CAP 5415 Computer Vision

Dr. Mubarak Shah

Univ. of Central Florida

Edge Detection

Lecture-3

Contents

Contents

- Gradient operators

Contents

- Gradient operators
  - Prewit

Contents

- Gradient operators
  - Prewit
  - Sobel

Contents

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

Contents

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

Edge detection

Source: D. Lowe

Edge detection

- **Goal:** Identify sudden changes (discontinuities) in an image

Source: D. Lowe

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

Source: D. Lowe

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

Source: D. Lowe

Example

Example

Example

Example

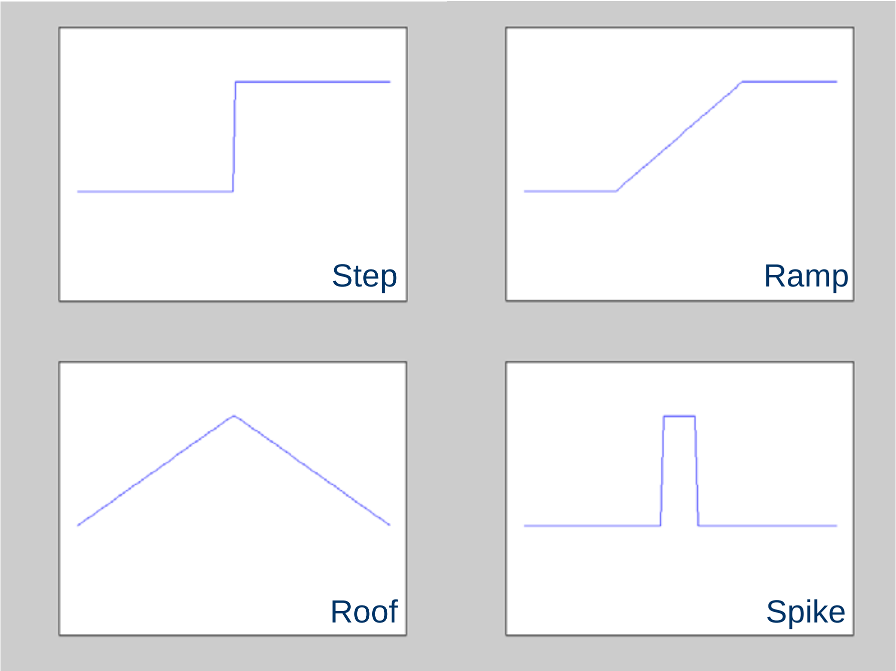
Example

An Application

- What is an object?
- How can we find it?

What is an Edge?

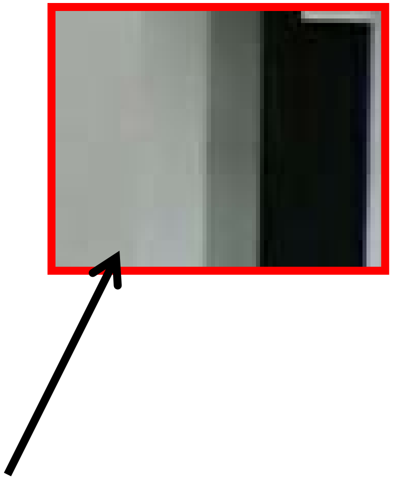
- Discontinuity of intensities in the image
- Edge models
  - Step
  - Roof
  - Ramp
  - Spike



Closeup of edges

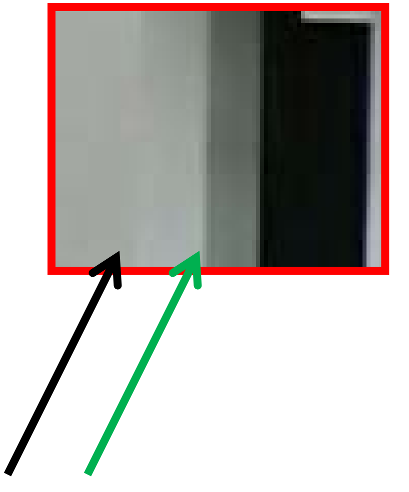
Source: D. Hoiem

Closeup of edges



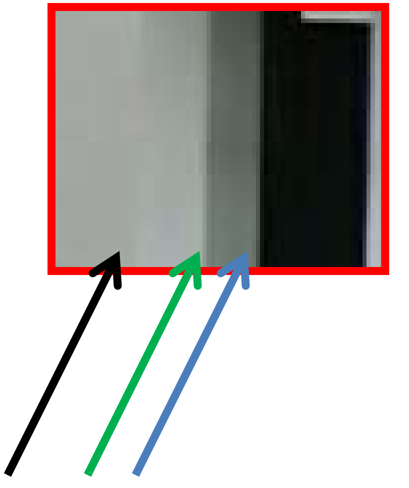
Source: D. Hoiem

Closeup of edges



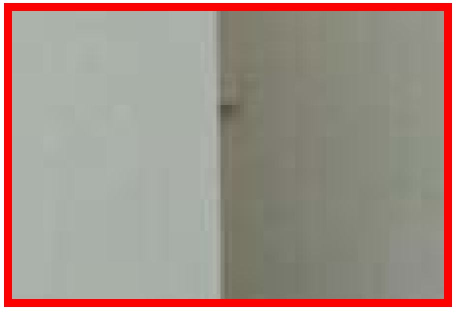
Source: D. Hoiem

Closeup of edges



Source: D. Hoiem

Closeup of edges



Source: D. Hoiem

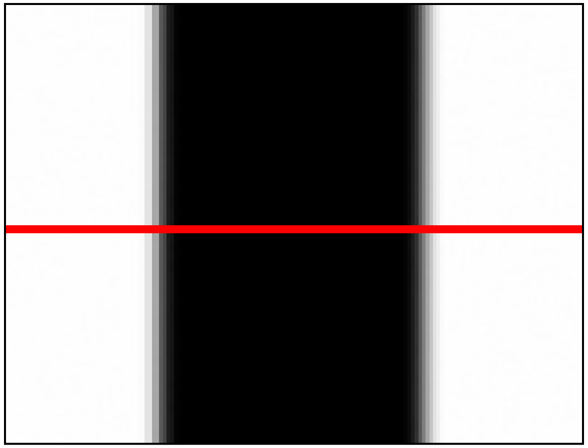
Closeup of edges

Source: D. Hoiem

Characterizing edges

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

image

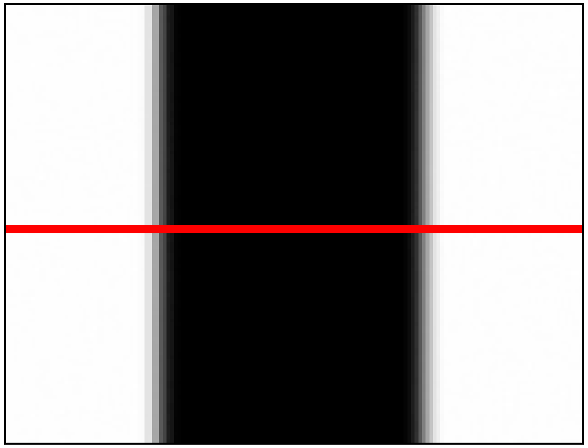


Slide Credit: James Hays

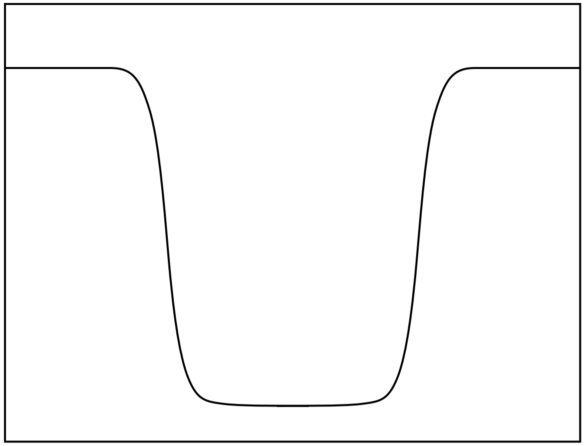
Characterizing edges

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

image



intensity function
(along horizontal scanline)

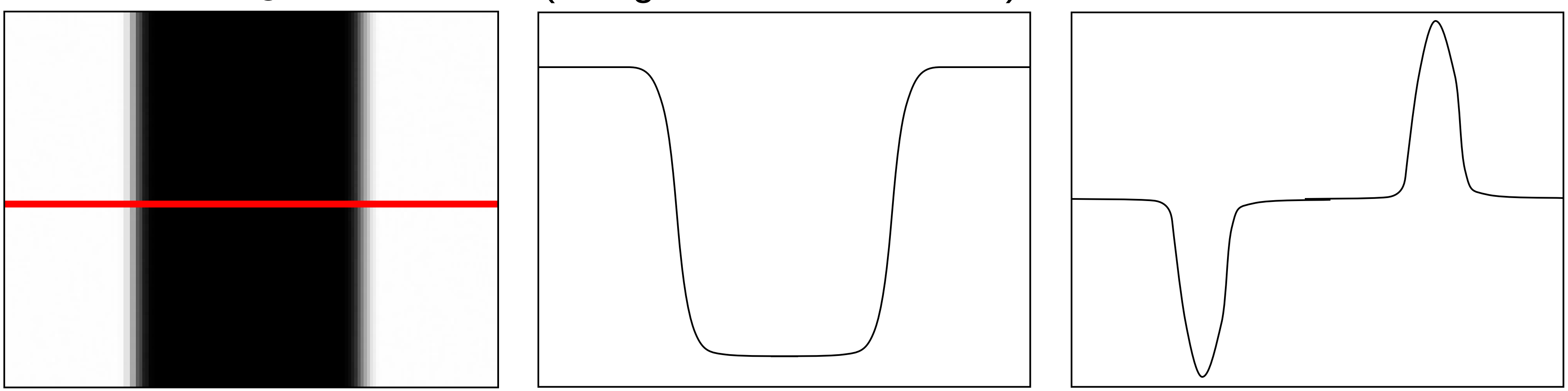


Slide Credit: James Hays

Characterizing edges

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

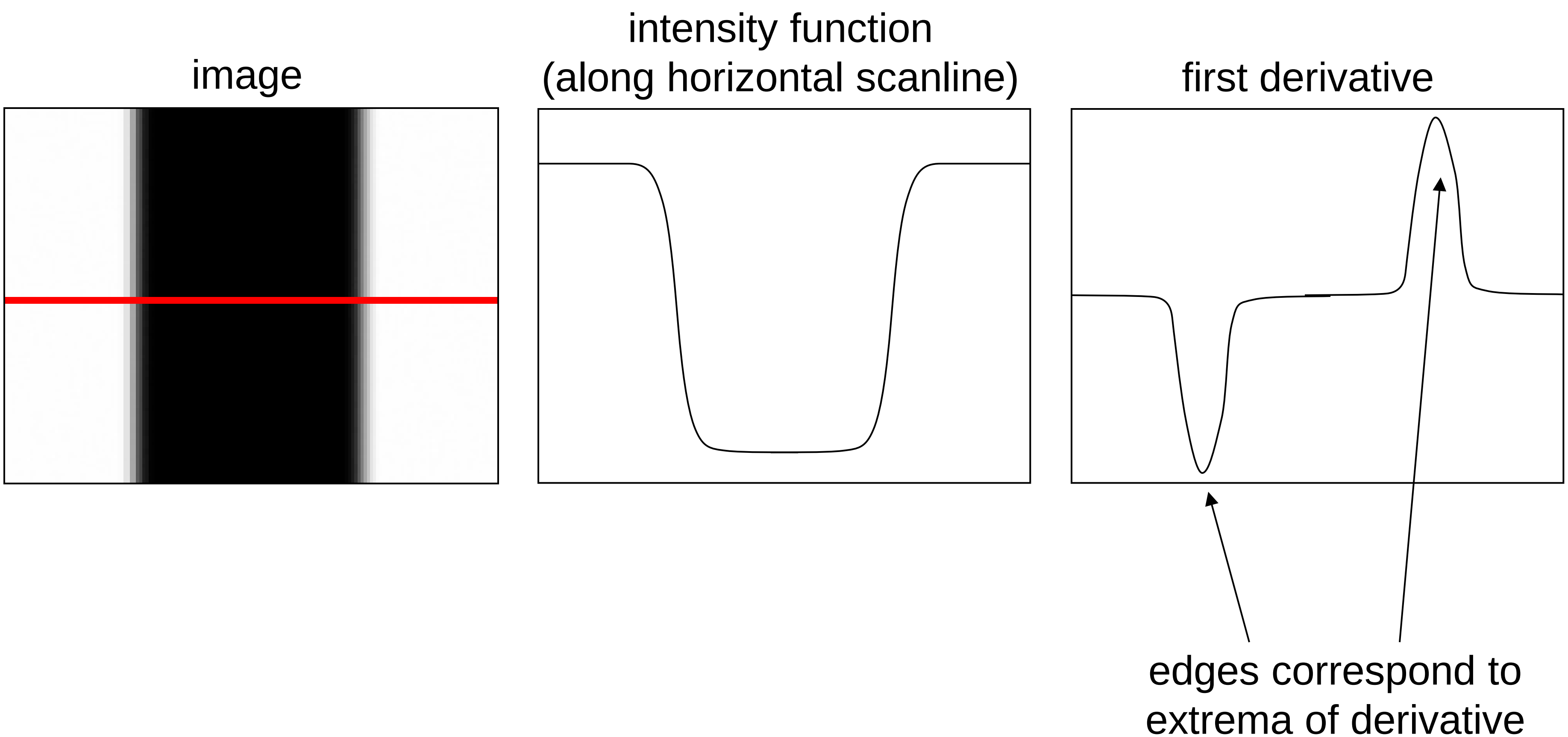
image | intensity function (along horizontal scanline) | first derivative



Slide Credit: James Hays

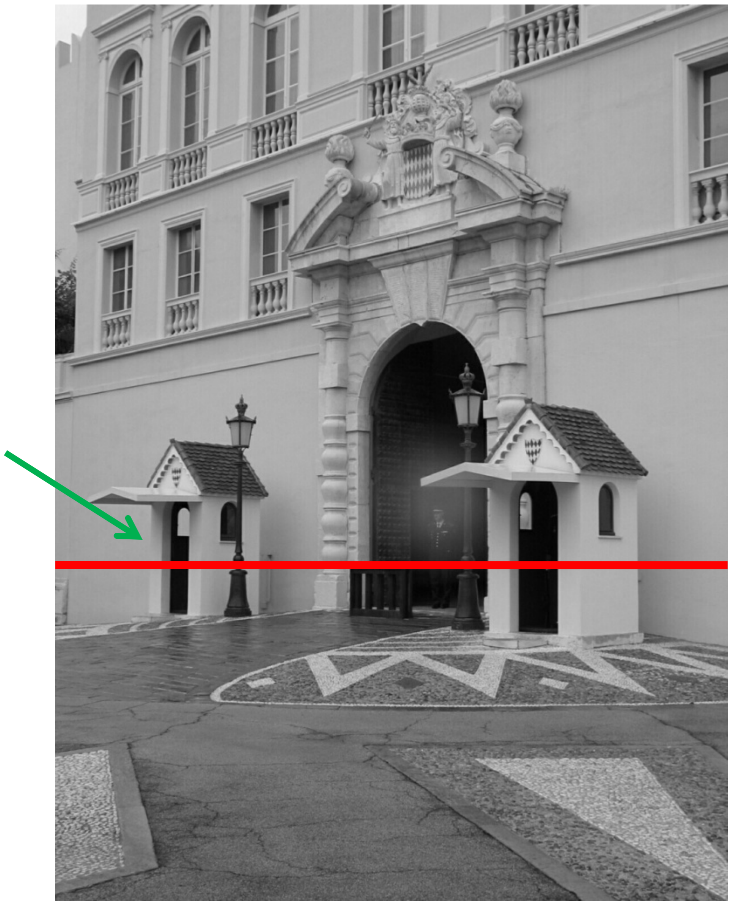
Characterizing edges

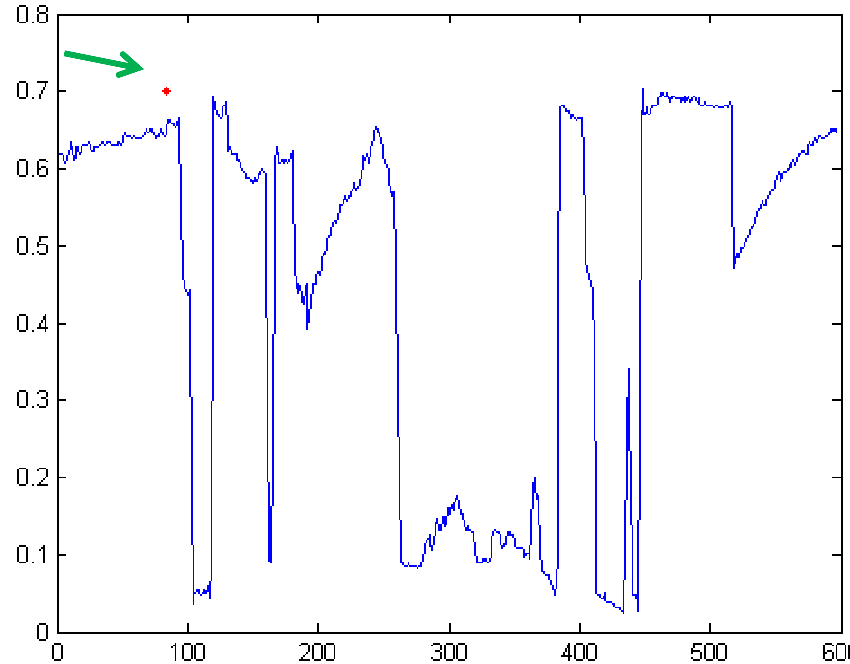
- An edge is a place of rapid change in the image intensity function



Slide Credit: James Hays

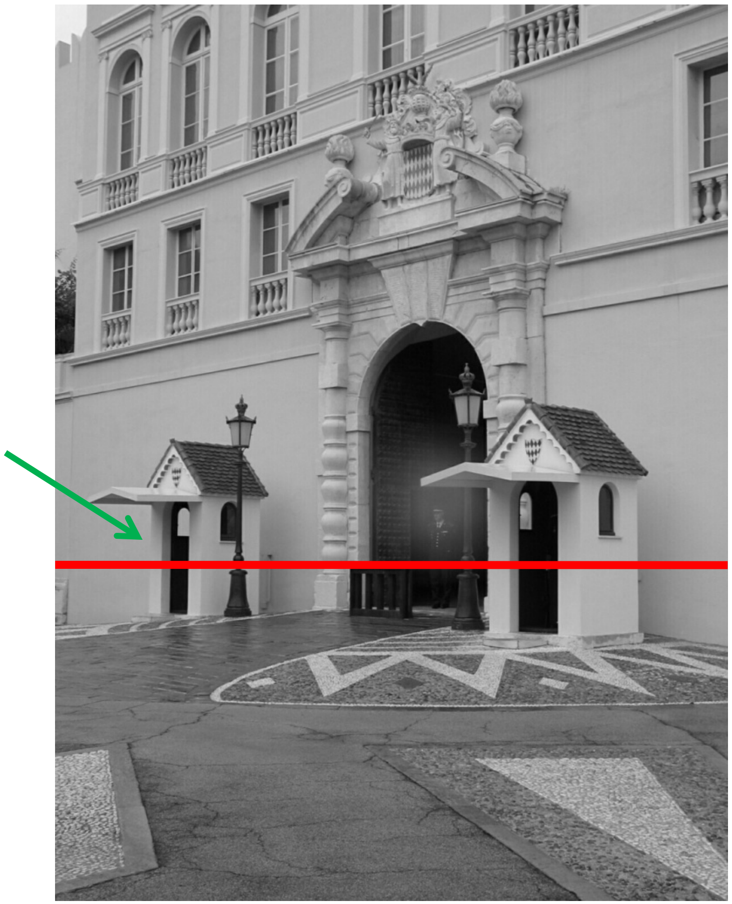
Intensity profile

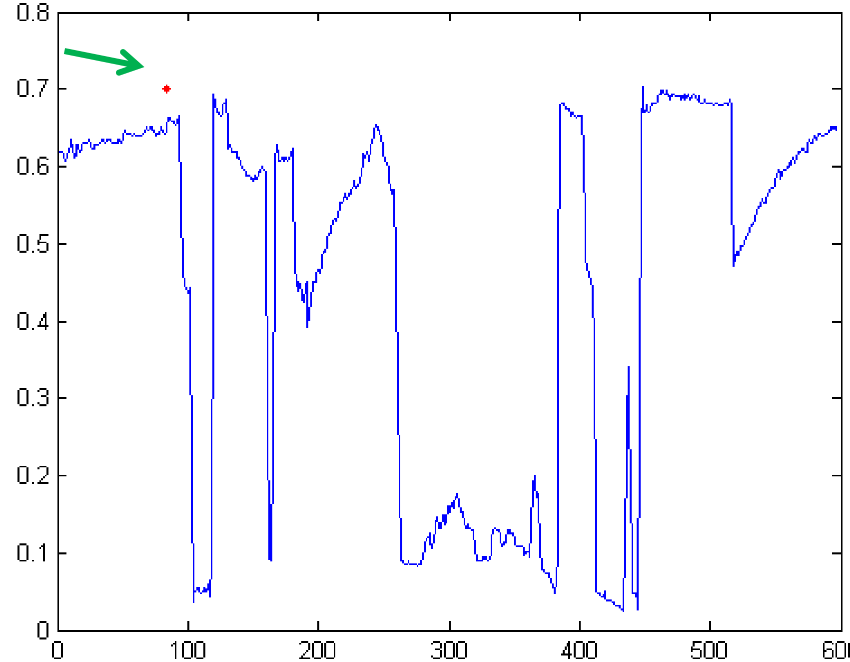


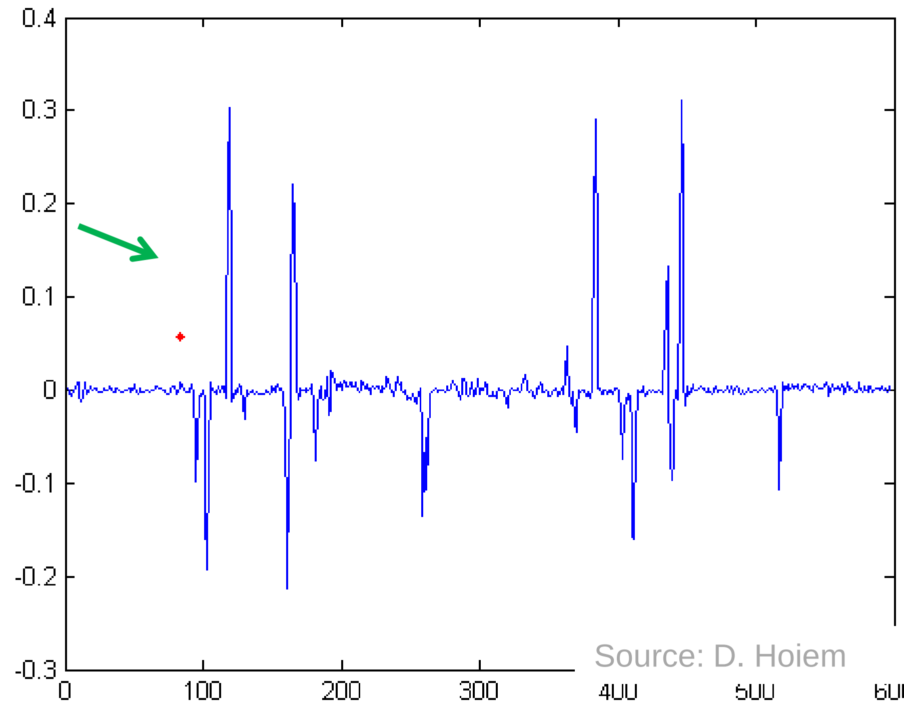


Source: D. Hoiem

Intensity profile

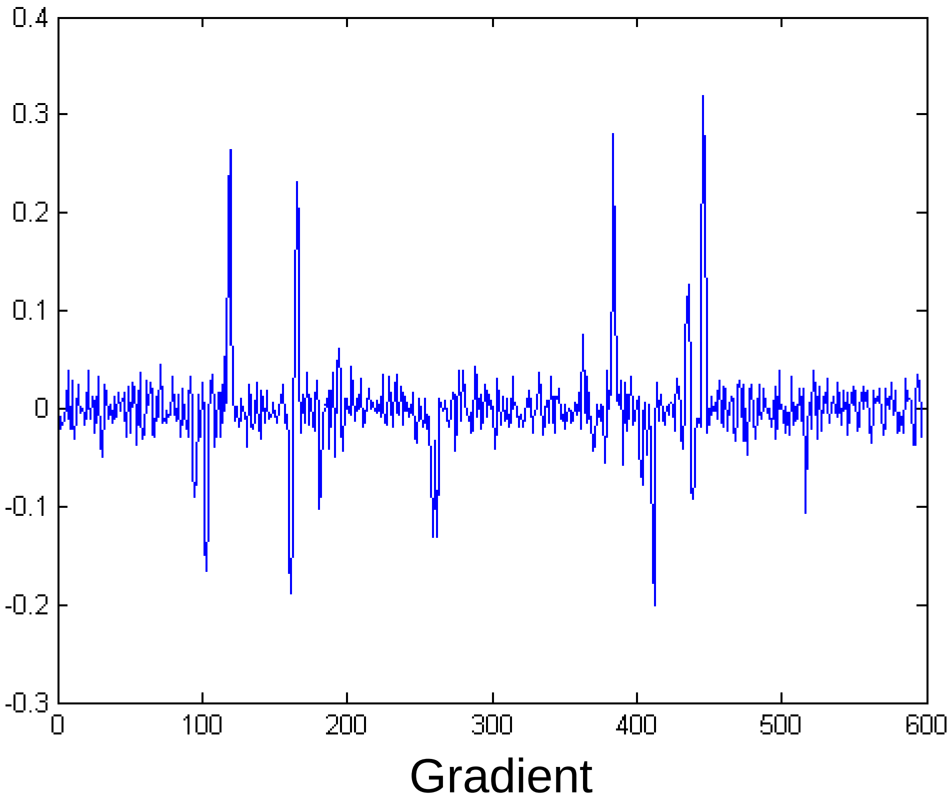






Source: D. Hoiem

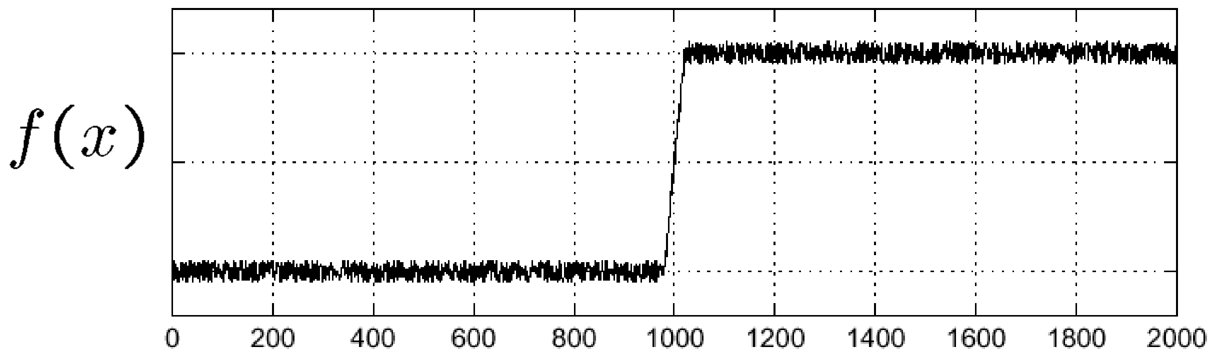
With a little Gaussian noise



Source: D. Hoiem

Effects of noise

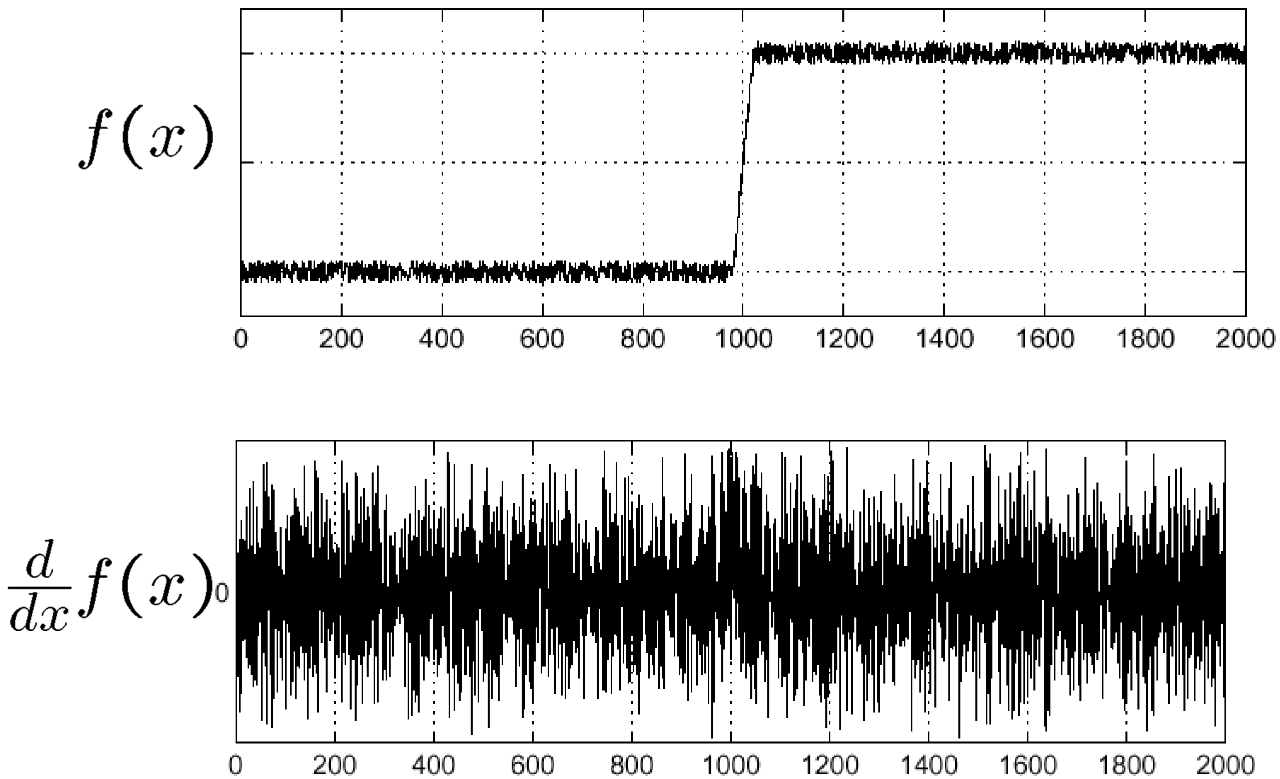
- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



Source: S. Seitz

Effects of noise

- Consider a single row or column of the image
  - Plotting intensity as a function of position gives a signal



Where is the edge?

Source: S. Seitz

Effects of noise

Source: D. Forsyth

Effects of noise

- Difference filters respond strongly to noise

Source: D. Forsyth

Effects of noise

- Difference filters respond strongly to noise
  – Image noise results in pixels that look very different from their neighbors

Source: D. Forsyth

Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response

Source: D. Forsyth

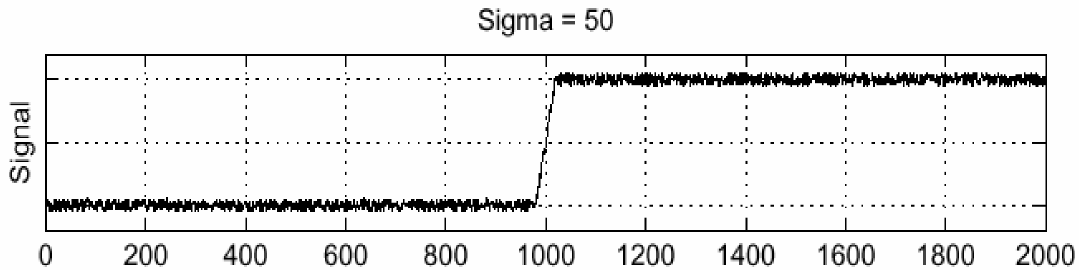
Effects of noise

- Difference filters respond strongly to noise
  - Image noise results in pixels that look very different from their neighbors
  - Generally, the larger the noise the stronger the response
- What can we do about it?

Source: D. Forsyth

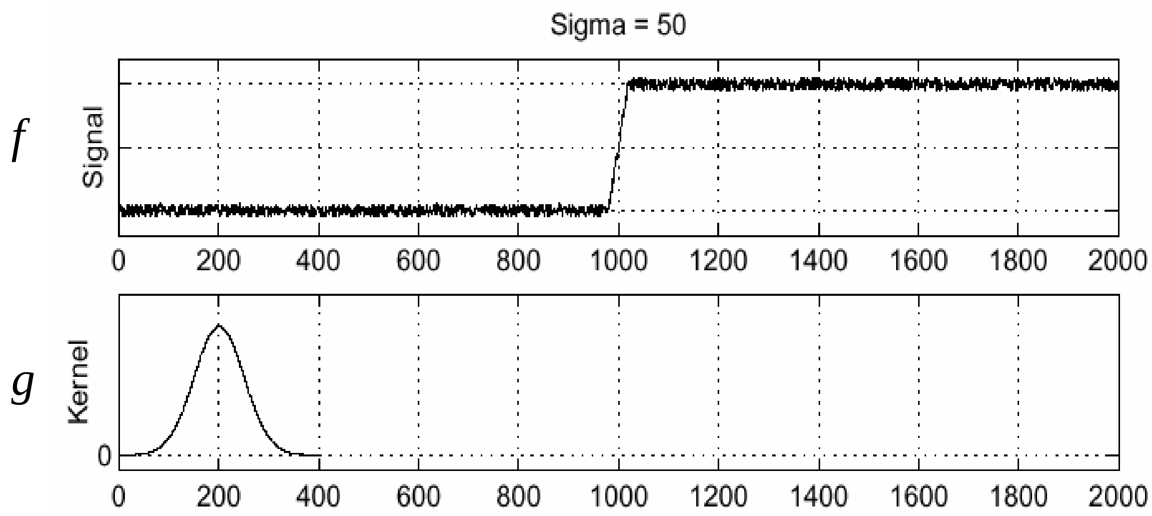
Solution: smooth first

f



Source: S. Seitz

Solution: smooth first



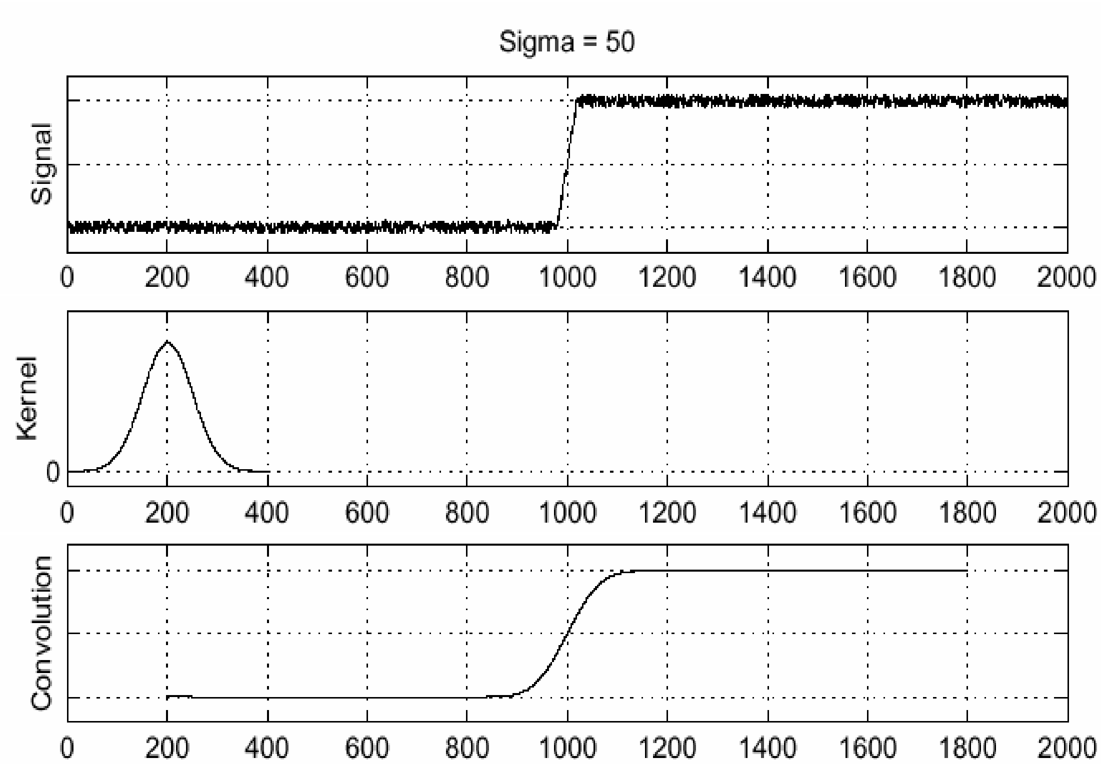
Source: S. Seitz

Solution: smooth first

f

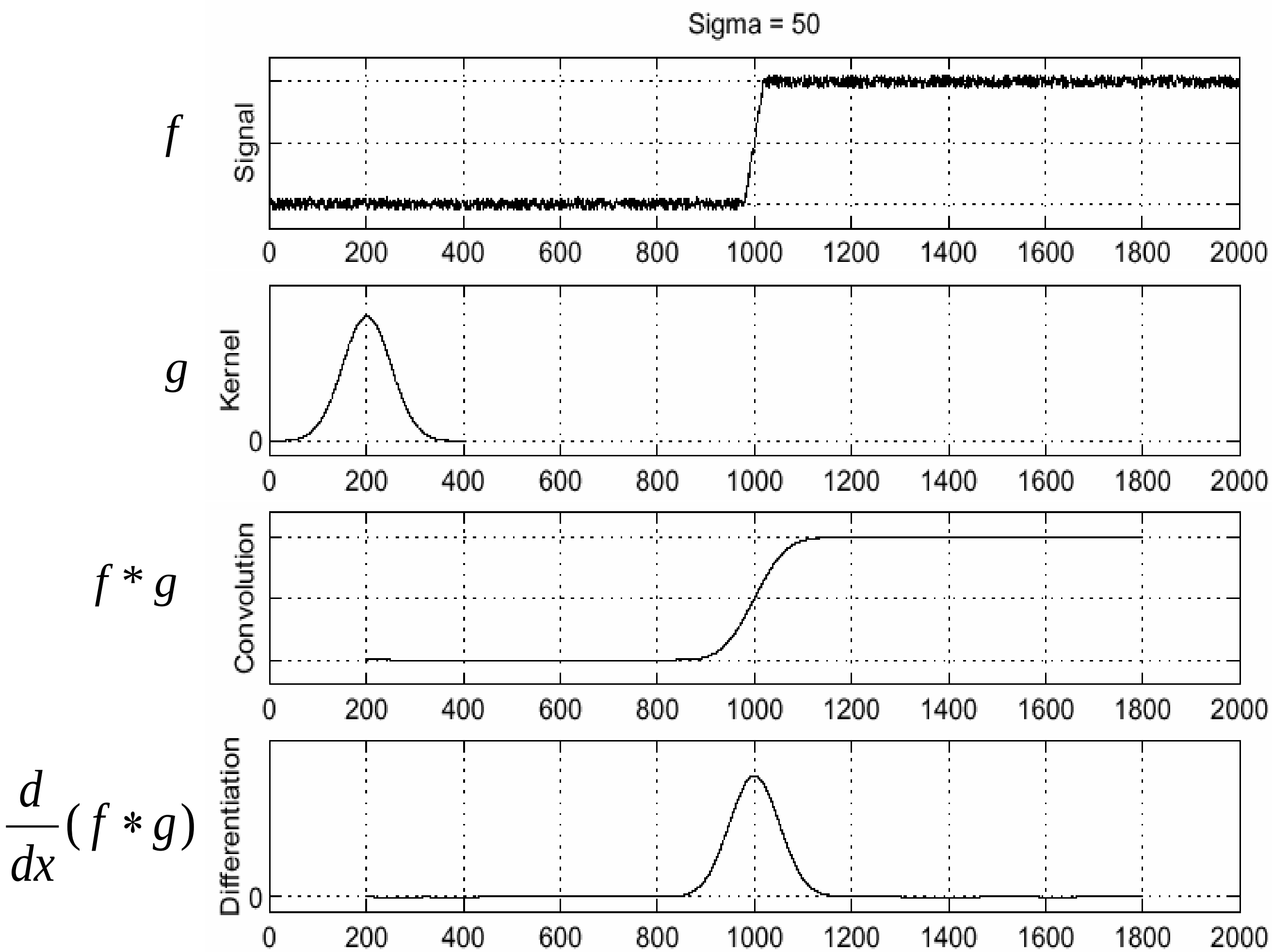
g

$f * g$



Source: S. Seitz

Solution: smooth first



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

Source: S. Seitz

Derivative theorem of convolution

Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

Source: S. Seitz

Derivative theorem of convolution

- Differentiation is convolution, and convolution is associative:

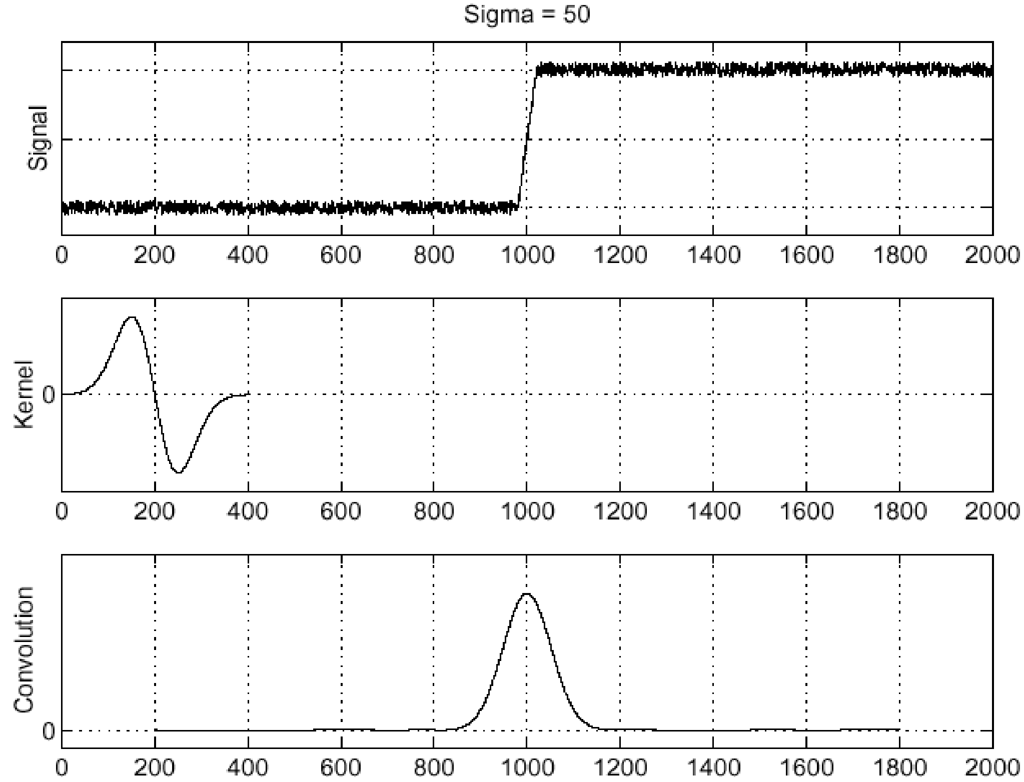
$$\frac{d}{dx}(f*g) = f*\frac{d}{dx}g$$

- This saves us one operation:

f

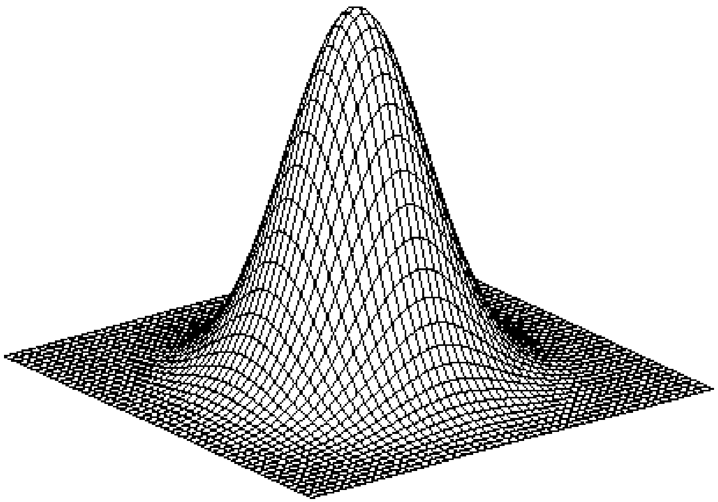
$\frac{d}{dx}g$

$f*\frac{d}{dx}g$

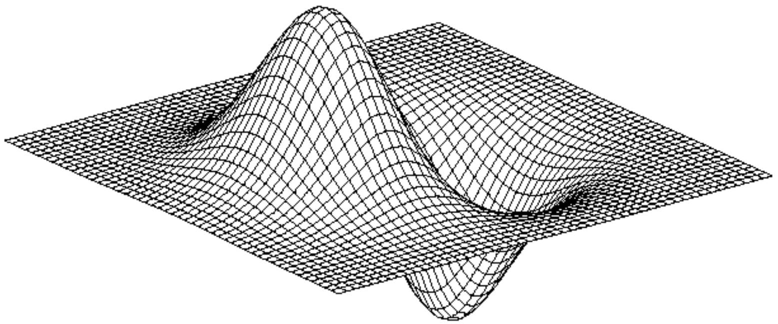


Source: S. Seitz

Derivative of Gaussian filter



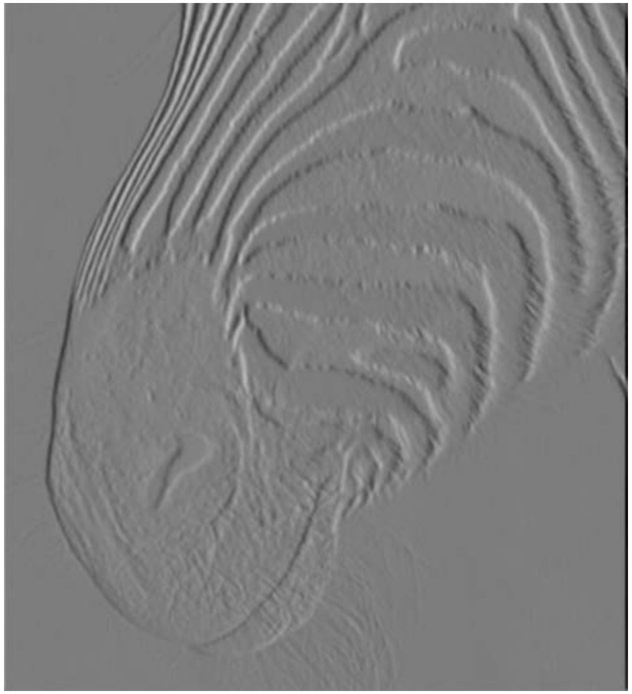
* [1 -1] =



Tradeoff between smoothing and localization

Source: D. Forsyth

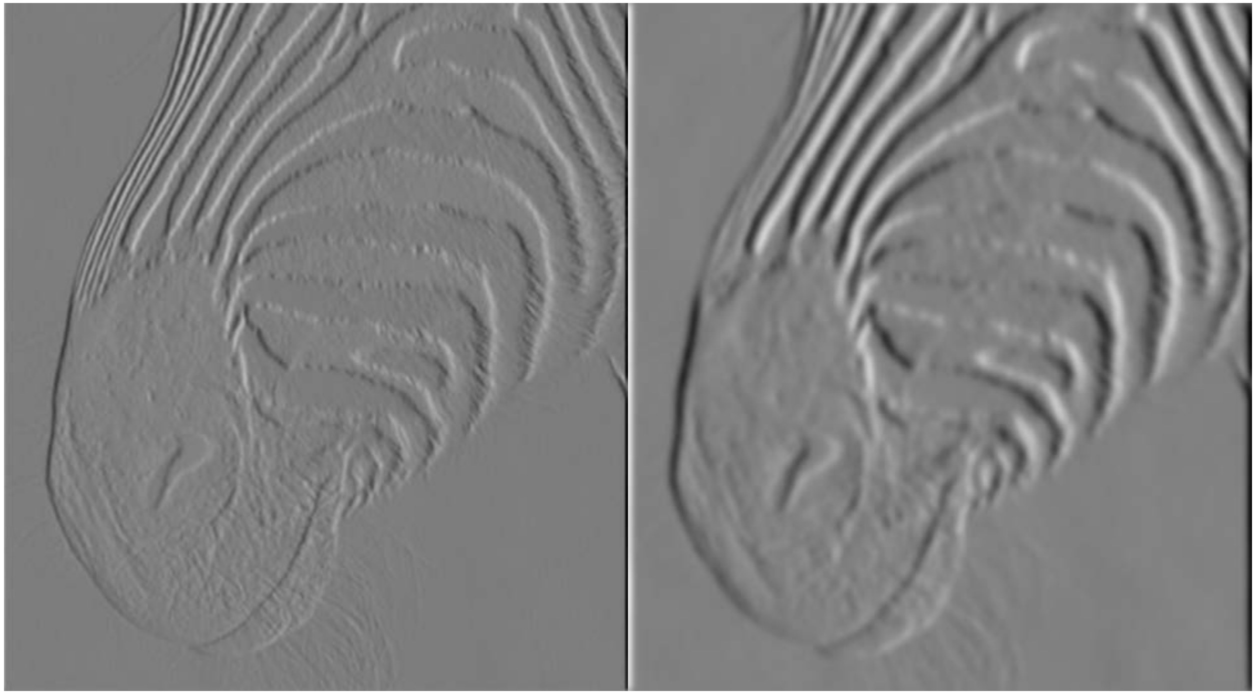
Tradeoff between smoothing and localization



1 pixel

Source: D. Forsyth

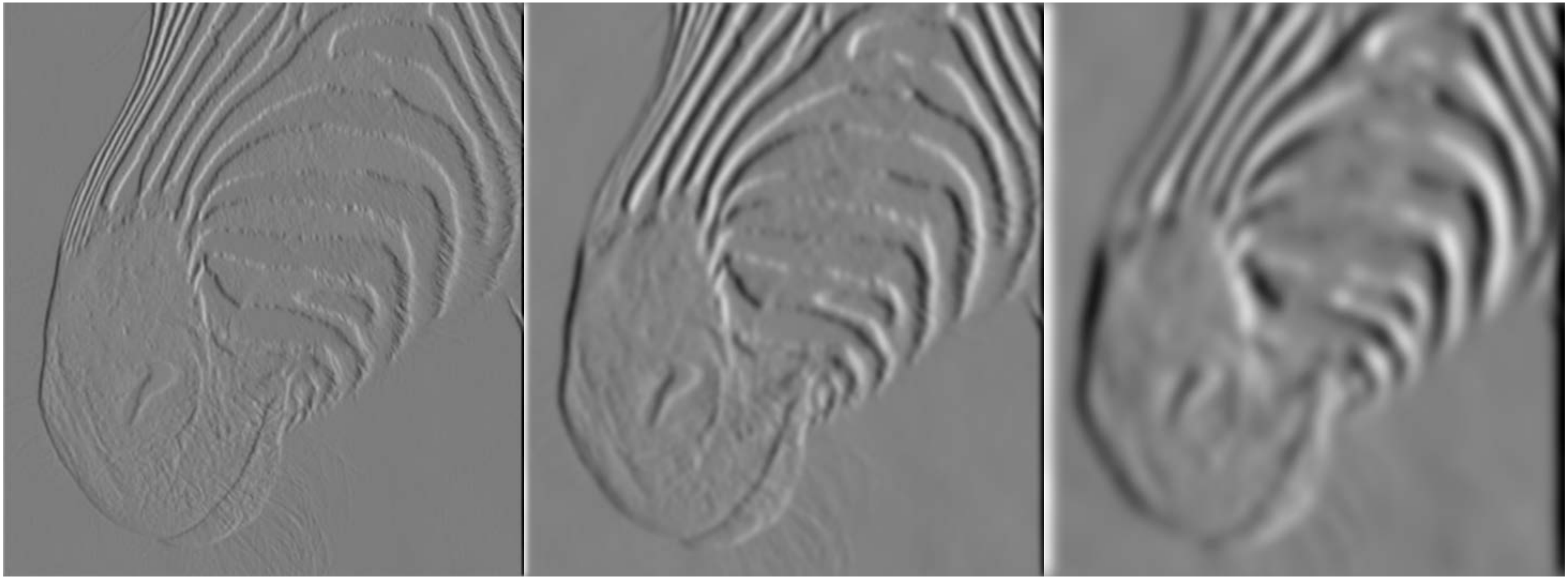
Tradeoff between smoothing and localization



1 pixel 3 pixels

Source: D. Forsyth

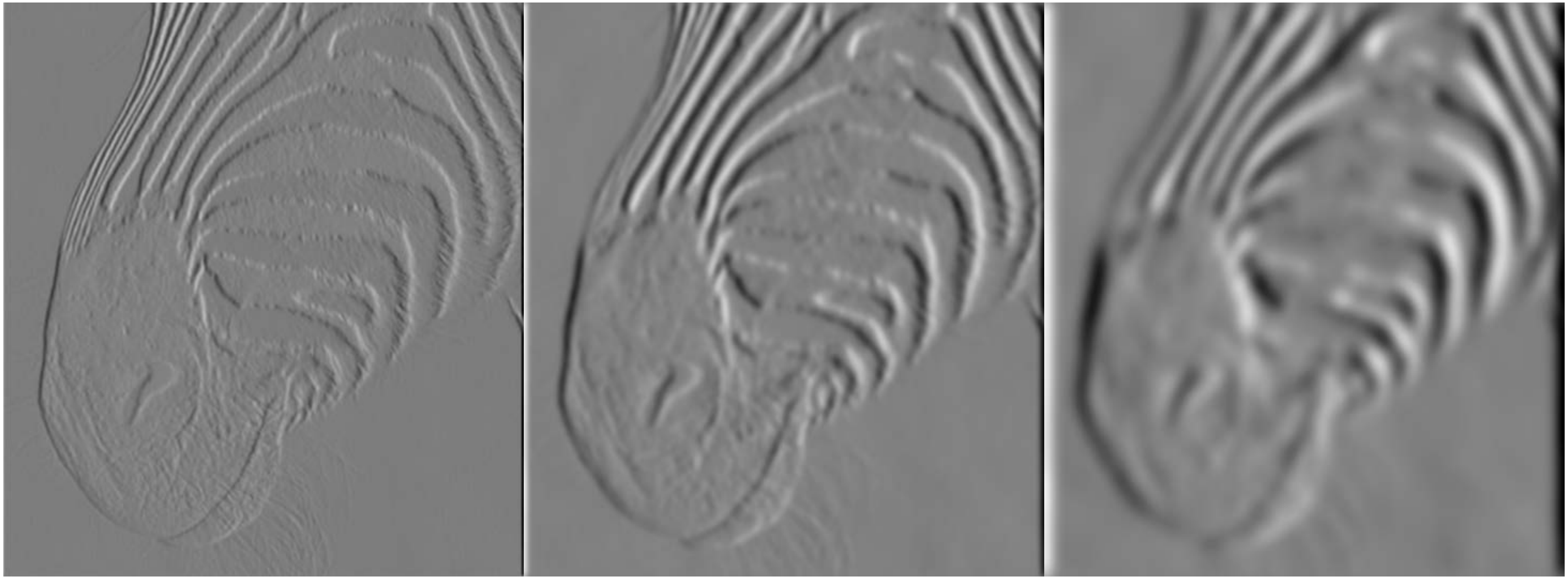
Tradeoff between smoothing and localization



1 pixel   3 pixels   7 pixels

Source: D. Forsyth

Tradeoff between smoothing and localization



1 pixel | 3 pixels | 7 pixels

- Smoothed derivative removes noise, but blurs edge. Also finds edges at different “scales”.

Source: D. Forsyth

Derivatives and Noise

Derivatives and Noise

Derivatives and Noise

- Strongly affected by noise

Derivatives and Noise

- Strongly affected by noise
    - obvious reason: image noise results in pixels that look very different from their neighbors

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

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

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?

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

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

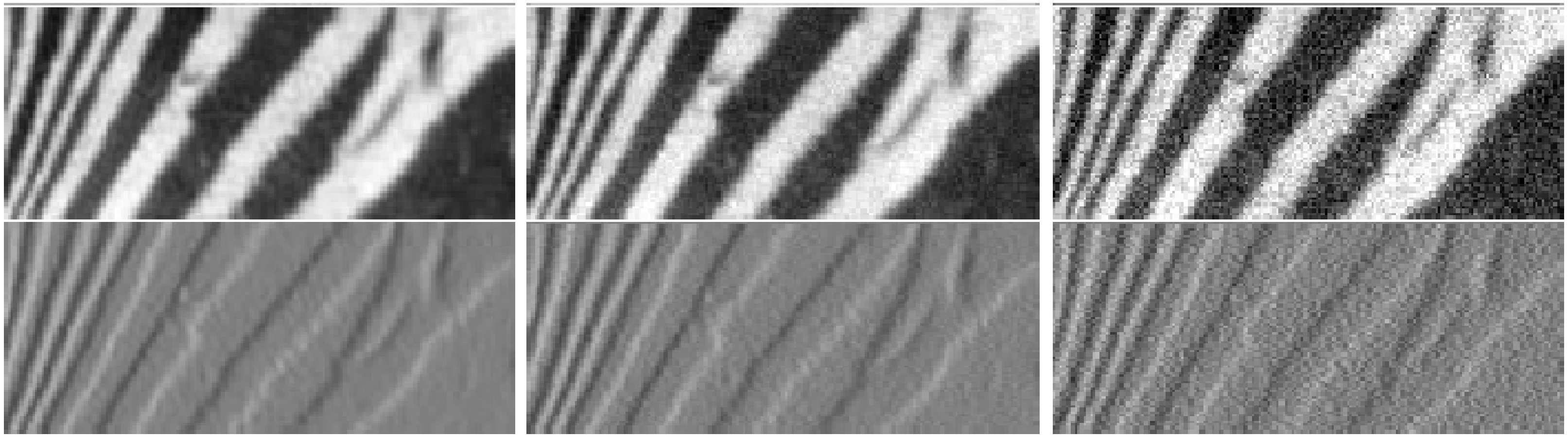
- What is to be done?
  - Neighboring pixels look alike
  - Pixel along an edge look alike

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 from their neighbors (possibly noise) to look like neighbors

Derivatives and Noise



Increasing noise →

Zero mean additive gaussian noise

Image Smoothing

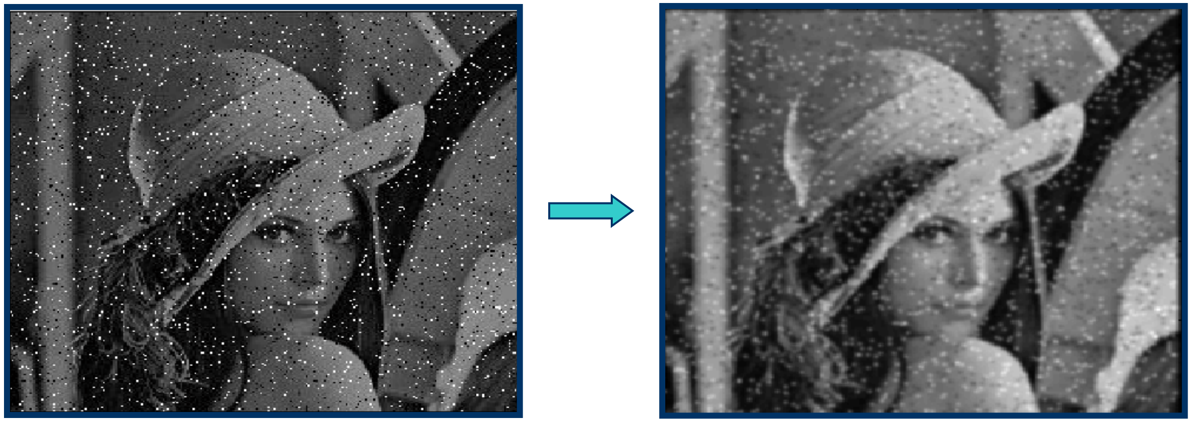
Image Smoothing

- Expect pixels to "**be like**" their neighbors
  - Relatively few reflectance changes

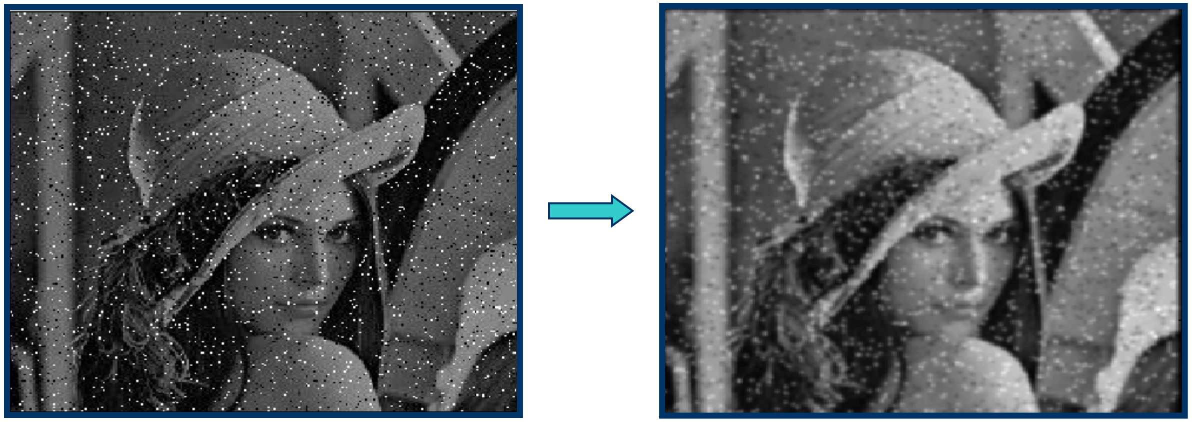
Image Smoothing

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

Gaussian Smoothing



Gaussian Smoothing



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

Gaussian Smoothing

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

- Scale of Gaussian σ

Gaussian Smoothing

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

- Scale of Gaussian σ
  - As σ increases, more pixels are involved in average

Gaussian Smoothing

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

- Scale of Gaussian σ
  - As σ increases, more pixels are involved in average
  - As σ increases, image is more blurred

Gaussian Smoothing

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

- 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

Edge Detectors

Edge Detectors

- Gradient operators
  - Prewit
  - Sobel

Edge Detectors

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

Edge Detectors

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

Prewitt and Sobel Edge Detector

Prewitt and Sobel Edge Detector

- Compute derivatives
  - In x and y directions

Prewitt and Sobel Edge Detector

- Compute derivatives
  - In x and y directions
- Find gradient magnitude

Prewitt and Sobel Edge Detector

- Compute derivatives
  - In x and y directions
- Find gradient magnitude
- Threshold gradient magnitude

Prewitt Edge Detector

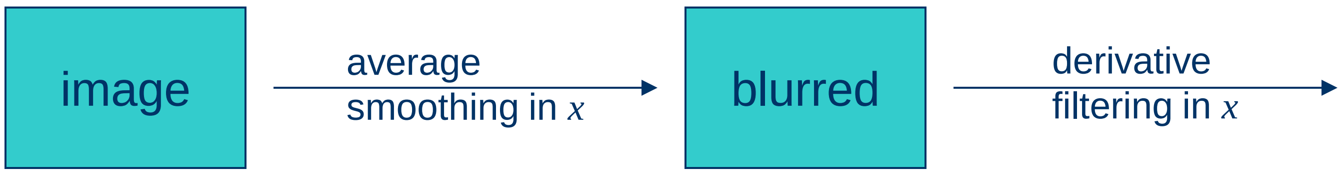
Prewitt Edge Detector

image

Prewitt Edge Detector

Prewitt Edge Detector

Prewitt Edge Detector



Prewitt Edge Detector

image $\xrightarrow[\text{smoothing in } x]{\text{average}}$ blurred $\xrightarrow[\text{filtering in } x]{\text{derivative}}$ edges in x

Prewitt Edge Detector

$$\text{image} \xrightarrow[\text{smoothing in } x]{\text{average}} \text{blurred} \xrightarrow[\text{filtering in } x]{\text{derivative}} \text{edges in } x$$

$$\boxed{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}}$$

Prewitt Edge Detector

image $\xrightarrow{\text{average smoothing in } x}$ blurred $\xrightarrow{\text{derivative filtering in } x}$ edges in x

$$\boxed{\begin{bmatrix}1\\1\\1\end{bmatrix}}$$

and

Prewitt Edge Detector

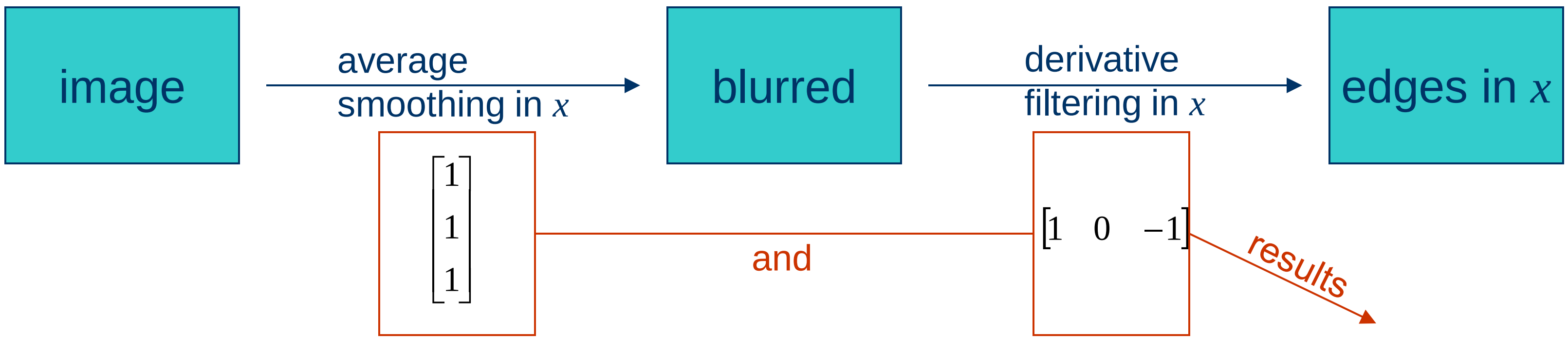
image $\xrightarrow[\text{smoothing in } x]{\text{average}}$ blurred $\xrightarrow[\text{filtering in } x]{\text{derivative}}$ edges in x

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

and

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

Prewitt Edge Detector

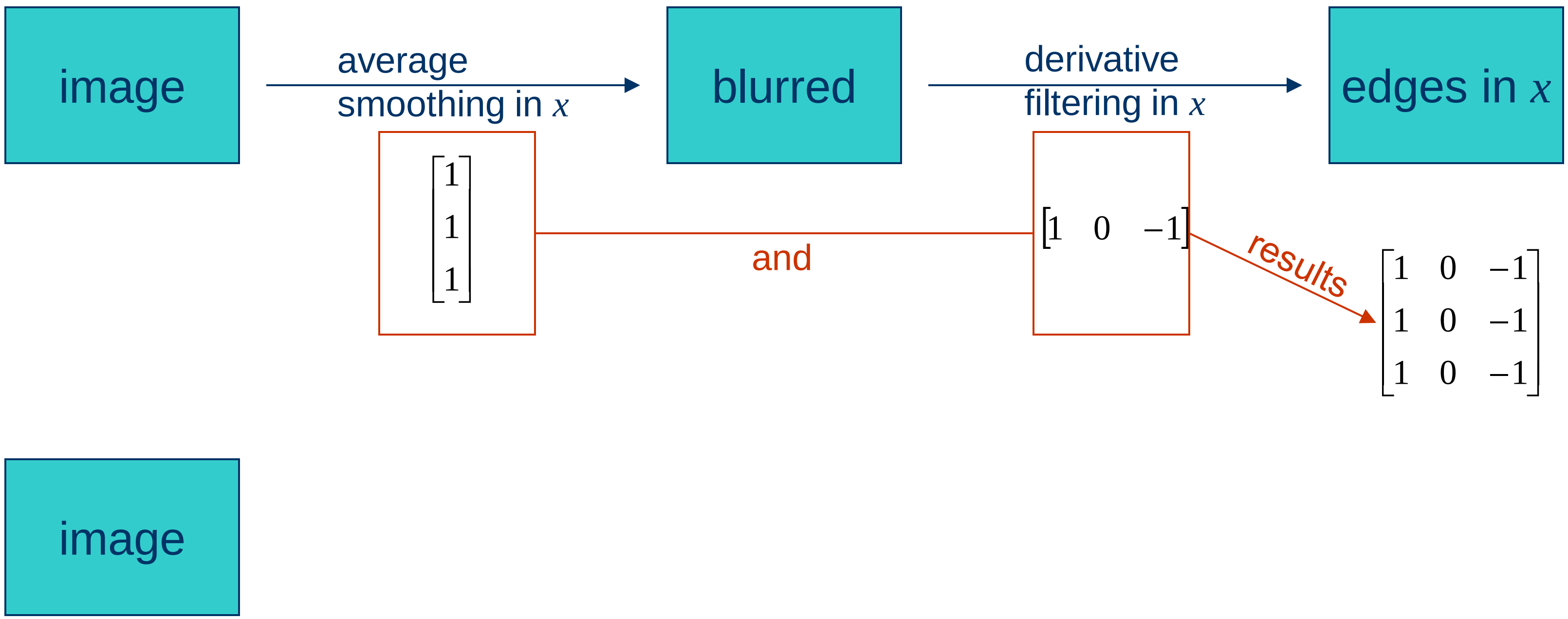


Prewitt Edge Detector

image $\xrightarrow{\text{average smoothing in } x}$ blurred $\xrightarrow{\text{derivative filtering in } x}$ edges in x

$$\boxed{\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}} \quad \text{and} \quad \boxed{\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}} \xrightarrow{\text{results}} \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

Prewitt Edge Detector



Prewitt Edge Detector

image $\xrightarrow{\text{average smoothing in } x}$ blurred $\xrightarrow{\text{derivative filtering in } x}$ edges in x

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix}$$

and

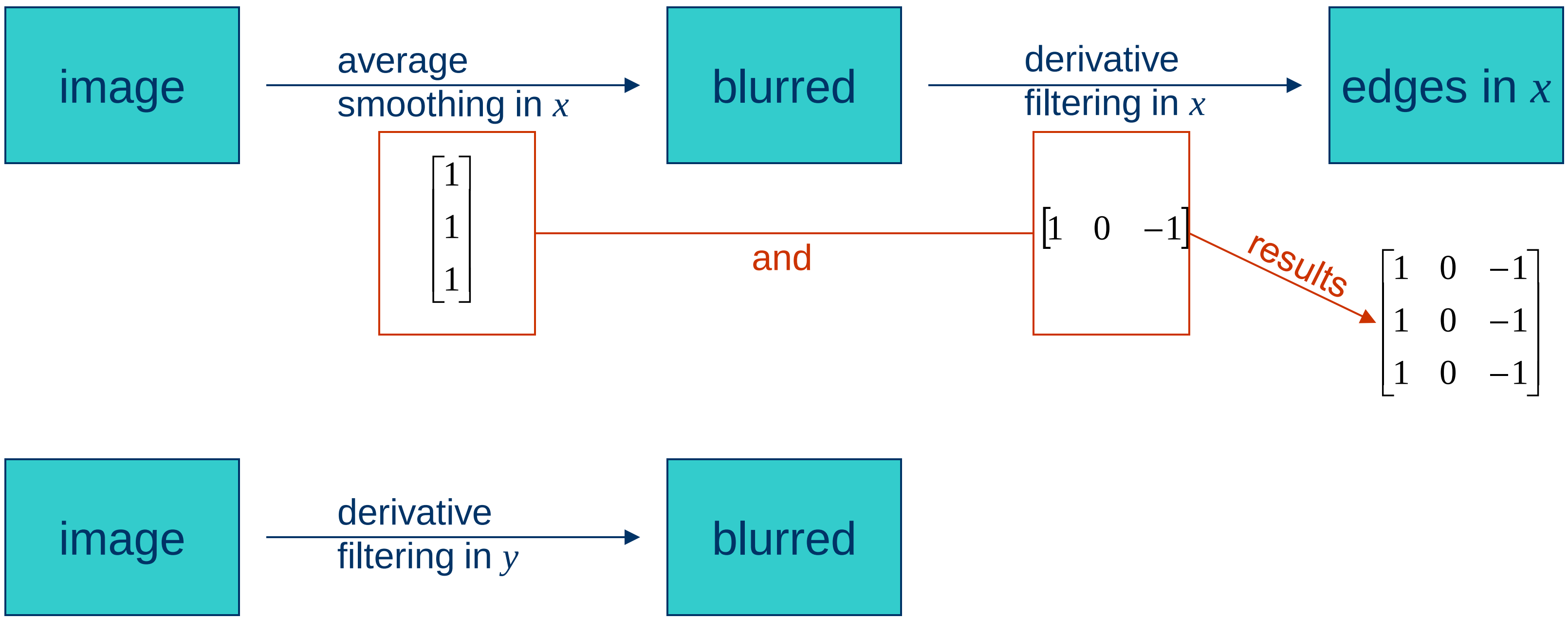
$$\begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

results

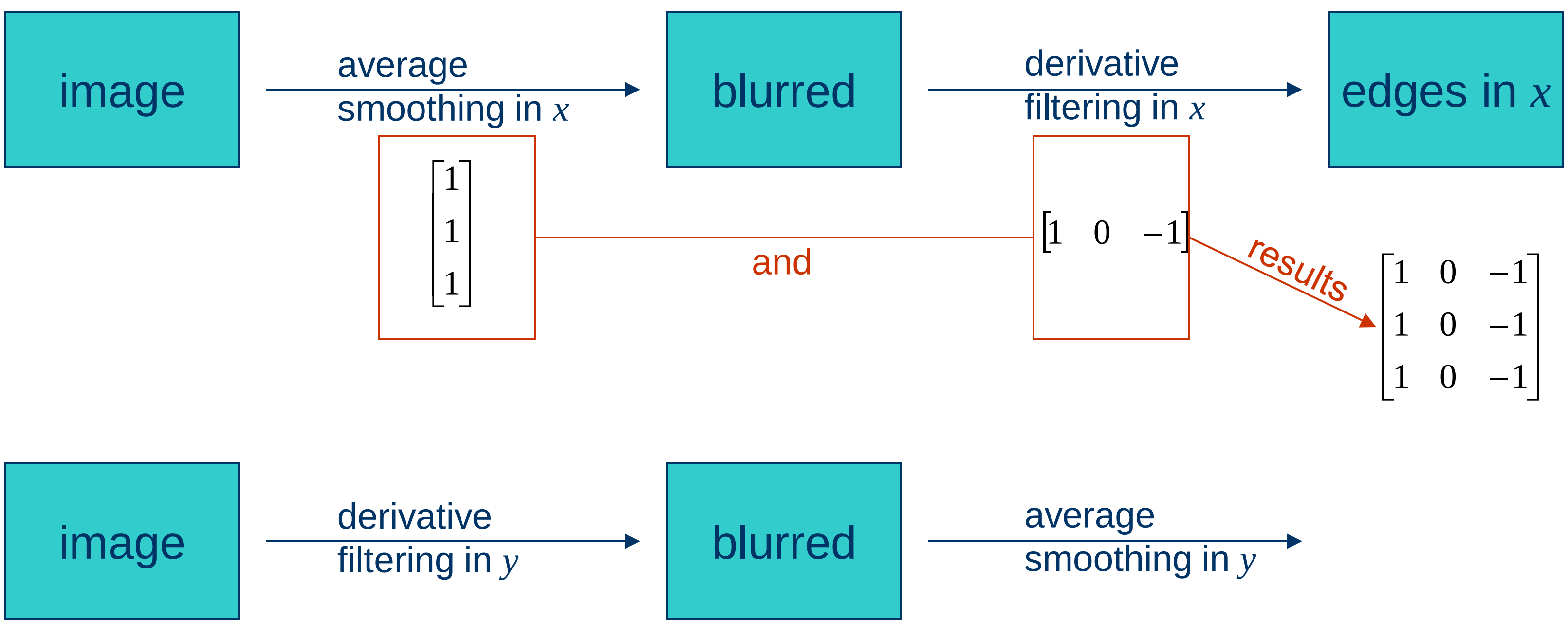
$$\begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

image $\xrightarrow{\text{derivative filtering in } y}$

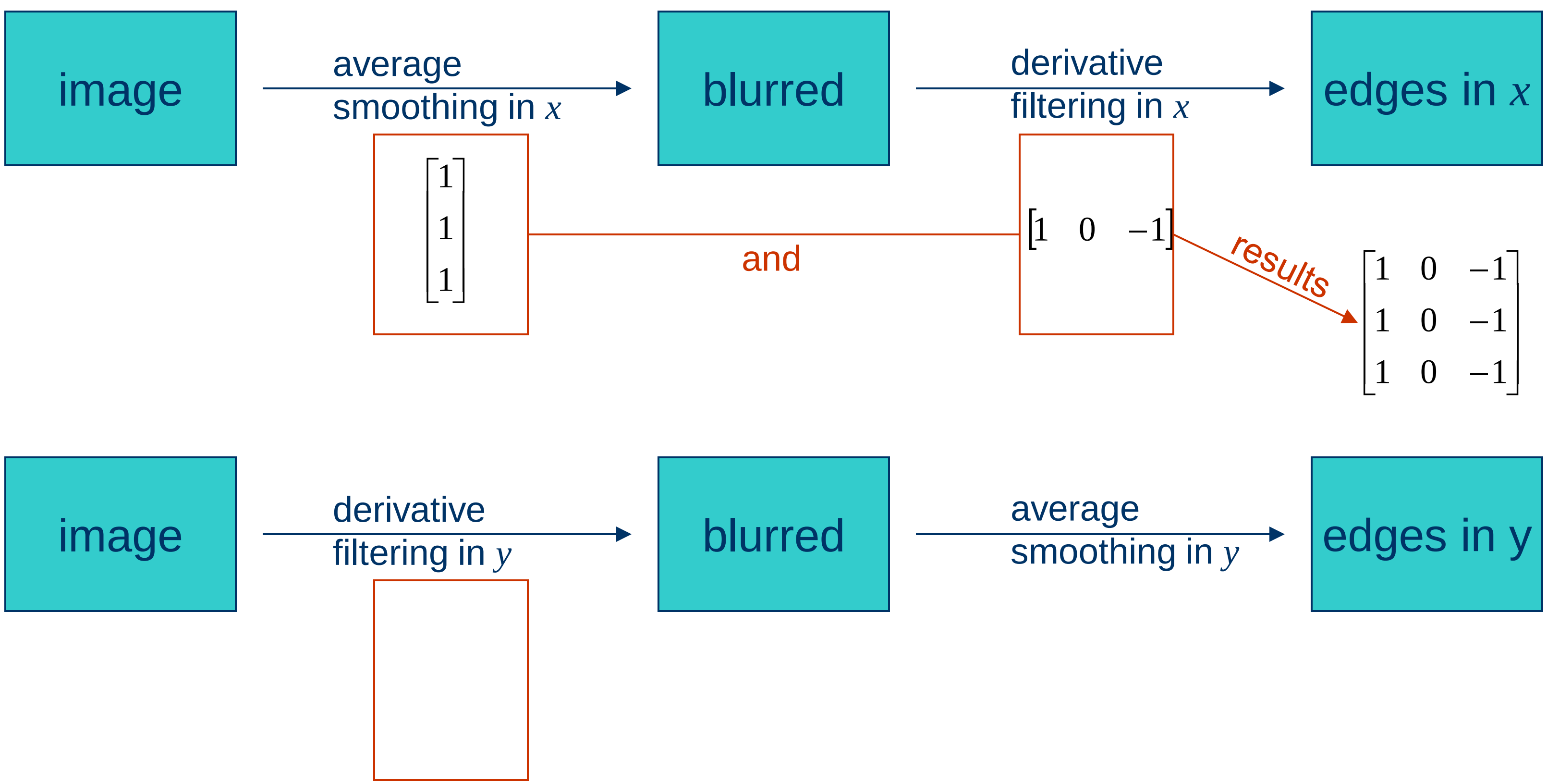
Prewitt Edge Detector



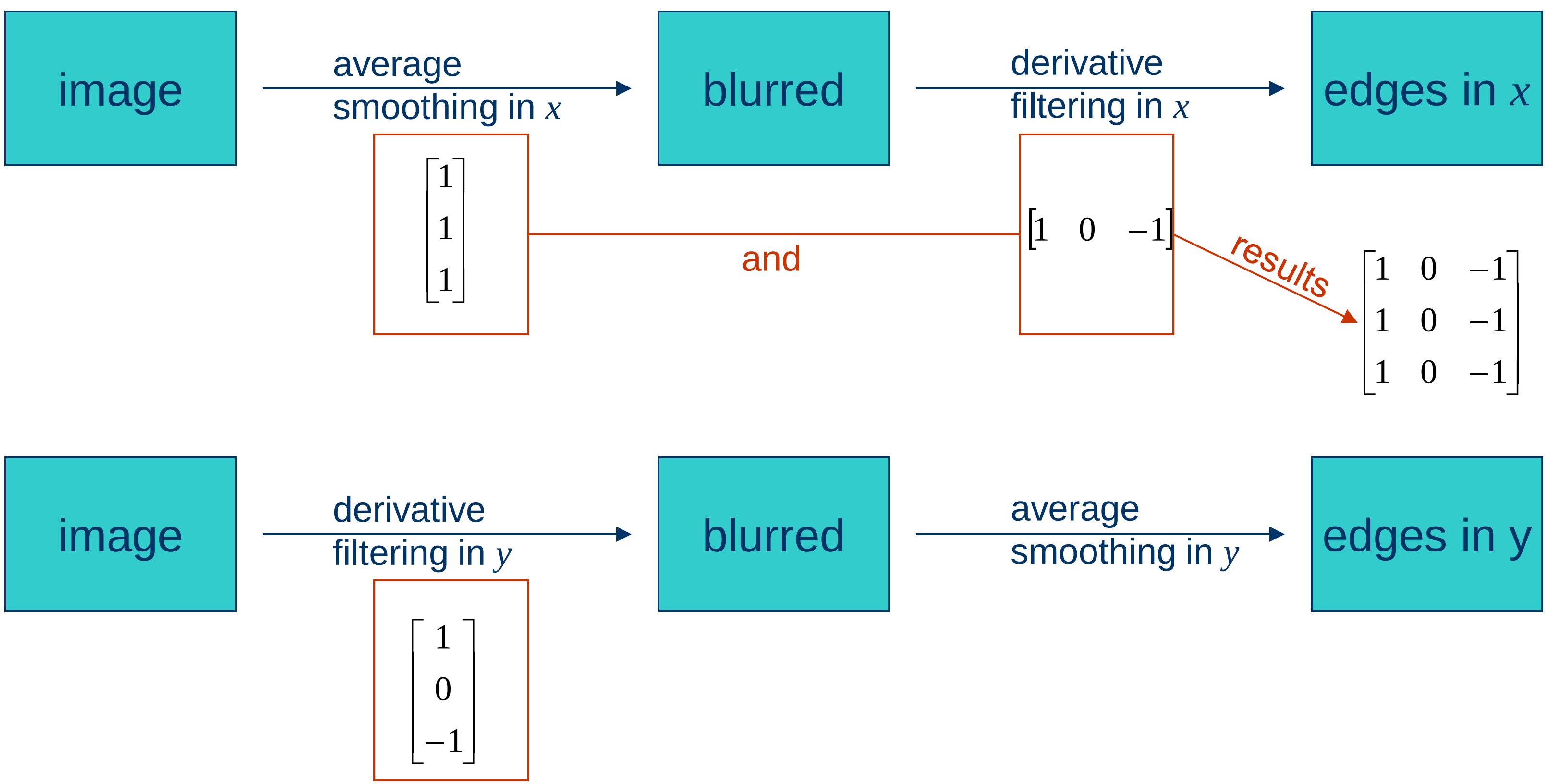
Prewitt Edge Detector



Prewitt Edge Detector



Prewitt Edge Detector



Prewitt Edge Detector

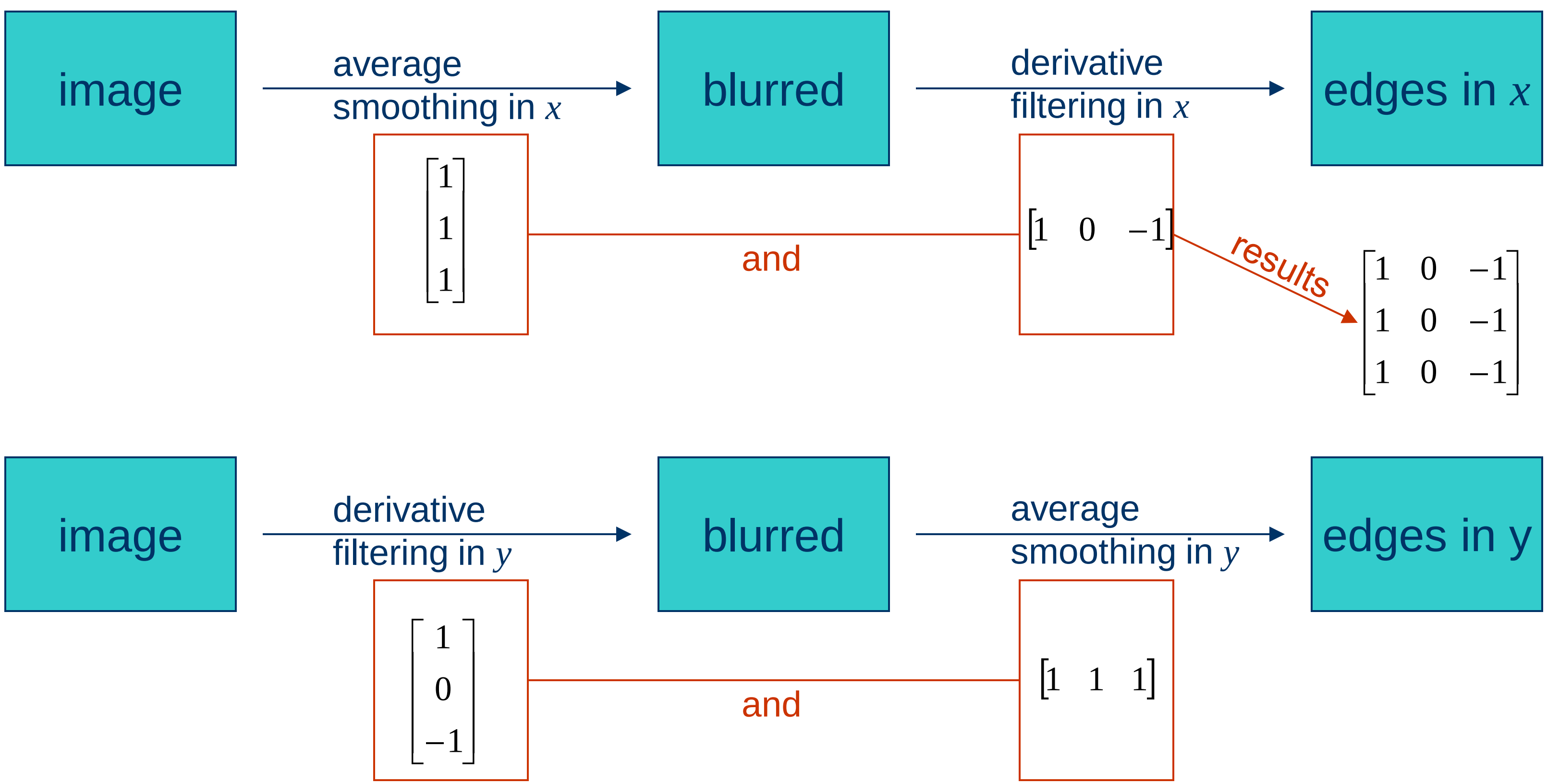
image $\xrightarrow{\text{average smoothing in } x}$ blurred $\xrightarrow{\text{derivative filtering in } x}$ edges in x

$$\begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 & -1 \end{bmatrix} \quad \xrightarrow{\text{results}} \quad \begin{bmatrix} 1 & 0 & -1 \\ 1 & 0 & -1 \\ 1 & 0 & -1 \end{bmatrix}$$

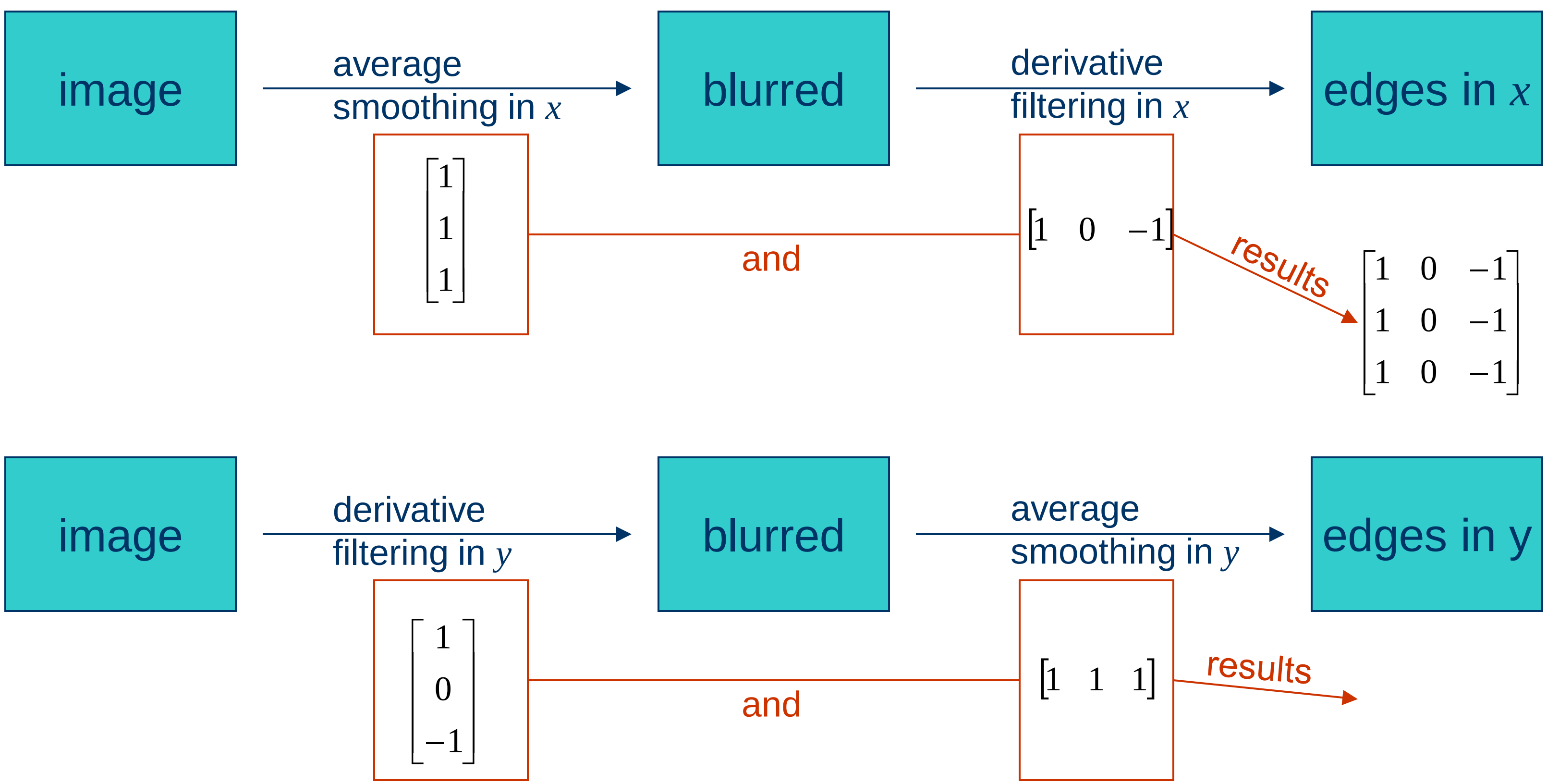
image $\xrightarrow{\text{derivative filtering in } y}$ blurred $\xrightarrow{\text{average smoothing in } y}$ edges in y

$$\begin{bmatrix} 1 \\ 0 \\ -1 \end{bmatrix} \quad \text{and}$$

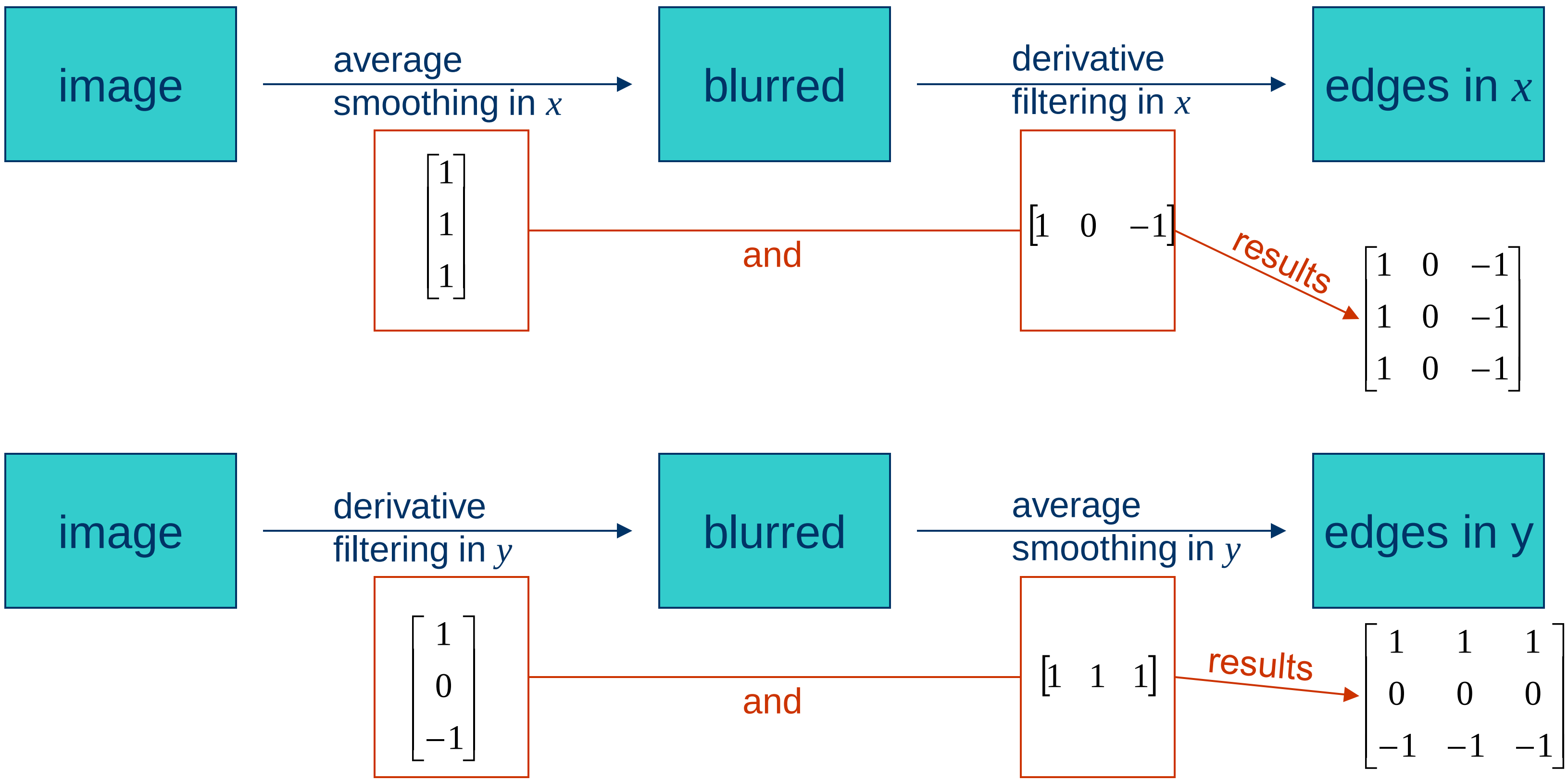
Prewitt Edge Detector



Prewitt Edge Detector



Prewitt Edge Detector



Sobel Edge Detector

Sobel Edge Detector

image

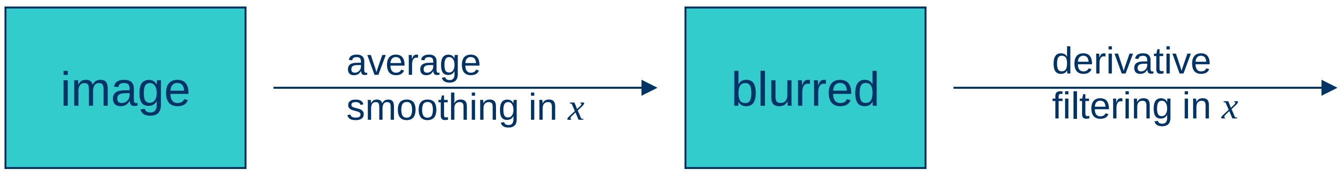
Sobel Edge Detector

image $\xrightarrow[\text{smoothing in } x]{\text{average}}$

Sobel Edge Detector

$$\boxed{\text{image}} \xrightarrow[\text{smoothing in } x]{\text{average}} \boxed{\text{blurred}}$$

Sobel Edge Detector



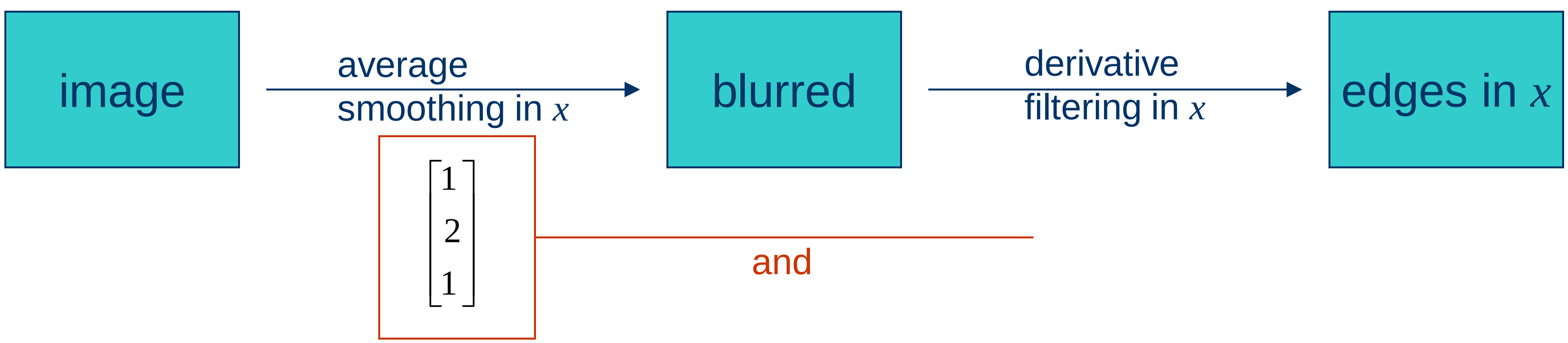
Sobel Edge Detector

Sobel Edge Detector

$$\text{image} \xrightarrow[\text{smoothing in } x]{\text{average}} \text{blurred} \xrightarrow[\text{filtering in } x]{\text{derivative}} \text{edges in } x$$

$$\boxed{\begin{bmatrix}1\\2\\1\end{bmatrix}}$$

Sobel Edge Detector

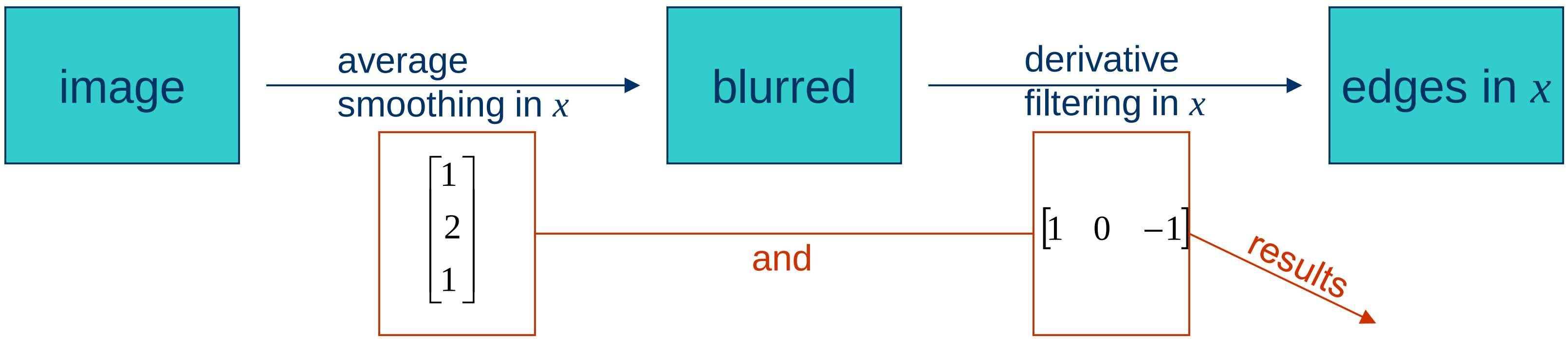


Sobel Edge Detector

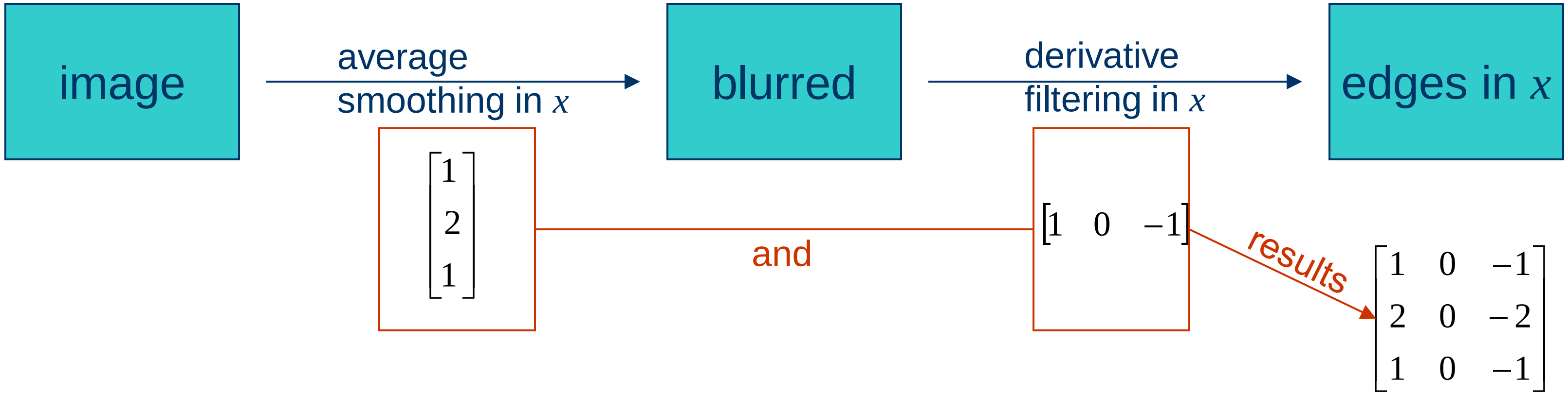
image $\xrightarrow{\text{average smoothing in } x}$ blurred $\xrightarrow{\text{derivative filtering in } x}$ edges in x

$$\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \quad \text{and} \quad \begin{bmatrix} 1 & 0 & -1 \end{bmatrix}$$

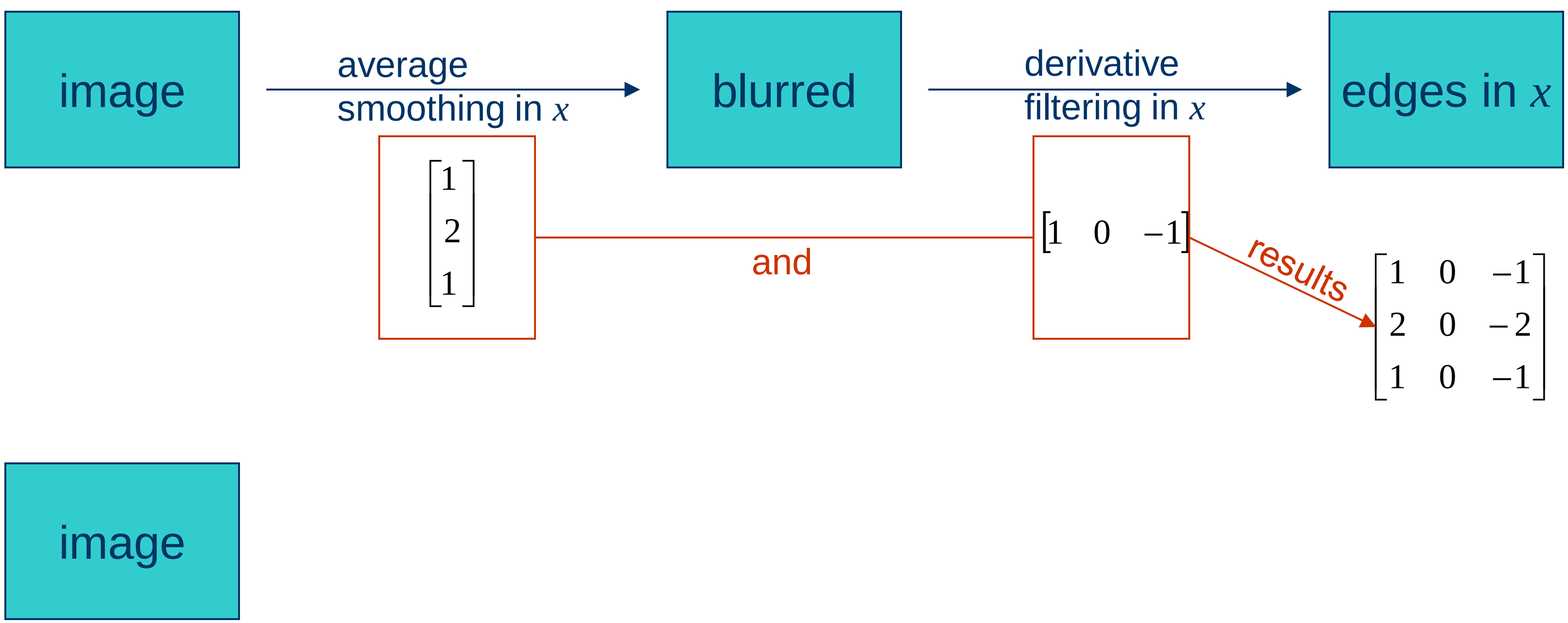
Sobel Edge Detector



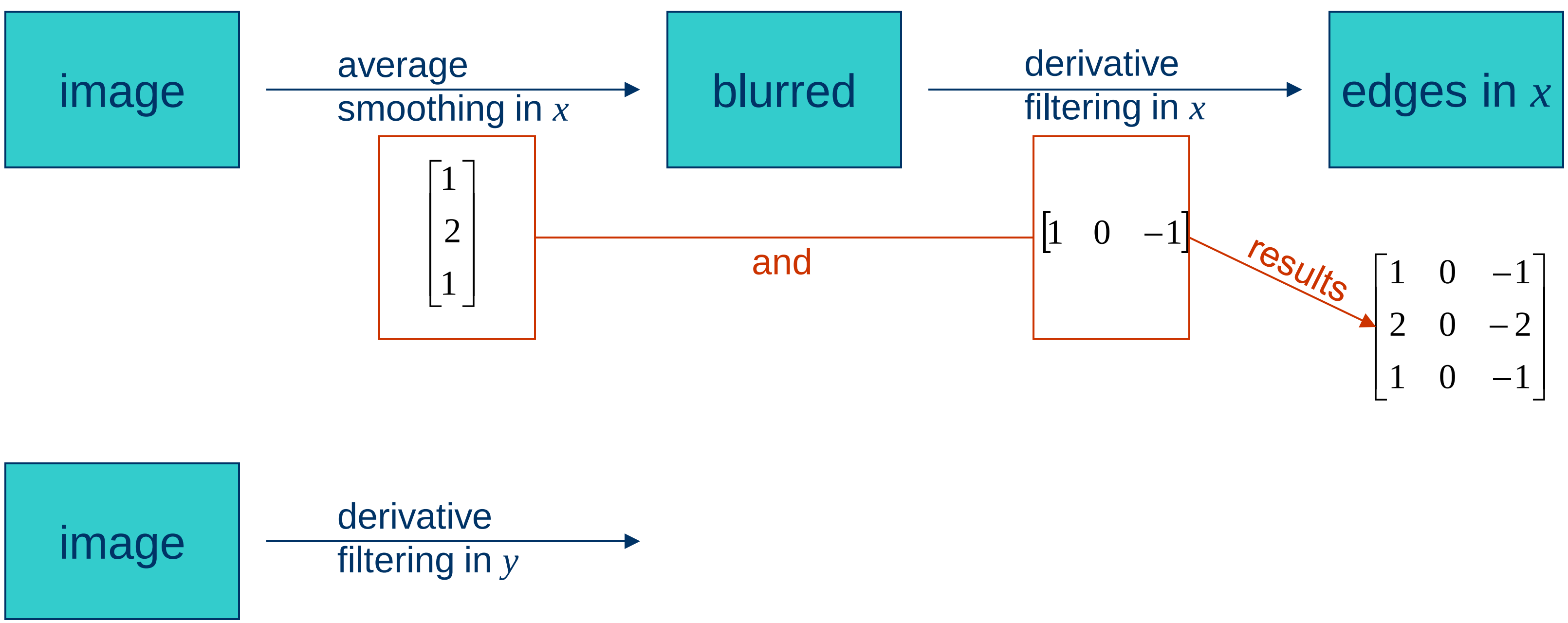
Sobel Edge Detector



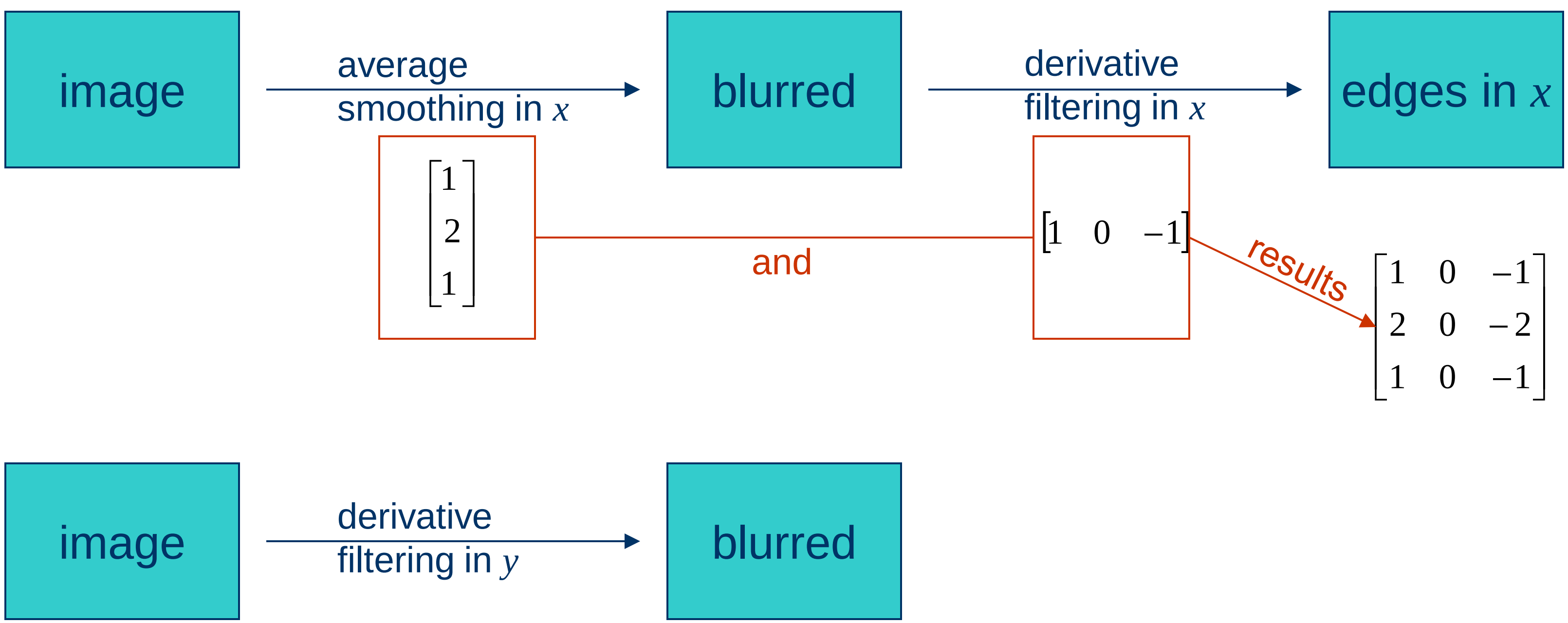
Sobel Edge Detector



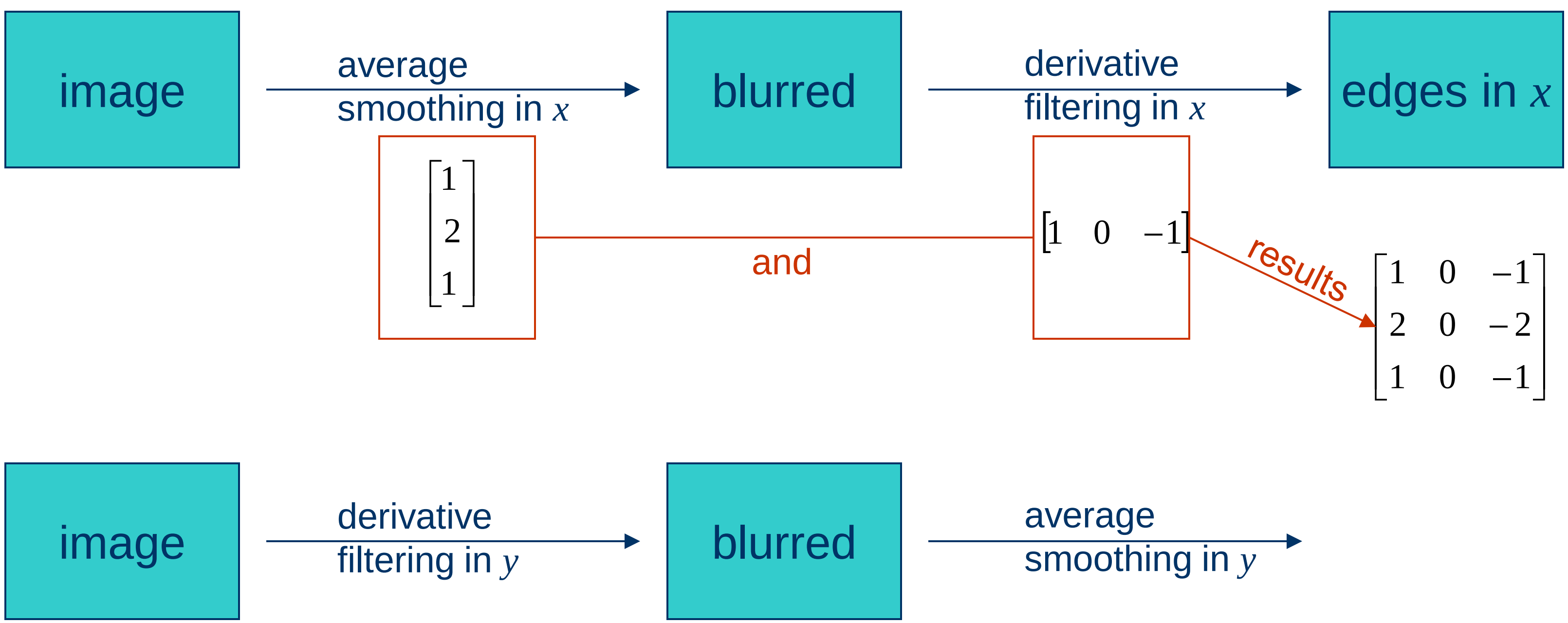
Sobel Edge Detector



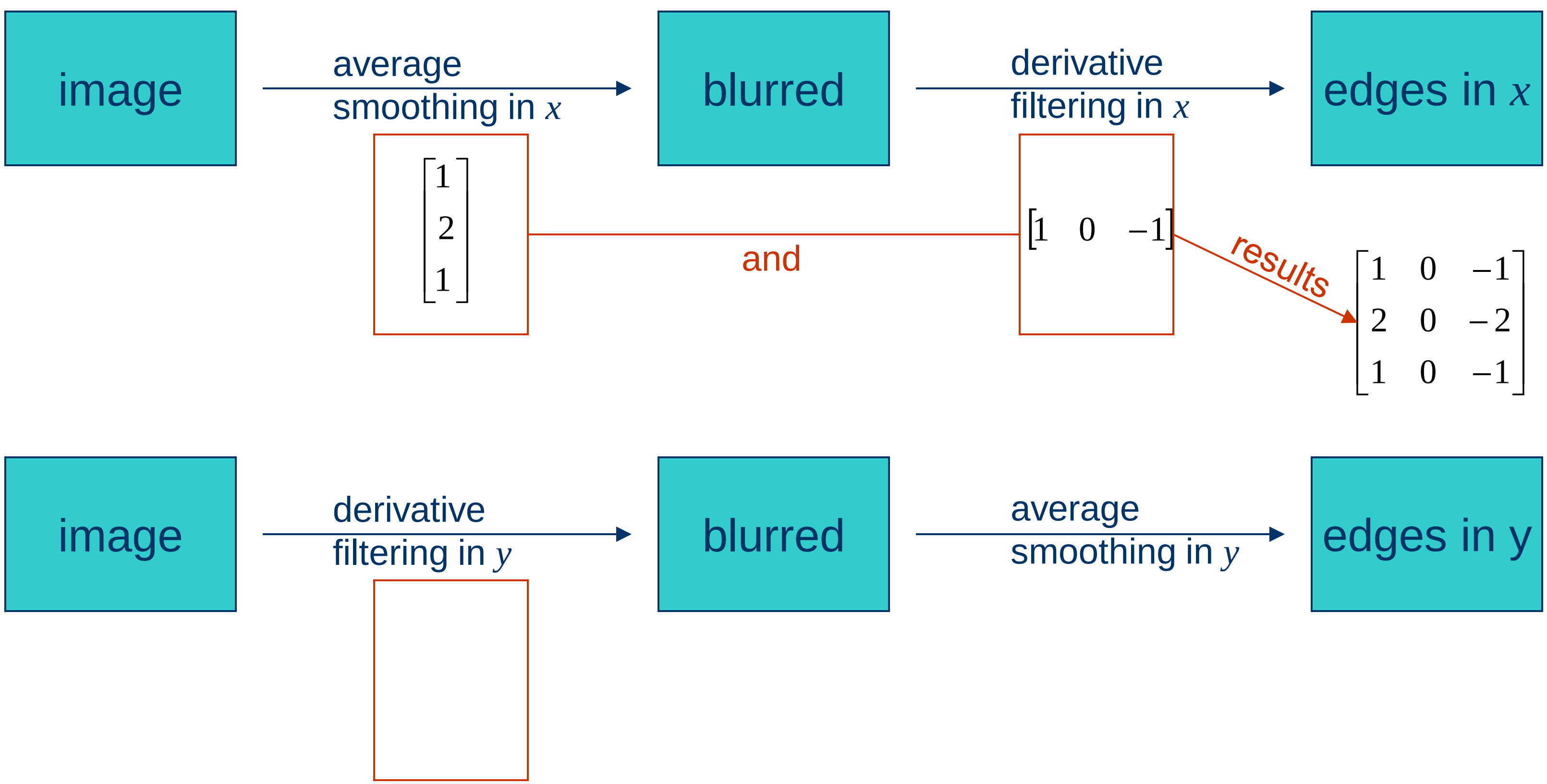
Sobel Edge Detector



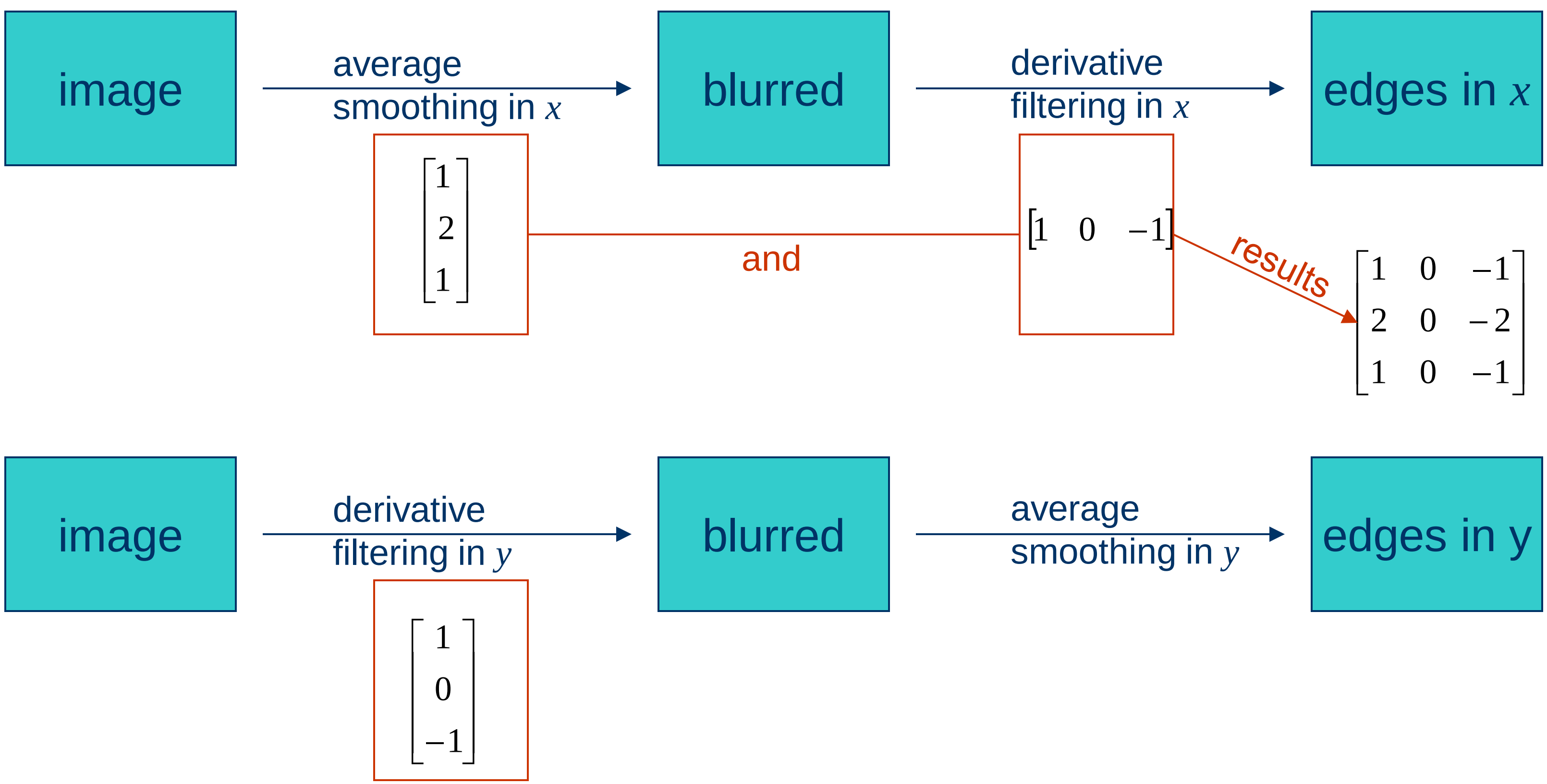
Sobel Edge Detector



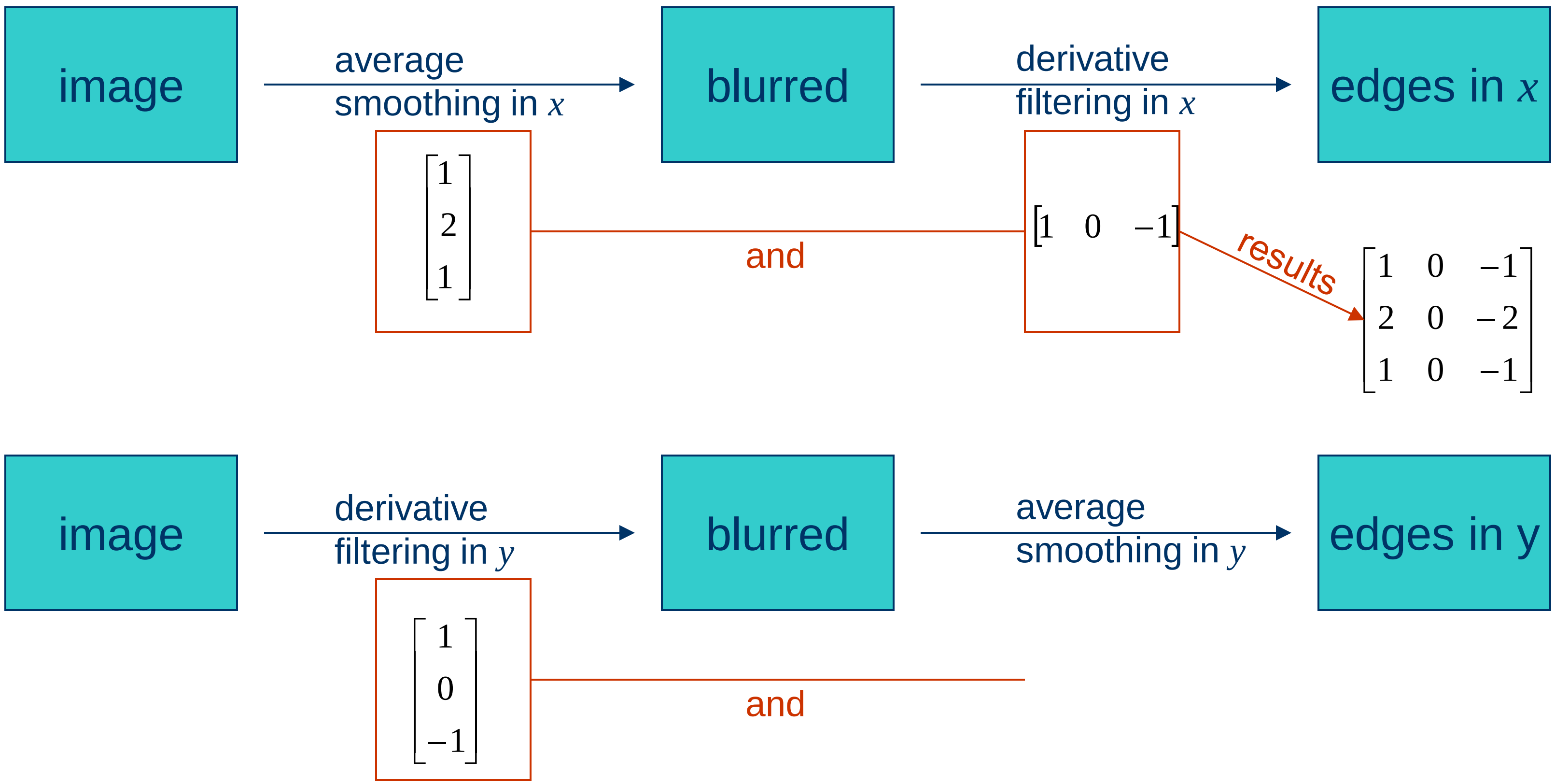
Sobel Edge Detector



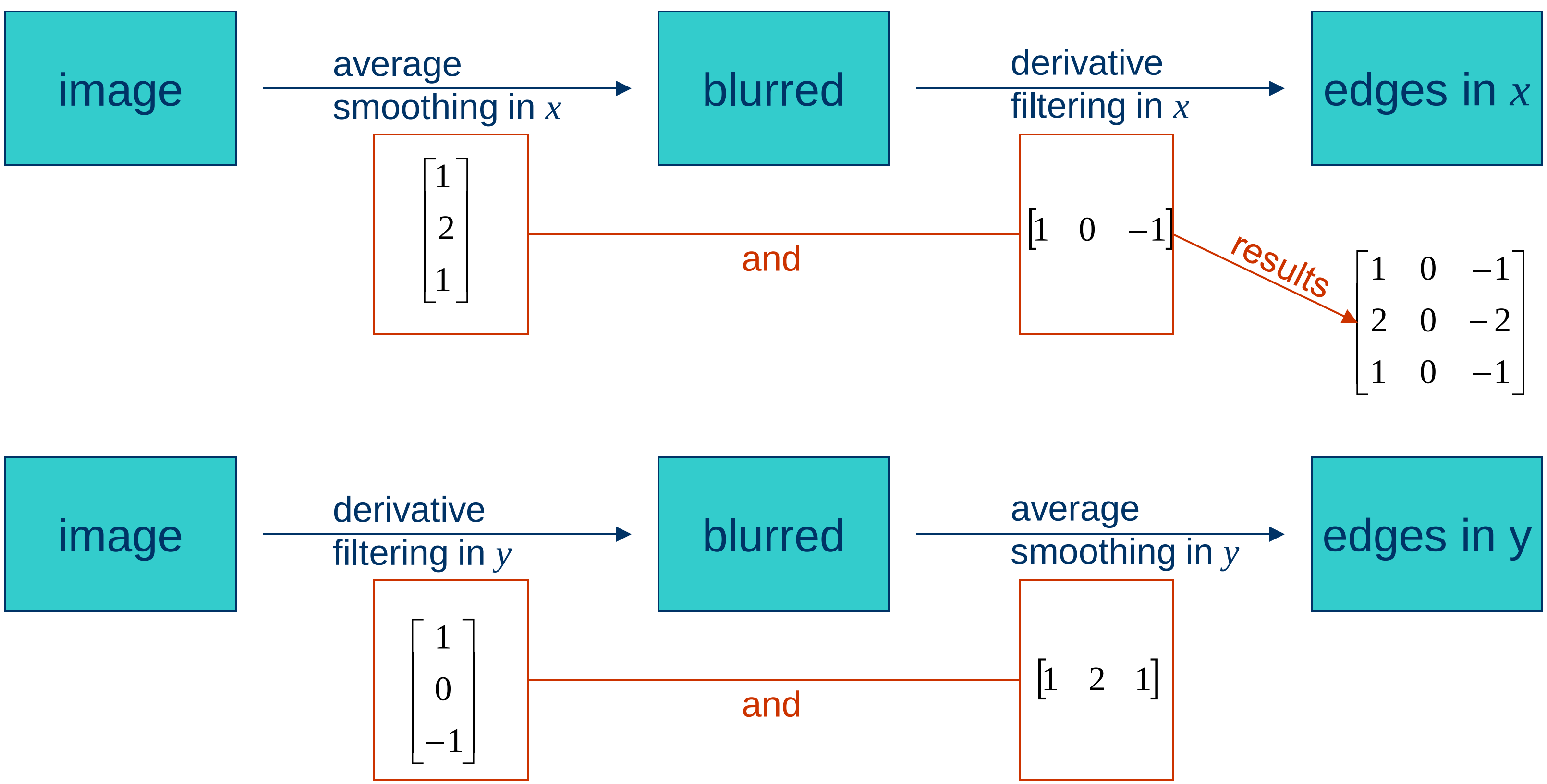
Sobel Edge Detector



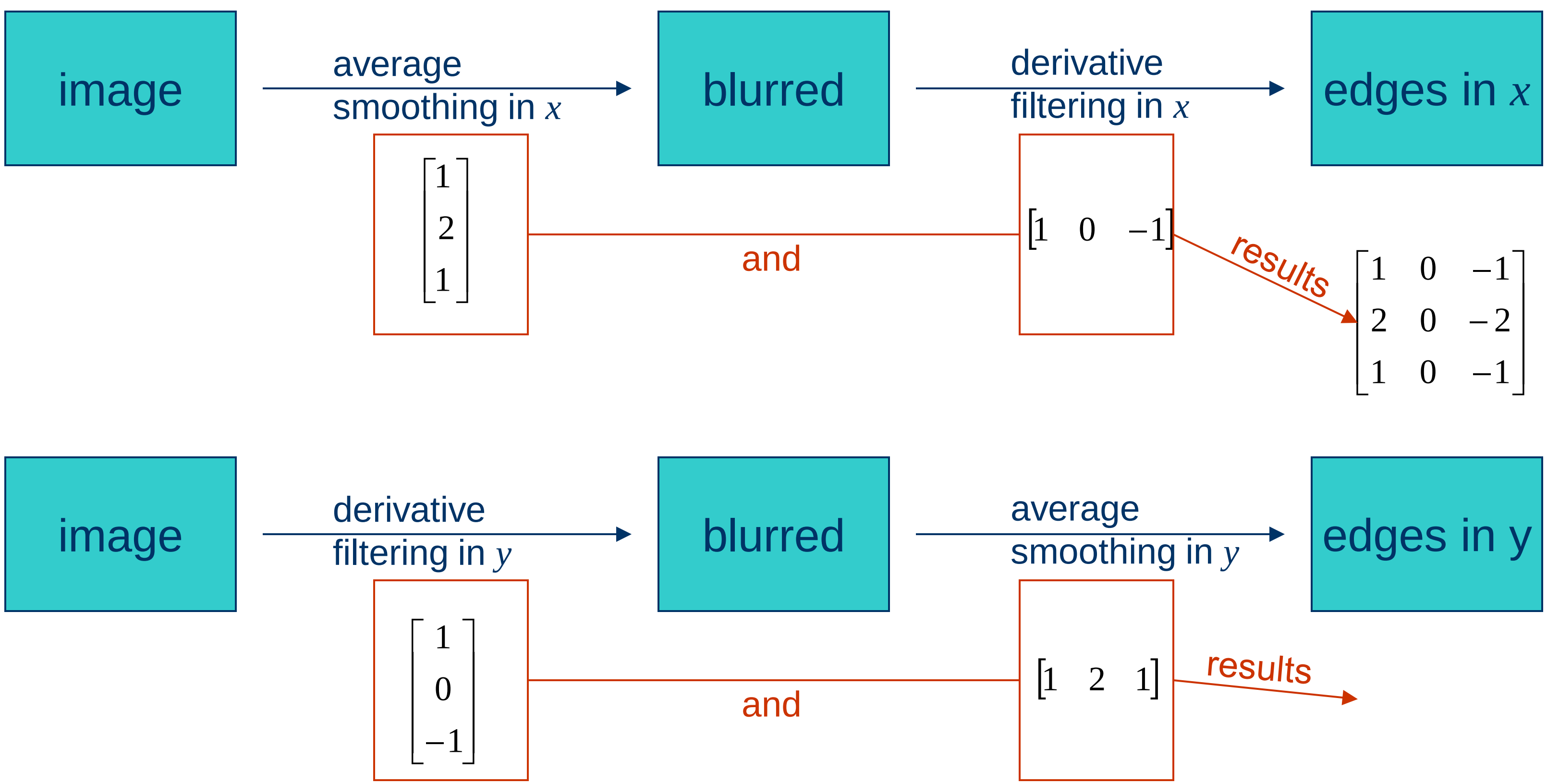
Sobel Edge Detector



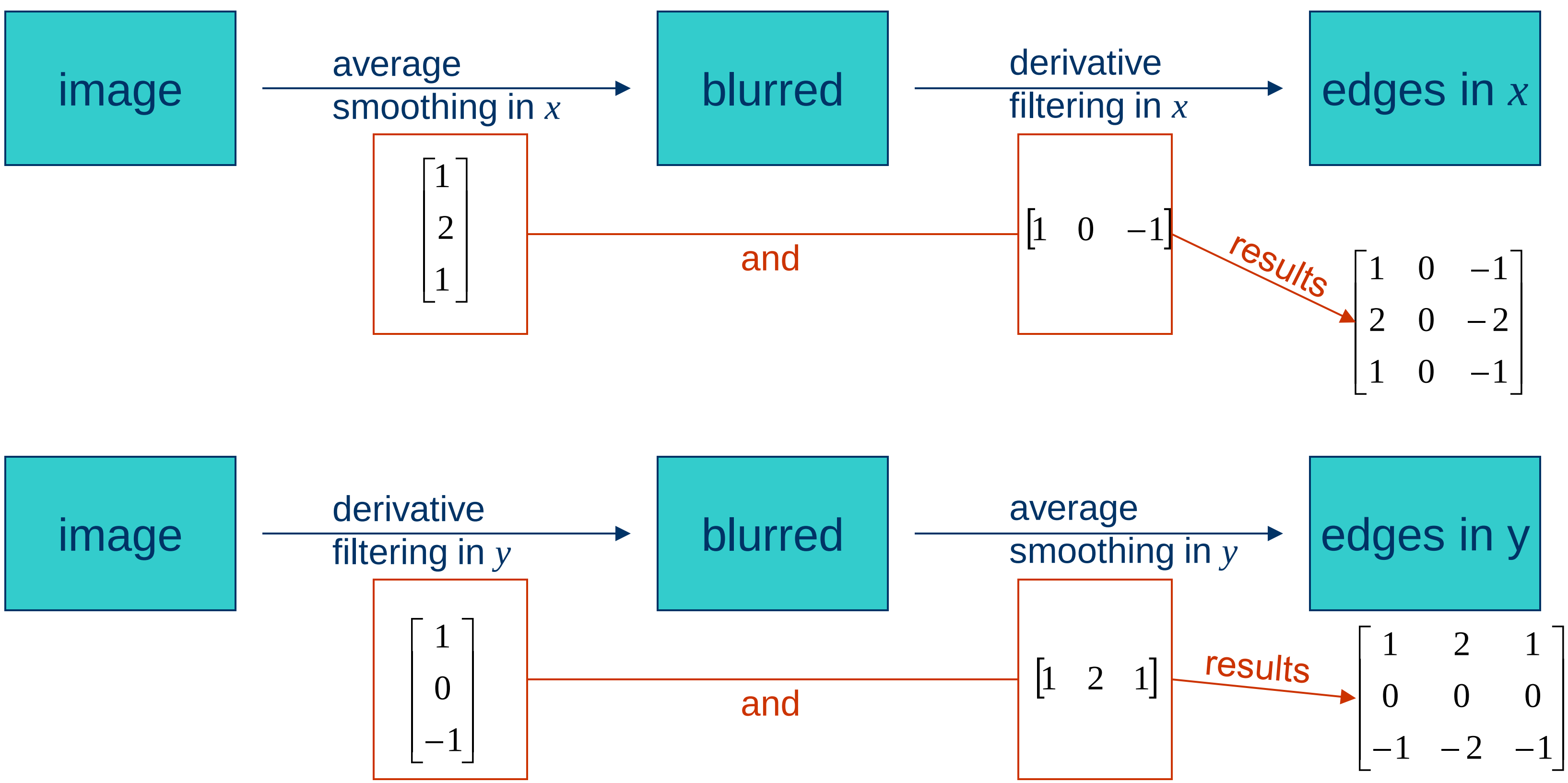
Sobel Edge Detector



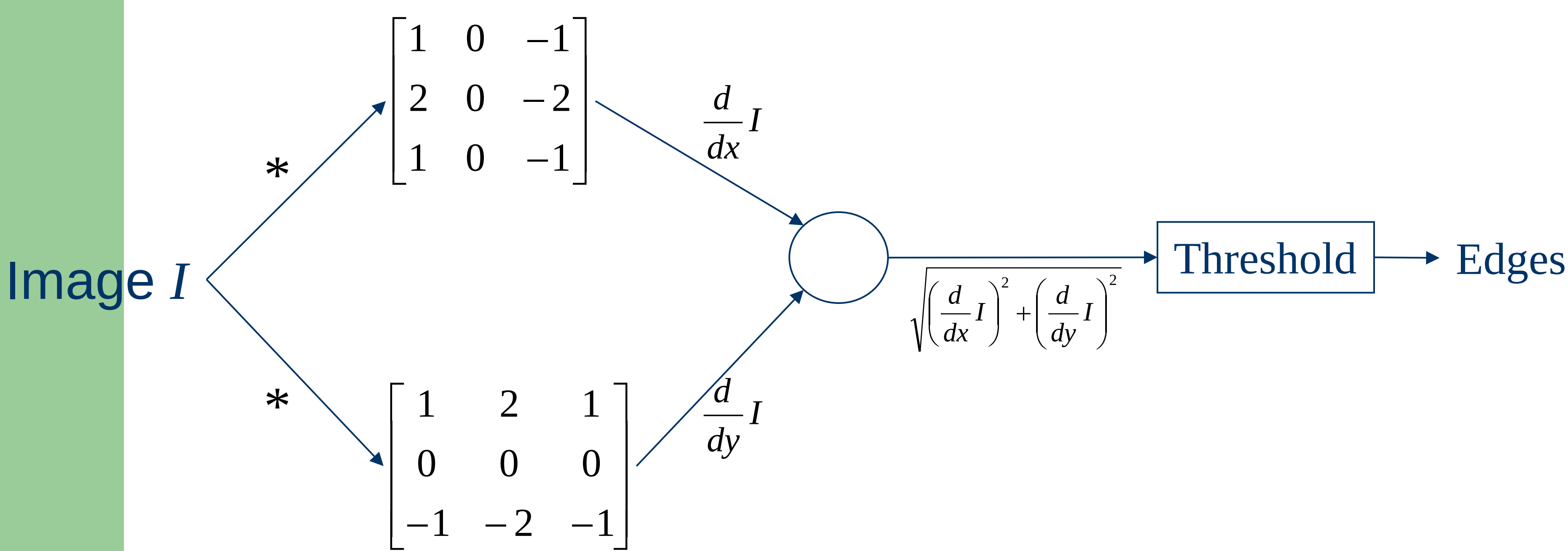
Sobel Edge Detector



Sobel Edge Detector

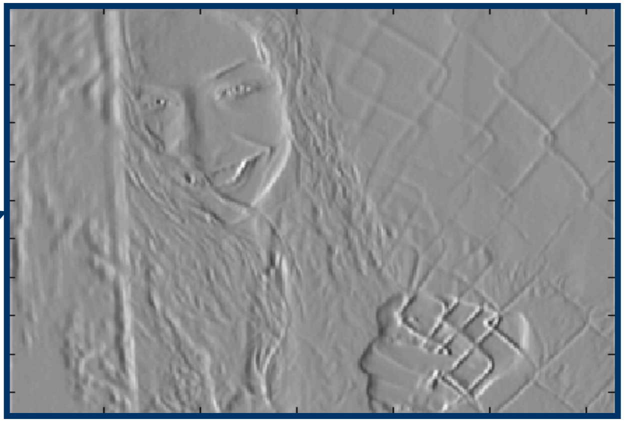


Sobel Edge Detector

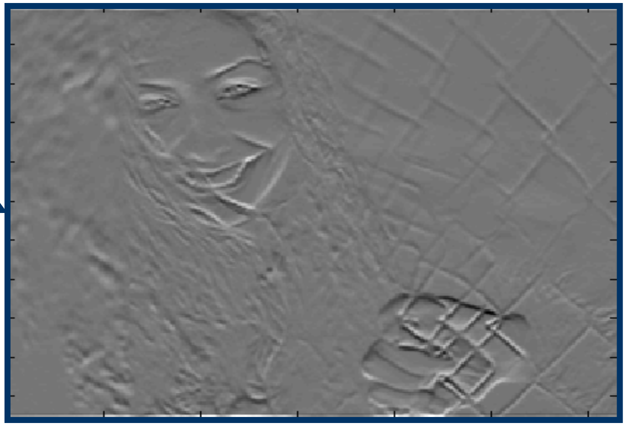


Sobel Edge Detector

$$\frac{d}{dx}I$$



$$\frac{d}{dy}I$$



Sobel Edge Detector

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

$$\Delta \geq Threshold = 100$$

Exercise:

You are provided with an image that has been corrupted by Gaussian noise. Your task is to detect edges within this noisy image using the Sobel operator. You must first apply a Gaussian filter to mitigate the noise and then use the Sobel operator to detect the edges. Here are the steps you will follow to complete this task:

1. Noise Reduction: Since the image is corrupted by Gaussian noise, apply a Gaussian filterto smooth the image. This will help reduce the noise and prevent false edge detection.
2. Sobel Operator: Apply the Sobel operator to the smoothed image to find the intensity gradients, which highlight edges.
3. Edge Strength: Calculate the magnitude of the gradient at each pixel, which correspondsto the strength of the edges.
4. Thresholding: Determine a threshold to distinguish between true edges and noise. Onlykeep the edges that have a strength above this threshold.
5. Prewitt Operator: Repeat steps 2-4 for the Prewitt operator as well and compare the results of 2 operators manually.

Marr Hildreth Edge Detector

Marr Hildreth Edge Detector

- Smooth image by Gaussian filter → S

Marr Hildreth Edge Detector

- Smooth image by Gaussian filter → S
- Apply Laplacian to S

Marr Hildreth Edge Detector

- Smooth image by Gaussian filter → S
- Apply Laplacian to S

Marr Hildreth Edge Detector

- Smooth image by Gaussian filter → S
- Apply Laplacian to S
- Find zero crossings

Marr Hildreth Edge Detector

- Smooth image by Gaussian filter → S
- Apply Laplacian to S
- Find zero crossings
  - Scan along each row, record an edge point at the location of zero-crossing.

Marr Hildreth Edge Detector

- Smooth image by Gaussian filter ➔ S
- Apply Laplacian to S
- 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

- Gaussian smoothing

- Find Laplacian

Marr Hildreth Edge Detector

- Gaussian smoothing

$$\overbrace{S}^{\text{smoothed image}} = \overbrace{g}^{\text{Gaussian filter}} * \overbrace{I}^{\text{image}}$$

- Find Laplacian

Marr Hildreth Edge Detector

- Gaussian smoothing

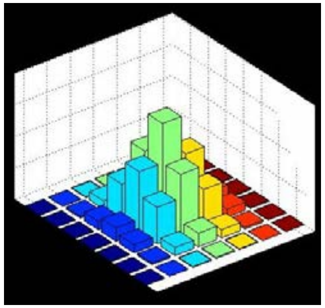
$$\overbrace{S}^{\text{smoothed image}} = \overbrace{g}^{\text{Gaussian filter}} * \overbrace{I}^{\text{image}} \qquad g = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2+y^2}{2\sigma^2}}$$

- Find Laplacian

Marr Hildreth Edge Detector

- Gaussian smoothing

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

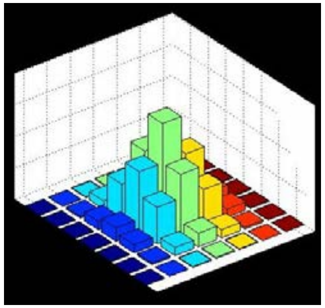


- Find Laplacian

Marr Hildreth Edge Detector

- Gaussian smoothing

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



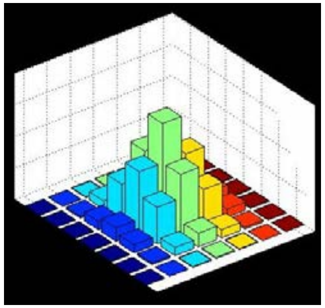
- Find Laplacian

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

Marr Hildreth Edge Detector

- Gaussian smoothing

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



- 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

- Deriving the Laplacian of Gaussian (LoG)

Marr Hildreth Edge Detector

- Deriving the Laplacian of Gaussian (LoG)

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

Marr Hildreth Edge Detector

- Deriving the Laplacian of Gaussian (LoG)

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

Marr Hildreth Edge Detector

- Deriving the Laplacian of Gaussian (LoG)

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

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

Marr Hildreth Edge Detector

- Deriving the Laplacian of Gaussian (LoG)

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

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

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

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

Marr Hildreth Edge Detector

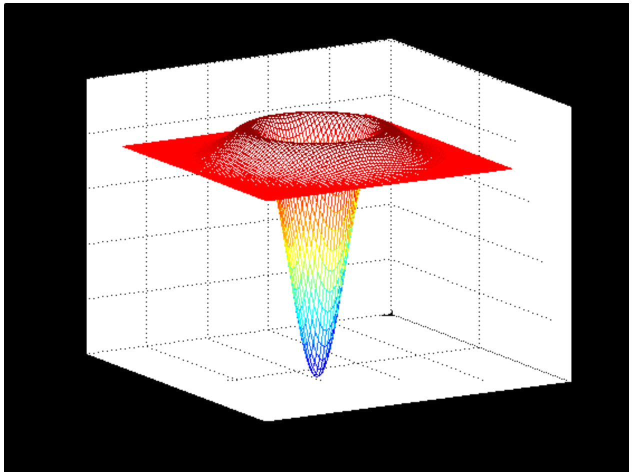
- Deriving the Laplacian of Gaussian (LoG)

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

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

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

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



Gaussian

Gaussian

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

Gaussian

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

Gaussian

$$g(x) = e^{\frac{-x^2}{2o^2}}$$

Standard deviation

Gaussian

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

Standard deviation

Gaussian

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

Standard deviation

x

Gaussian

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

Standard deviation

x

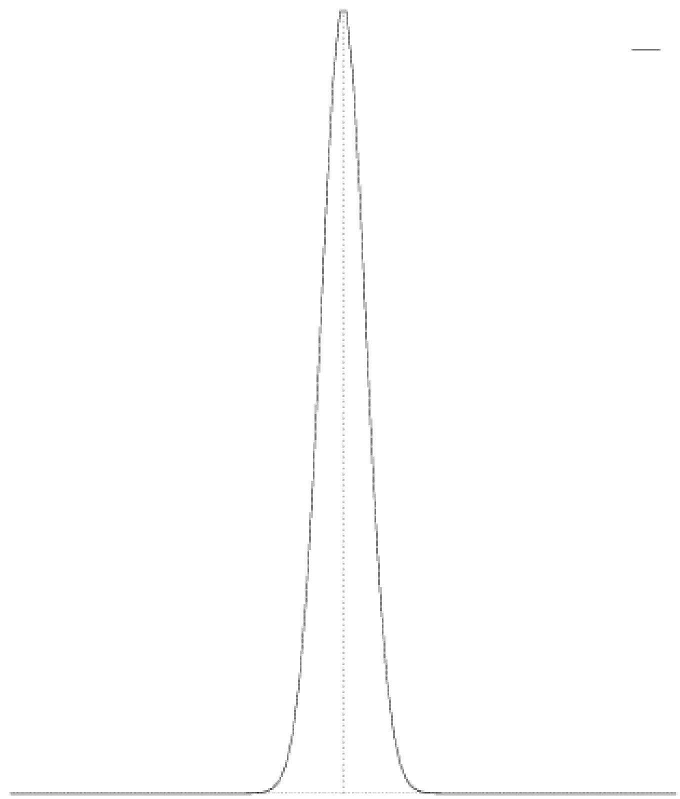
g(x)

Gaussian

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

Standard deviation

x	-3	-2	-1	0	1	2	3
g(x)	.011	.13	.6	1	.6	.13	.011



2-D Gaussian

2-D Gaussian

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

2-D Gaussian

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

$$\sigma = 2$$

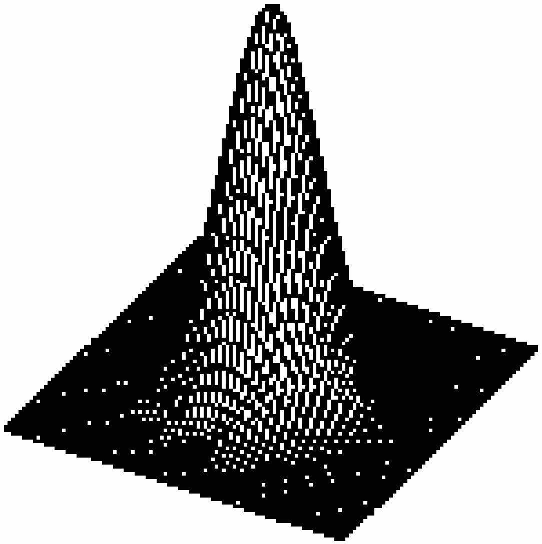
2-D Gaussian

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

0	0	0	0	1	2	2	2	1	0	0	0	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	3	11	26	50	73	82	73	50	26	11	3	0
1	6	20	50	93	136	154	136	93	50	20	6	1
2	9	30	73	136	198	225	198	136	73	30	9	2
2	11	34	82	154	225	255	225	154	82	34	11	2
2	9	30	73	136	198	225	198	136	73	30	9	2
1	6	20	50	93	136	154	136	93	50	20	6	1
0	3	11	26	50	73	82	73	50	26	11	3	0
0	1	4	11	20	30	34	30	20	11	4	1	0
0	0	1	3	6	9	11	9	6	3	1	0	0
0	0	0	0	1	2	2	2	1	0	0	0	0

$$\sigma = 2$$

2-D Gaussian



LoG Filter

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

LoG Filter

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

Y

0.0008	0.0066	0.0215	0.031	0.0215	0.0066	0.0008	
0.0066	0.0438	0.0982	0.108	0.0982	0.0438	0.0066	
0.0215	0.0982	0	-0.242	0	0.0982	0.0215	
0.031	0.108	-0.242	-0.7979	-0.242	0.108	0.031	X
0.0215	0.0982	0	-0.242	0	0.0982	0.0215	
0.0066	0.0438	0.0982	0.108	0.0982	0.0438	0.0066	
0.0008	0.0066	0.0215	0.031	0.0215	0.0066	0.0008	

Finding Zero Crossings

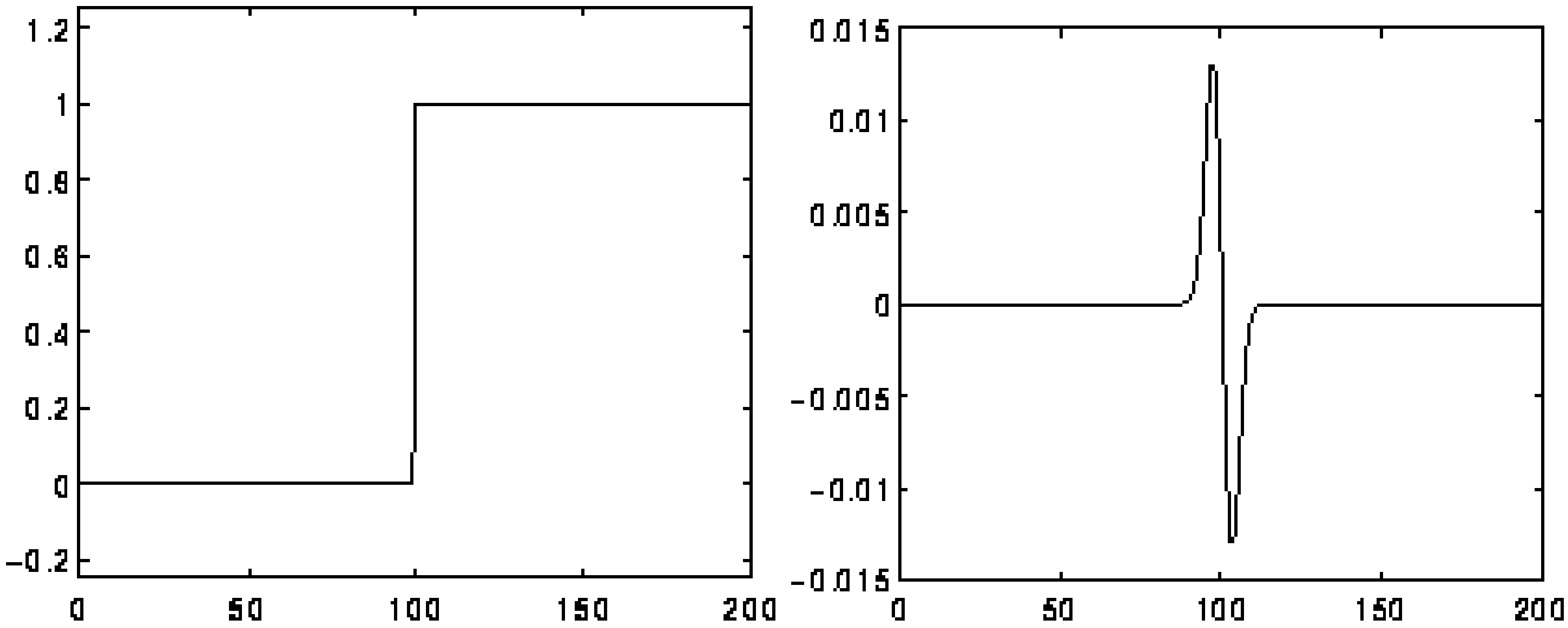


Figure 1 Response of 1-D LoG filter to a step edge.

The left hand graph shows a 1-D image line scan, 200 pixels long, containing a step edge. The right hand graph shows the response of a 1-D LoG filter with Gaussian standard deviation 3 pixels. Note that first derivative is not shown here.

Finding Zero Crossings

- Four cases of zero-crossings :

Finding Zero Crossings

- Four cases of zero-crossings :
  - {+,-}

Finding Zero Crossings

- Four cases of zero-crossings :
  - {+,-}
  - {+,0,-}

Finding Zero Crossings

- Four cases of zero-crossings :
  - {+,-}
  - {+,0,-}
  - {-,+}

Finding Zero Crossings

- Four cases of zero-crossings :
  - {+,-}
  - {+,0,-}
  - {-,+}
  - {-,0,+}

Finding Zero Crossings

- Four cases of zero-crossings :
  - {+,-}
  - {+,0,-}
  - {-,+}
  - {-,0,+}
- Slope of zero-crossing {a, -b} is |a+b|.

Finding Zero Crossings

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

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

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

Example

I

Example

I

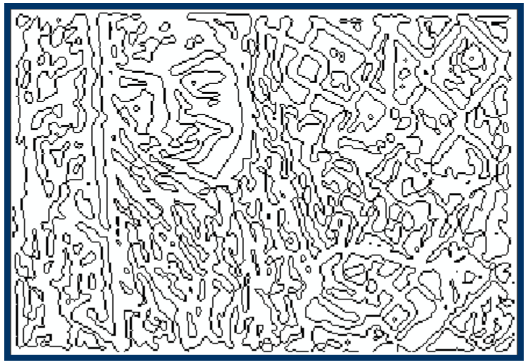
$I * \left(\Delta^2 g\right)$

Example

I

$I * \left(\Delta^2 g\right)$

Zero crossings of $\Delta^2 S$



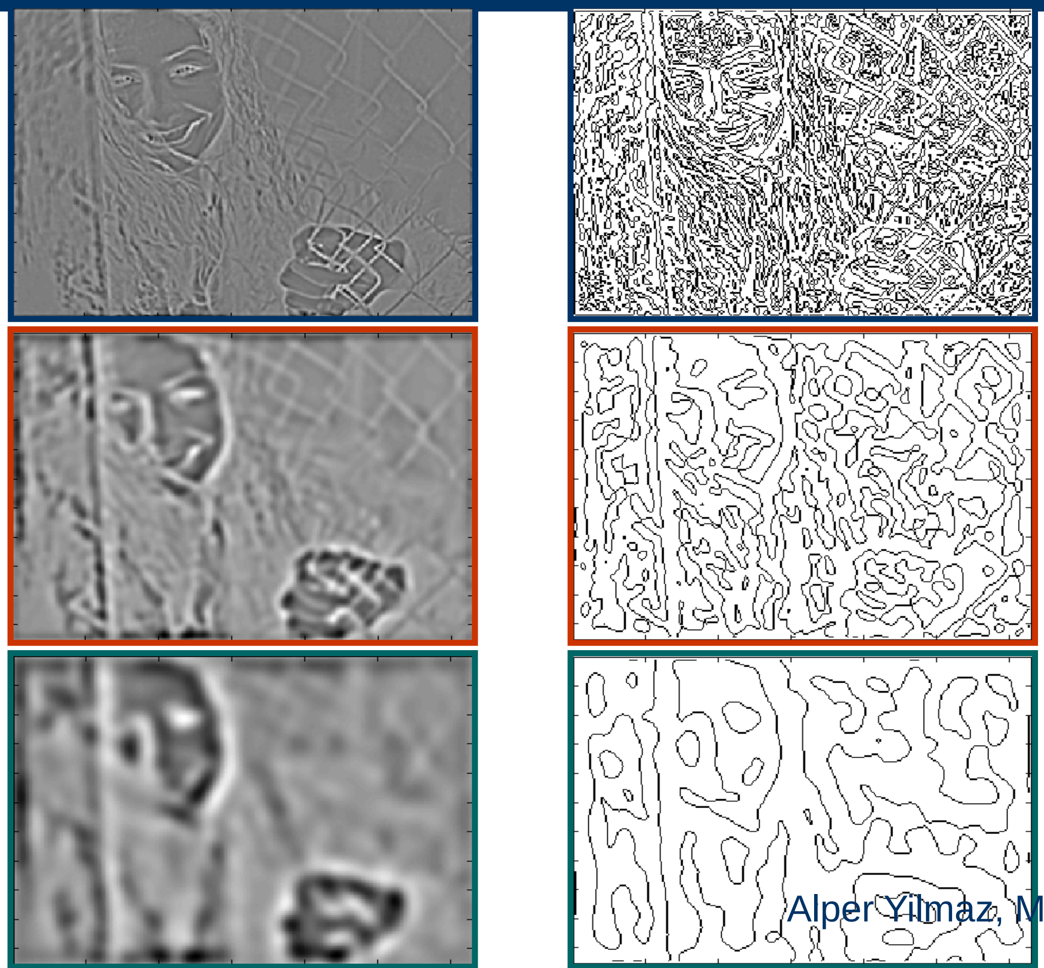
Example

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

$\sigma = 1$

$\sigma = 3$

$\sigma = 6$



LoG Algorithm

LoG Algorithm

- Apply LoG to the Image

LoG Algorithm

- Apply LoG to the Image
- Find zero-crossings from each row

LoG Algorithm

- Apply LoG to the Image
- Find zero-crossings from each row
- Find slope of zero-crossings

LoG Algorithm

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

Quality of an Edge

Quality of an Edge

- Robust to noise

Quality of an Edge

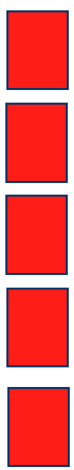
- Robust to noise
- Localization

Quality of an Edge

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

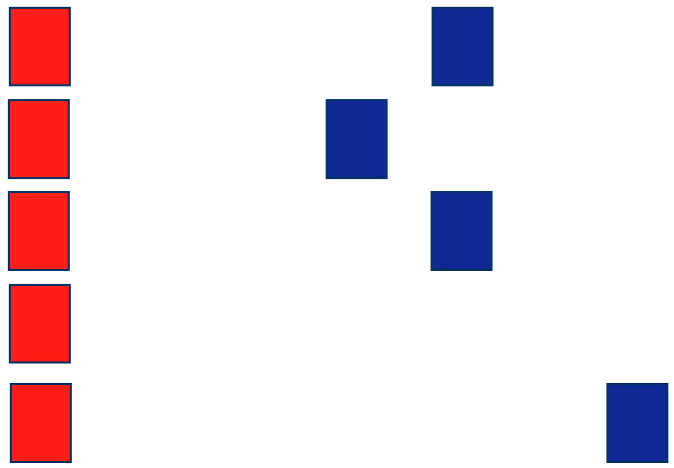
Quality of an Edge

Quality of an Edge



True
edge

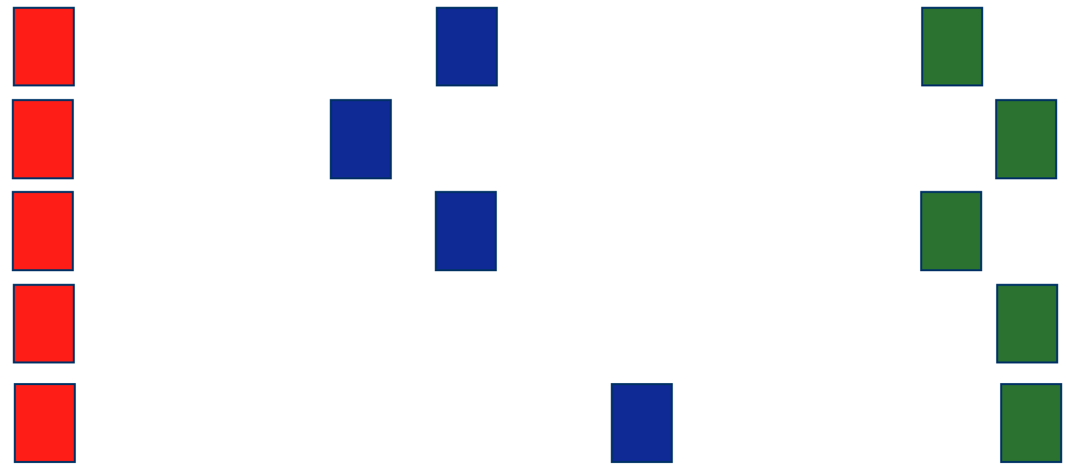
Quality of an Edge



True edge

Poor robustness to noise

Quality of an Edge

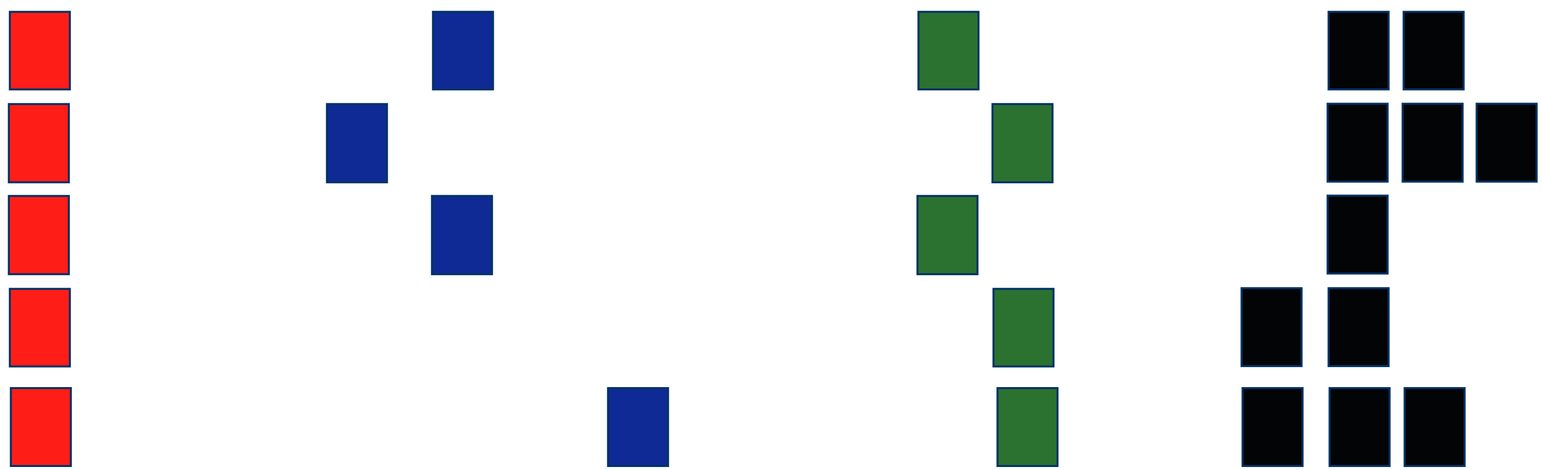


True edge

Poor robustness to noise

Poor localization

Quality of an Edge



True edge

Poor robustness to noise

Poor localization

Too many responses

Canny Edge Detector

Canny Edge Detector

- Criterion 1: Good Detection: The optimal detector must minimize the probability of false positives as well as false negatives.

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.

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

Canny Edge Detector Steps

1. Smooth image with Gaussian filter

Canny Edge Detector Steps

1. Smooth image with Gaussian filter
2. Compute derivative of filtered image

Canny Edge Detector Steps

1. Smooth image with Gaussian filter
2. Compute derivative of filtered image
3. Find magnitude and orientation of gradient

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”

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
- Derivative

Canny Edge Detector
First Two Steps

- Smoothing

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

- Derivative

Canny Edge Detector
First Two Steps

- Smoothing

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

- Derivative

Canny Edge Detector First Two Steps

- Smoothing

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

- Derivative

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

Canny Edge Detector
First Two Steps

- Smoothing

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

- Derivative

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

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

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

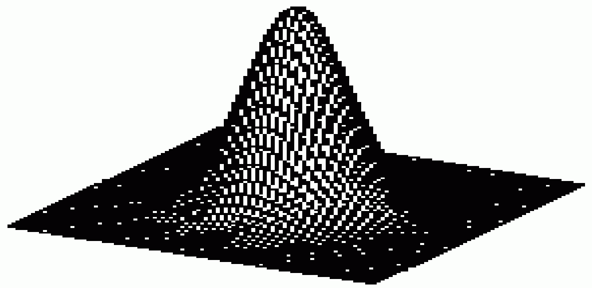
$$\nabla S = \nabla(g * I) = (\nabla g) * I$$

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

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

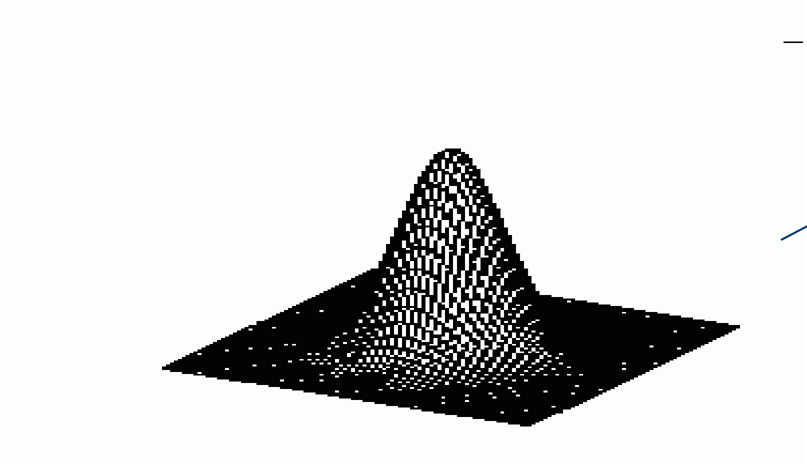
Canny Edge Detector
Derivative of Gaussian

Canny Edge Detector
Derivative of Gaussian



$g(x, y)$

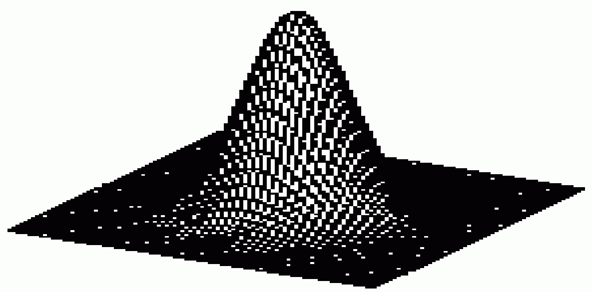
Canny Edge Detector
Derivative of Gaussian



$g(x, y)$

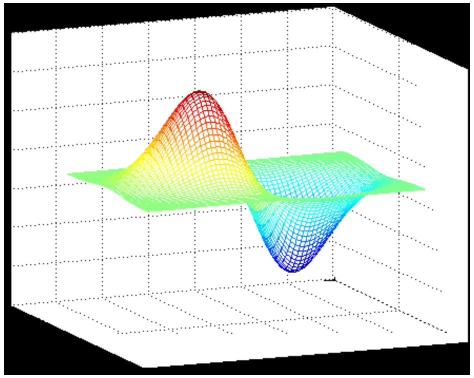
$g_x(x, y)$

Canny Edge Detector
Derivative of Gaussian

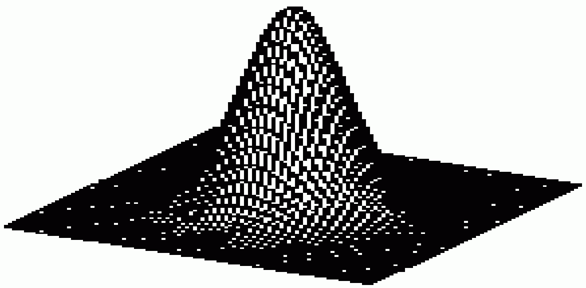


$$g(x, y)$$

$$g_x(x, y)$$

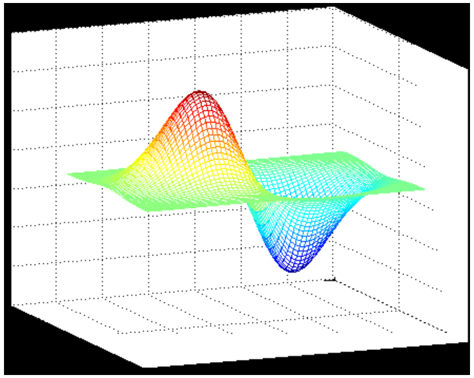


Canny Edge Detector
Derivative of Gaussian



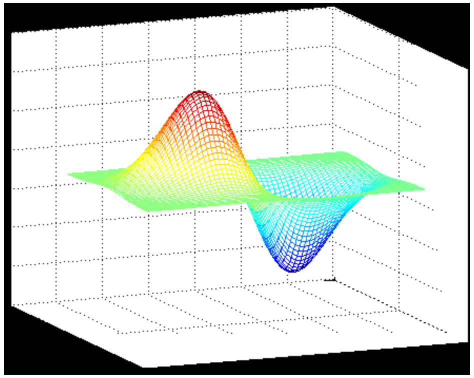
$g(x,y)$

$g_x(x,y)$

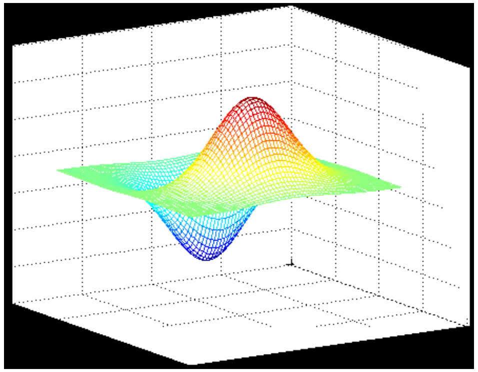


$g_y(x,y)$

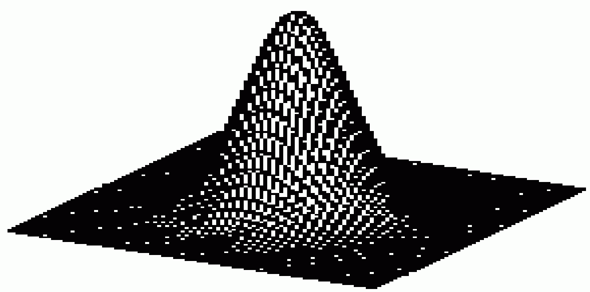
Canny Edge Detector
Derivative of Gaussian



$g_x(x, y)$



$g_y(x, y)$



$g(x, y)$

Canny Edge Detector
First Two Steps

Canny Edge Detector
First Two Steps

I

Canny Edge Detector
First Two Steps

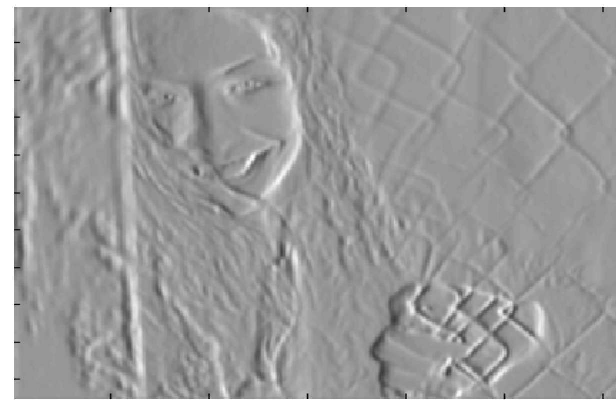
I

S_x

Canny Edge Detector
First Two Steps

I

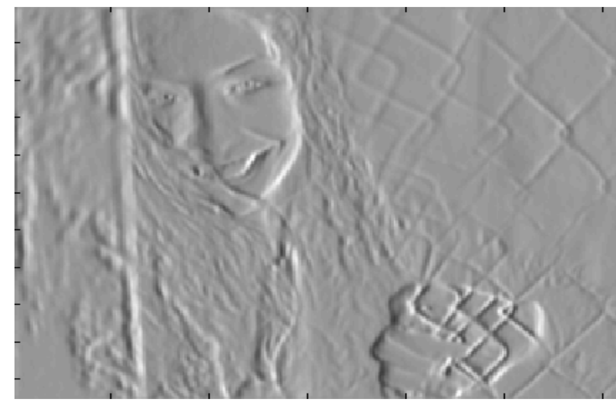
S_x



Canny Edge Detector
First Two Steps

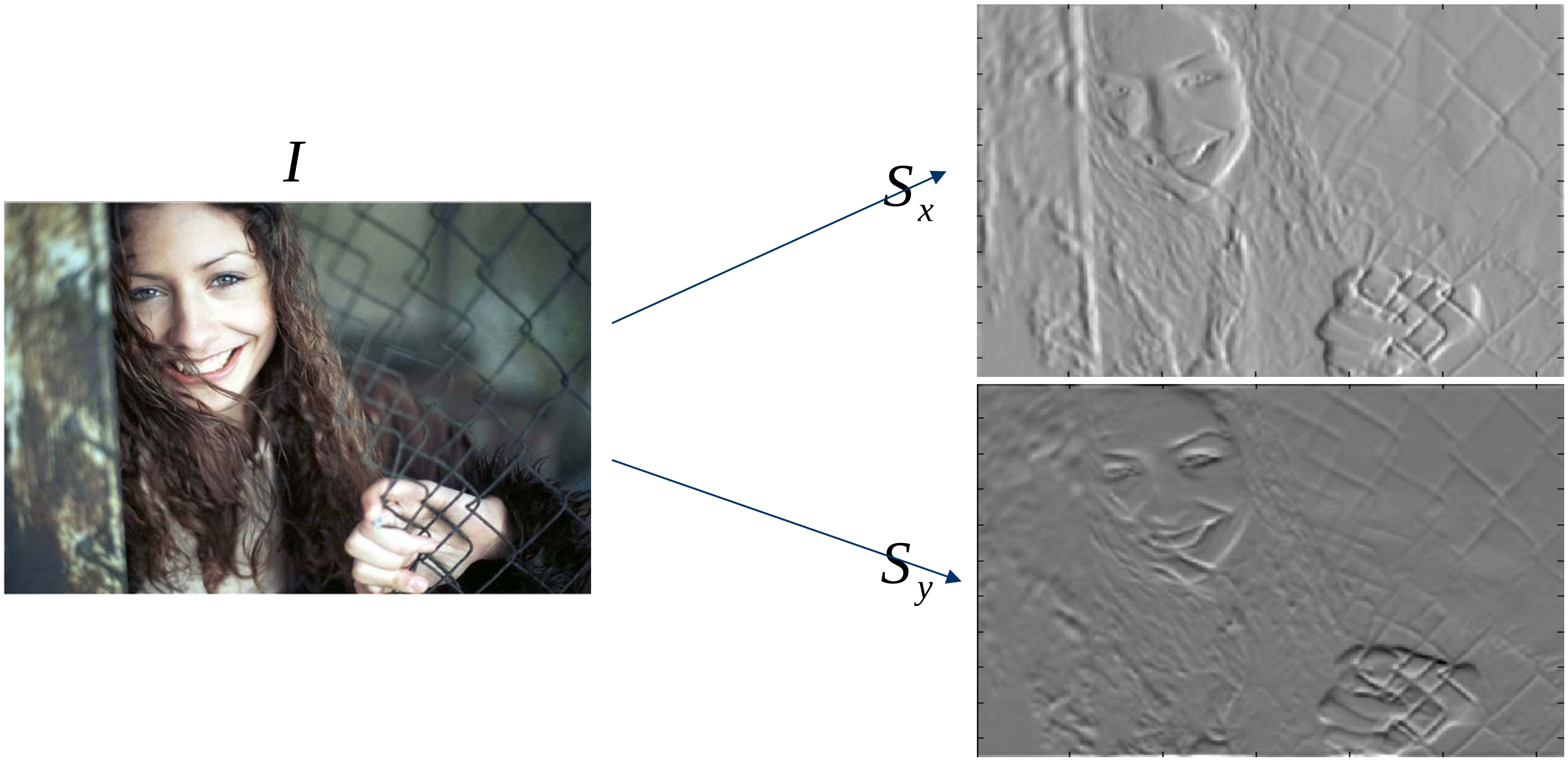
I

S_x



S_y

Canny Edge Detector
First Two Steps



Canny Edge Detector
Third Step

- Gradient magnitude and gradient direction

Canny Edge Detector
Third Step

- Gradient magnitude and gradient direction

(S_x, S_y) Gradient Vector

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

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

Canny Edge Detector
Third Step

- 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

Canny Edge Detector
Third Step

- Gradient magnitude and gradient direction

(S_x, S_y) Gradient Vector

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

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

image

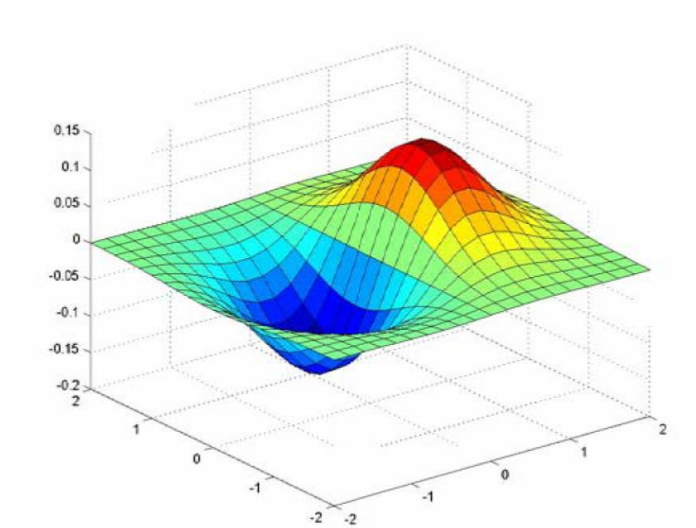
gradient magnitude

Example

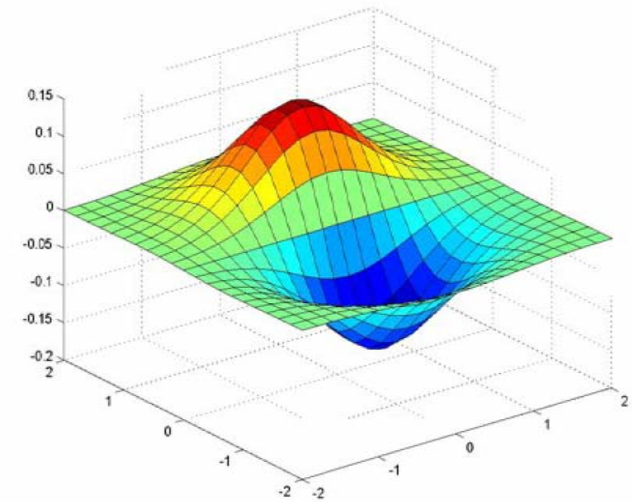
original image (Lena)

Slide Credit: James hays

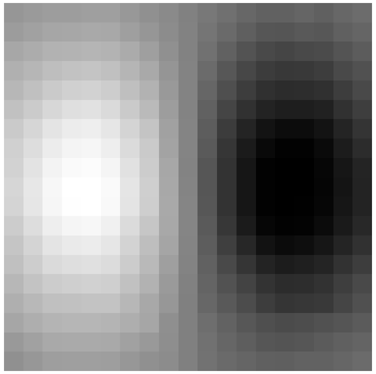
Derivative of Gaussian filter

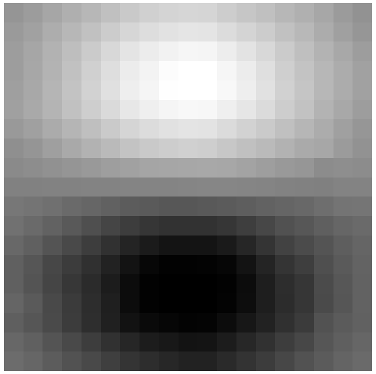


x-direction



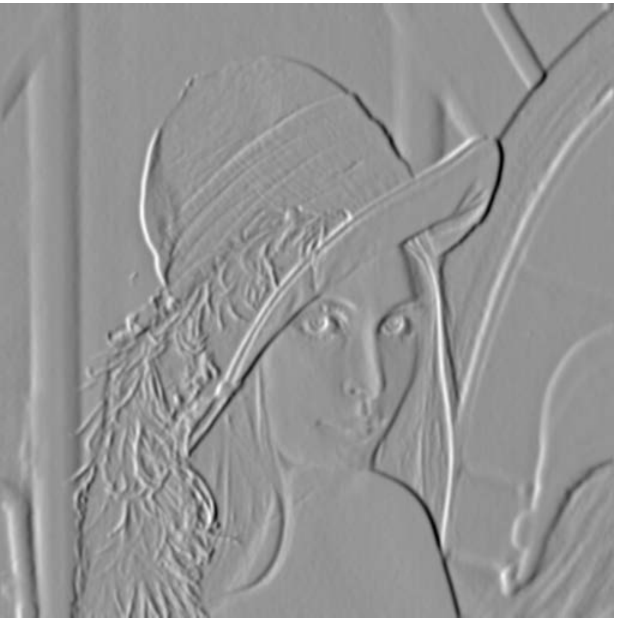
y-direction





Slide Credit: James hays

Compute Gradients



X-Derivative of Gaussian

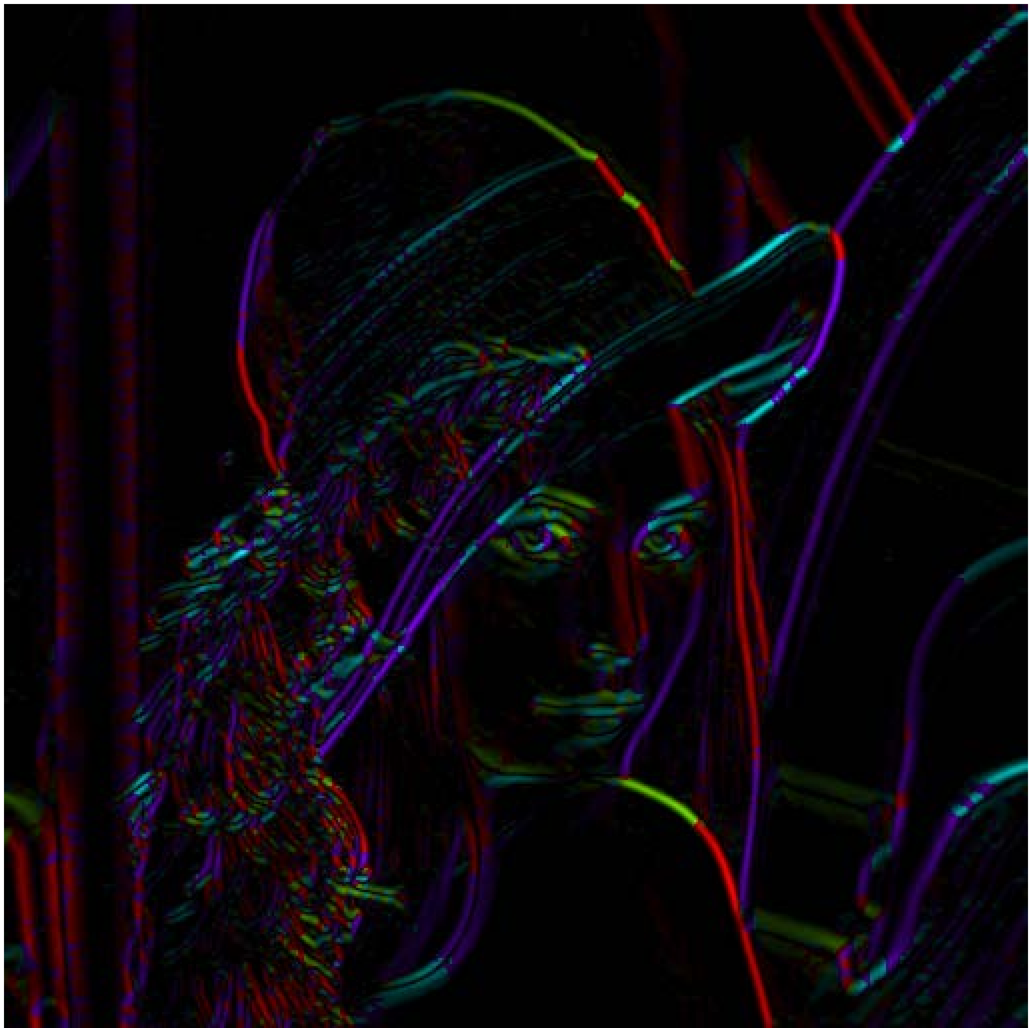
Y-Derivative of Gaussian

Gradient Magnitude

Slide Credit: James hays

Get Orientation at Each Pixel

- Threshold at minimum level
- Get orientation



theta = atan2(gy, gx)

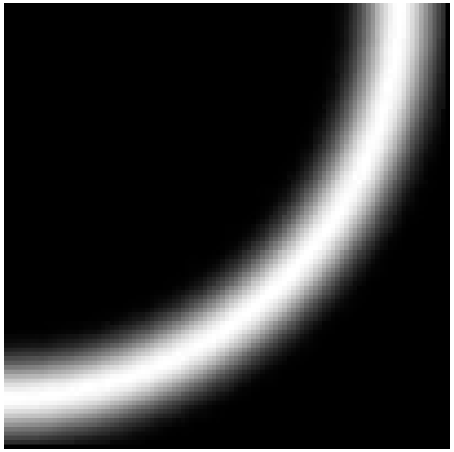
Slide Credit: James hays

Canny Edge Detector
Fourth Step

- Non maximum suppression

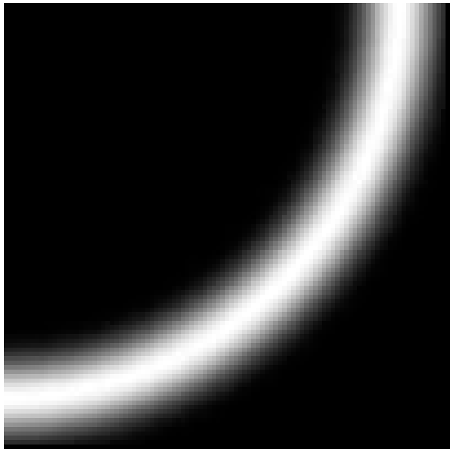
Canny Edge Detector
Fourth Step

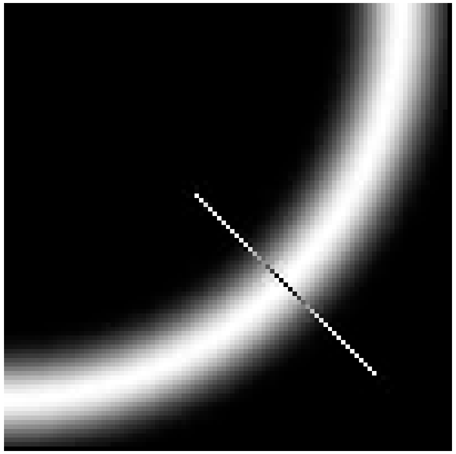
- Non maximum suppression



Canny Edge Detector
Fourth Step

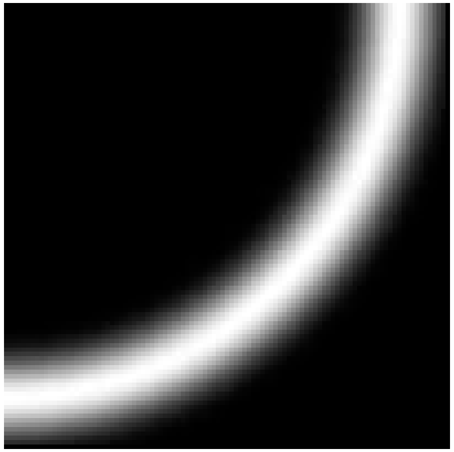
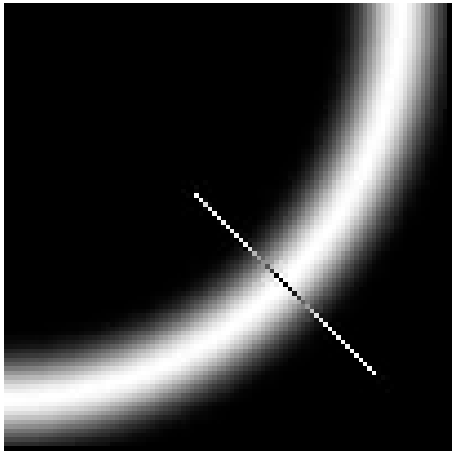
- Non maximum suppression





Canny Edge Detector
Fourth Step

- Non maximum suppression

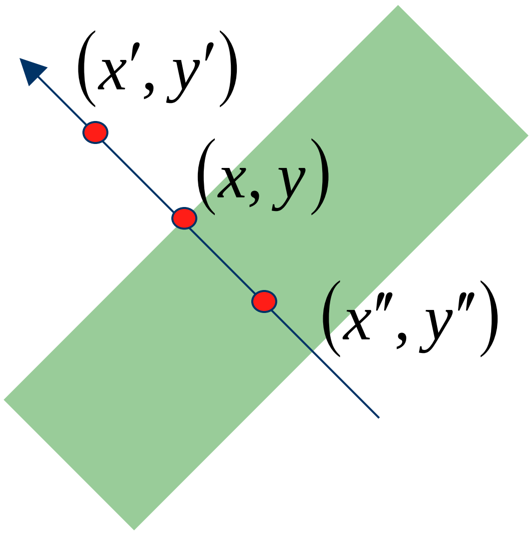
We wish to mark points along the curve where the **magnitude is largest**. We can do this by looking for a maximum along a slice normal to the curve (non-maximum suppression). These points should form a curve. There are then two algorithmic issues: at which point is the maximum, and where is the next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

Canny Edge Detector
Non-Maximum Suppression

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

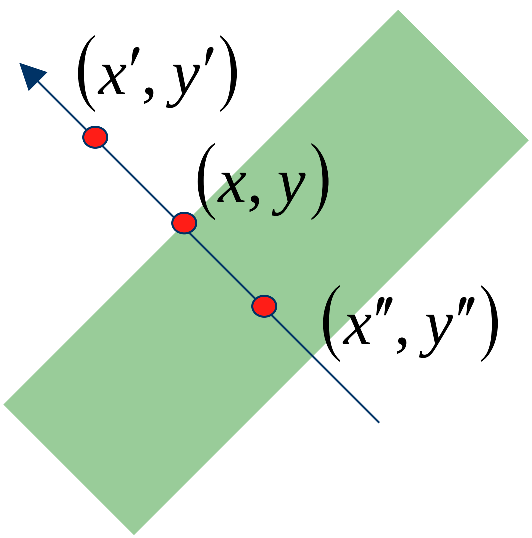
Canny Edge Detector
Non-Maximum Suppression

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



Canny Edge Detector
Non-Maximum Suppression

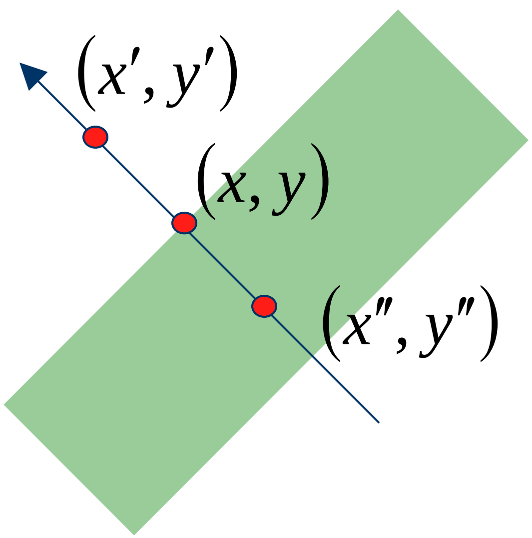
- Suppress the pixels in $|\nabla S|$ which are not local maximum



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

Canny Edge Detector
Non-Maximum Suppression

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



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

$\mathbf{x}$’ and $\mathbf{x}$’’ are the neighbors of $\mathbf{x}$ along normal direction to an edge

Canny Edge Detector
Non-Maximum Suppression

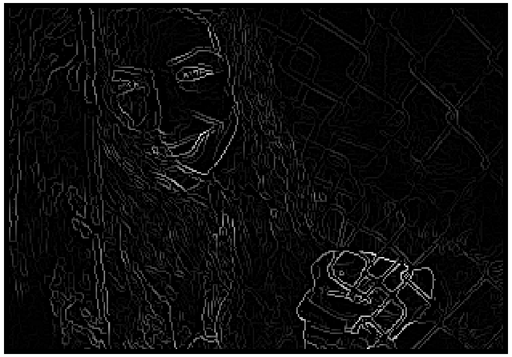
Canny Edge Detector
Non-Maximum Suppression

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

Canny Edge Detector
Non-Maximum Suppression

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

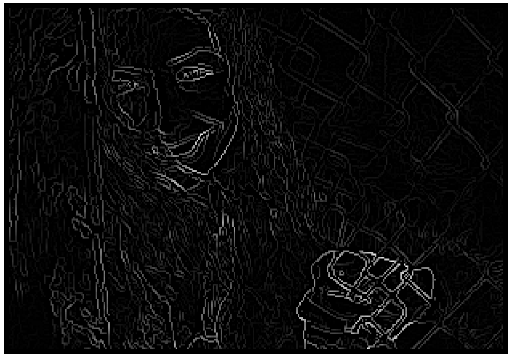
M



Canny Edge Detector
Non-Maximum Suppression

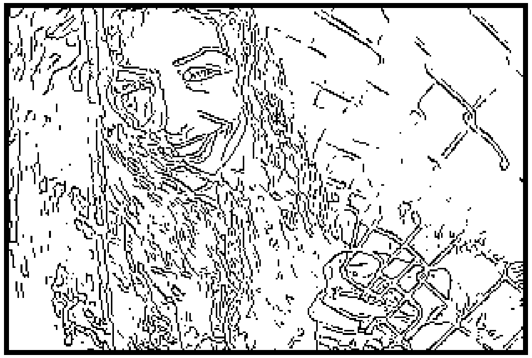
$$|\Delta S| = \sqrt{S_x^2 + S_y^2}$$

M



For visualization

$M \geq Threshold = 25$



Canny Edge Detector
Hysteresis Thresholding

Canny Edge Detector
Hysteresis Thresholding

- If the gradient at a pixel is

Canny Edge Detector
Hysteresis Thresholding

- If the gradient at a pixel is
  - above “**High**”, declare it as an ‘**edge pixel**’

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**"

Canny Edge Detector
Hysteresis Thresholding

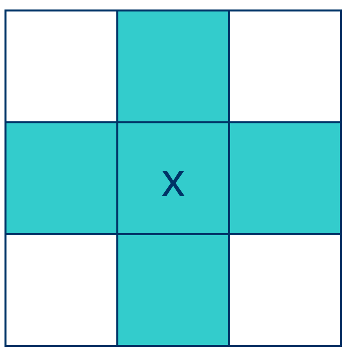
- If the gradient at a pixel is
  - above “**High**”, declare it as an ‘**edge pixel**’
  - below “**Low**”, declare it as a “**non-edge-pixel**”
  - **between** “low” and “high”
    - Consider its neighbors iteratively then declare it an “edge pixel” if it is **connected** to an ‘edge pixel’ **directly** or via pixels **between** “low” and “high”.

Canny Edge Detector
Hysteresis Thresholding

- Connectedness

Canny Edge Detector
Hysteresis Thresholding

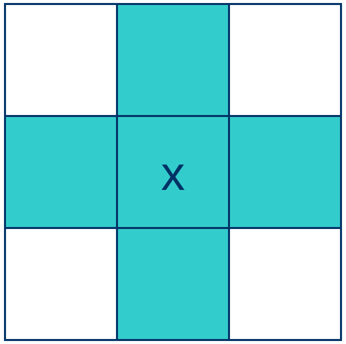
- Connectedness



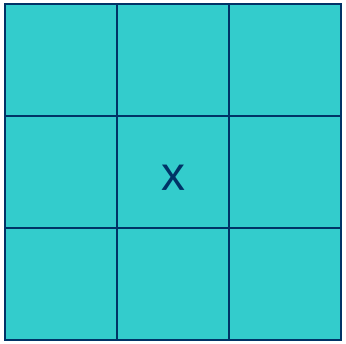
4 connected

Canny Edge Detector
Hysteresis Thresholding

- Connectedness



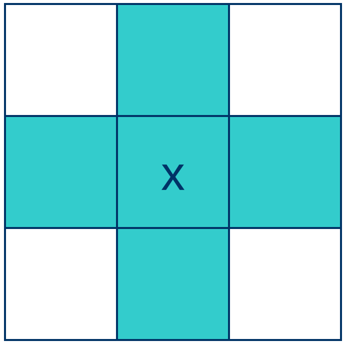
4 connected



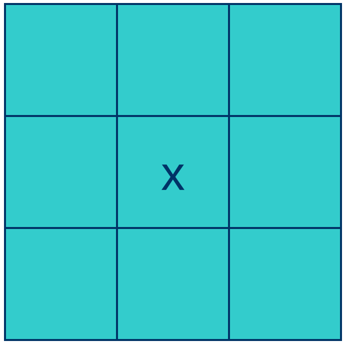
8 connected

Canny Edge Detector
Hysteresis Thresholding

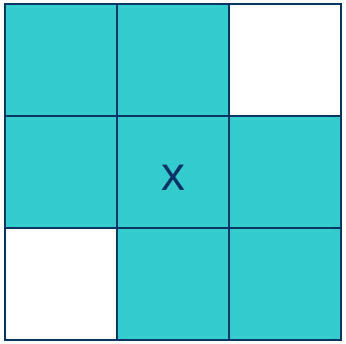
- Connectedness



4 connected



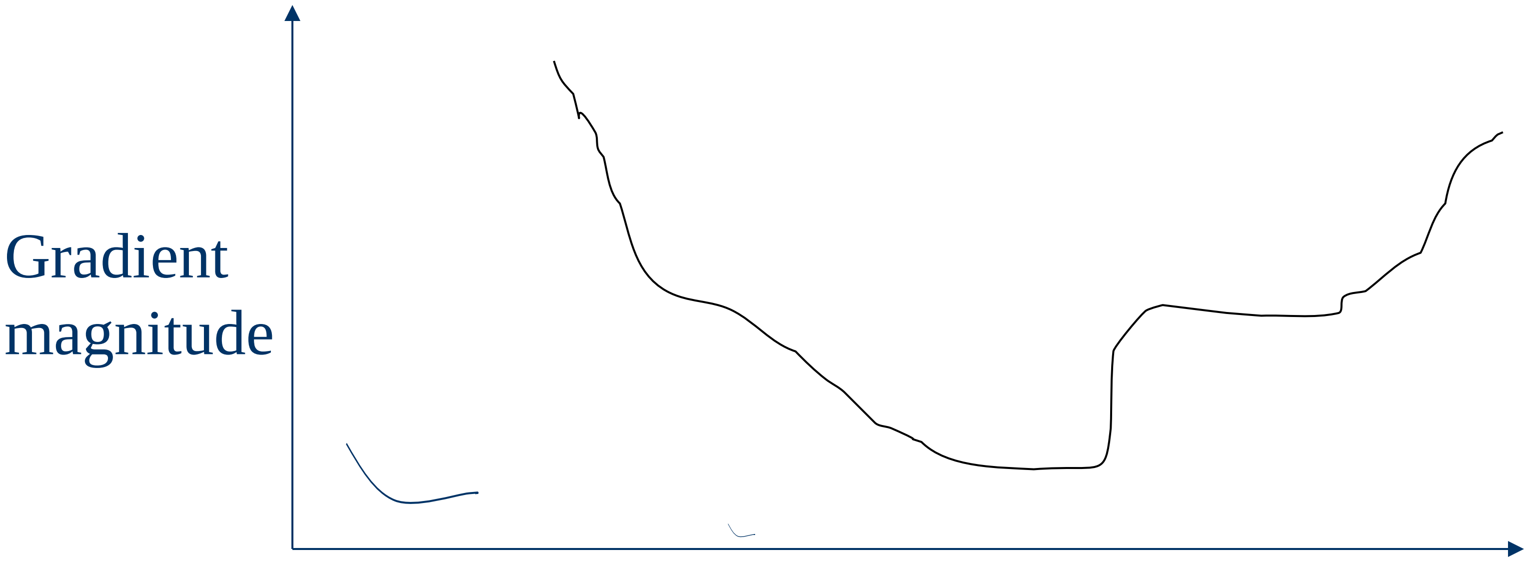
8 connected



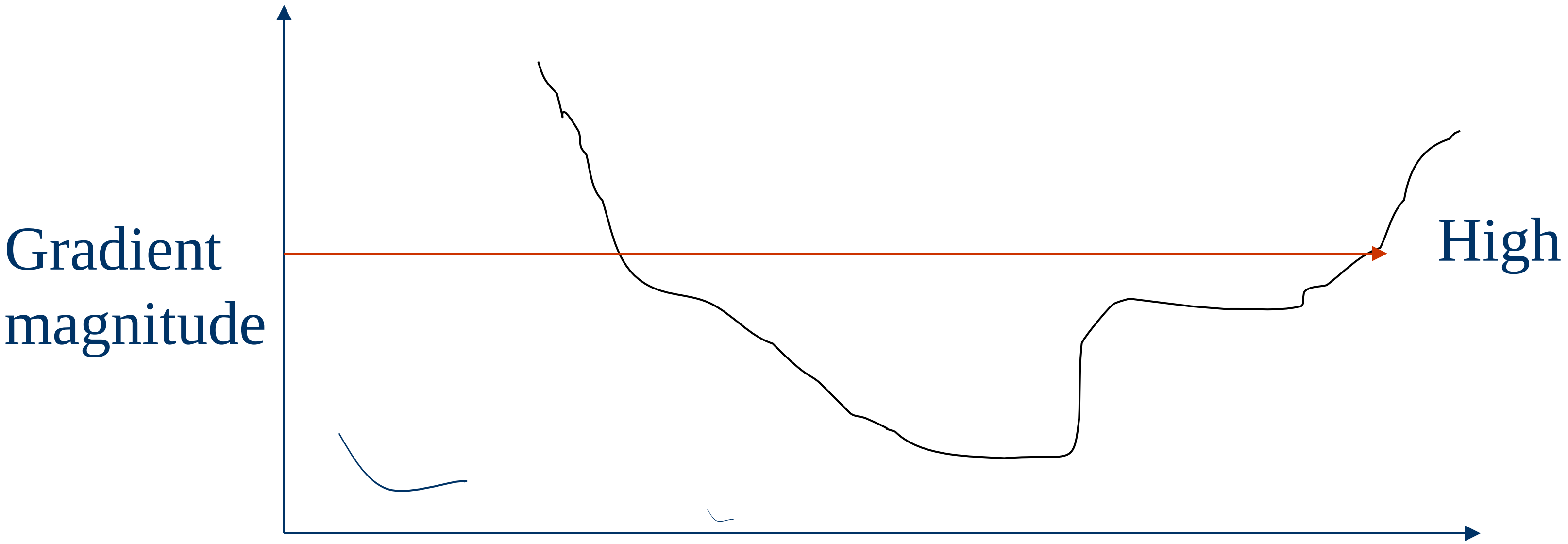
6 connected

Canny Edge Detector
Hysteresis Thresholding

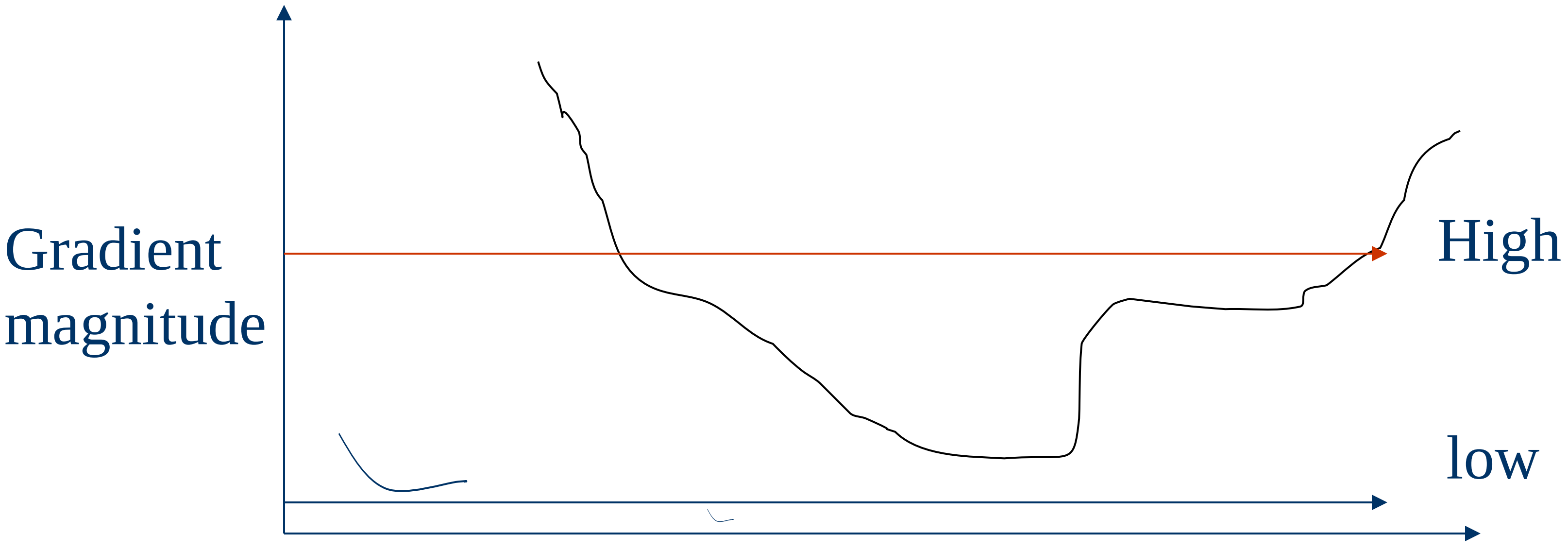
Canny Edge Detector Hysteresis Thresholding



Canny Edge Detector
Hysteresis Thresholding



Canny Edge Detector
Hysteresis Thresholding



Canny Edge Detector
Hysteresis Thresholding

Canny Edge Detector Hysteresis Thresholding

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

Canny Edge Detector
Hysteresis Thresholding

- 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

Canny Edge Detector
Hysteresis Thresholding

- 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.

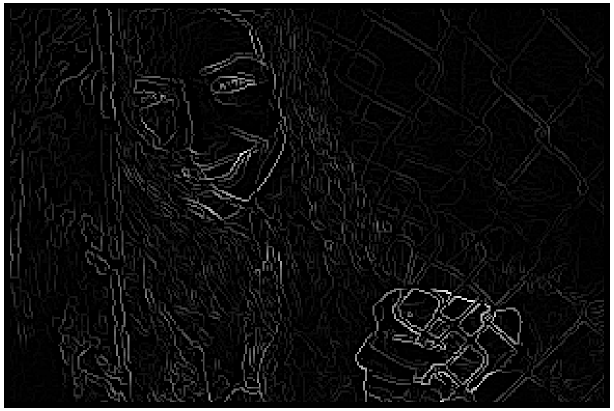
Canny Edge Detector Hysteresis Thresholding

- 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

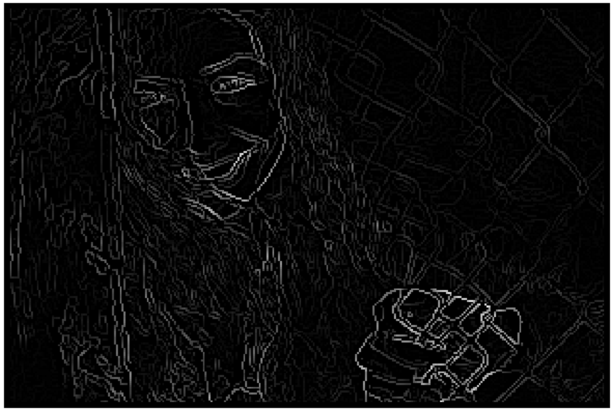
Canny Edge Detector
Hysteresis Thresholding

M



Canny Edge Detector Hysteresis Thresholding

M

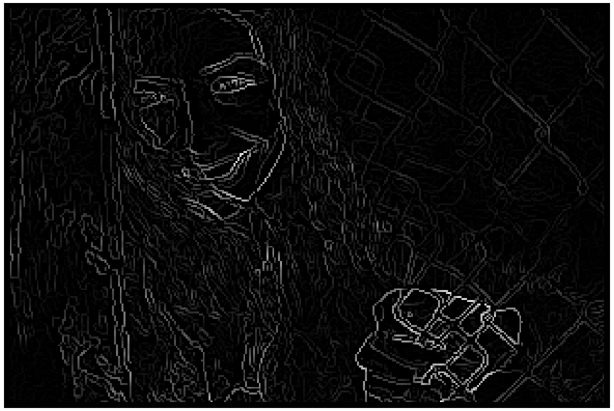


regular

$M \geq 25$

Canny Edge Detector
Hysteresis Thresholding

M



regular

$M \geq 25$

Hysteresis

$High = 35$

$Low = 15$

Before Non-max Suppression

Slide Credit: James hays

After non-max suppression

Slide Credit: James hays

Hysteresis thresholding

- Threshold at low/high levels to get weak/strong edge pixels
- Do connected components, starting from strong edge pixels

Slide Credit: James hays

Final Canny Edges

Slide Credit: James hays

Effect of σ (Gaussian kernel spread/size)

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

Source: S. Seitz

Suggested Reading

Suggested Reading

- Chapter 2, Mubarak Shah, “Fundamentals of Computer Vision”

Suggested Reading

- Chapter 2, Mubarak Shah, “Fundamentals of Computer Vision”
- Richard Szeliski, "Computer Vision: Algorithms and Applications".
  - 4.2