CMP362/CMPN446: Image Processing and Computer Vision

Lecture 04: Image Preprocessing
Local Preprocessing - Edge Detection

Mayada Hadhoud
Computer Engineering Department
Cairo University

Agenda

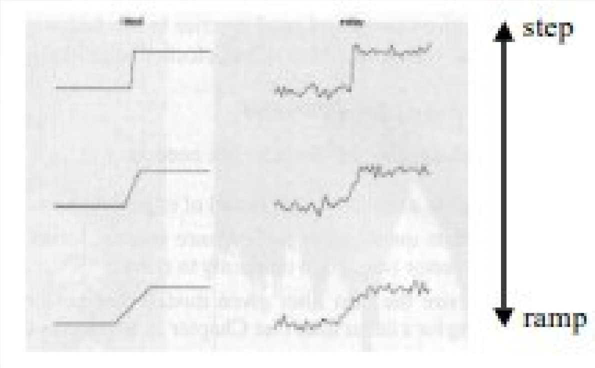
- What is Edge Detection?
- Types of Edges
- Gradient Operators
- Operators that approximate 1^{st} derivative
- Operators based on zero crossing of the 2^{nd} derivative
- Canny Edge Detector

What is Edge Detection?

"The ability to measure gray-level transitions in a meaningful way."

*(R.C. Gonzales & R. E. Woods – **Digital Image Processing**, 2nd Edition, Prentice-Hall, 2001)*

Types of Edges



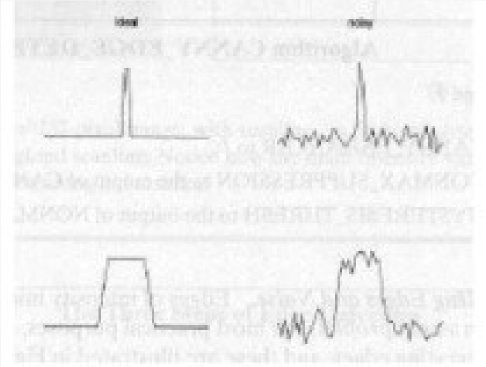
Step Edge :

abrupt change from one value to a different value in the opposite side

Ramp Edge:

a step edge where the intensity change is not instantaneous but occurs over a finite distance.

Types of Edges

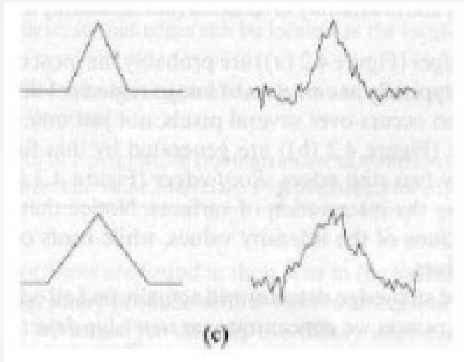


Ridge Edge :

- the image intensity abruptly changes value but then returns
  to the starting value within some short distance

– generated usually by **lines**

Types of Edges



Roof Edge :

- a ridge edge where the intensity change is not instantaneous but occurs over a finite distance

– generated usually by **the intersection of surfaces**

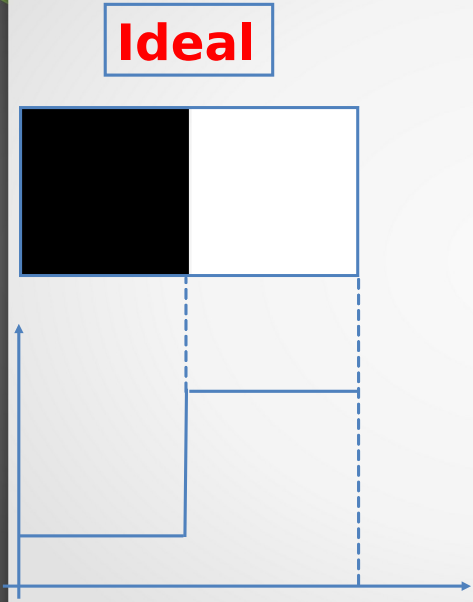
Examples

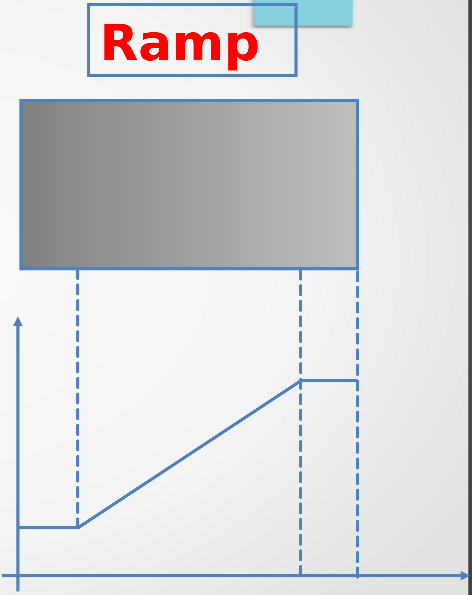
Step Edge

Examples

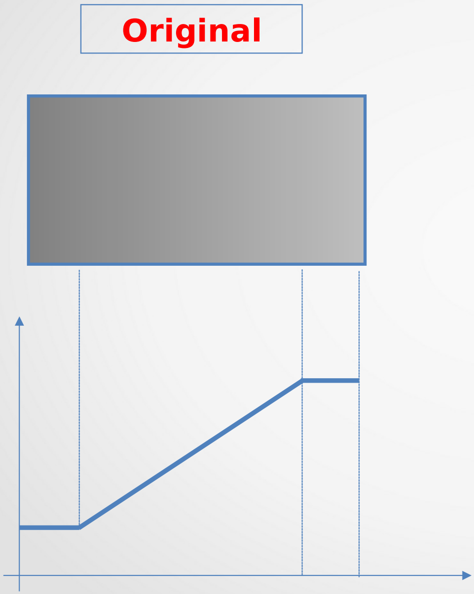
Ridge Edge

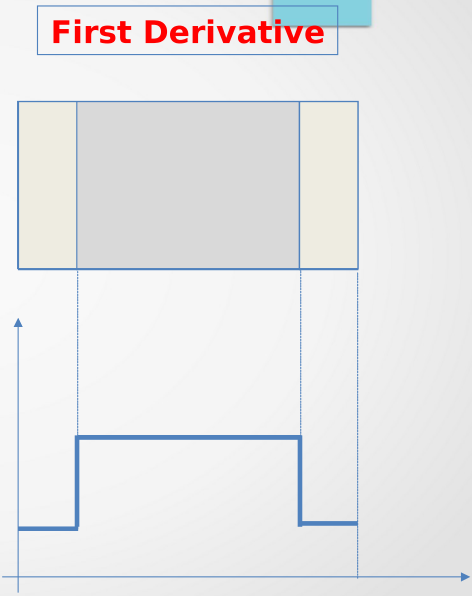
Gray Level Transition



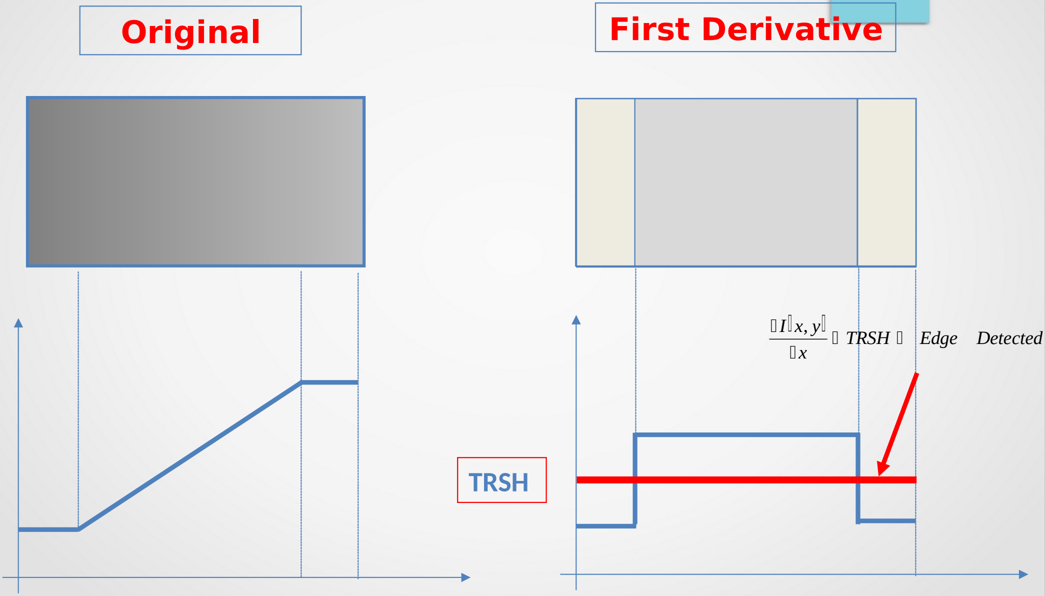


The First Derivative





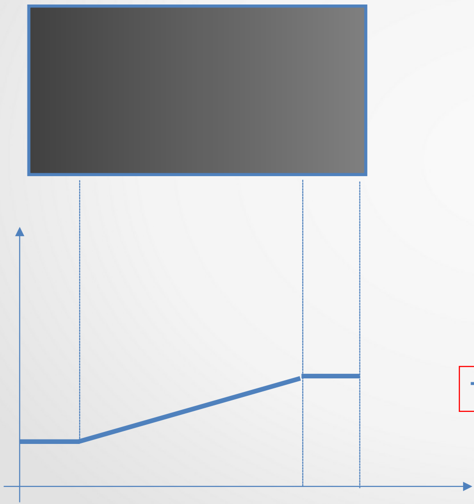
Detecting the Edge

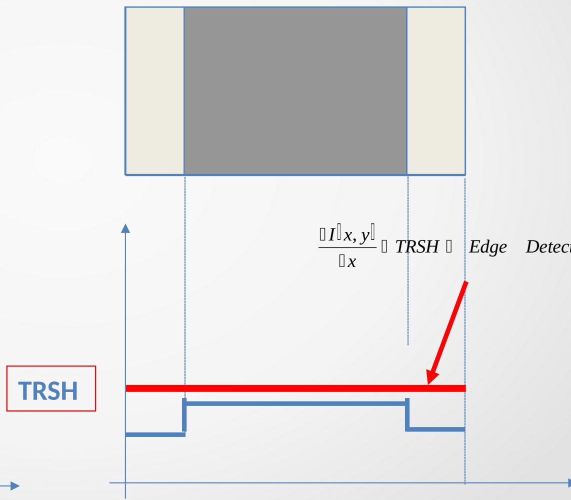


Detecting the Edge

Original

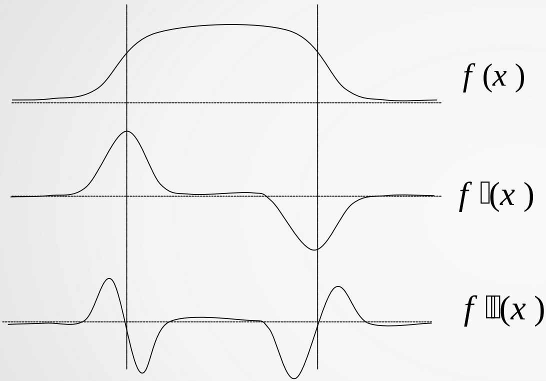
First Derivative

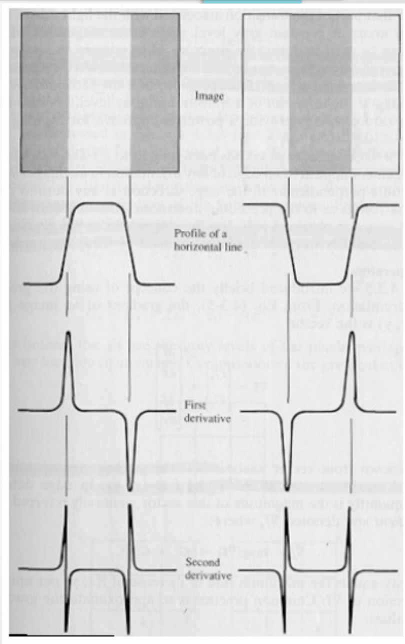




$$\frac{\partial I(x,y)}{\partial x} > TRSH \Rightarrow Edge\quad Detected$$

Edge Detection





Edges can be characterized as either:

- local extrema of $f'(x)$
- zero-crossings of $f''(x)$

Gradient Operators

- The gradient of the image $I(x,y)$ at location (x,y), is the vector:

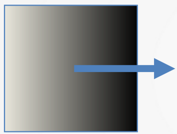
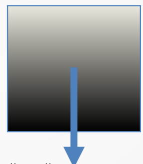
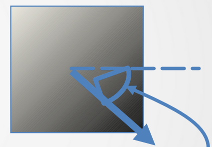
$$\overline{\nabla I} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial I(x,y)}{\partial x} \\ \dfrac{\partial I(x,y)}{\partial y} \end{bmatrix}$$

- The magnitude of the gradient: $\nabla I = \left\| \overline{\nabla I} \right\| = \sqrt{\left[G_x^2 + G_y^2 \right]}$

- The direction of the gradient vector: $\theta(x,y) = \tan^{-1}\left[\dfrac{G_x}{G_y} \right]$

What is the meaning of Gradient?

- It represents the direction of the strongest variation in intensity

	Vertical	Horizontal	Generic
			
Edge Strength:	$\|\nabla I\| = \|G_x\|$	$\|\nabla I\| = \|G_y\|$	$\|\nabla I\| = \sqrt{G_x^2 + G_y^2}$
Edge Direction:	$\alpha(x, y) = 0$	$\alpha(x, y) = -\dfrac{\pi}{2}$	$\alpha(x, y) = \tan^{-1}\left(\dfrac{G_y}{G_x}\right)$

The direction of the edge at location *(x,y)* is perpendicular to the gradient vector at that point

Operators that approximate 1^{st} derivative

- examine **small local neighborhoods**
- can be expressed by **convolution masks**.
- Operators which are **able to detect edge direction** as well are represented by a collection of masks, each corresponding to a certain direction.

Operators that approximate 1st derivative

– **Roberts operator**

Pixel location (i,j)

$$h_1 = \begin{bmatrix} 1 & 0 \\ 0 & -1 \end{bmatrix} \qquad h_2 = \begin{bmatrix} 0 & 1 \\ -1 & 0 \end{bmatrix}$$

– the magnitude of the edge is computed as

$$\left|g(i,j) - g(i+1,j+1)\right| + \left|g(i,j+1) - g(i+1,j)\right|$$

– **Primary disadvantage**: **high sensitivity to noise**, as very few pixels are used for approximation.

Operators that approximate 1st derivative

- Prewitt operator
  - estimated in eight possible directions
  - direction is given by the mask giving maximal response

$$h_1 = \begin{bmatrix} 1 & 1 & 1 \\ 0 & 0 & 0 \\ -1 & -1 & -1 \end{bmatrix} \quad h_2 = \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix} \quad h_3 = \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

Exampe: Roberts and Prewitt

(a) (b) (c)

Operators that approximate 1st derivative

– **Sobel operator**

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

h_x

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

h_y

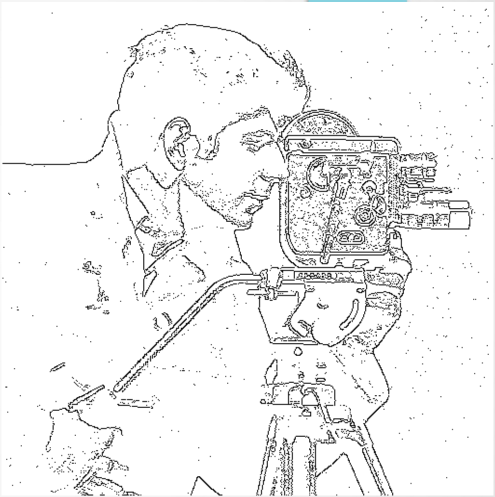
then edge strength=

and direction y/x

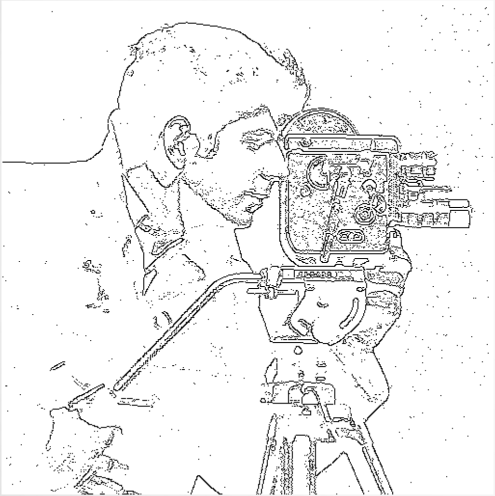
Sobel example

Discrete Operators Compared

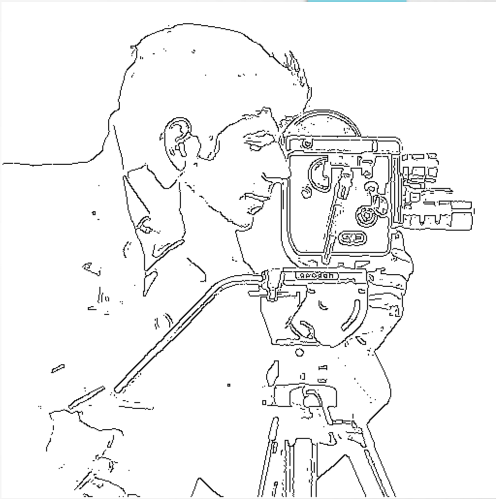
Original



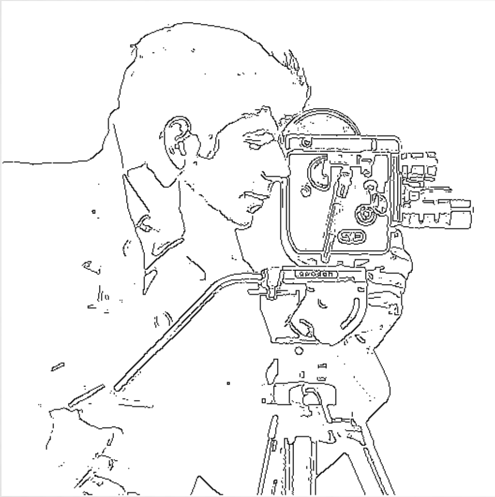
Roberts

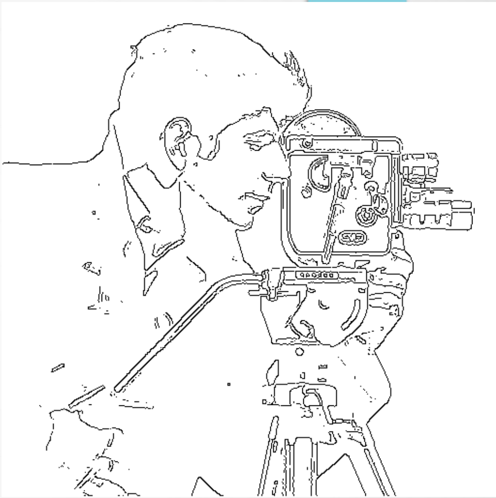


Roberts



Prewitt





Prewitt

Sobel

Need thresholding to thin output edges and remove small edges

Operators based on zero crossing of the 2nd derivative

- The first derivative of the image function should have an extreme at the position corresponding to the edge in the image, and so the second derivative should be zero at the same position.
- It is easier and more precise to find a zero crossing position than an extreme

Operators based on zero crossing of the 2nd derivative

- **Laplace operator:**
  - a very popular operator approximating the second derivative which gives the gradient magnitude only.
  - is approximated in digital images by a convolution sum. A 3 x 3 mask for 4-neighborhoods and 8-neighborhood

$$h = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$

$$h = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Laplace Operator Example

5	5	5	5	5	5	5
4	5	5	5	5	5	5
3	4	5	5	5	5	5
3	3	4	5	5	5	5
3	3	3	4	4	4	4
3	3	3	3	3	3	3
3	3	3	3	3	3	3

Original Image

0	−1	0
−1	4	−1
0	−1	0

2	0	0	0	0
0	2	0	0	0
−2	0	2	1	1
0	−2	1	0	0
0	0	−1	−1	−1

−1	−1	−1
−1	8	−1
−1	−1	−1

4	1	0	0	0
0	4	1	0	0
−4	0	5	3	3
−1	−4	2	0	0
0	−1	−2	−3	−3

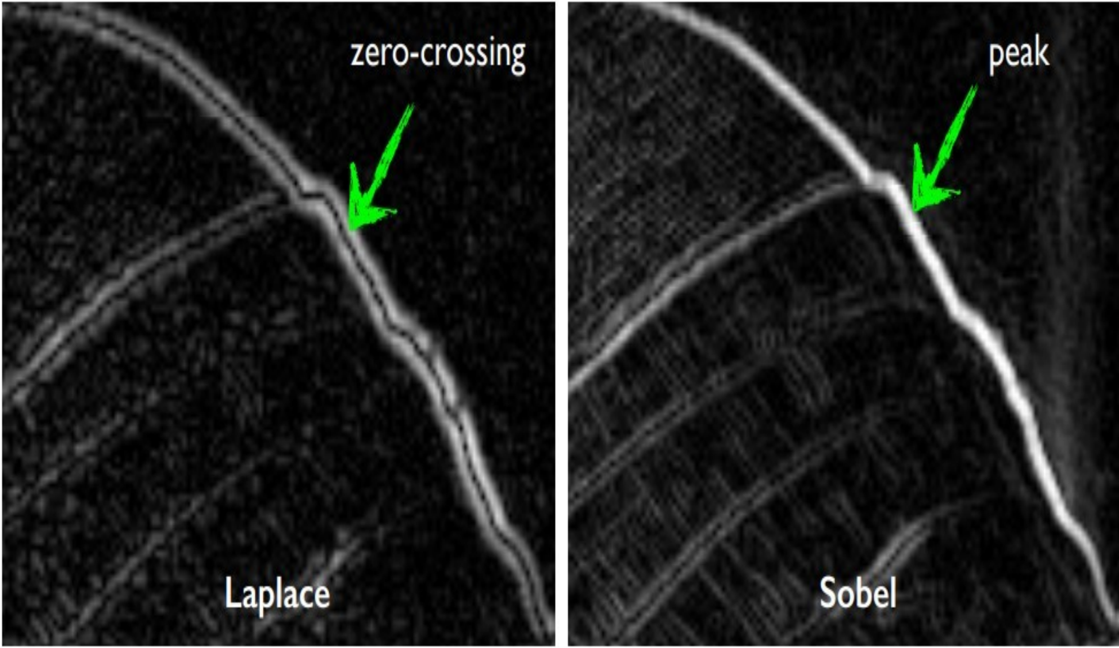
Laplacian of Gaussian (LoG)

- As Laplace operator may detect edges as well as noise
- **LoG**
  - Smooth Image with Gaussian Filter
  - Applying the Laplacian for a Gaussian-filtered image can be done in one step of convolution.
  - Find zero-crossings
  - Find slope of zero-crossings
  - Apply threshold to slope and mark edges


Laplace filter
with smoothing

Laplace filter
without smoothing

What's different between the two results?

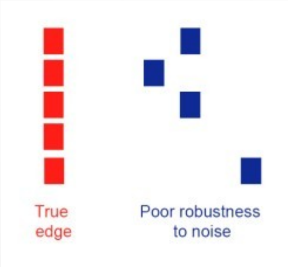


Zero crossings are more accurate at localizing edges

Designing an edge detector

Criteria for an "optimal" edge detector:

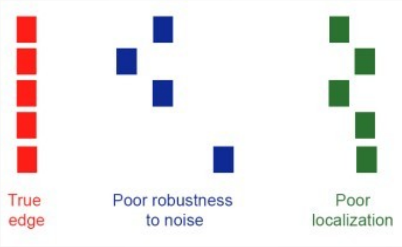
– <u>**Good detection:**</u> the optimal detector must minimize the probability of false positives (detecting spurious edges caused by noise), as well as that of false negatives (missing real edges)



Designing an edge detector

Criteria for an “optimal” edge detector:

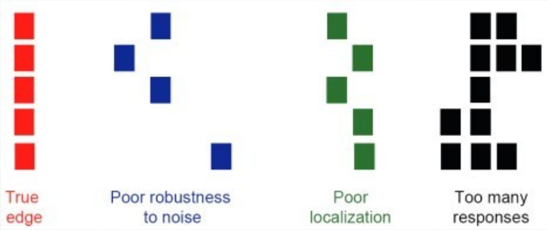
– **Good localization:** the edges detected must be as close as possible to the true edges



Designing an edge detector

Criteria for an "optimal" edge detector:

– <u>**Single response:**</u> the detector must return one point only for each true edge point; that is, minimize the number of local maxima around the true edge



Designing an edge detector

Criteria for an "optimal" edge detector:

– Good detection

- Good localization

- Single response

Canny Edge Detector

- **Steps:**
  - **Smooth** the Image with Gaussian Filter
  - Compute the **Gradient** Magnitude and Orientation using finite-difference approximations for the partial derivatives,
  - Apply **non-maxima suppression** to the gradient magnitude to **thin edge**
  - Use the double **thresholding** algorithm to detect and link edges

Canny Edge Detector

- Smooth by Gaussian

$$S = G_\sigma * I \qquad G_\sigma = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Compute *x* and *y* derivatives

$$\nabla S = \left[\frac{\partial}{\partial x} S \quad \frac{\partial}{\partial y} S \right]^T = \left[S_x \quad S_y \right]^T$$

- Compute gradient magnitude and orientation

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

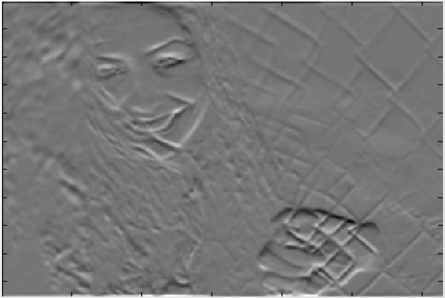
$$\theta = \tan^{-1} \frac{S_y}{S_x}$$

Canny Edge Detector

S_x

I

S_y



Canny Edge Detector

$$\left|\nabla S\right| = \sqrt{S_x^2 + S_y^2}$$

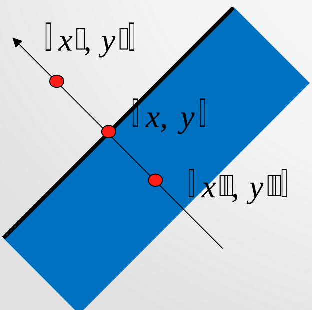
I

$$\left|\nabla S\right| \geq Threshold = 25$$

Non-Maxima Suppression

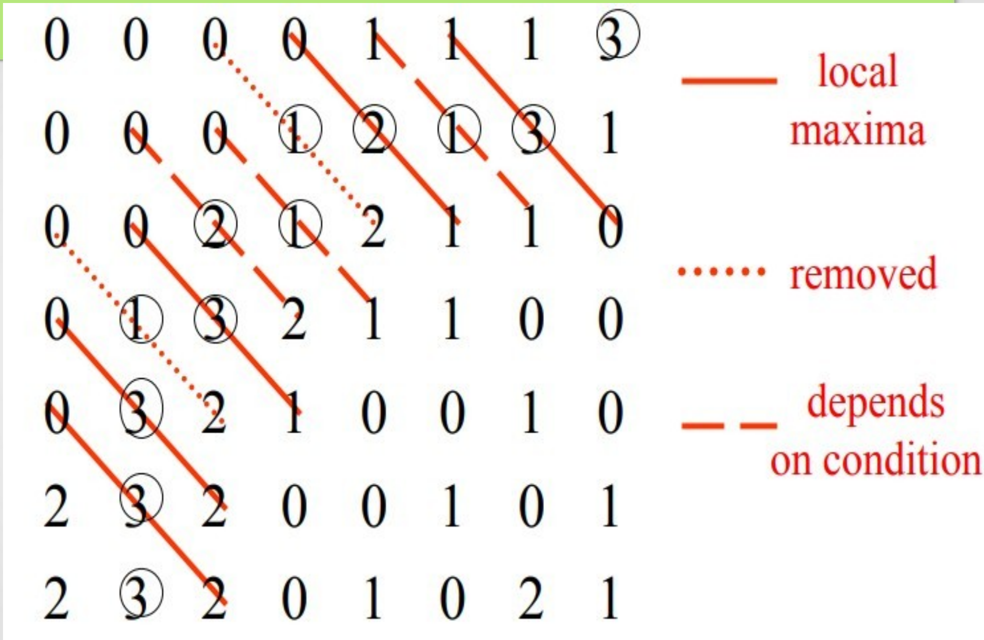
- Suppress the pixels in 'Gradient Magnitude Image' which are not local maximum

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



(x',y') and (x'',y'') are the neighbors of (x,y) in $|\nabla S|$ along the direction normal to an edge

Non-maximum suppression obtains points where the gradient magnitude is at a maximum along the direction of the gradient.!


0 0 0 0 1 1 1 3
0 0 0 1 2 1 3 1
0 0 2 1 2 1 1 0
0 1 3 2 1 1 0 0
0 3 2 1 0 0 1 0
2 3 2 0 0 1 0 1
2 3 2 0 1 0 2 1
local maxima
removed
depends on condition

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

false edges

- The suppressed magnitude image will contain many false edges caused by noise or fine texture

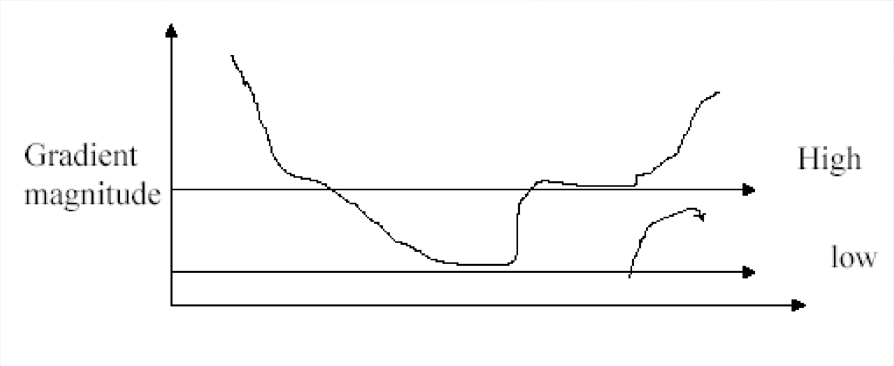
Non-Maximum Suppression

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

M

$$M > Threshold = 25$$

Hysteresis Thresholding



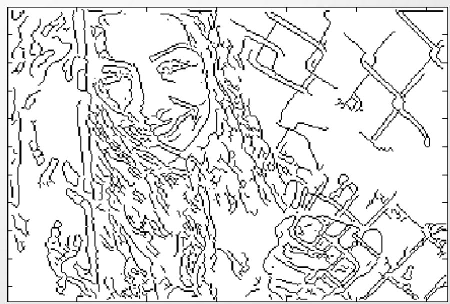
Hysteresis Thresholding

- If the gradient at a pixel is above ‘High’, declare it an ‘edge pixel’
- If the gradient at a pixel is below ‘Low’, declare it a ‘non-edge-pixel’
- If the gradient at a pixel is between ‘Low’ and ‘High’ then declare it an ‘edge pixel’ if and only if it is connected to an ‘edge pixel’ directly or via pixels between ‘Low’ and ‘ High’

Hysteresis Thresholding

M

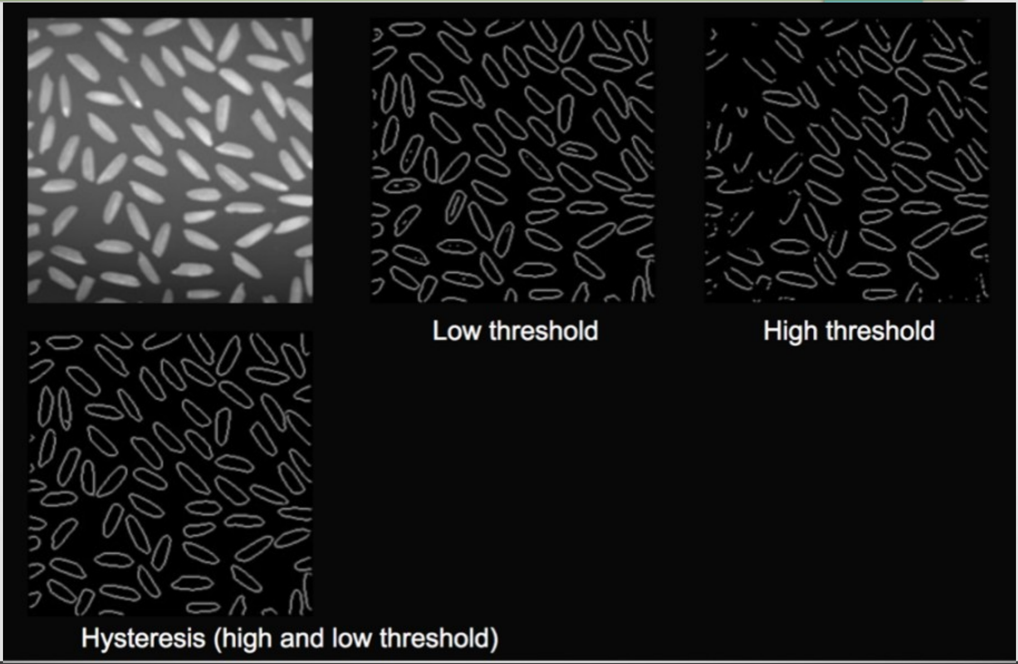
$M \geq Threshold = 25$



$High = 35$

$Low = 15$

Example: Hysteresis Thresholding



Guidelines

The effect of the Canny operator is determined by three parameters:

– the width of the Gaussian kernel used in the smoothing phase (σ)

– the upper threshold of hysteresis

– and the lower threshold used by the tracker.

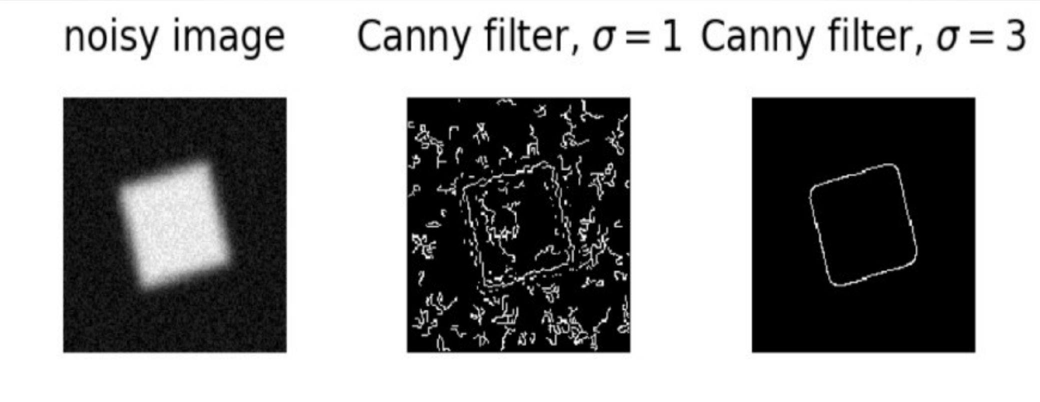
Effect of Gaussian Kernel (smoothing)

original — Canny with $\sigma = 1$ — Canny with $\sigma = 2$

The choice of σ depends on desired behavior

- large σ detects large scale edges
- small σ detects fine features

Effect of Gaussian Kernel (smoothing)



Guidelines

- **Increasing** the width of the Gaussian kernel **reduces** the detector's sensitivity to noise, at the expense of **losing** some of the finer detail in the image.
- The **localization error** in the detected edges also **increases** slightly as the **Gaussian width is increased**.
- The upper tracking threshold can be set quite high, and the lower threshold quite low for good results.
- Setting the lower threshold too high will cause noisy edges to break up.
- Setting the upper threshold too low increases the number of spurious and undesirable edge fragments appearing in the output.

The detection of edges is based on comparing the edge gradient with a threshold. This threshold value can be chosen low enough only when there is no noise in the image, so
that all true edges can be detected without miss.
Practically Thigh ~=1.5 Tlow