CAP 4453 Robot Vision

Dr. Gonzalo Vaca-Castaño

gonzalo.vacacastano@ucf.edu

Administrative details

- Homework 2 questions?
- Any Doubts from last classes?

Robot Vision

5. Edge detection I

Credits

- Some slides comes directly from:
  - Yogesh S Rawat (UCF)
  - Noah Snavely (Cornell)
  - Ioannis (Yannis) Gkioulekas (CMU)
  - Mubarak Shah (UCF)
  - S. Seitz
  - James Tompkin
  - Ulas Bagci
  - L. Lazebnik
  - D. Hoeim

Outline

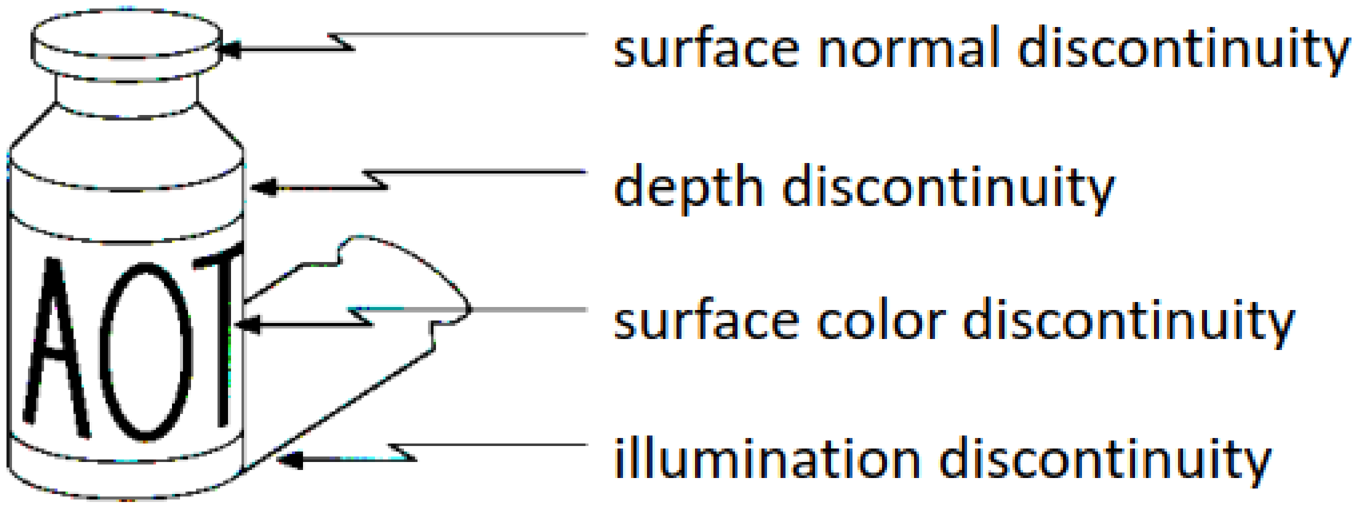
- Image as a function
- Extracting useful information from Images
  - ~~Histogram~~
  - ~~Filtering (linear)~~
  - ~~Smoothing/Removing noise~~
  - ~~Convolution/Correlation~~
  - ~~Image Derivatives/Gradient~~
  - Edges

Edge Detection

- Identify sudden changes in an image
  - Semantic and shape information
  - Mark the border of an object
  - More compact than pixels

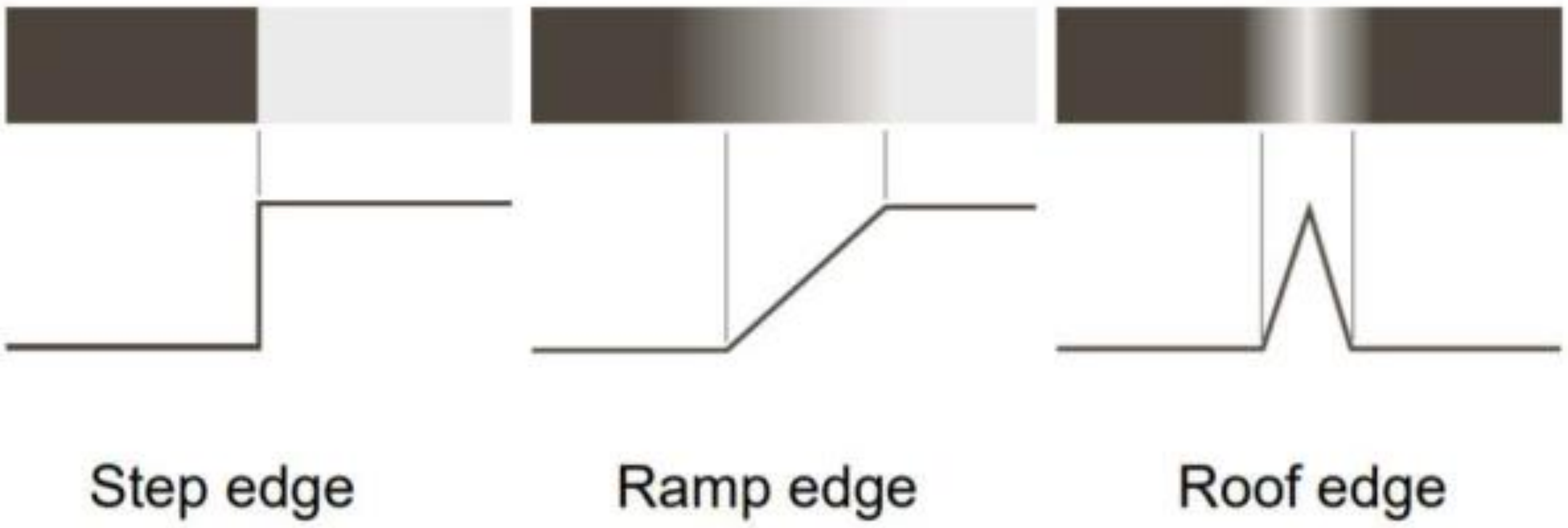
Origin of edges

- Edges are caused by a variety of factors



Type of edges

- Edge models



Why edge detection ?

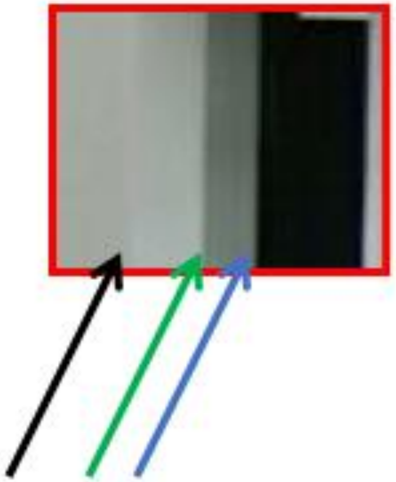
- Extract useful information from images
  - Recognizing objects
- Recover geometry



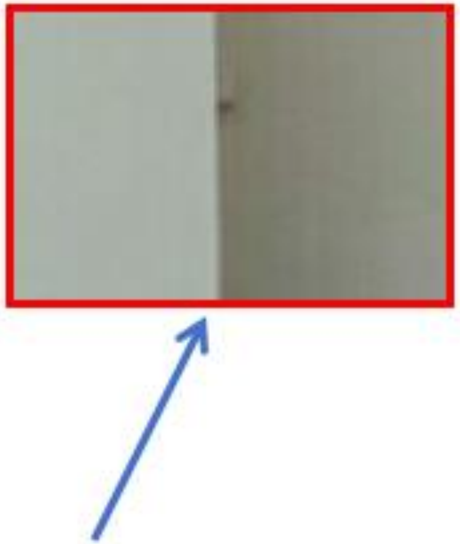


Close up of edges

Close up of edges

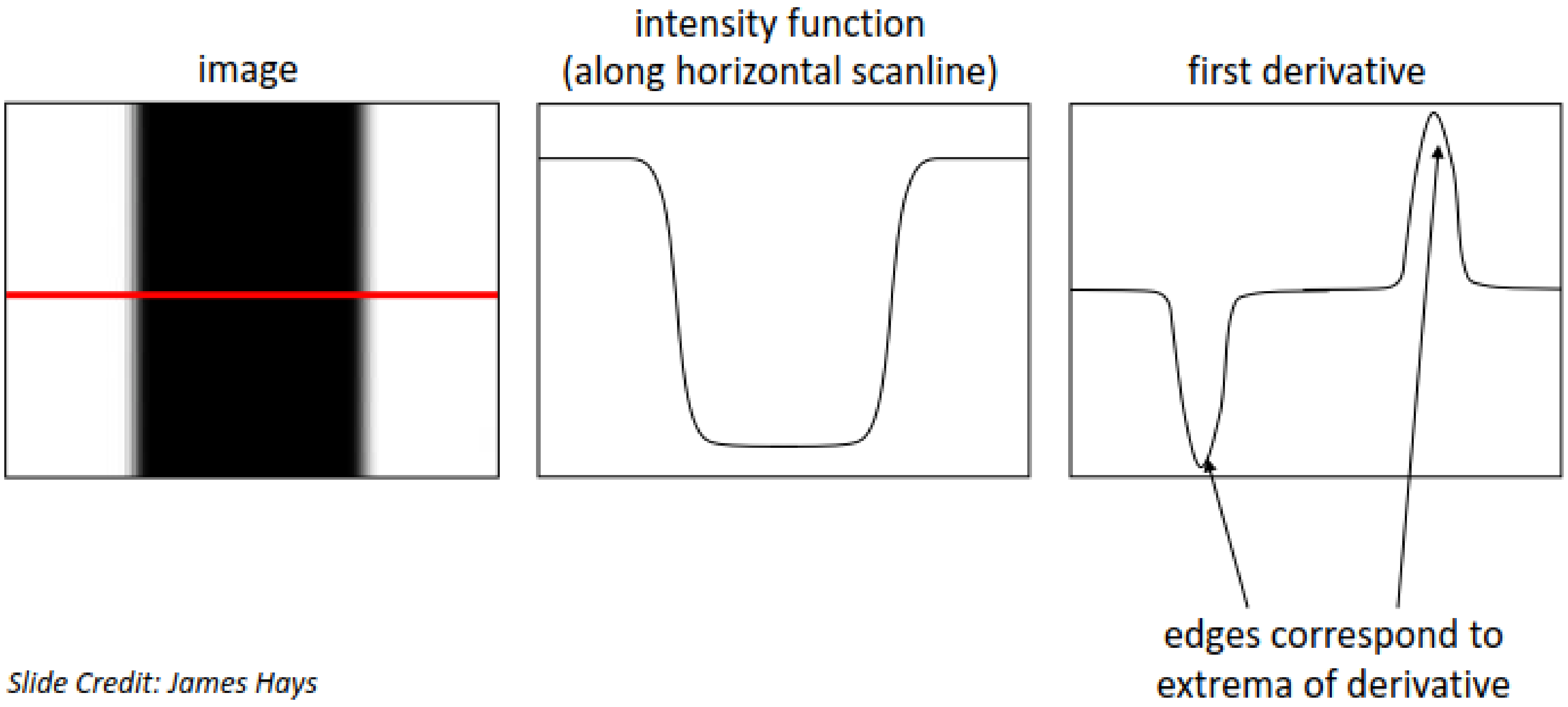


Close up of edges



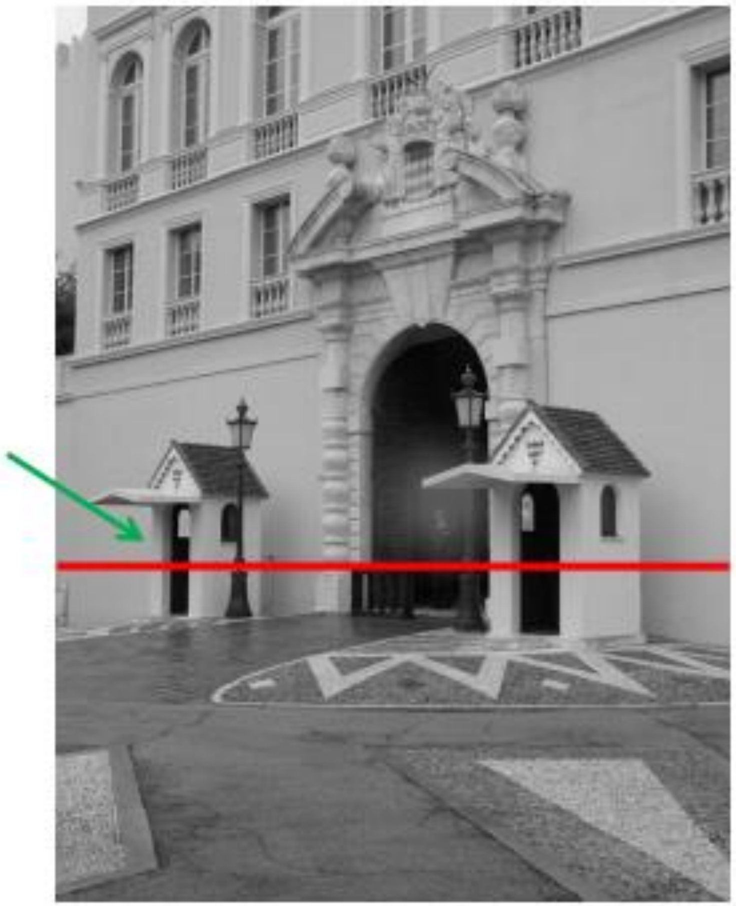
Close up of edges

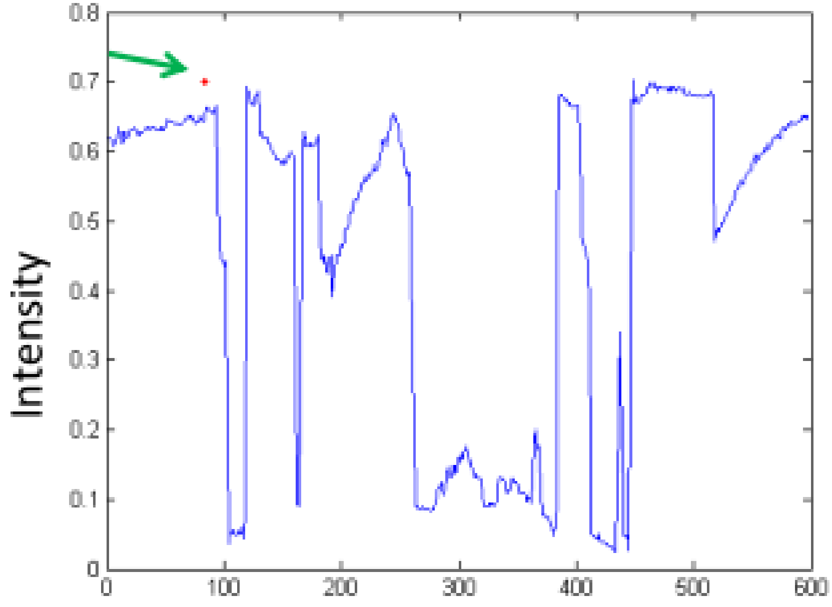
Characterizing edges

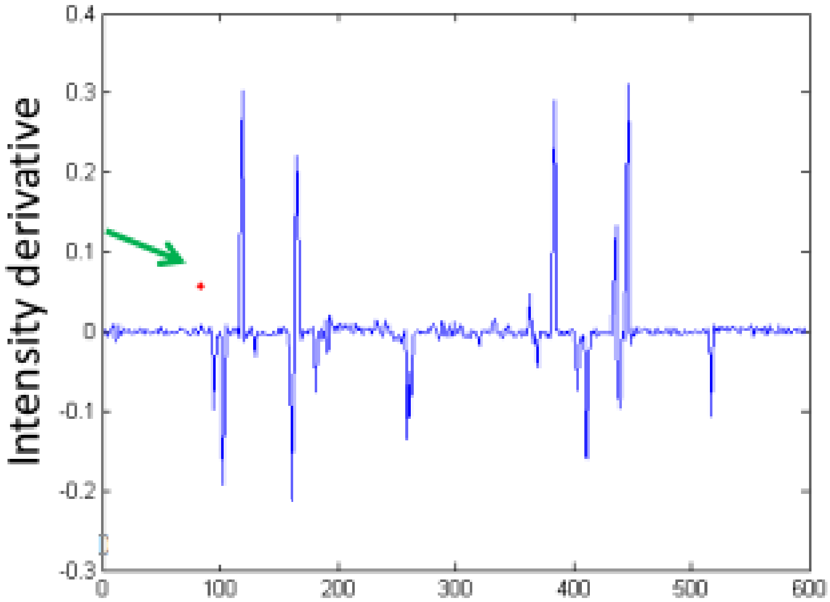


Slide Credit: James Hays

Intensity profile



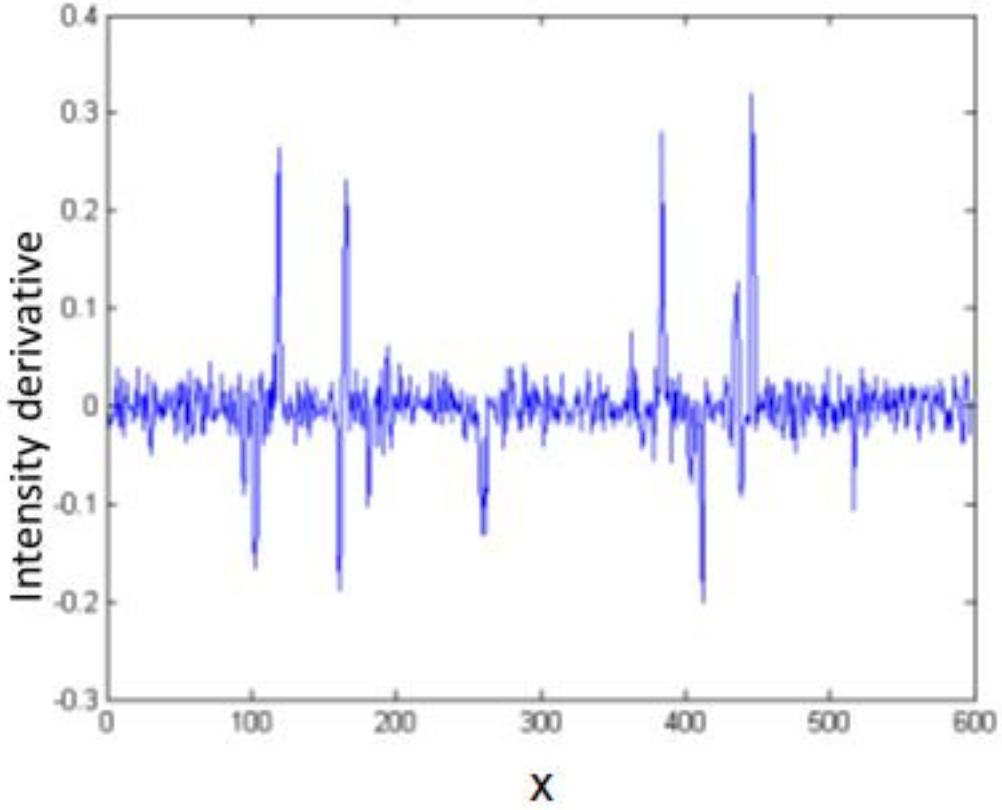




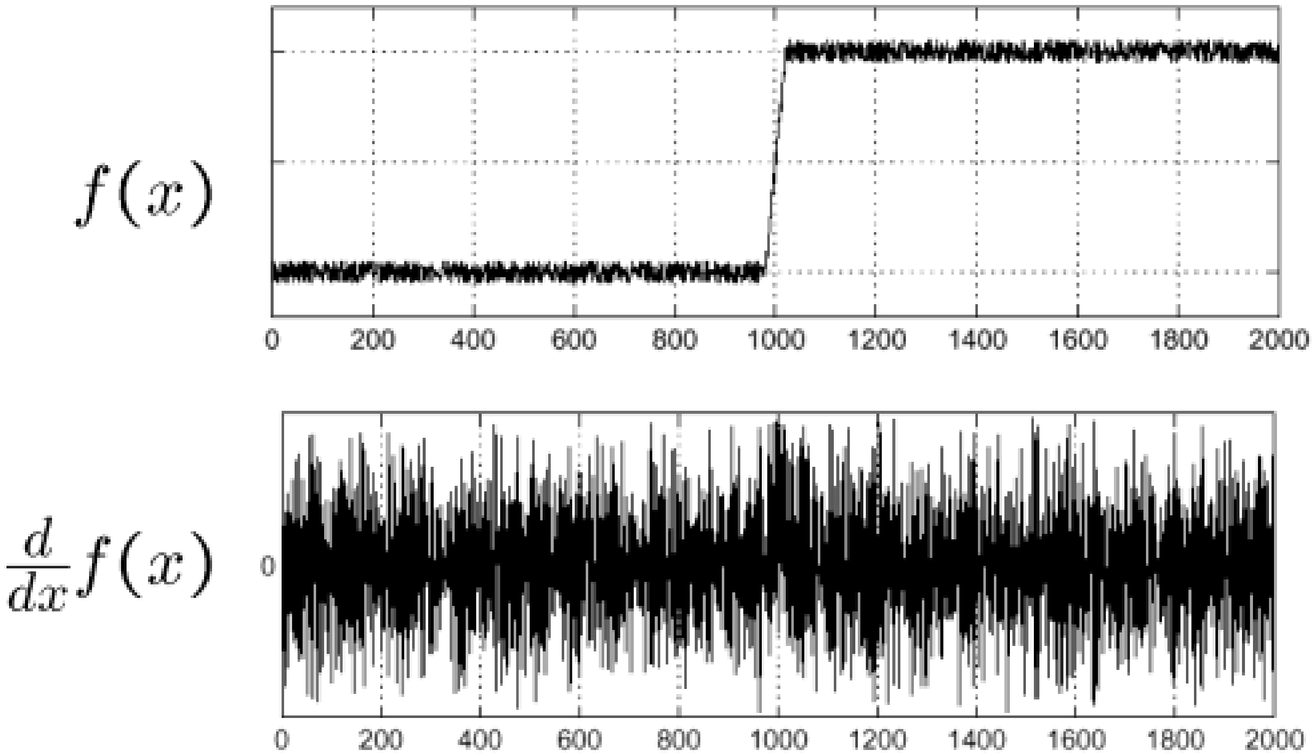
1D derivative filter

1	0	-1

With a little bit of gaussian noise

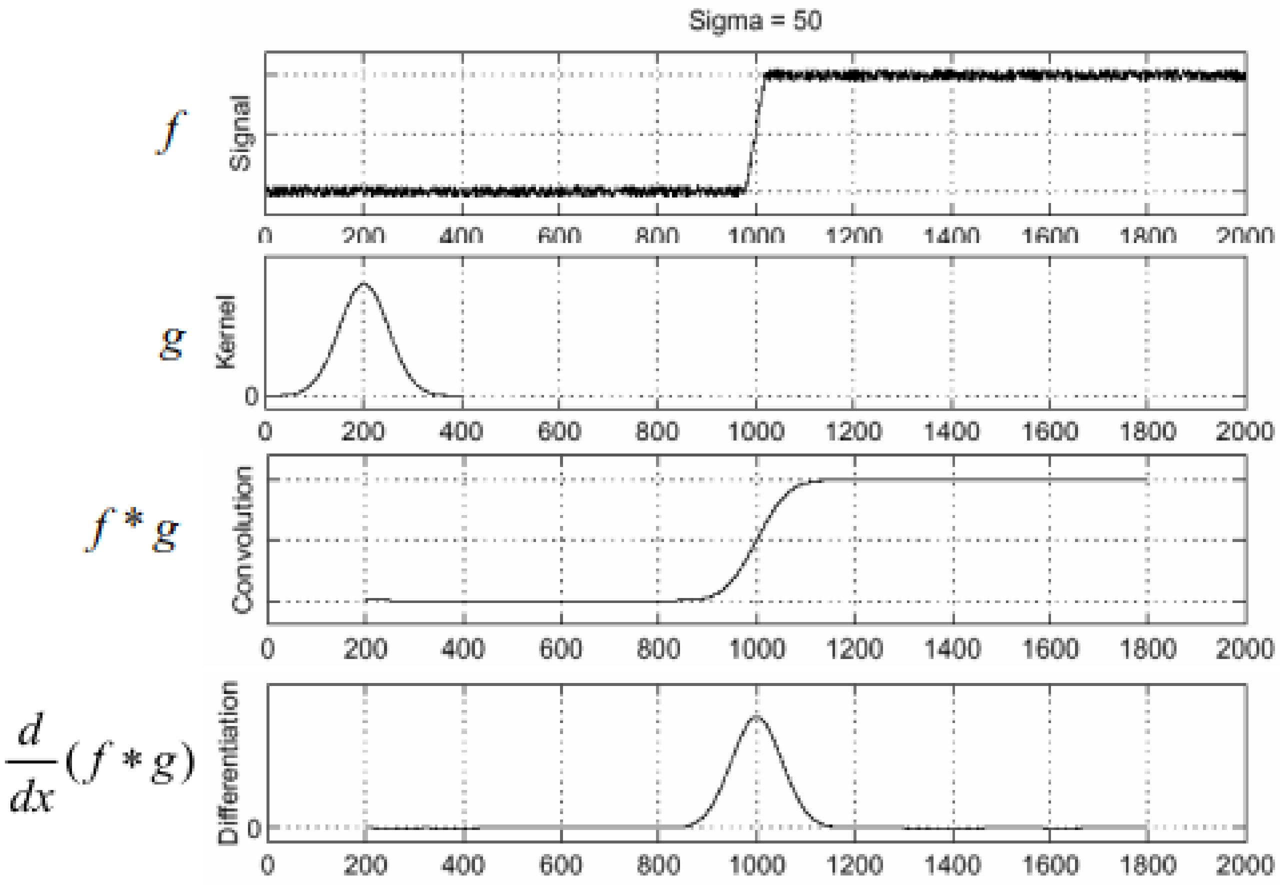


An extreme case



Where is the edge?

Solution: smooth and derivate



To find edges, look for peaks in $\frac{d}{dx}(f*g)$

The Sobel filter

$$\underset{\text{Sobel filter}}{\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}} = \underset{\text{Smoothing}}{\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}} * \underset{\text{1D derivative filter}}{\begin{bmatrix}1 & 0 & -1\end{bmatrix}}$$

$$\underset{\text{Sobel filter}}{\begin{bmatrix}1 & 0 & -1\\ 2 & 0 & -2\\ 1 & 0 & -1\end{bmatrix}} = \underset{\text{1D derivative filter}}{\begin{bmatrix}1 & 0 & -1\end{bmatrix}} * \underset{\text{Smoothing}}{\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}}$$

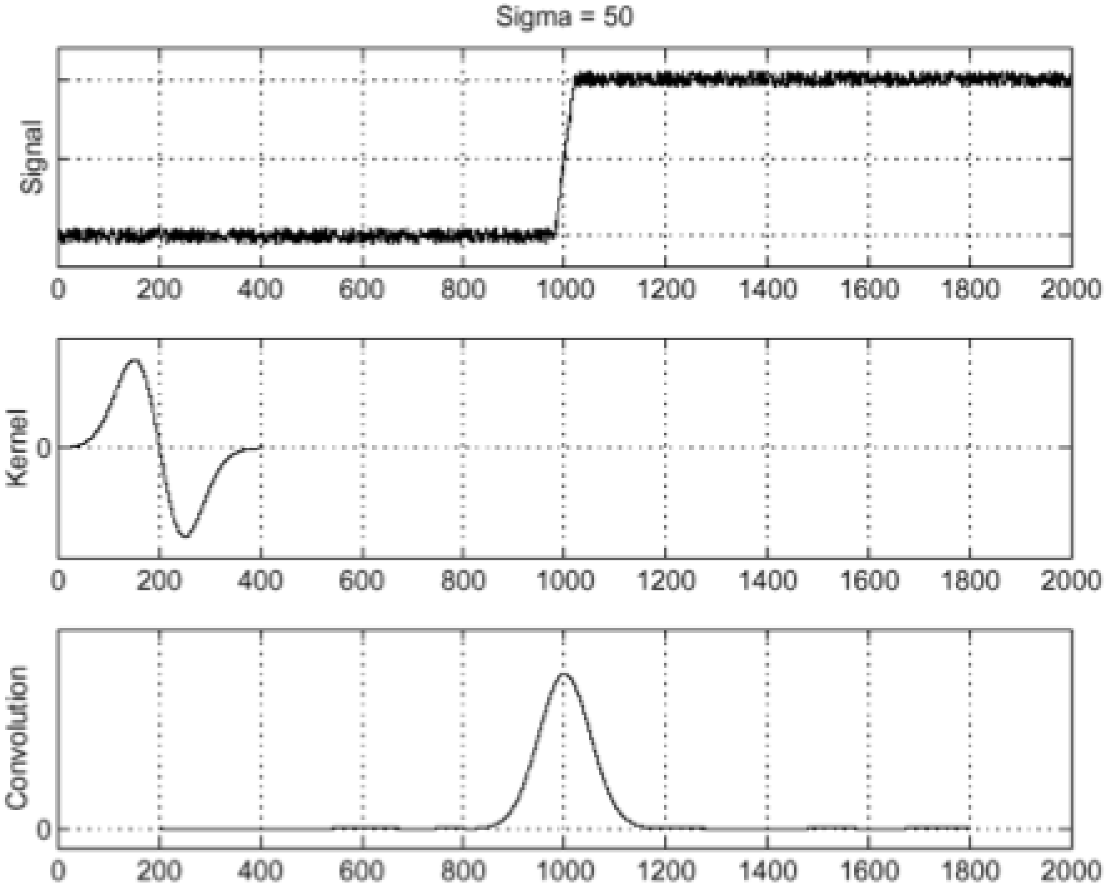
Derivative of Gaussian (DoG) filter

Derivative theorem of convolution: $\frac{\partial}{\partial x}(h \star f) = (\frac{\partial}{\partial x}h) \star f$

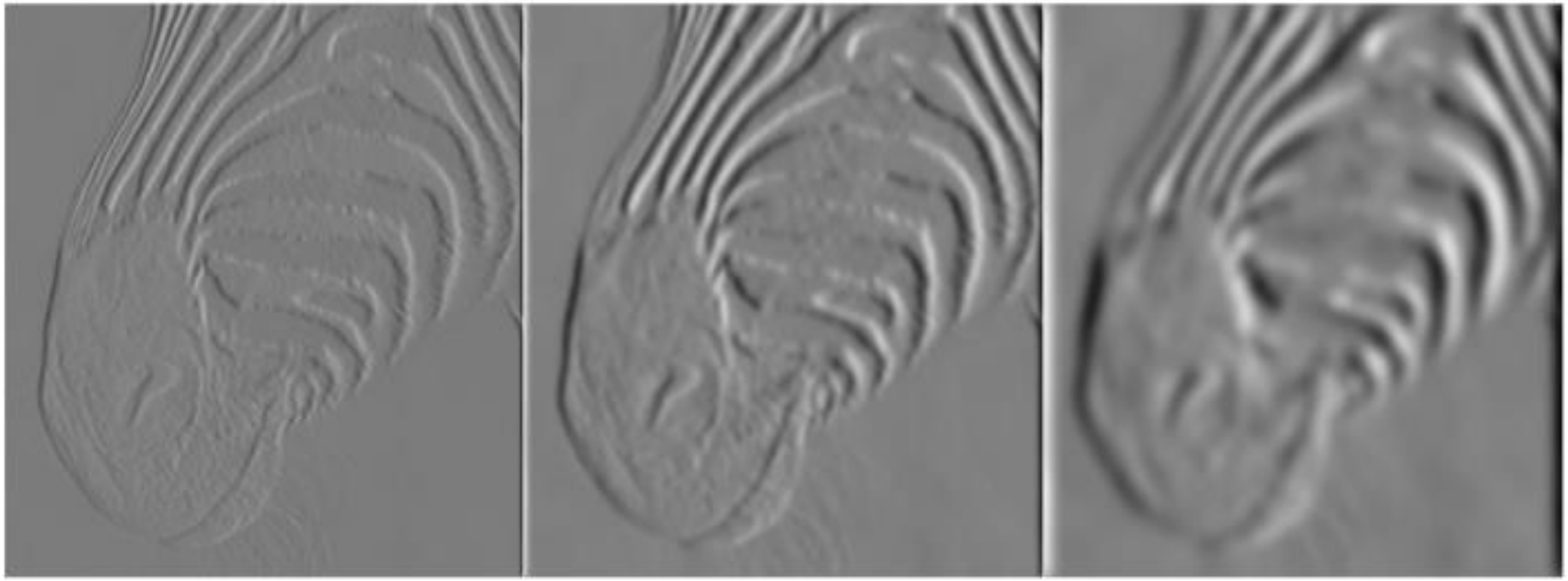
input

derivative of Gaussian

output (same as before)



Solution: smoothing



1 pixel | 3 pixels | 7 pixels

Smoothing remove noise, but also blur the edge

How to obtain the edges of an image?

Several derivative filters

Sobel

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

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

Scharr

3	0	-3
10	0	-10
3	0	-3

3	10	3
0	0	0
-3	-10	-3

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Roberts

0	1
-1	0

1	0
0	-1

- How are the other filters derived and how do they relate to the Sobel filter?
- How would you derive a derivative filter that is larger than 3x3?

Edge detectors

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

Gradient operators edge detector algorithm

1. Compute derivatives
   - In x and y directions
   - Use Sobel or Prewitt filters
2. Find gradient magnitude
3. Threshold gradient magnitude

Sobel

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

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

Prewitt

1	0	-1
1	0	-1
1	0	-1

1	1	1
0	0	0
-1	-1	-1

Computing image gradients

1. Select your favorite derivative filters.

$$\boldsymbol{S}_x = \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} \qquad \boldsymbol{S}_y = \begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 0 & 0 & 0 \\ \hline -1 & -2 & -1 \\ \hline \end{array}$$

2. Convolve with the image to compute derivatives.

$$\frac{\partial \boldsymbol{f}}{\partial x} = \boldsymbol{S}_x \otimes \boldsymbol{f} \qquad \frac{\partial \boldsymbol{f}}{\partial y} = \boldsymbol{S}_y \otimes \boldsymbol{f}$$

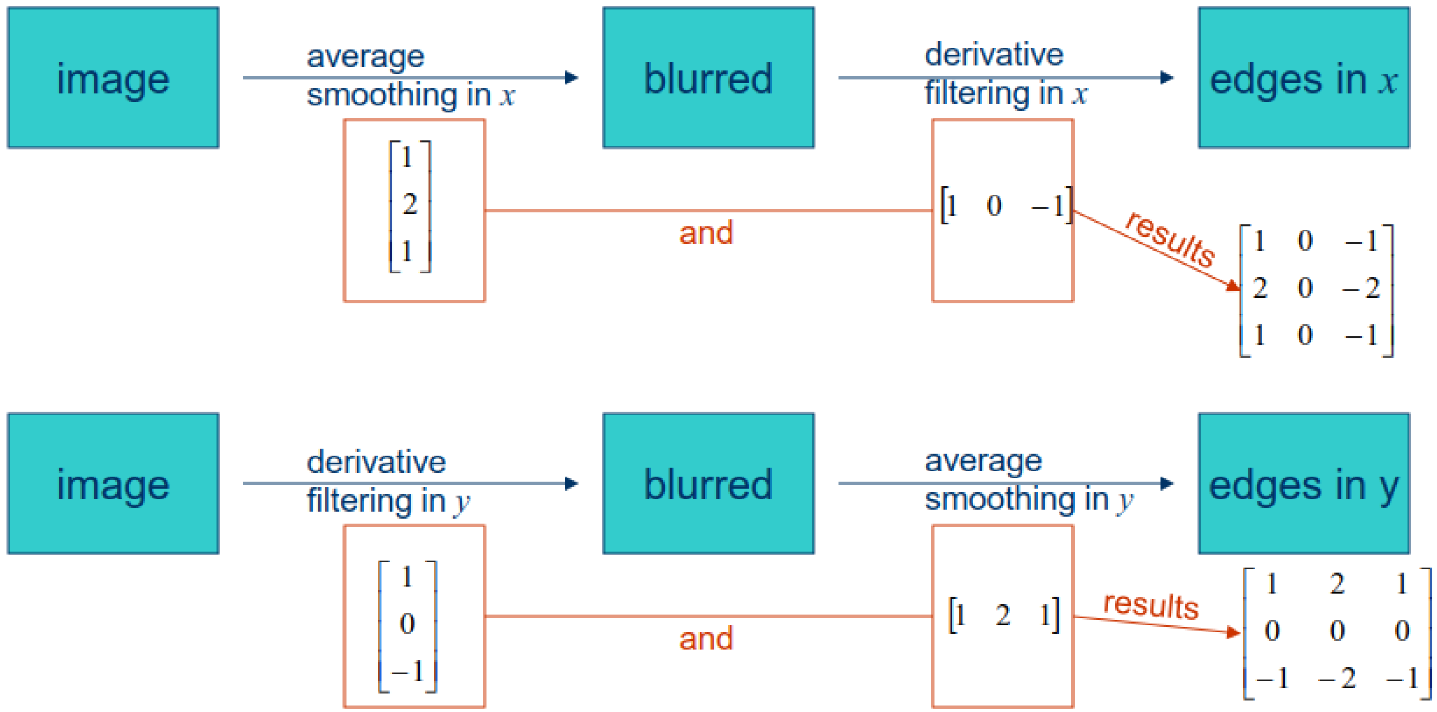
3. Form the image gradient, and compute its direction and amplitude.

$$\nabla \boldsymbol{f} = \left[\frac{\partial \boldsymbol{f}}{\partial x}, \frac{\partial \boldsymbol{f}}{\partial y}\right] \qquad \theta = \tan^{-1}\left(\frac{\partial f}{\partial y} \Big/ \frac{\partial f}{\partial x}\right) \qquad ||\nabla f|| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

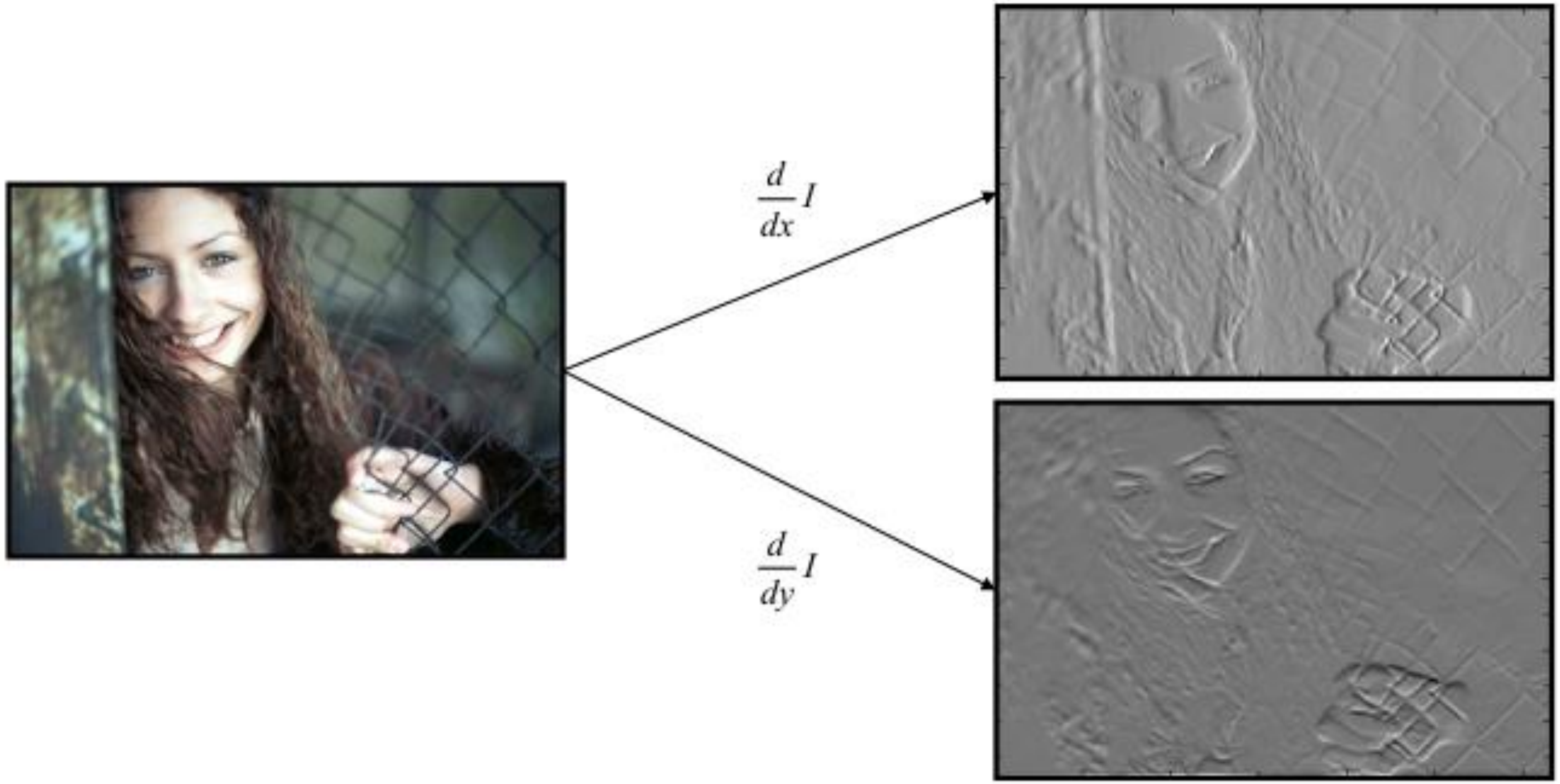
gradient direction amplitude

Sobel edge detector

1. Compute derivatives



Step 1



Sobel edge detector

2. Find gradient magnitude

Step 1

Image I

$$I * \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \rightarrow \frac{d}{dx} I$$

$$I * \begin{bmatrix} 1 & 2 & 1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix} \rightarrow \frac{d}{dy} I$$

Step 2.
Take each pixel of the derivative of the image and square it
Add the two resulting images
Take a square root of each pixel

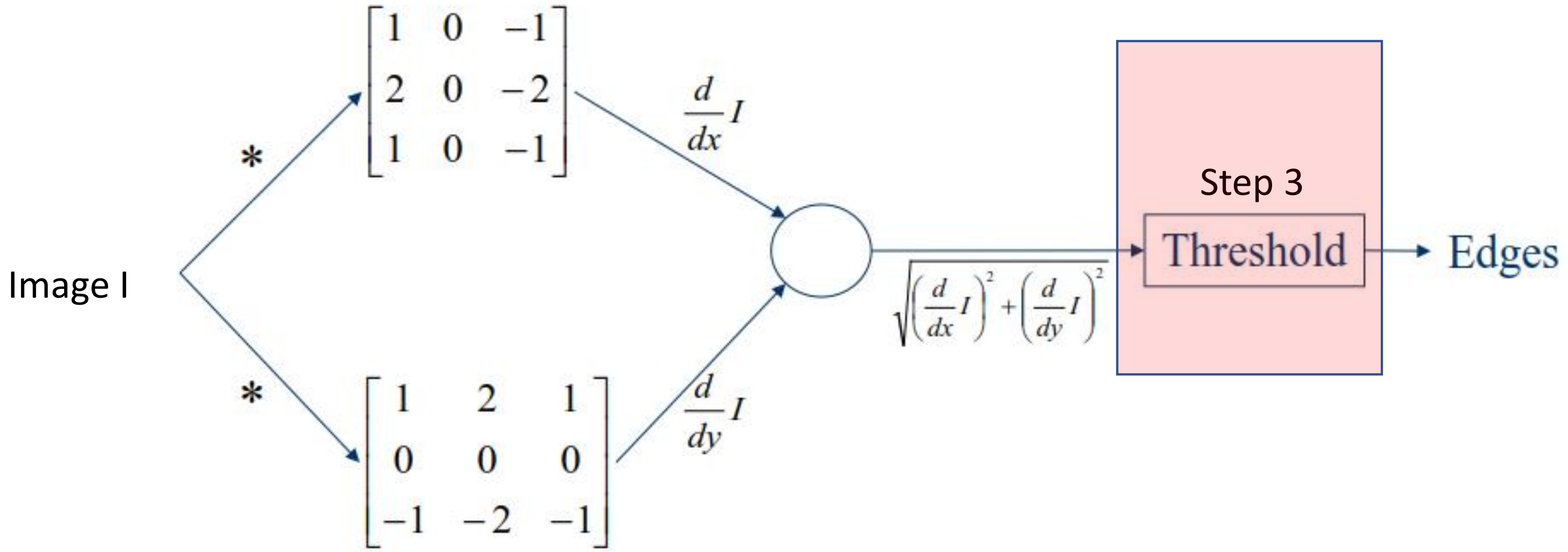
$$\sqrt{\left(\frac{d}{dx} I\right)^2 + \left(\frac{d}{dy} I\right)^2}$$

Step 2

$$\Delta = \sqrt{\left(\frac{d}{dx}I\right)^2 + \left(\frac{d}{dy}I\right)^2}$$

Sobel edge detector

3. Threshold

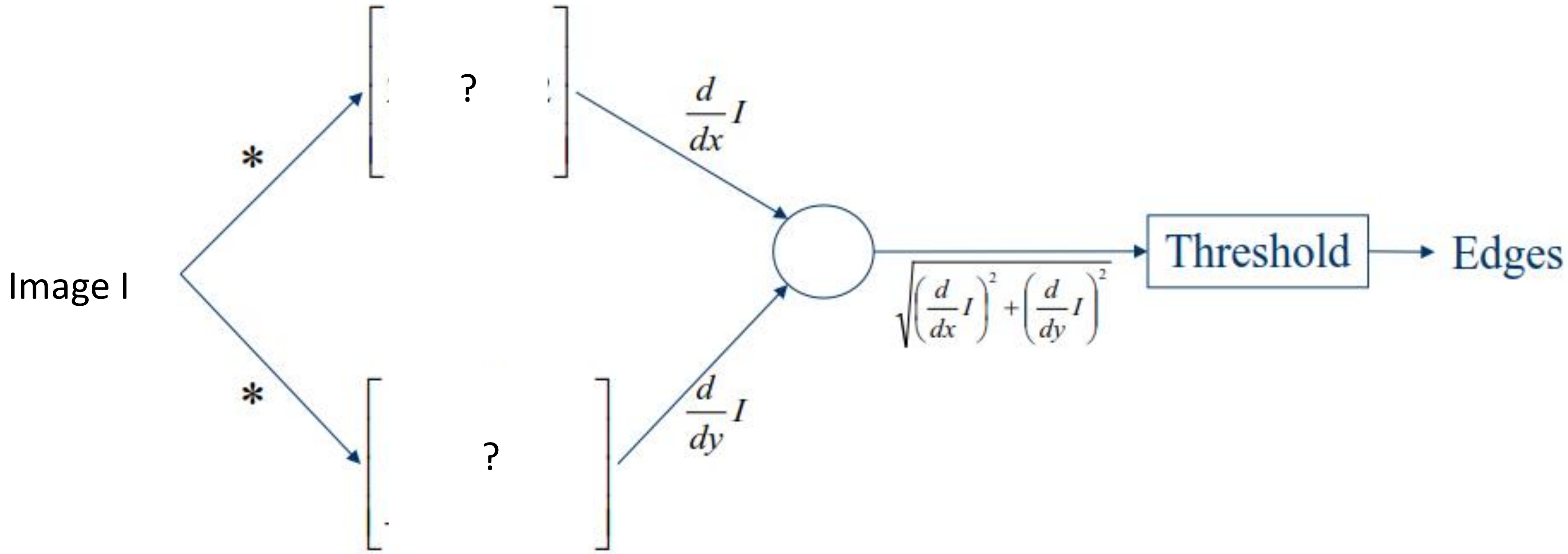


Sobel Edge Detector

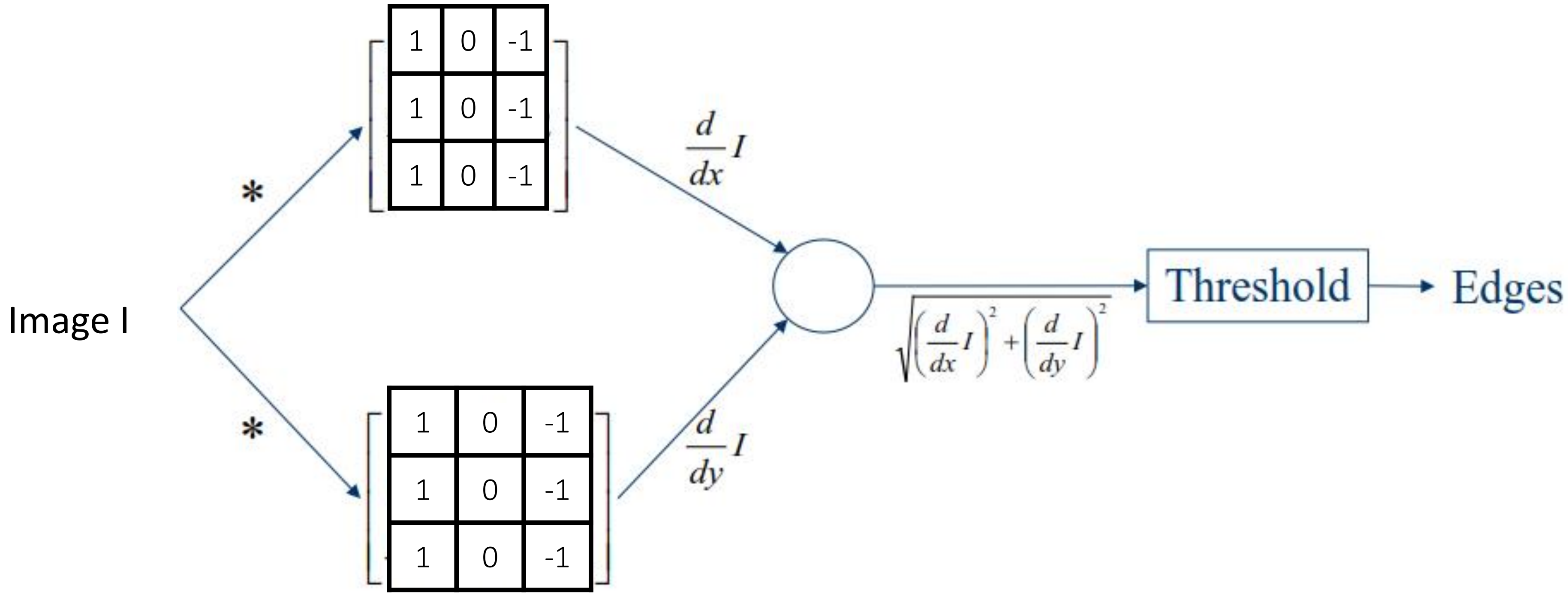
$$\Delta = \sqrt{\left(\frac{d}{dx}I\right)^2 + \left(\frac{d}{dy}I\right)^2}$$

$$\Delta \geq Threshold = 100$$

Prewitt edge detector



Prewitt edge detector

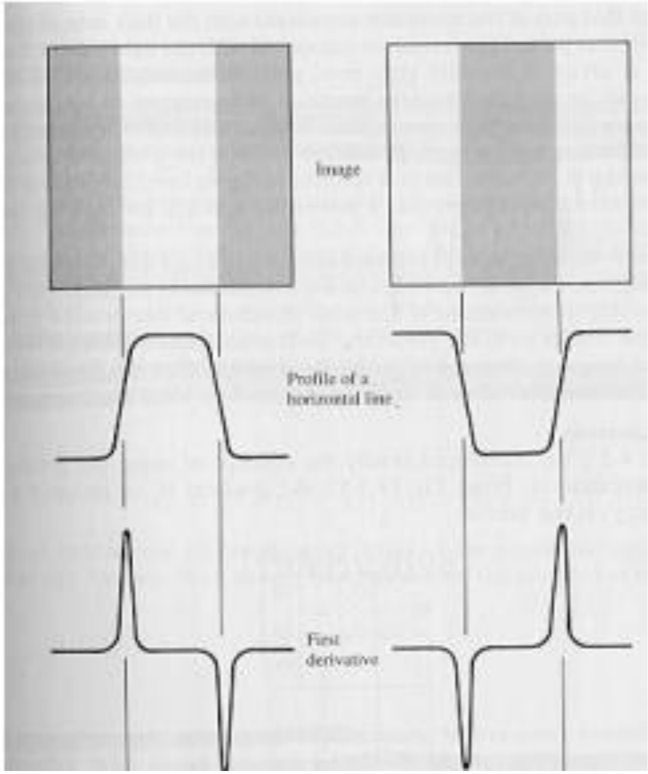


Edge detectors

- Gradient operators
  - Prewit
  - Sobel
- **Marr-Hildreth (Laplacian of Gaussian)**
- Canny (Gradient of Gaussian)

Where are the edges ?

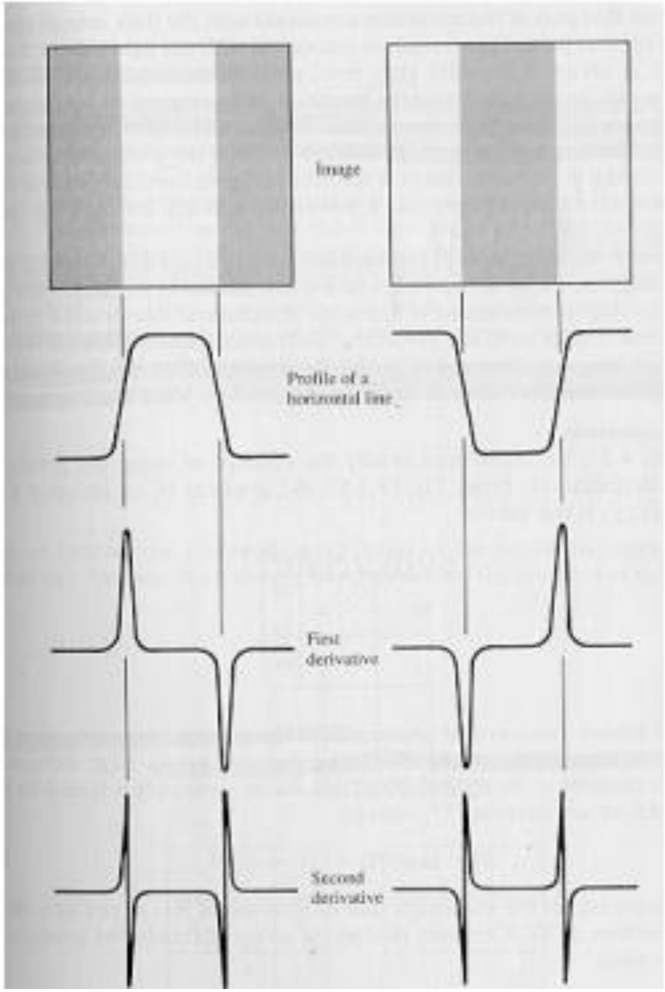
- First derivative ?
  - Maxima or minima



Where are the edges ?

- First derivative ?
  - Maxima or minima

- Second derivative?
  - Zero-crossing



Laplace filter

Basically a second derivative filter.

- We can use finite differences to derive it, as with first derivative filter.

first-order finite difference

$$f'(x) = \lim_{h \to 0} \frac{f(x + 0.5h) - f(x - 0.5h)}{h} \longrightarrow$$

1D derivative filter

1	0	-1

second-order finite difference

$$f''(x) = \lim_{h \to 0} \frac{f(x + h) - 2f(x) + f(x - h)}{h^2} \longrightarrow$$

Laplace filter

1	-2	1

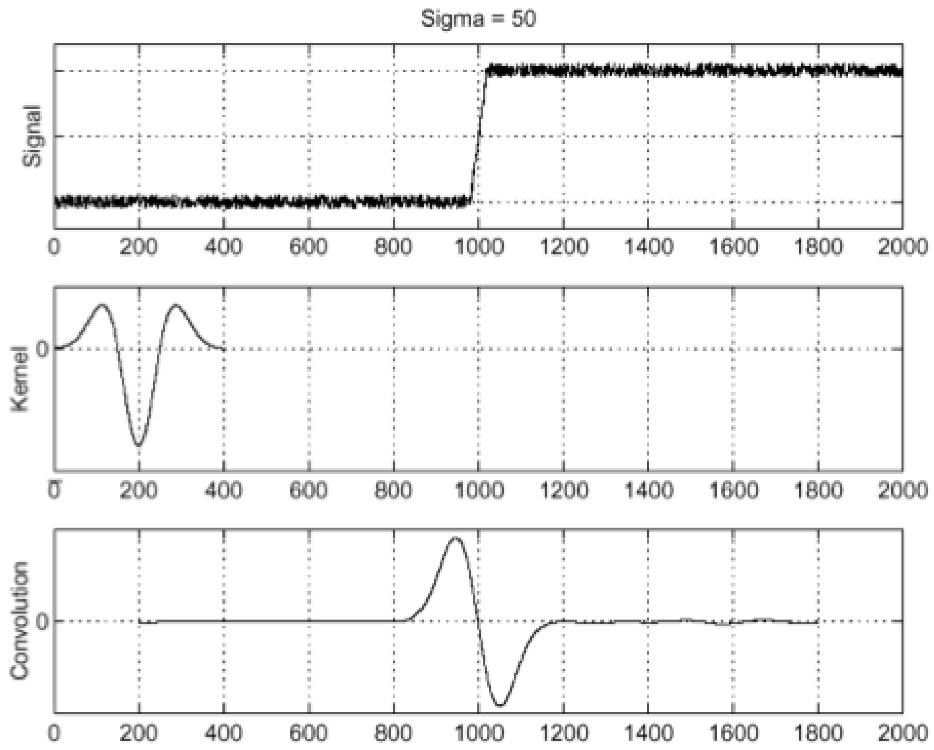
Laplacian of Gaussian (LoG) filter

As with derivative, we can combine Laplace filtering with Gaussian filtering

input

Laplacian of Gaussian

output



"zero crossings" at edges

Laplace and LoG filtering examples

Laplacian of Gaussian filtering

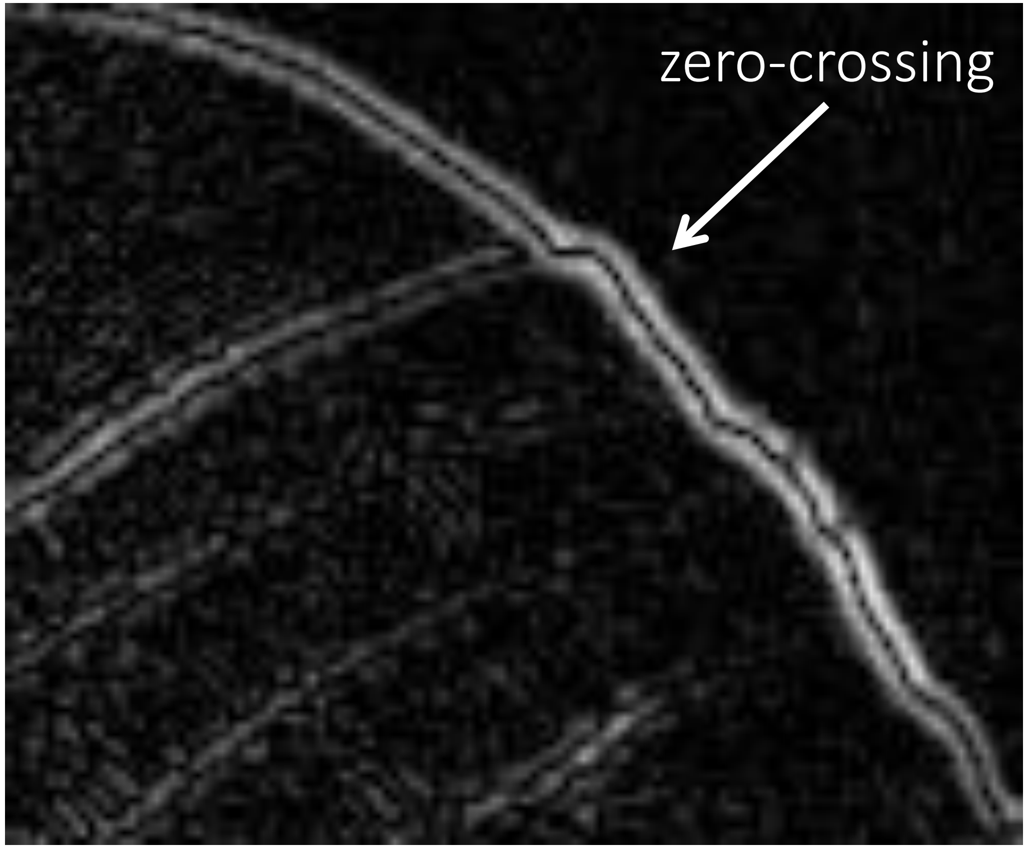
Laplace filtering

Laplacian of Gaussian vs Derivative of Gaussian

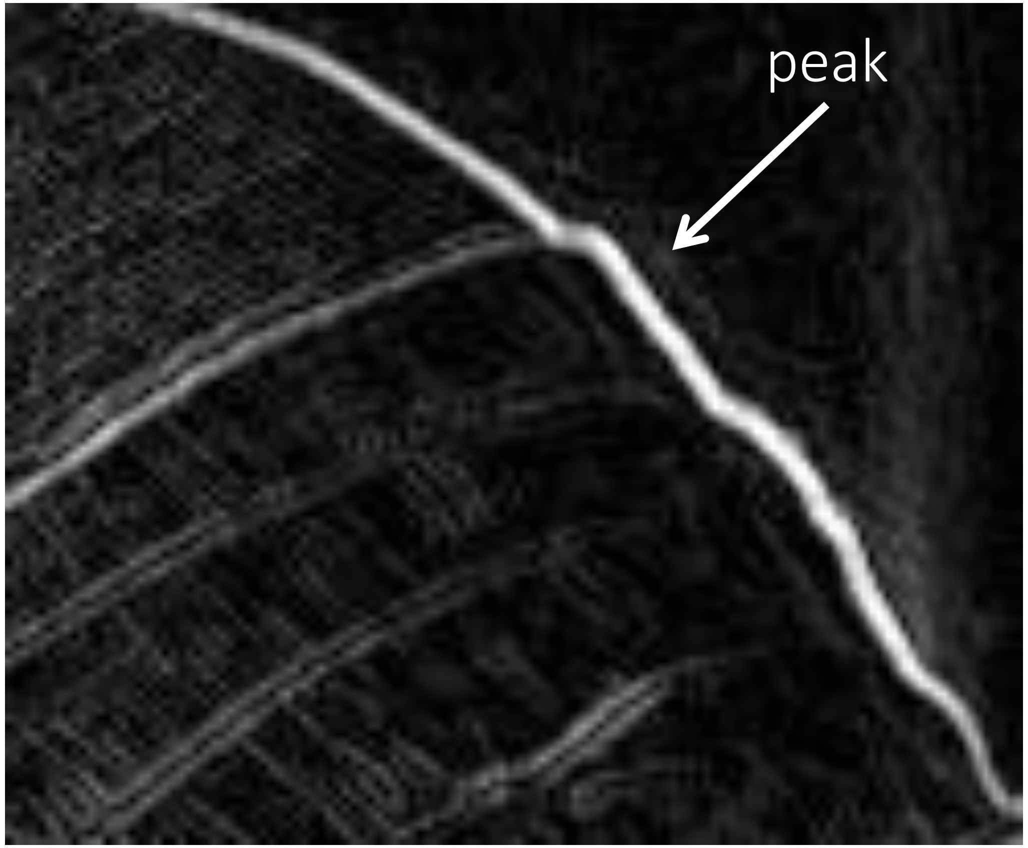
Laplacian of Gaussian filtering

Derivative of Gaussian filtering

Laplacian of Gaussian vs Derivative of Gaussian



Laplacian of Gaussian filtering



Derivative of Gaussian filtering

Zero crossings are more accurate at localizing edges

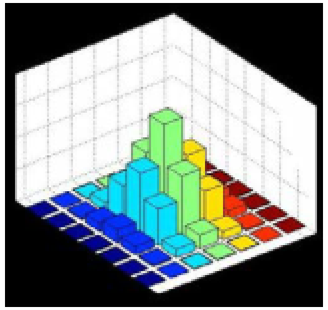
Marr-Hildreth edge detector algorithm

1. Smooth image by Gaussian filtering
2. Apply Laplacian to smoothed image
   - Used in mechanics, electromagnetics, wave theory, quantum mechanics
3. Find Zero crossings

Marr-Hildreth edge detector algorithm

1. Smooth image by Gaussian filtering
   - Gaussian smoothing

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



2. Apply Laplacian to smoothed image
   - 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 (first derivative)
- Δ^2 is used for Laplacian (Second derivative)

Marr-Hildreth edge detector algorithm

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

- ∇ is used for gradient (first derivative)
- Δ^2 is used for Laplacian (Second derivative)

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

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

This is more efficient computationally

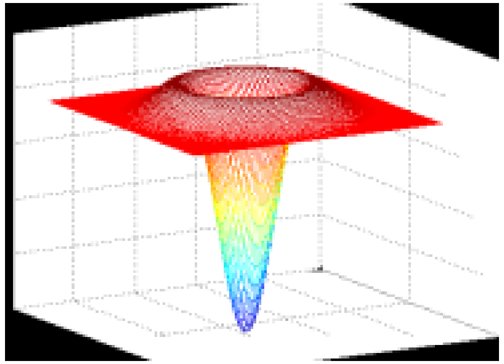
Marr-Hildreth edge detector algorithm

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

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

The second derivative of a gaussian

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



Marr-Hildreth edge detector algorithm

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

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

The second derivative of a gaussian

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

Marr-Hildreth edge detector algorithm

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

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

The second derivative of a gaussian

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

Given a σ, Compute LoG for each x,y to obtain a Kernel

Marr-Hildreth edge detector algorithm

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

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

The second derivative of a gaussian

For $\sigma = 1.4$

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

$$LoG(0,0) \approx -0.1624$$

Given a σ, Compute LoG for each x,y to obtain a Kernel

Marr-Hildreth edge detector algorithm

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

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

The second derivative of a gaussian

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

Given a σ, Compute LoG for each x,y to obtain a Kernel

For σ = 1.4

$$\begin{pmatrix} 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 2 & 5 & 0 & -23 & -40 & -23 & 0 & 5 & 2 \\ 2 & 5 & 3 & -12 & -23 & -12 & 3 & 5 & 2 \\ 3 & 3 & 5 & 3 & 0 & 3 & 5 & 3 & 3 \\ 0 & 2 & 3 & 5 & 5 & 5 & 3 & 2 & 0 \\ 0 & 0 & 3 & 2 & 2 & 2 & 3 & 0 & 0 \end{pmatrix}$$

Marr-Hildreth edge detector algorithm

1. Smooth image by Gaussian filtering
2. Apply Laplacian to smoothed image
   - Used in mechanics, electromagnetics, wave theory, quantum mechanics
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 algorithm

3. Find Zero crossings

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

```
from skimage.filters import laplace
import numpy as np

lap = np.sign(laplace(image))
lap = np.pad(lap, ((0, 1), (0, 1)))
diff_x = lap[:-1, :-1] - lap[:-1, 1:] < 0
diff_y = lap[:-1, :-1] - lap[1:, :-1] < 0

edges =  np.logical_or(diff_x, diff_y).astype(float)
```

Marr-Hildreth edge detector algorithm

3. Find Zero crossings

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

```
from skimage.filters import laplace
import numpy as np

lap = np.sign(laplace(image))
lap = np.pad(lap, ((0, 1), (0, 1)))
diff_x = lap[:-1, :-1] - lap[:-1, 1:] < 0
diff_y = lap[:-1, :-1] - lap[1:, :-1] < 0

edges =  np.logical_or(diff_x, diff_y).astype(float)
```

returns -1 if x < 0, 0 if x==0, 1 if x > 0.

Marr-Hildreth edge detector algorithm

3. Find Zero crossings

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

```
from skimage.filters import laplace
import numpy as np

lap = np.sign(laplace(image))
lap = np.pad(lap, ((0, 1), (0, 1)))
diff_x = lap[:-1, :-1] - lap[:-1, 1:] < 0
diff_y = lap[:-1, :-1] - lap[1:, :-1] < 0

edges =  np.logical_or(diff_x, diff_y).astype(float)
```

returns -1 if x < 0, 0 if x==0, 1 if x > 0.

Add extra row, and extra column

Marr-Hildreth edge detector algorithm

3. Find Zero crossings

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

```
from skimage.filters import laplace
import numpy as np

lap = np.sign(laplace(image))
lap = np.pad(lap, ((0, 1), (0, 1)))
diff_x = lap[:-1, :-1] - lap[:-1, 1:] < 0
diff_y = lap[:-1, :-1] - lap[1:, :-1] < 0

edges =  np.logical_or(diff_x, diff_y).astype(float)
```

returns -1 if x < 0, 0 if x==0, 1 if x > 0.

Add extra row, and extra column

Zero-crossing X direction

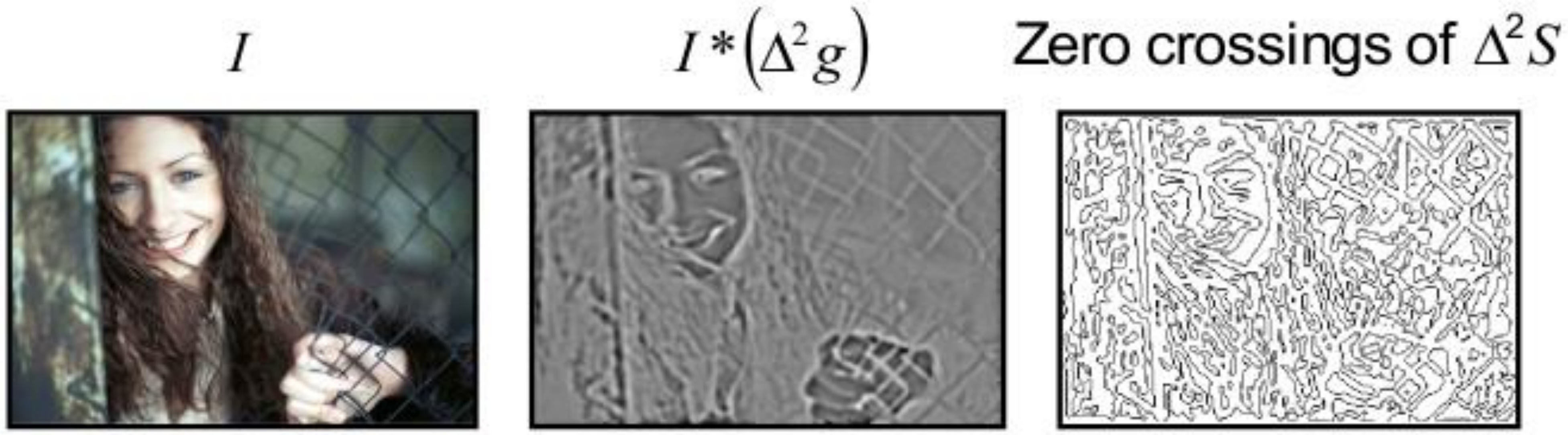
Zero-crossing Y direction

Marr-Hildreth edge detector algorithm

3. Find Zero crossings (Another implementation)

- 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

Example

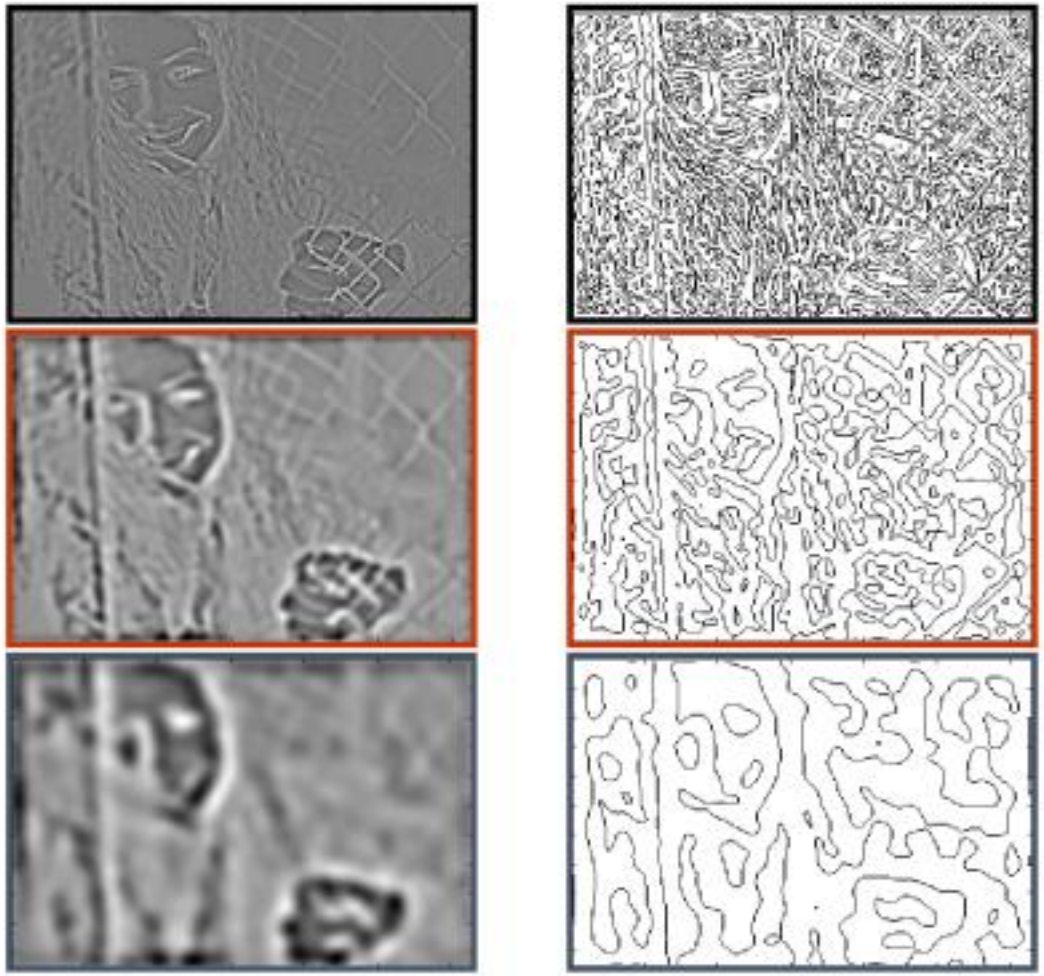


Example

$\sigma = 1$

$\sigma = 3$

$\sigma = 6$

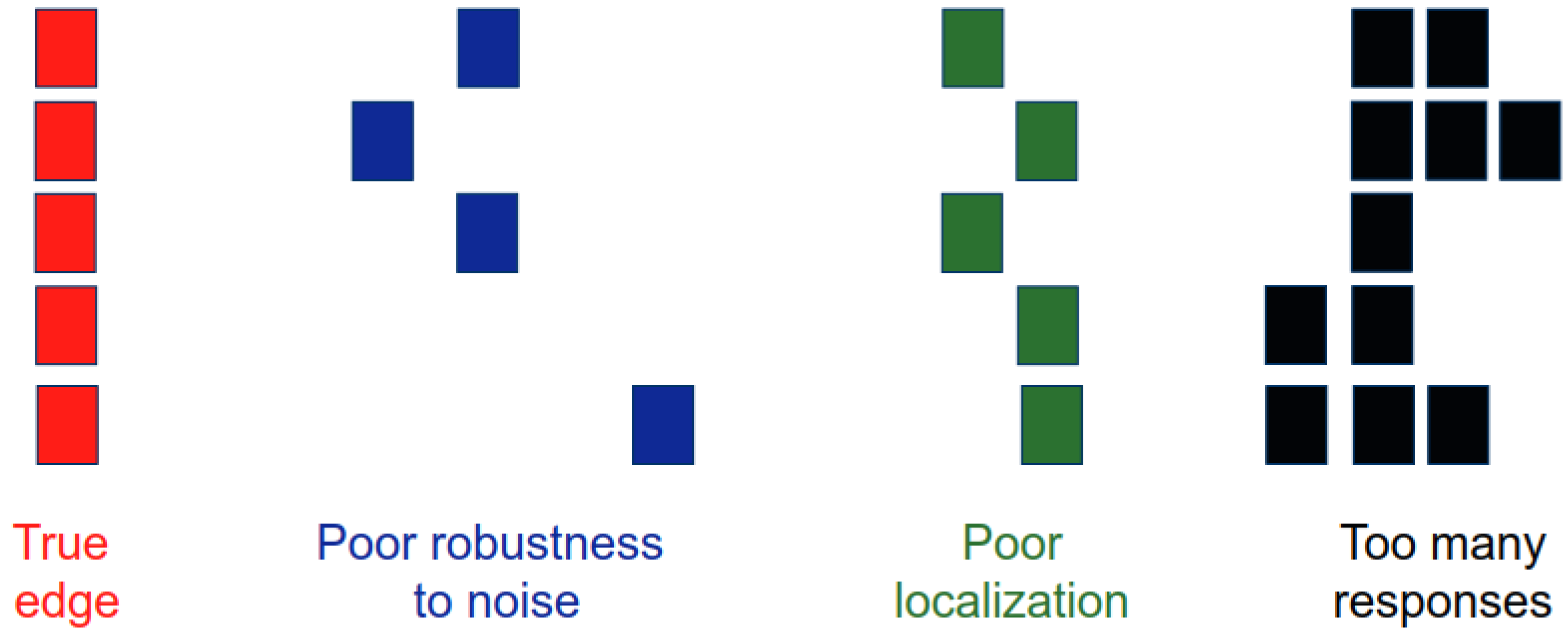


Edge detectors

- Gradient operators
  - Prewit
  - Sobel
- Marr-Hildreth (Laplacian of Gaussian)
- **Canny (Gradient of Gaussian)**

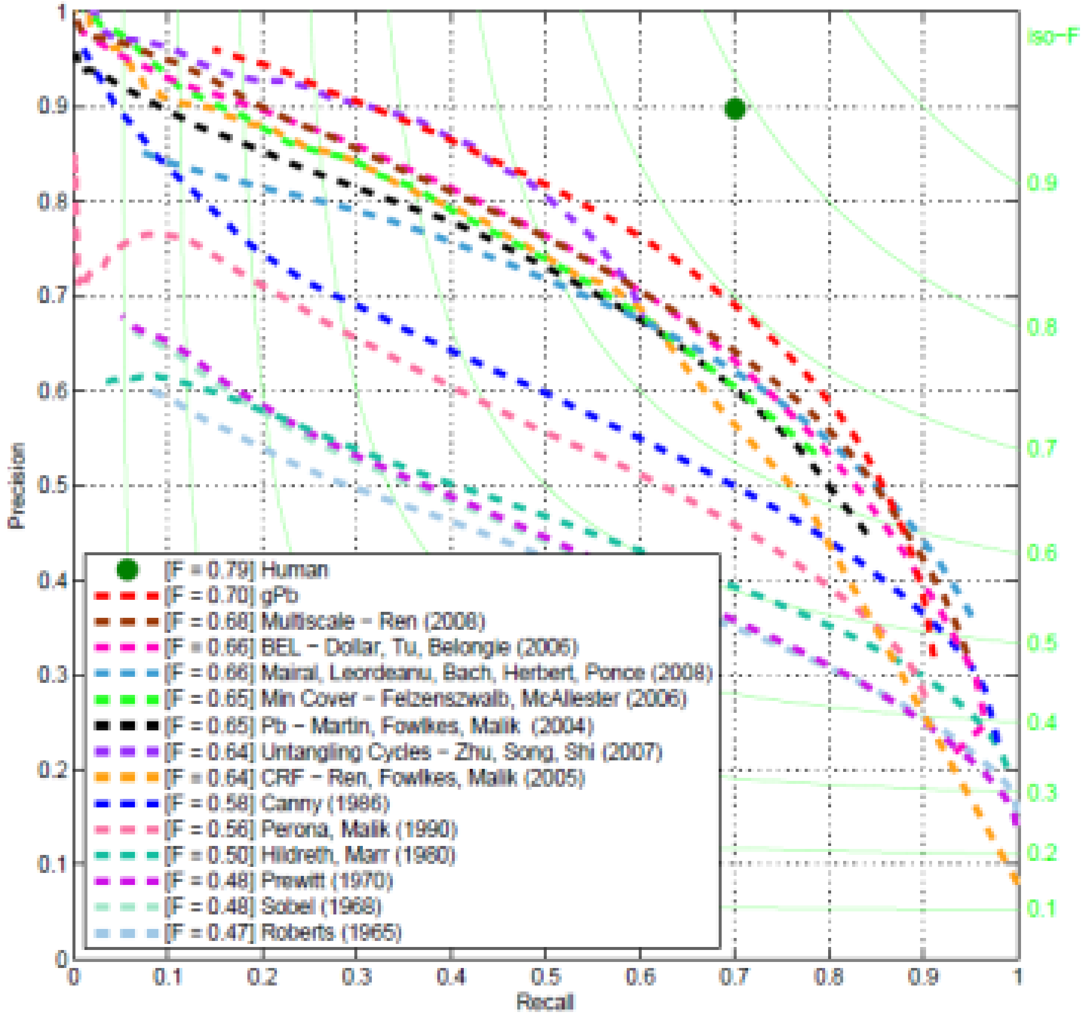
Design Criteria for Edge Detection

- Good detection: find all real edges, ignoring noise or other artifacts
- Good localization
  - as close as possible to the true edges
  - one point only for each true edge point



45 years of boundary detection

[Pre deep learning]



Questions ?