Computer Vision

L03: Feature Detection and Matching (Edge Detection)

Prof. Dr. Mohammed Elmogy

Mansoura University
Faculty of Computers & Information
Dept. of Information Technology

elmogy@gmail.com

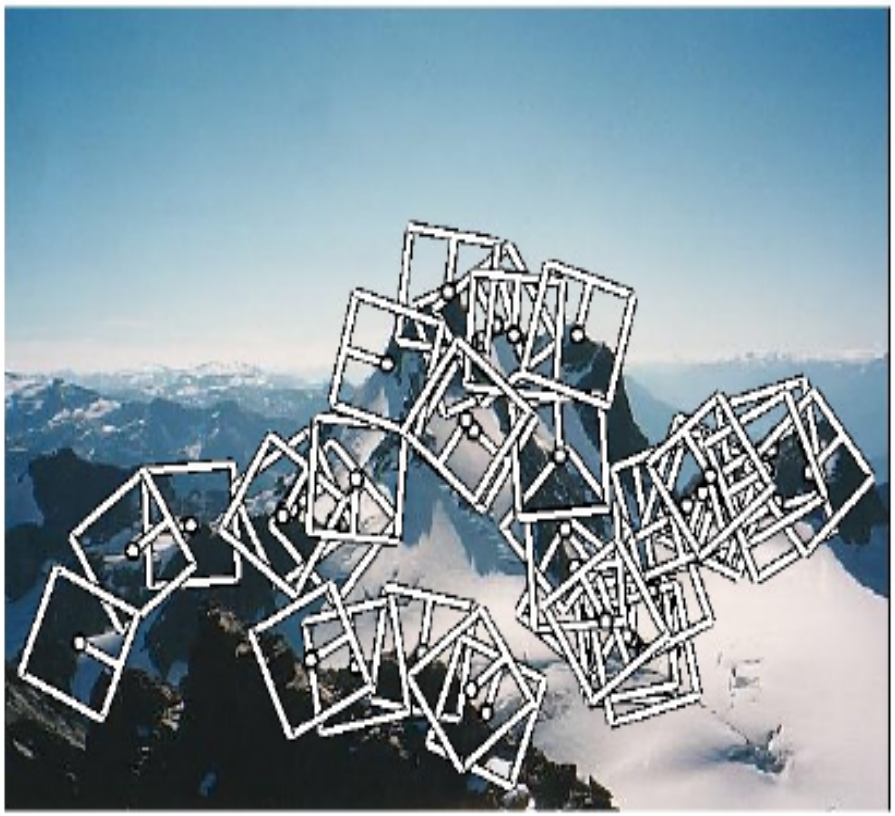
05. October 2025

Outline

Outline

A Variety of Feature Detectors and Descriptors

Point-like & Region-like Interest Operators



A Variety of Feature Detectors and Descriptors (cont'd)

Edges & Straight Lines

Where Can We Use It?

Feature detection and matching are an essential component of many computer vision applications:

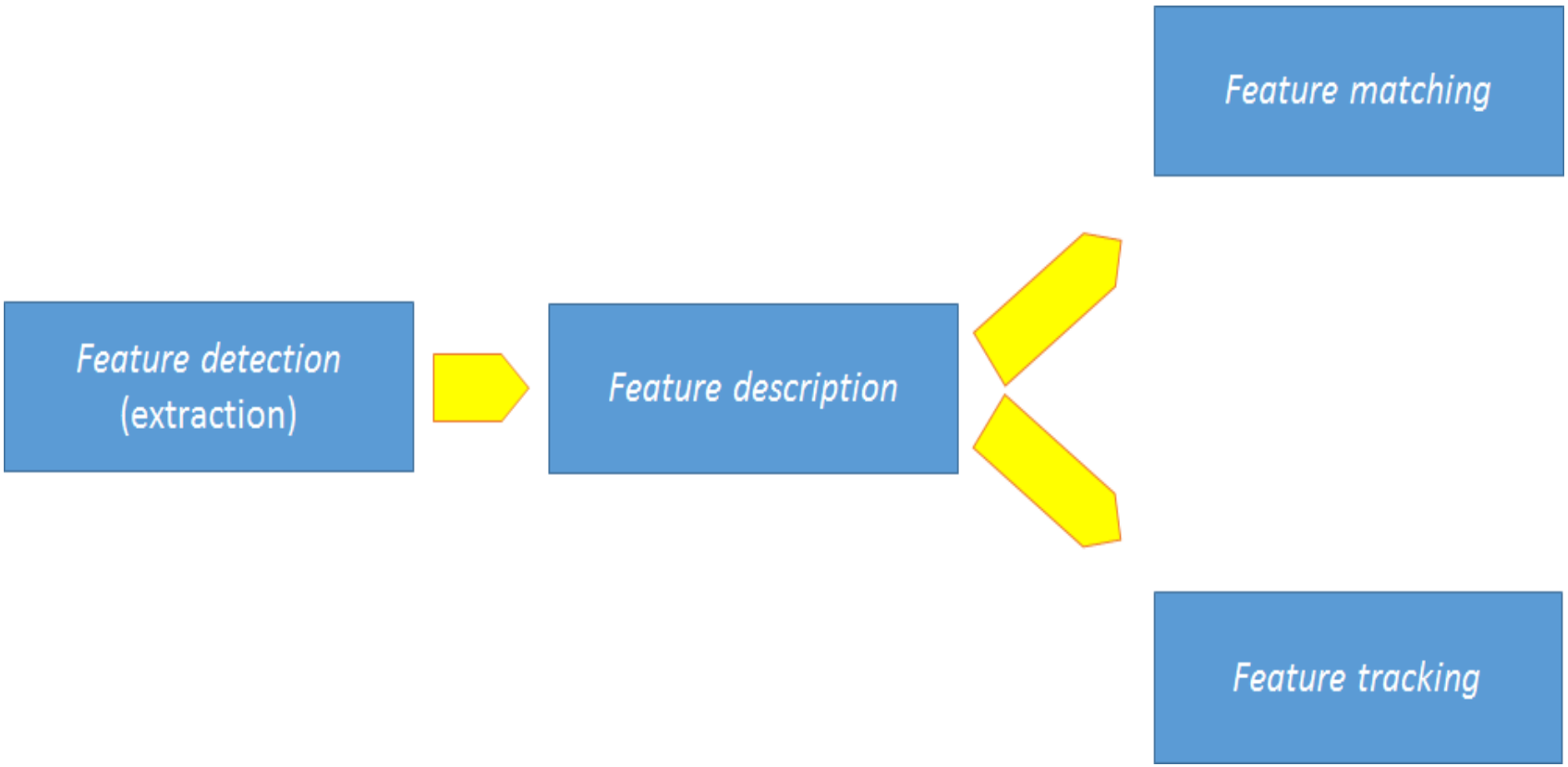
- Automate object tracking
- Point matching for computing disparity
- Stereo calibration and Estimation of fundamental matrix
- Motion based segmentation
- Recognition
- 3D object reconstruction
- Robot navigation
- Image retrieval and indexing

Main Approaches

1. Find features in one image that can be accurately tracked using a local search technique, such as correlation or least squares.
   - When images are **taken from nearby viewpoints** or in **rapid succession** (e.g., video sequences).

2. Independently detect features in all the images under consideration and then match features based on their local appearance.
   - A large amount of **motion or appearance change** is expected.
   - Stitching together **panoramas**.
   - **Establishing correspondences** in wide baseline stereo.
   - Performing **object recognition**.

Keypoint Detection and Matching Pipeline



Outline

Properties of Interest Point Detectors

- Detect all (or most) true interest points.
- No false interest points.
- Well localized.
- Robust with respect to noise.
- Efficient detection.

A key advantage of interest point (keypoints)

They permit matching even in the presence of **clutter (occlusion)**, **large scale**, and **orientation changes**.

Properties of Interest Point Detectors

- Detect all (or most) true interest points.
- No false interest points.
- Well localized.
- Robust with respect to noise.
- Efficient detection.

A key advantage of interest point (keypoints)

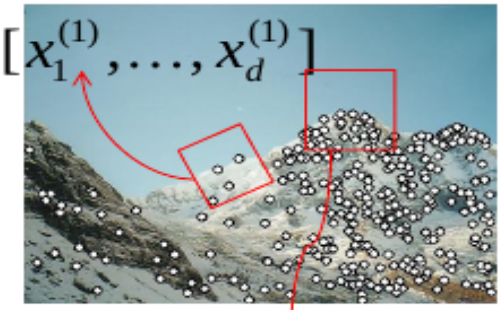
They permit matching even in the presence of **clutter (occlusion)**, **large scale**, and **orientation changes**.

Local Features: Main Components

1) **Detection:** Identify the interest points

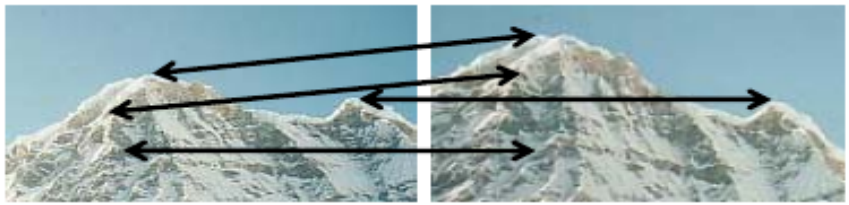
2) **Description** :Extract feature vector descriptor surrounding each interest point.

$$\mathbf{x}_1 = [x_1^{(1)}, \ldots, x_d^{(1)}]$$



$$\mathbf{x}_2 = [x_1^{(2)}, \ldots, x_d^{(2)}]$$

3) **Matching:** Determine correspondence between descriptors in two views



Goal: Interest Operator Repeatability

- We want to detect (at least some of) the same points in both images.

No chance to find true matches!

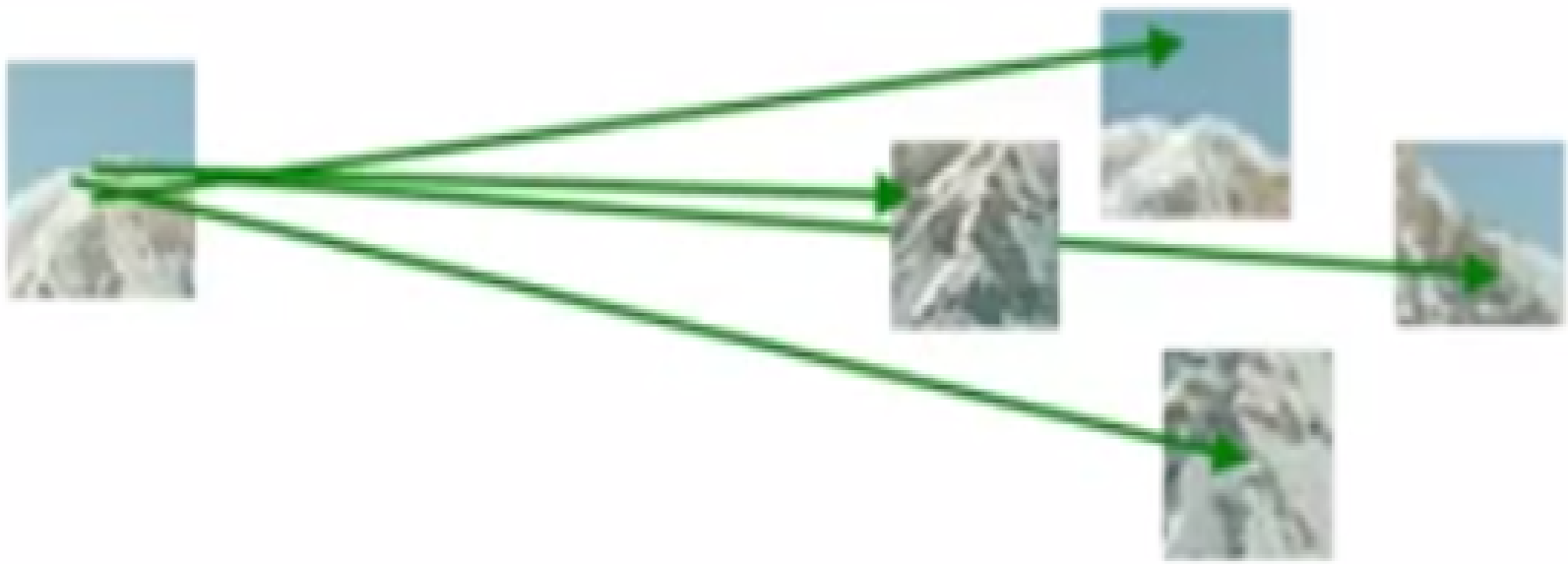
- Yet we have to be able to run the detection procedure *independently* per image.

Goal: Descriptor Distinctiveness

We want to be able to reliably determine which point goes with which.

Goal: Descriptor Distinctiveness

- We want to be able to reliably determine which point goes with which.
- Must provide some invariance to geometric and photometric differences between the two views.



Outline

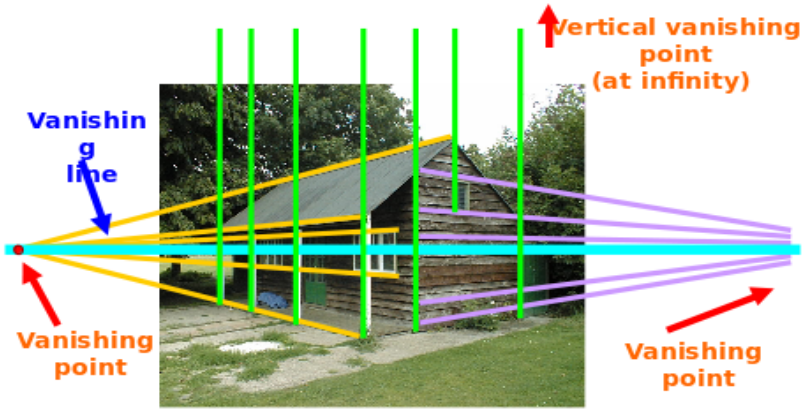
Edge Detection

- **Goal: Identify sudden changes** (discontinuities) in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges.
  - More compact than pixels.

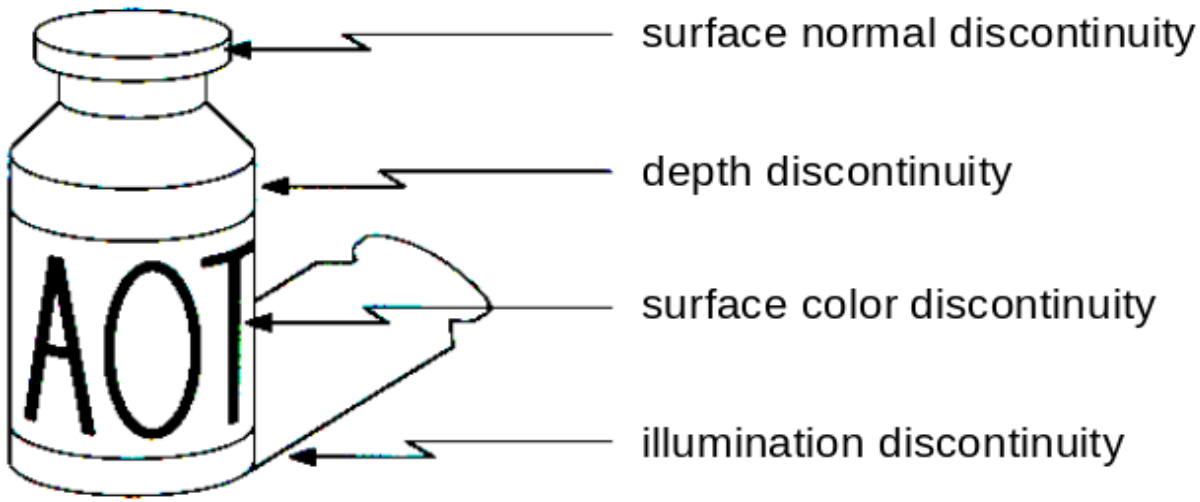
Why Do We Care About Edges?

- Extract information, recognize objects

- Recover geometry and viewpoint



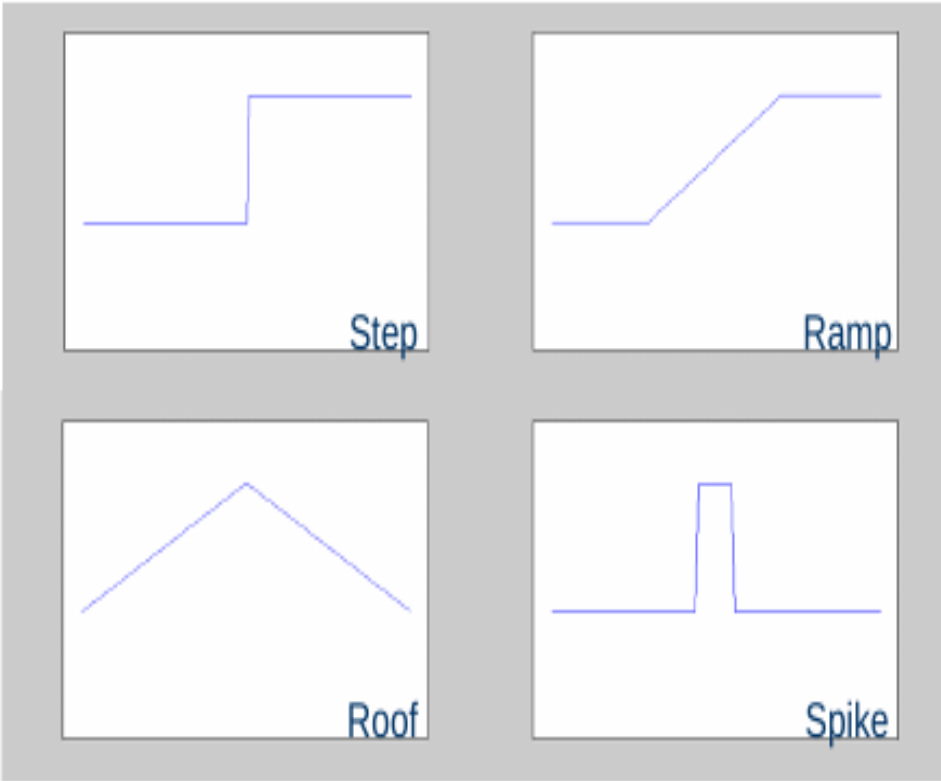
Origin of Edges



- Edges are caused by a variety of factors

What is an Edge?

- Discontinuity of intensities in the image
- Edge models
  - Step
  - Roof
  - Ramp
  - Spike



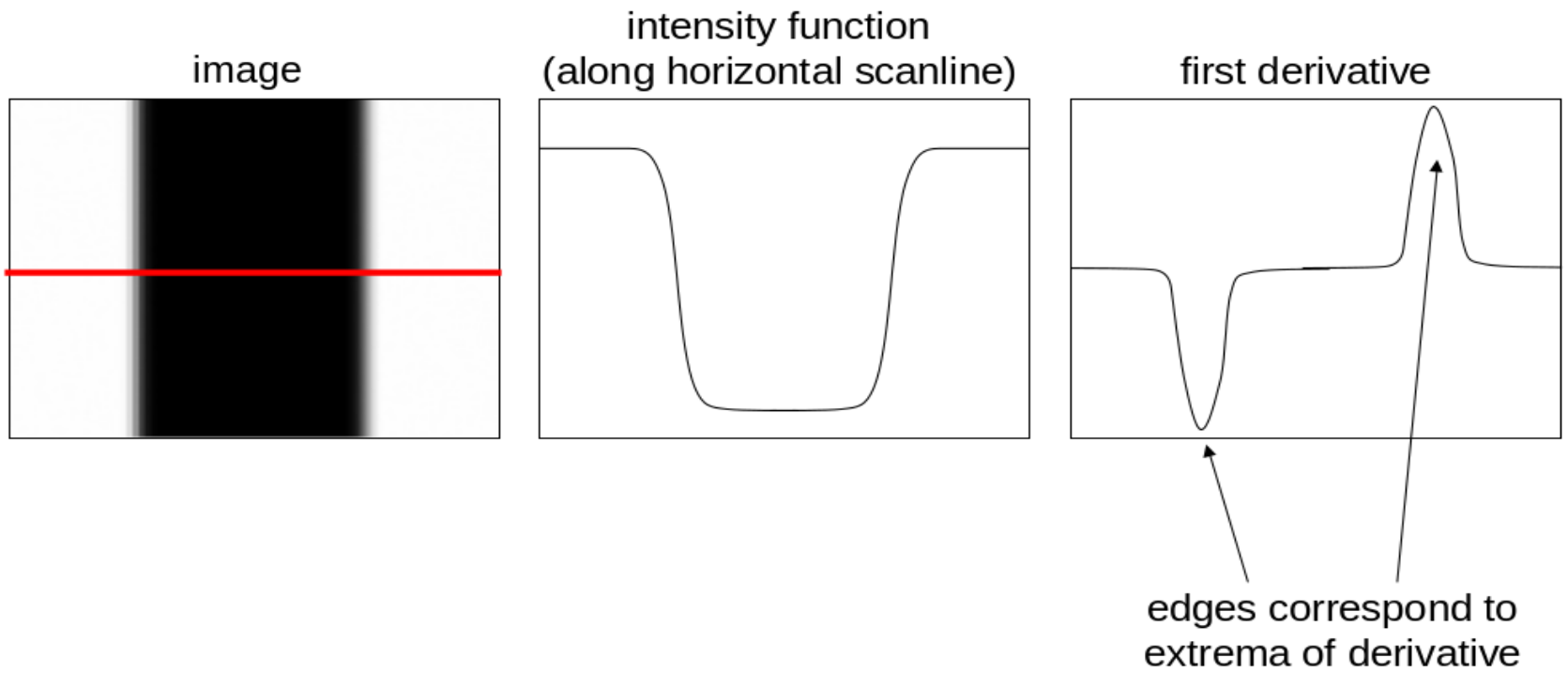
General Notes

- Edge points are far more plentiful and often carry important semantic associations.
- Qualitatively, edges occur at **boundaries between regions** of different color, intensity, or texture.
- A reasonable edge detection approach is to **define an edge as a location of rapid intensity variation**.

Outline

Characterizing edges

- An edge is a place of rapid change in the image intensity function



Detecting Discontinuities

- Image derivatives

$$\frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon) - f(x)}{\varepsilon}\right) \Rightarrow \frac{\partial f}{\partial x} \approx \frac{f(x_{n+1}) - f(x)}{\Delta x}$$

- Convolve image with derivative filters

Backward difference	[-1 1]
Forward difference	[1 -1]
Central difference	[-1 0 1]

Derivative in Two-Dimensions

- Definition

$$\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0}\left(\frac{f(x+\varepsilon,y)-f(x,y)}{\varepsilon}\right) \qquad \frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0}\left(\frac{f(x,y+\varepsilon)-f(x,y)}{\varepsilon}\right)$$

- Approximation

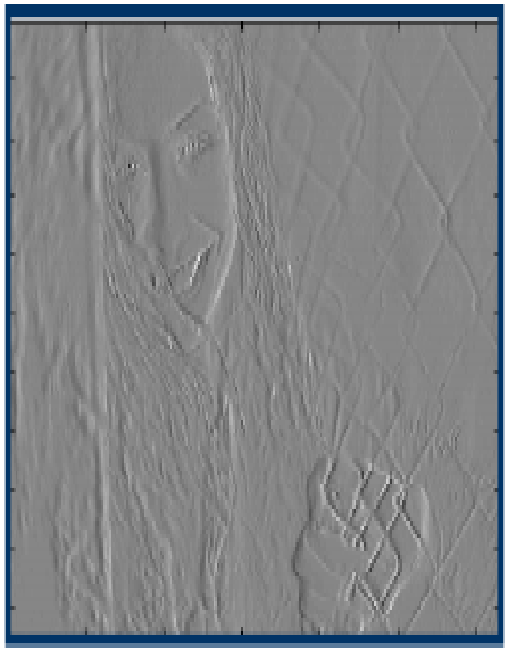
$$\frac{\partial f(x,y)}{\partial x} \approx \frac{f(x_{n+1},y_m)-f(x_n,y_m)}{\Delta x} \qquad \frac{\partial f(x,y)}{\partial y} \approx \frac{f(x_n,y_{m+1})-f(x_n,y_m)}{\Delta x}$$

- Convolution kernels

$$f_x = \begin{bmatrix} 1 & -1 \end{bmatrix} \qquad f_y = \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Image Derivatives

Image I



$$I_x = I * \begin{bmatrix} 1 & -1 \end{bmatrix}$$

$$I_y = I * \begin{bmatrix} 1 \\ -1 \end{bmatrix}$$

Outline

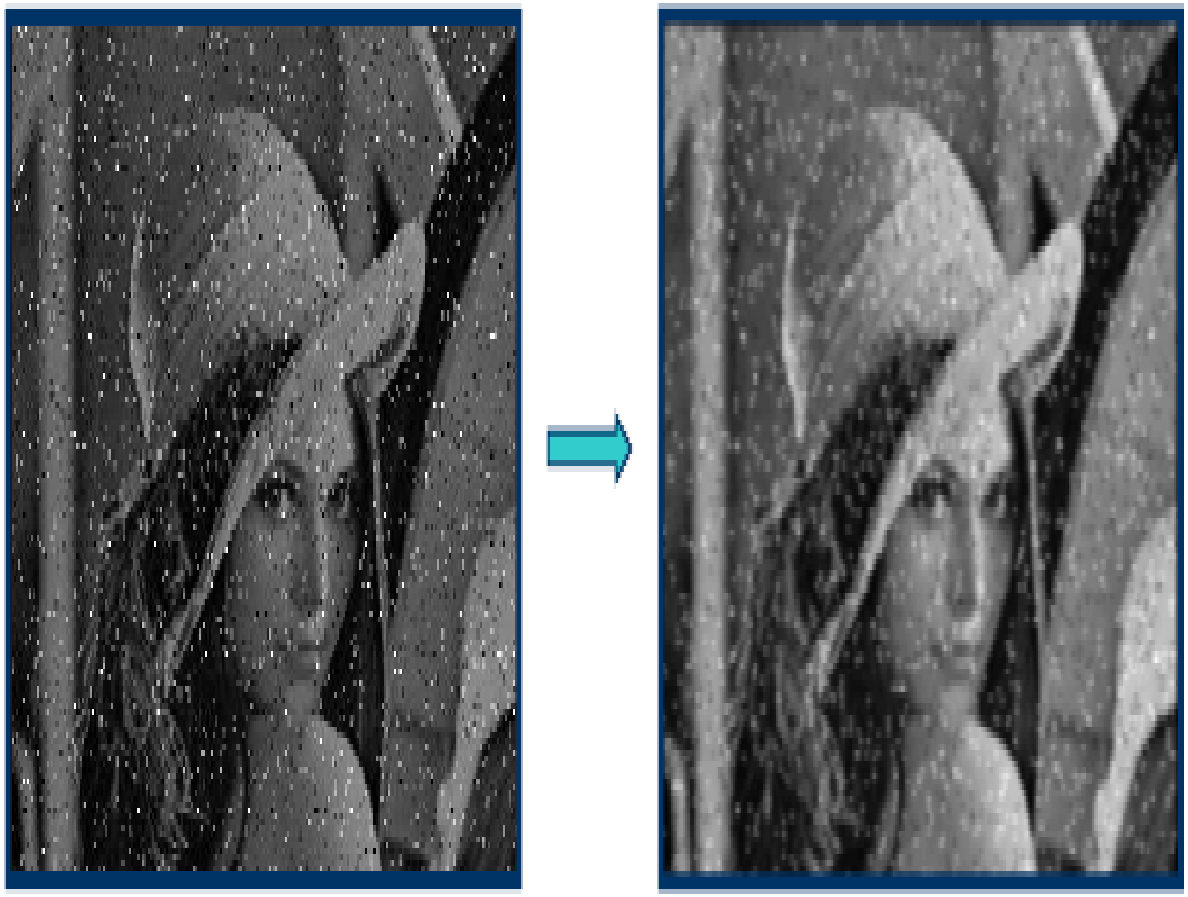
Derivatives and Noise

- Strongly affected by noise:
  - **Obvious reason**: image noise results in pixels that look very different from their neighbors.
  - The larger the noise is the stronger the response.
- What is to be done?
  - Neighboring pixels look alike.
  - Pixel along an edge look alike.
  - Image smoothing should help. Force pixels different to their neighbors (possibly noise) to look like neighbors.

Image Smoothing

- Expect pixels to "be like" their neighbors $\Longrightarrow$ **Relatively few reflectance changes**.
- Generally expect noise to be independent from pixel to pixel $\Longrightarrow$ **Smoothing suppresses noise**.

Gaussian Smoothing



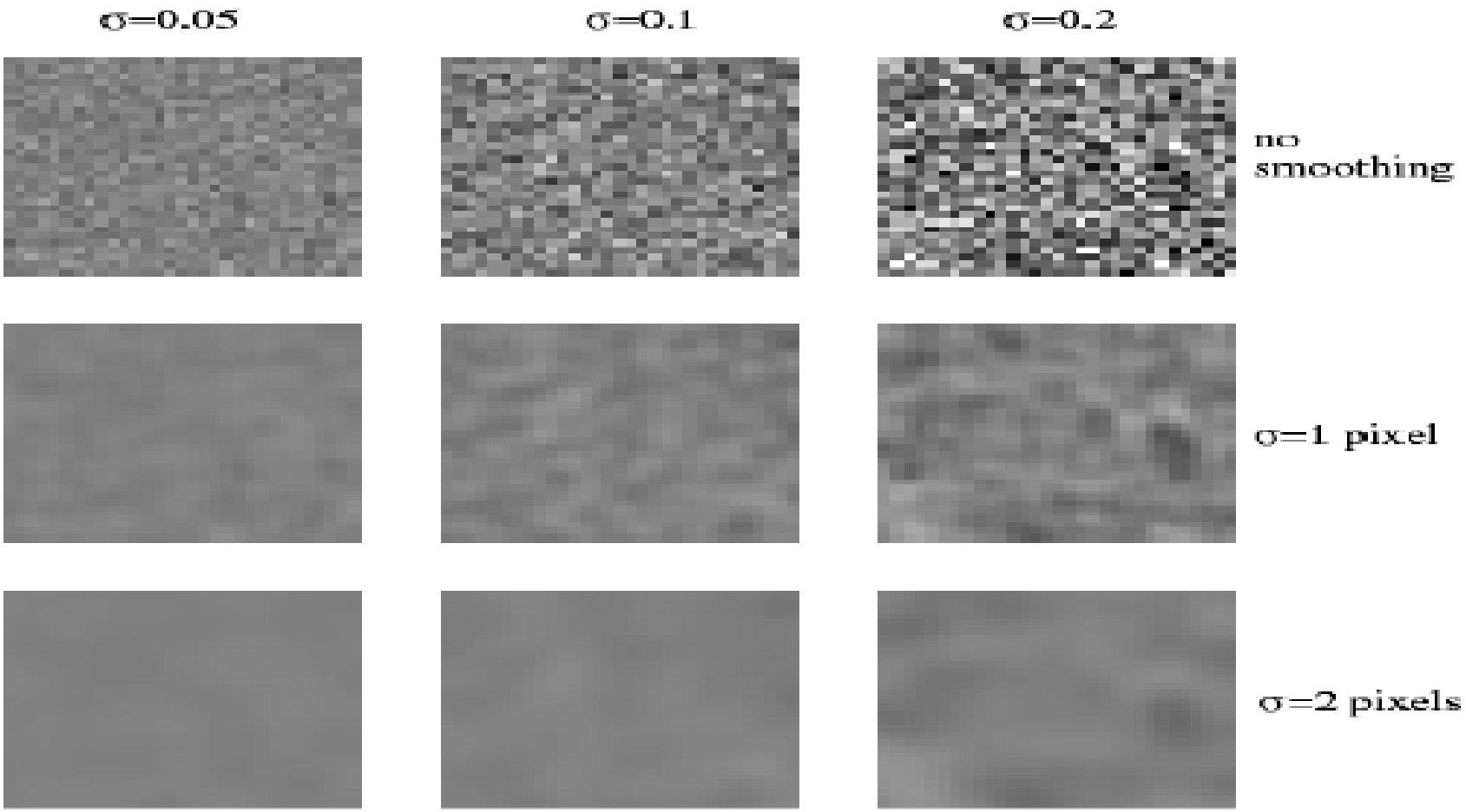
$$g(x,y) = e^{\frac{-(x^2+y^2)}{2\sigma^2}}$$

Gaussian Smoothing (cont'd)

Scale of Gaussian σ

- As σ increases, more pixels are involved in average.
- As σ increases, image is more blurred.
- As σ increases, noise is more effectively suppressed.

Gaussian Smoothing (Examples)



Outline

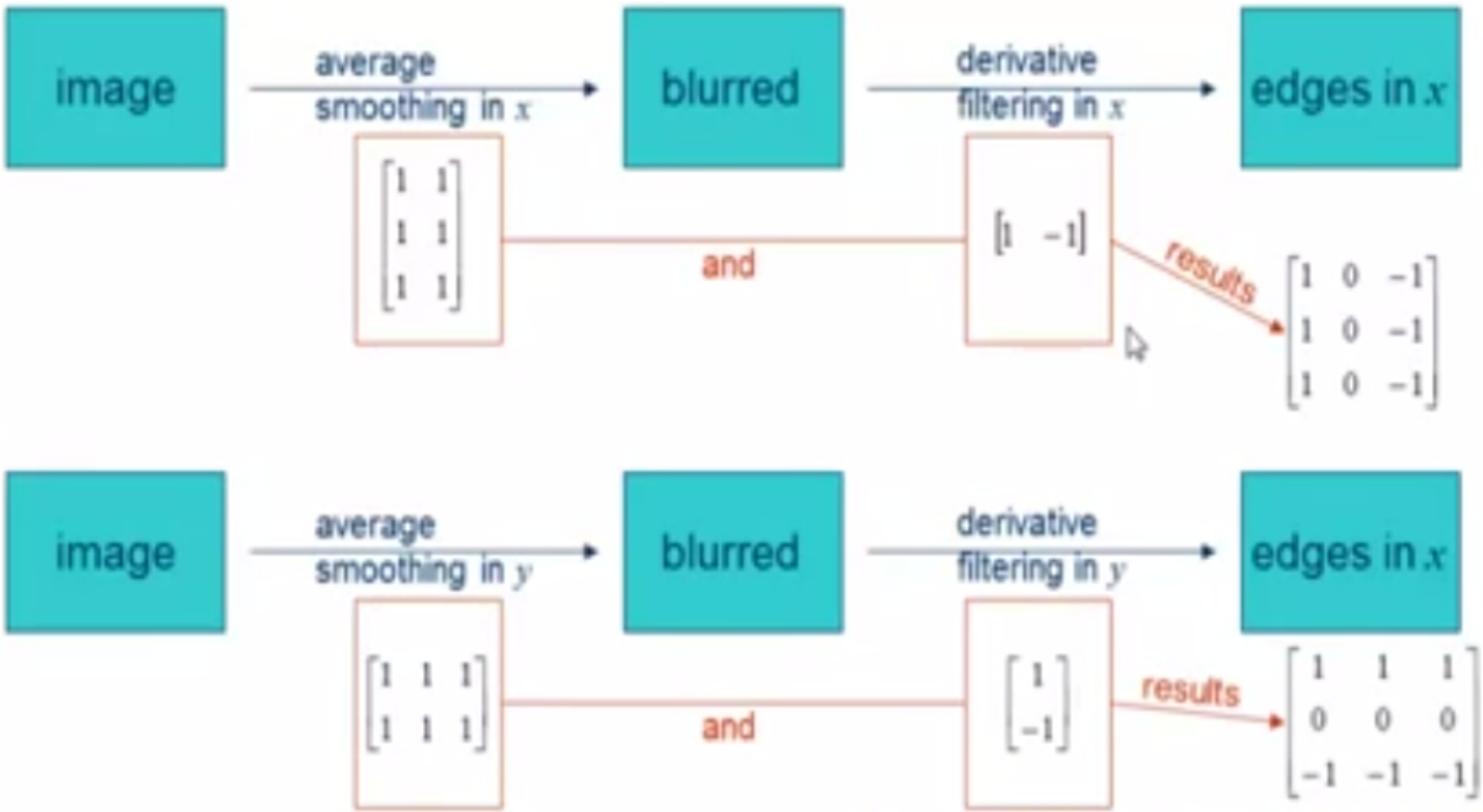
Edge Detectors

- Gradient operators:
  - Prewitt
  - Sobel
- Laplacian of Gaussian (Marr-Hildreth)
- Gradient of Gaussian (Canny)

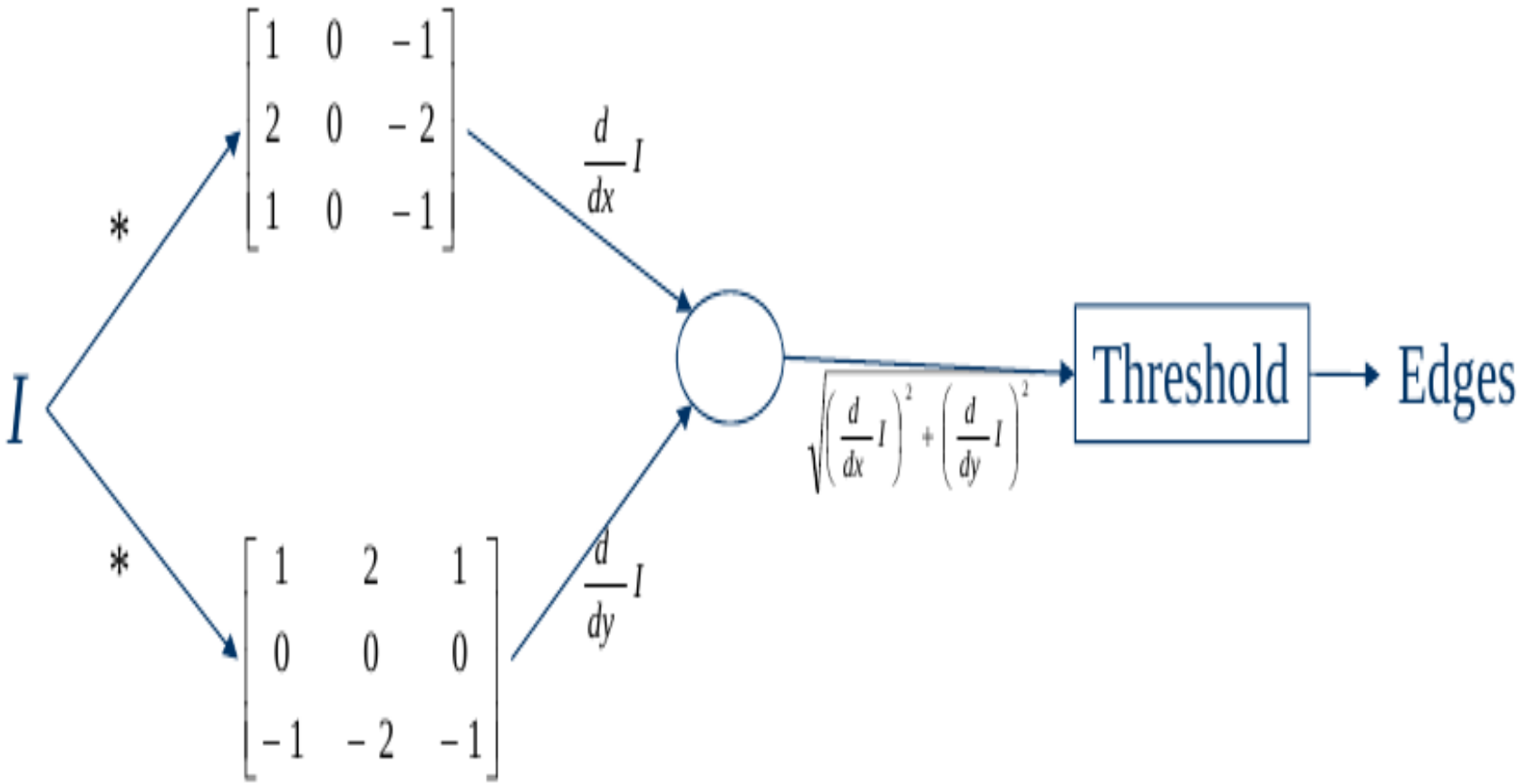
Prewitt and Sobel Edge Detector

1. Compute derivatives $\Longrightarrow$ in x and y directions.
2. Find gradient magnitude.
3. Threshold gradient magnitude.

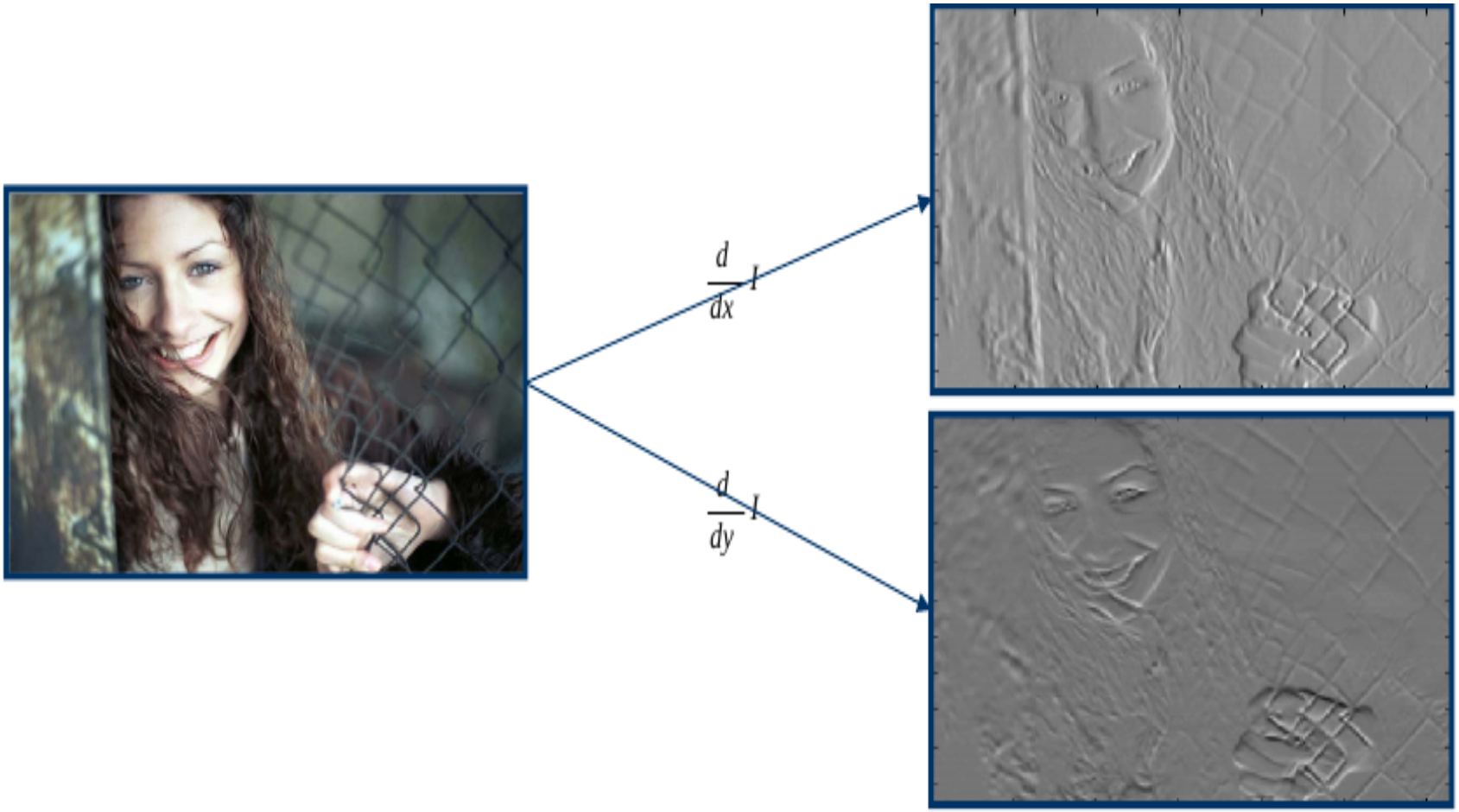
Prewitt Edge Detector



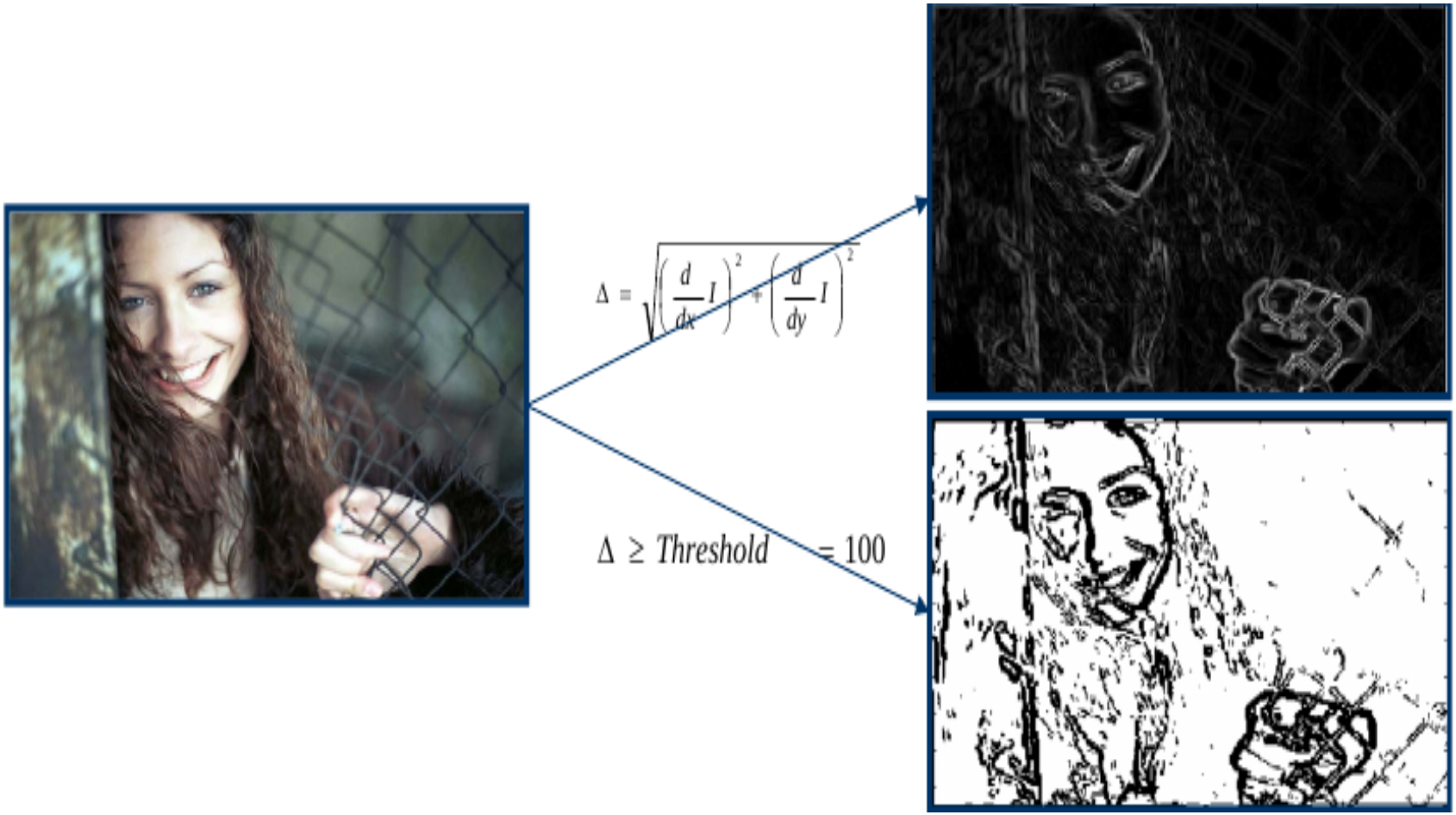
Sobel Edge Detector



Sobel Edge Detector (cont'd)



Sobel Edge Detector (cont'd)



Marr Hildreth Edge Detector

1. Smooth image by Gaussian filter $\Longrightarrow$ **S**.

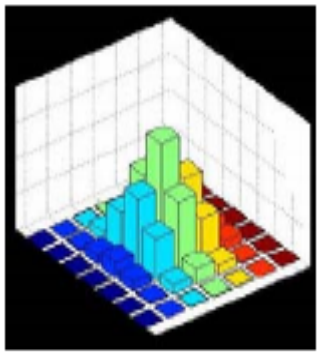
2. Apply Laplacian to S: Used in mechanics, electromagnetics, wave theory, quantum mechanics and Laplace equation.

3. Find zero crossings:
   - Scan along each row, record an edge point at the location of zero-crossing.
   - Repeat above step along each column.

Marr Hildreth Edge Detector (cont'd)

- **Gaussian smoothing**

$$\underbrace{S}_{\text{smoothed image}} = \underbrace{g}_{\text{Gaussian filter}} * \underbrace{I}_{\text{image}} \qquad g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$



- **Find Laplacian**

$$\Delta^2 S = \overbrace{\frac{\partial^2}{\partial x^2} S}^{\text{second order derivative in } x} + \overbrace{\frac{\partial^2}{\partial y^2} S}^{\text{second order derivative in } y}$$

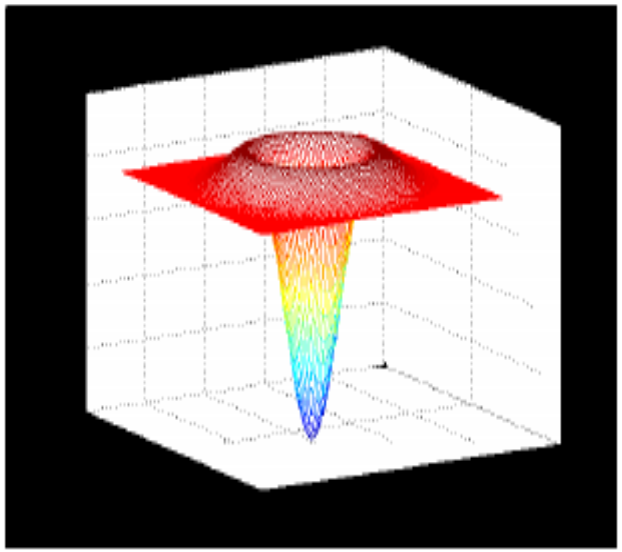
- ∇ is used for gradient (derivative)
- Δ is used for Laplacian

Marr Hildreth Edge Detector (cont'd)

- Deriving the Laplacian of Gaussian (LoG)

$$\Delta^2 S = \Delta^2 (g * I) = (\Delta^2 g) * I$$

$$\Delta^2 g = -\frac{1}{\sqrt{2\pi}\sigma^3}\left(2 - \frac{x^2 + y^2}{\sigma^2}\right) e^{-\frac{x^2+y^2}{2\sigma^2}}$$



Finding Zero Crossings

- Four cases of zero-crossings :
  - {+,-}
  - {+,0,-}
  - {-,+}
  - {-,0,+}
- Slope of zero-crossing {a, -b} is |a+b|.
- To mark an edge
  - compute slope of zero-crossing
  - Apply a threshold to slope

Finding Zero Crossings (cont'd)

- An easy way to detect and represent zero crossings is to look for **adjacent pixel locations** x_i and x_j where the sign changes value, i.e., $[S(x_i) > 0] \neq [S(x_j) > 0]$.
- The sub-pixel location of this crossing can be obtained by computing the "**x-intercept**" of the "**line**" connecting $S(x_i)$ and $S(x_j)$,

$$x_z = \frac{x_i S(x_j) - x_j S(x_i)}{S(x_j) - S(x_i)}$$

- The **orientation** and **strength** of such edgels can be obtained by linearly interpolating the gradient values computed on the original pixel grid.

On the Separability of LoG

- Similar to separability of Gaussian filter
  - Two-dimensional Gaussian can be separated into 2 one-dimensional Gaussians

$$h(x,y) = I(x,y) * g(x,y) \qquad n^2 \text{ multiplications}$$

$$h(x,y) = \left(I(x,y) * g_1(x)\right) * g_2(y) \qquad 2n \text{ multiplications}$$

$$g(x) = e^{-\left(\frac{x^2}{2\sigma^2}\right)}$$

$$g_1 = g(x) = \begin{bmatrix} .011 & .13 & .6 & 1 & .6 & .13 & .011 \end{bmatrix}$$

$$g_2 = g(y) = \begin{bmatrix} .011 \\ .13 \\ .6 \\ 1 \\ .6 \\ .13 \\ .011 \end{bmatrix}$$

On the Separability of LoG (cont'd)

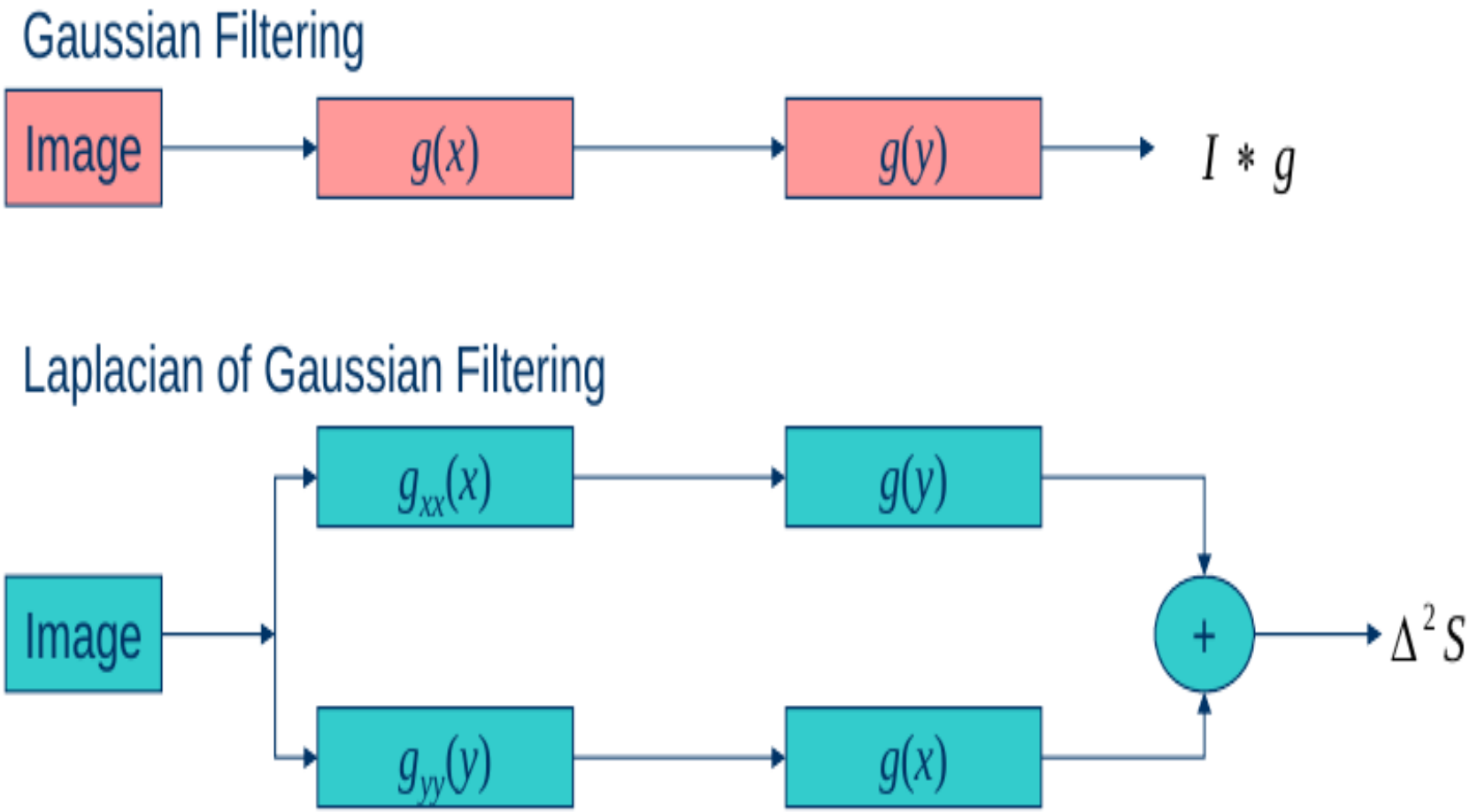
$$\Delta^2 S = \Delta^2 (g * I) = (\Delta^2 g) * I = I * (\Delta^2 g)$$

Requires n^2 multiplications

$$\Delta^2 S = (I * g_{xx}(x)) * g(y) + (I * g_{yy}(y)) * g(x)$$

Requires $4n$ multiplications

Seperability



Algorithm

- Compute LoG
  - Use 2D filter
  - Use 4 1D filters

$$\Delta^2 g(x,y)$$

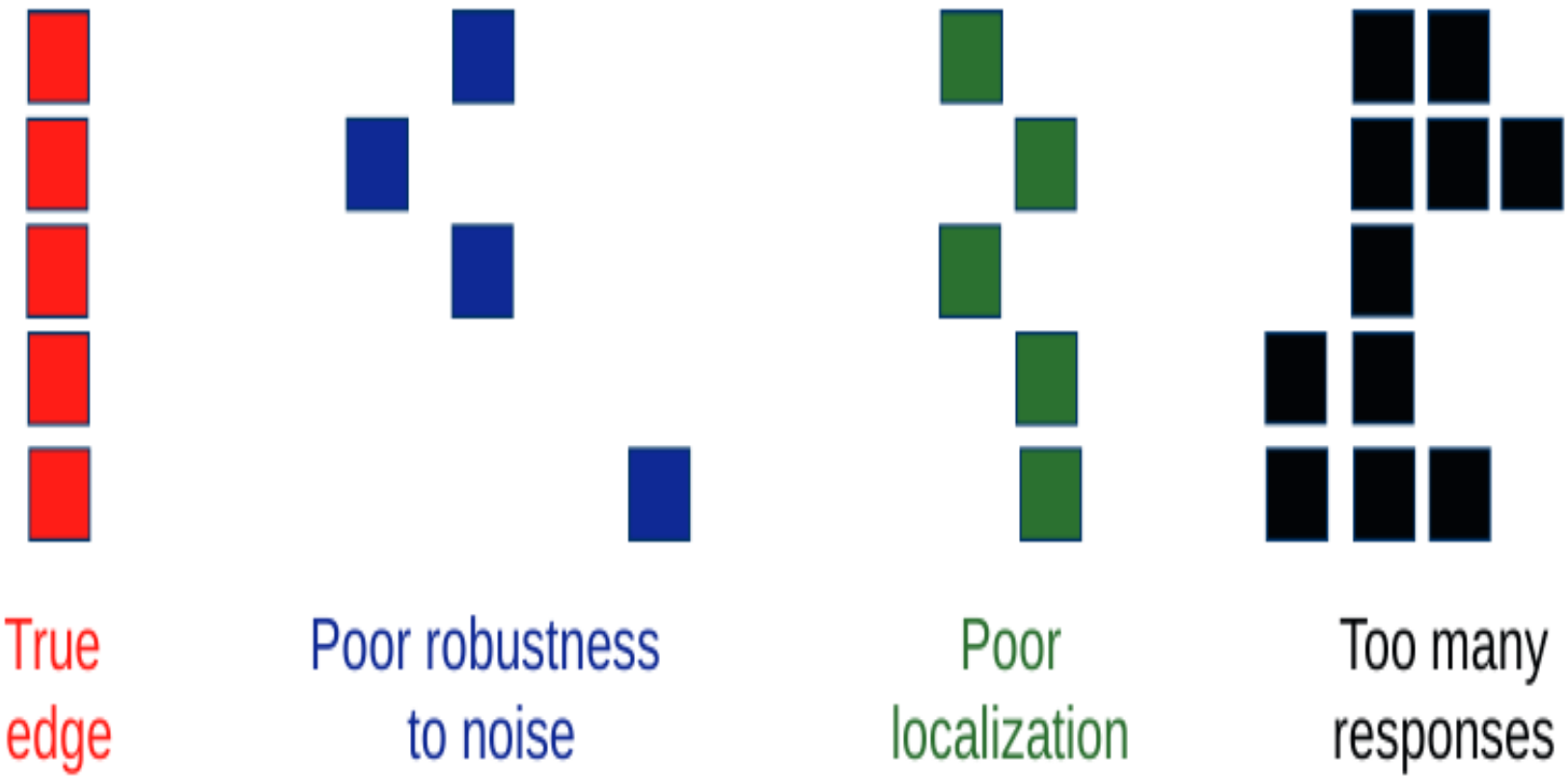
$$g(x),\ g_{xx}(x),\ g(y),\ g_{yy}(y)$$

- Find zero-crossings from each row
- Find slope of zero-crossings
- Apply threshold to slope and mark edges

Quality of an Edge

- Robust to noise
- Localization
- Too many or too less responses

Quality of an Edge



Canny Edge Detector

- **Criterion 1**: Good Detection: The optimal detector must minimize the probability of false positives as well as false negatives.
- **Criterion 2**: Good Localization: The edges detected must be as close as possible to the true edges.
- Single Response Constraint: The detector must return one point only for each edge point.

Canny Edge Detector Steps

1. Smooth image with Gaussian filter.
2. Compute derivative of filtered image.
3. Find magnitude and orientation of gradient.
4. Apply “Non-maximum Suppression”.
5. Apply “Hysteresis Threshold”.

Canny Edge Detector: First Two Steps

- **Smoothing**

$$S = I * g(x,y) = g(x,y) * I$$

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

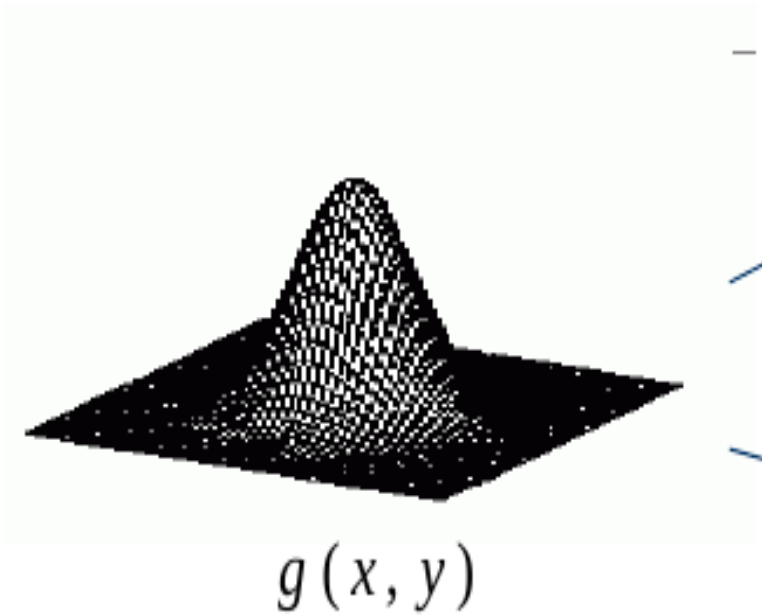
- **Derivative**

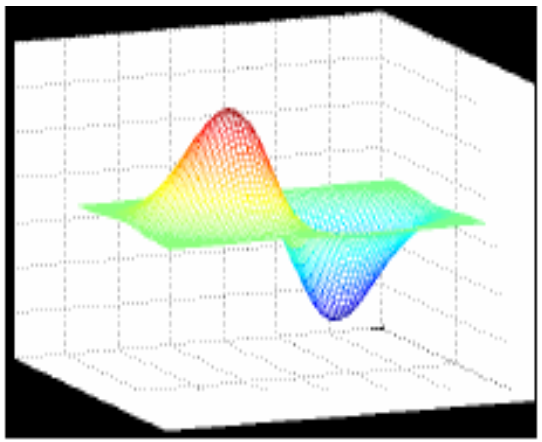
$$\nabla S = \nabla (g * I) = (\nabla g) * I$$

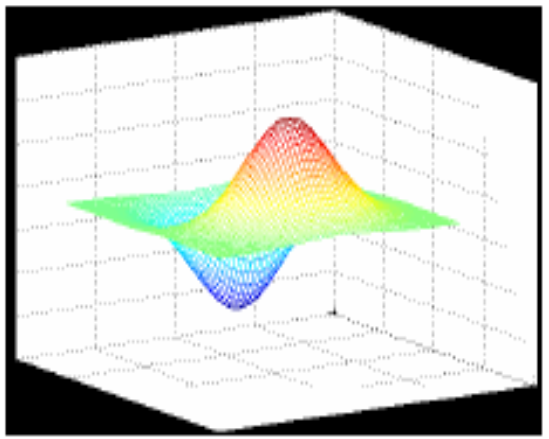
$$\nabla S = \begin{bmatrix} g_x \\ g_y \end{bmatrix} * I = \begin{bmatrix} g_x * I \\ g_y * I \end{bmatrix}$$

$$\nabla g = \begin{bmatrix} \frac{\partial g}{\partial x} \\ \frac{\partial g}{\partial y} \end{bmatrix} = \begin{bmatrix} g_x \\ g_y \end{bmatrix}$$

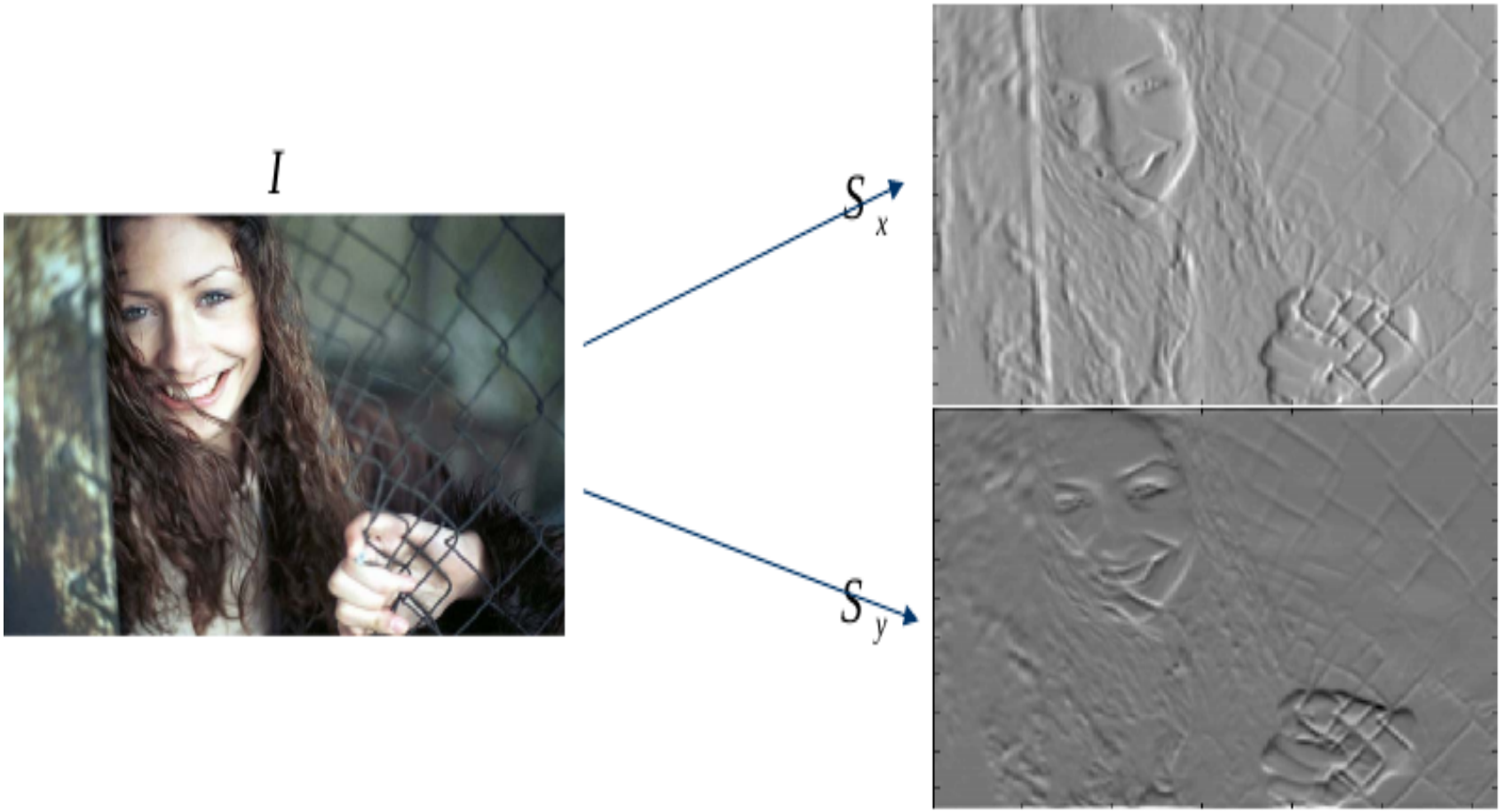
Canny Edge Detector Derivative of Gaussian







Canny Edge Detector: First Two Steps (cont'd)



Canny Edge: Third Step Detector

- Gradient magnitude and gradient direction

(S_x, S_y) Gradient Vector

$$\text{magnitude} = \sqrt{(S_x^2 + S_y^2)}$$

$$\text{direction} = \theta = \tan^{-1}\frac{S_y}{S_x}$$

image

gradient magnitude

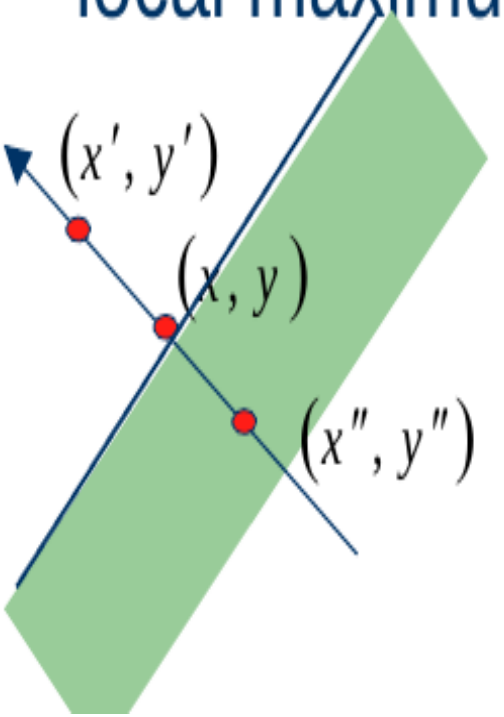
Canny Edge Detector: Fourth Step

- Non maximum suppression

We wish to mark points along the curve where the **magnitude is largest**. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next one?

Canny Edge Detector: Non-Maximum Suppression

- Suppress the pixels in $|\nabla S|$ which are not local maximum



$$M(x,y) = \begin{cases} |\nabla S|(x,y) & \text{if } |\nabla S|(x,y) > |\Delta S|(x',y') \\ & \& \ |\Delta S|(x,y) > |\Delta S|(x'',y'') \\ 0 & \text{otherwise} \end{cases}$$

x' and **x"** are the neighbors of **x** along normal direction to an edge

Canny Edge Detector: Non-Maximum Suppression (cont'd)

$$|\Delta S| = \sqrt{S_x^2 + S_y^2}$$

M

For visual ization

$M \geq Threshold = 25$

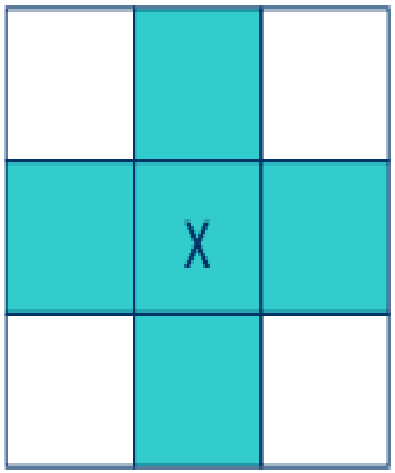
Canny Edge Detector: Hysteresis Thresholding

If the gradient at a pixel is:

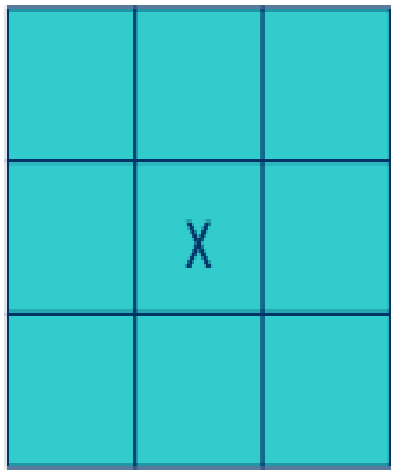
- above High, declare it as an edge pixel.
- below Low, declare it as a non-edge-pixel.
- between low and high: Consider its neighbors iteratively then declare it an "edge pixel" if it is connected to an 'edge pixel' directly or via pixels between "low" and "high".

Canny Edge Detector: Hysteresis Thresholding (cont'd)

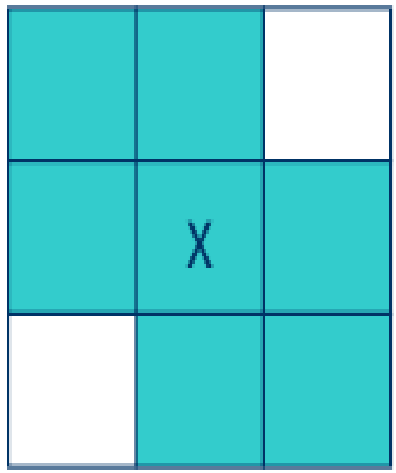
- Connectedness



4 connected

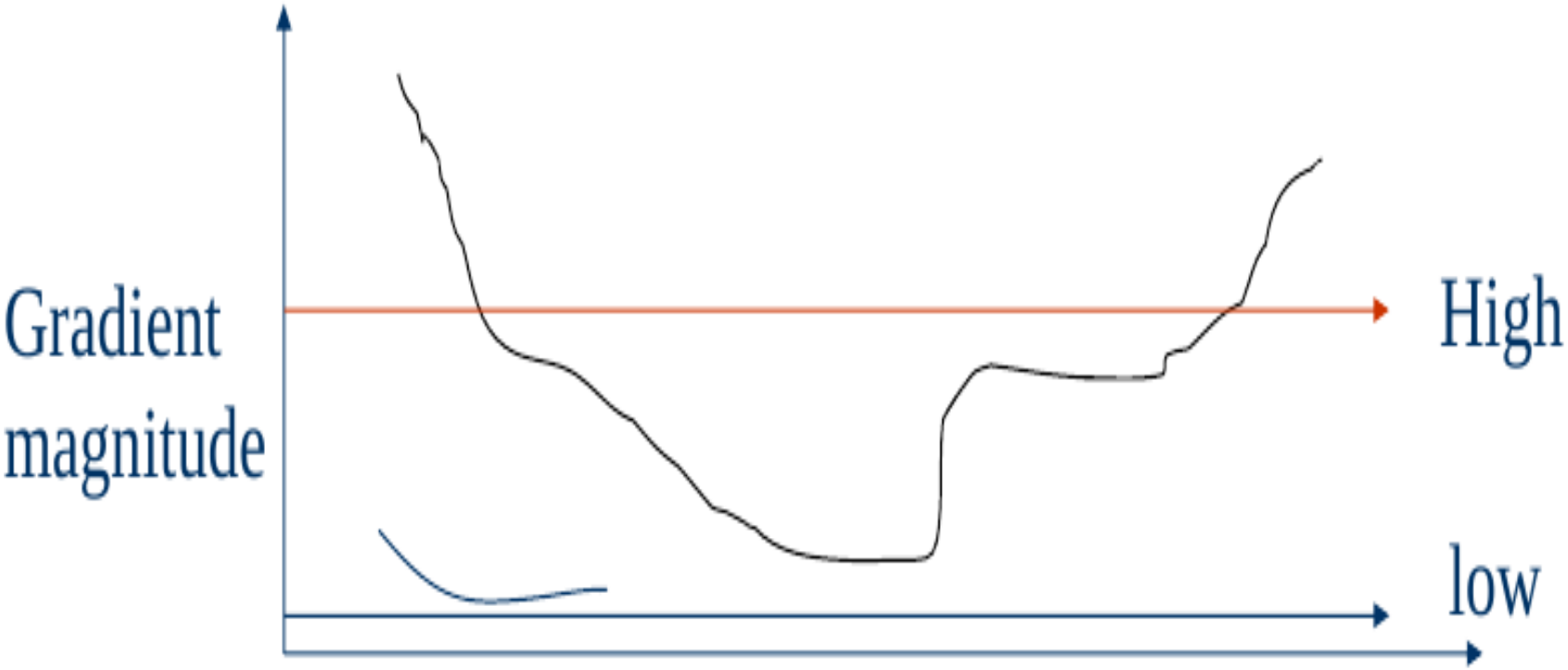


8 connected



6 connected

Canny Edge Detector: Hysteresis Thresholding (cont'd)



Canny Edge Detector: Hysteresis Thresholding (cont'd)

Scan the image from left to right, top-bottom.

- The gradient magnitude at a pixel is above a high threshold declare that as an edge point.
- Then recursively consider the neighbors of this pixel. If the gradient magnitude is above the low threshold declare that as an edge pixel.

Canny Edge Detector: Hysteresis Thresholding (cont'd)

M

regular

$M \geq 25$

Hysteresis

$High = 35$

$Low = 15$

Outline

Steerable Filters as Edge Detector

- In applications where the **accuracy of the edge orientation is more important**, higher-order steerable filters can be used.
- They are **more selective for more elongated edges** and also have the **possibility of better modeling curve intersections** because they can represent multiple orientations at the same pixel.
- Their disadvantage is that they are more expensive to compute and the directional derivative of the edge strength does not have a simple closed form solution.

Outline

Color Edge Detection

1. One simple approach is to combine the outputs of grayscale detectors run on each color band separately.
   - If we simply sum up the gradients in each of the color bands, the signed gradients may actually cancel each other.

2. **Detect edges independently** in each band and then take the union of these.
   - This might lead to thickened or doubled edges that are hard to link.

Color Edge Detection (cont'd)

3. Compute the oriented energy in each band using a second-order steerable filter, and then **sum up the orientation-weighted energies** and find their **joint best orientation**.
   - The directional derivative of this energy may not have a closed form solution , so a simple zero crossing-based strategy cannot be used.

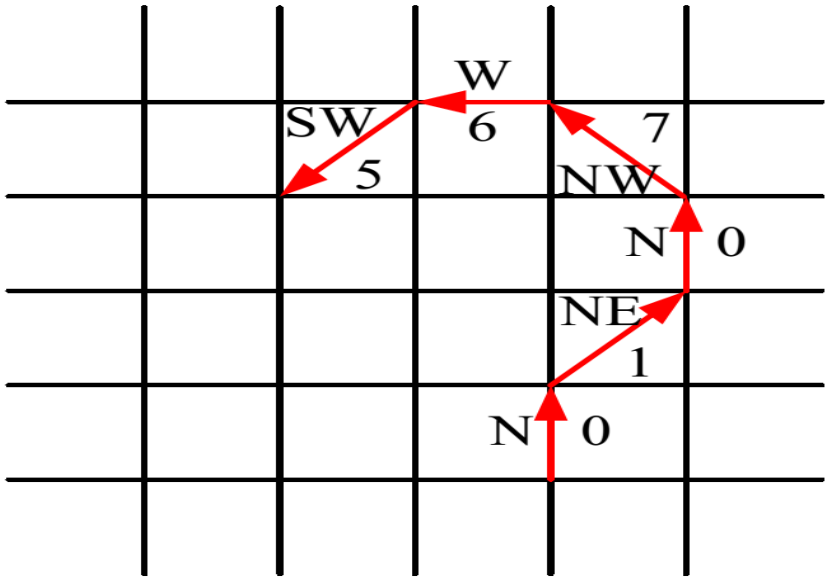
4. Estimate local color statistics in regions around each pixel.
   - This has the advantage that more sophisticated techniques (e.g., 3D color histograms) can be used to compare regional statistics and that additional measures, such as texture, can also be considered.

Outline

Edge Linking

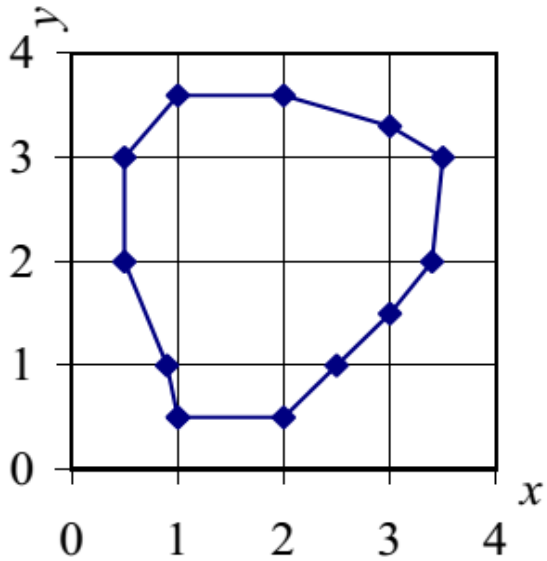
- Isolated edges become even more useful when linked into continuous contours.
- Linking the edgels into chains involves picking up an unlinked edgel and following its neighbors in both directions.
- Either a sorted list of edgels or a 2D array can be used to accelerate the neighbor finding.

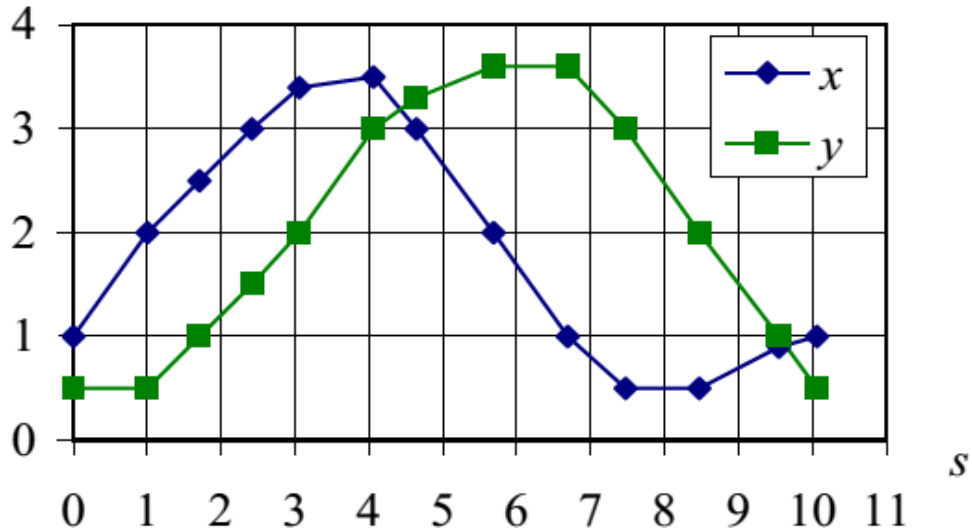
Chain Code



It encodes a list of connected points lying on a grid using a three-bit code corresponding to the eight cardinal directions (N, NE, E, SE, S, SW, W, NW) between a point and its successor.

Arc-length Parameterization of a Contour





It makes matching and processing (e.g., smoothing) operations **much easier**.

References

1. Richard Szeliski, Computer Vision: Algorithms and Applications, 2nd Ed., Springer-Verlag London, 2020.

2. Mubarak Shah CV Course: Lecture 3 (https://www.youtube.com/watch?v=7mEiTU-XgCo&list=PLd3hlSJsX_ImKP68wfKZJVIPTd8Ie5u-9&index=3)

3. David Forsyth and Jean Ponce: Computer Vision: A Modern Approach, 2nd Ed., Pearson Education, Inc., 2012.