \frac{\partial^2 G}{\partial y^2}(x,y) = \frac{1}{2\pi\sigma^4}\left[\frac{r^2}{\sigma^2} - 2\right] e^{-\frac{r^2}{2\sigma^2}},\ r^2 = x^2 + y^2
\end{aligned}
$$

Mexican Hat filter



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

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

- Thanks to **separability of Gaussian**, computing LOG boils down to **four 1D convolutions**, which is **substantially faster** ($4d$ rather than d^2 operations per pixel):

$$\begin{aligned} I(x,y) * \nabla^2 G(x,y) &= I(x,y) * (G''(x)G(y) + G''(y)G(x)) \\ &= I(x,y) * (G''(x)G(y)) + I(x,y) * (G''(y)G(x)) \\ &= (I(x,y) * G''(x)) * G(y) + (I(x,y) * G''(y)) * G(x) \end{aligned}$$

8.5. Canny’s Edge Detector

- Canny proposed to set forth quantitative criteria to measure performance of an edge detector and then to find optimal filter w.r.t. such criteria:

  1. **Good Detection**: filter should correctly extract edges in noisy images.
  2. **Good Localization**: distance between found edge and “true” edge should be minimum.
  3. **Good Response to One Edge**: filter should detect one single edge pixel at each “true” edge.

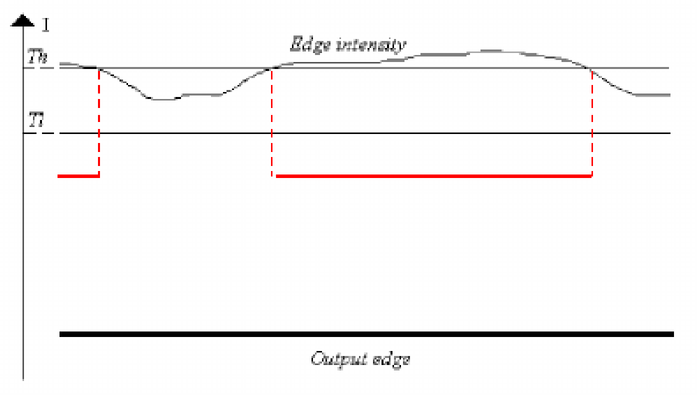
- In the 1D case of an edge modeled as noisy step, the optimal edge detection operation consists in finding local extrema of the convolution of the signal by a first order Gaussian derivative ($G'(x)$).

- In 2D, the local extrema should be sought for in the gradient’s direction, thus a Canny’s Edge Detector can be achieved by **Gaussian smoothing followed by gradient computation and NMS along gradient’s direction**.

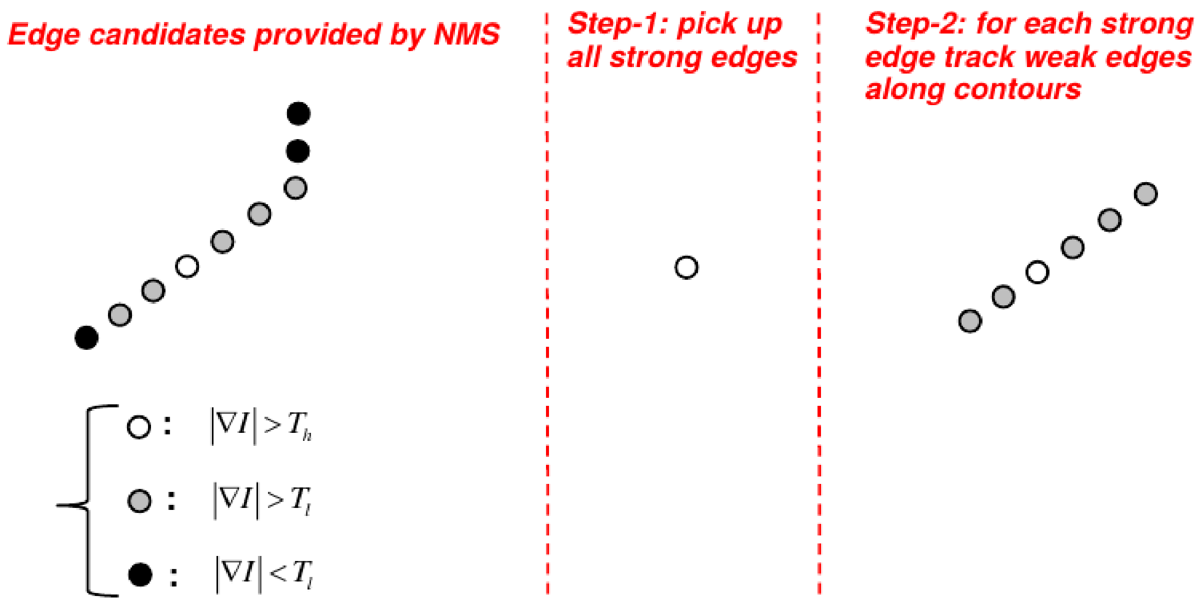
- 2D convolution by a Gaussian can be slow, so **separability** is leveraged to speed-up computation:

$$\tilde{I}_x(x,y) = \frac{\partial}{\partial x}(I(x,y) * G(x,y)) = I(x,y) * \frac{\partial G}{\partial x}(x,y)$$
$$\tilde{I}_y(x,y) = \frac{\partial}{\partial y}(I(x,y) * G(x,y)) = I(x,y) * \frac{\partial G}{\partial y}(x,y)$$
$$G(x,y) = G(x)G(y)$$
$$\Downarrow$$
$$\tilde{I}_x(x,y) = I(x,y) * (G'(x)G(y)) = (I(x,y) * G'(x)) * G(y)$$
$$\tilde{I}_y(x,y) = I(x,y) * (G'(y)G(x)) = (I(x,y) * G'(y)) * G(x)$$

- After NMS thresholding is applied, but **one threshold may be not enough**, since **magnitude can vary** along object's contour (*edge streaking*).



- Thus, an **hysteresis thresholding approach** is employed, relying on a higher (T_h) and a lower (T_l) threshold; the pixel is taken as an edge if gradient's magnitude is:
  - higher than T_h;
  - higher than T_l, and such pixel is a neighbour of an already detected edge.



- Overall flow-chart: