Welcome to Computer Vision

Computer Vision

Dr. Muhammad Tahir

DEPARTMENT OF COMPUTER SCIENCE,

FAST-NUCES, Peshawar

Course Details

LECTURES: Tuesday
& Wednesday

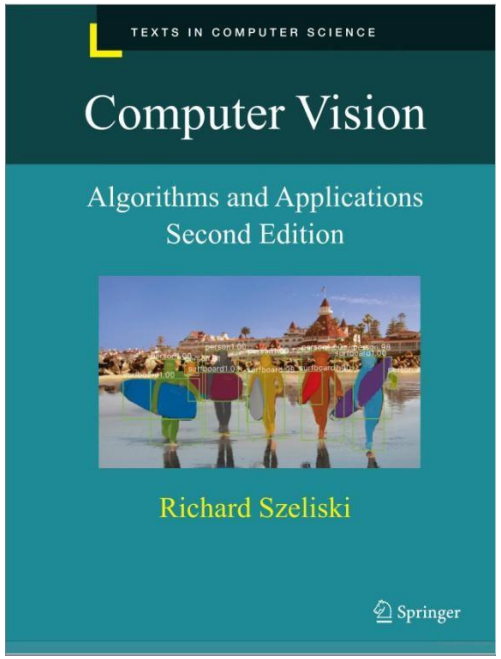
TIMINGS:

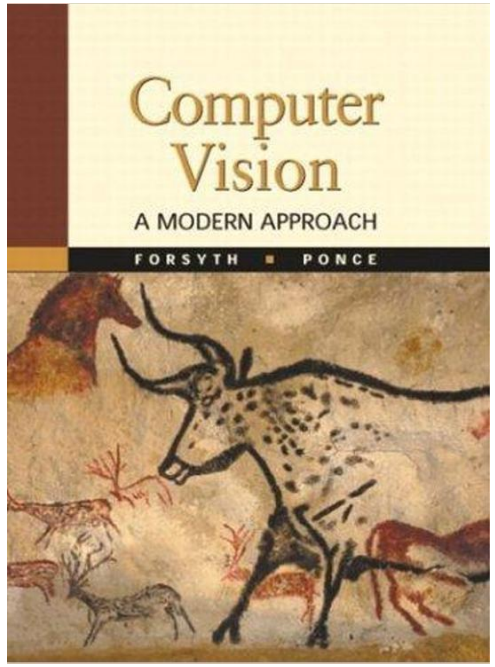
8:00 am – 9:30 am

MY OFFICE:

OFFICE HOURS:

EMAIL: m.tahir@nu.edu.pk





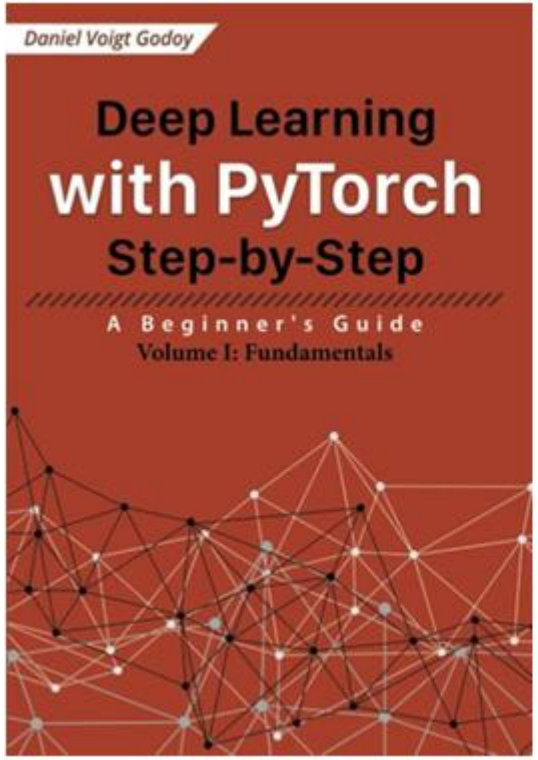
References

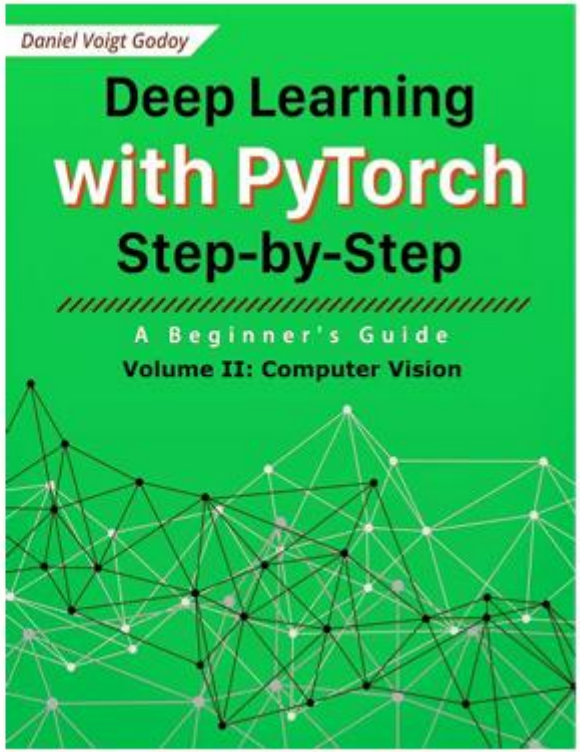
The material in these slides are based on:

1. Rick Szeliski's book: Computer Vision: Algorithms and Applications
2. Forsythe and Ponce: Computer Vision: A Modern Approach

Recommended Books

Deep Learning with PyTorch Step-by-Step by Daniel Voigt Godoy





Course Learning Outcomes

No	CLO (Tentative)	Domain	Taxonomy Level	PLO
1	Understanding basics of Computer Vision: algorithms, tools, and techniques	Cognitive	2	
2	Develop solutions for image/video understanding and recognition	Cognitive	3	
3	Design solutions to solve practical Computer Vision problems	Cognitive	3	

Outline

- Image Representation
- Image Filtering
- Gradients
- Convolution
- Correlation

Spatial Filtering for Image Sharpening

Background: to highlight fine detail in an image or to enhance blurred detail

Applications: electronic printing, medical imaging, industrial inspection, autonomous target detection (smart weapons).....

Foundation:

Blurring/smoothing is performed by spatial averaging (**equivalent to integration**)

Sharpening is performed by noting only the **gray level changes** in the image; that is the **differentiation**

Spatial Filtering for Image Sharpening

- **Operation of Image Differentiation**
  - Enhance edges and discontinuities (magnitude of output gray level >>0)
  - De-emphasize areas with slowly varying gray-level values (output gray level: 0)

- Previously we have looked at smoothing filters which remove fine detail
- Sharpening spatial filters seek to highlight fine detail
  - Remove blurring from images
  - Highlight edges
- Sharpening filters are based on spatial differentiation

First and Second Order Derivatives

$$\frac{\partial f}{\partial x} = f(x+1) - f(x)$$

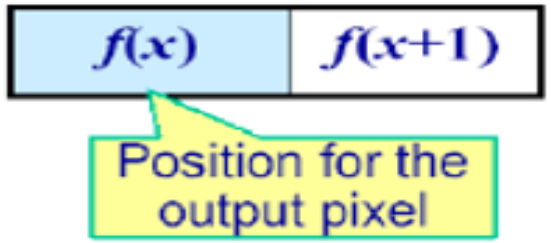
$$\frac{\partial^2 f}{\partial x^2} = f'(x+1) - f'(x)$$

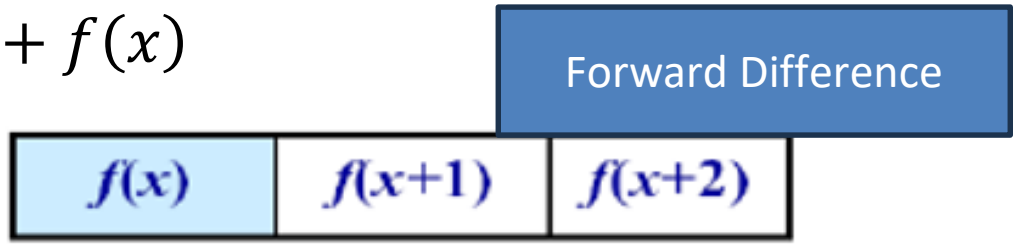
$$\frac{\partial^2 f}{\partial x^2} = [f(x+2) - f(x+1)] - [f(x+1) - f(x)]$$

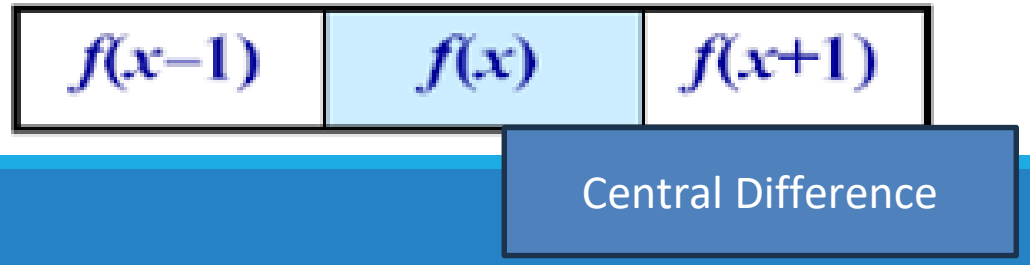
$$\frac{\partial^2 f}{\partial x^2} = f(x+2) - f(x+1) - f(x+1) + f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+2) - 2f(x+1) + f(x)$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$



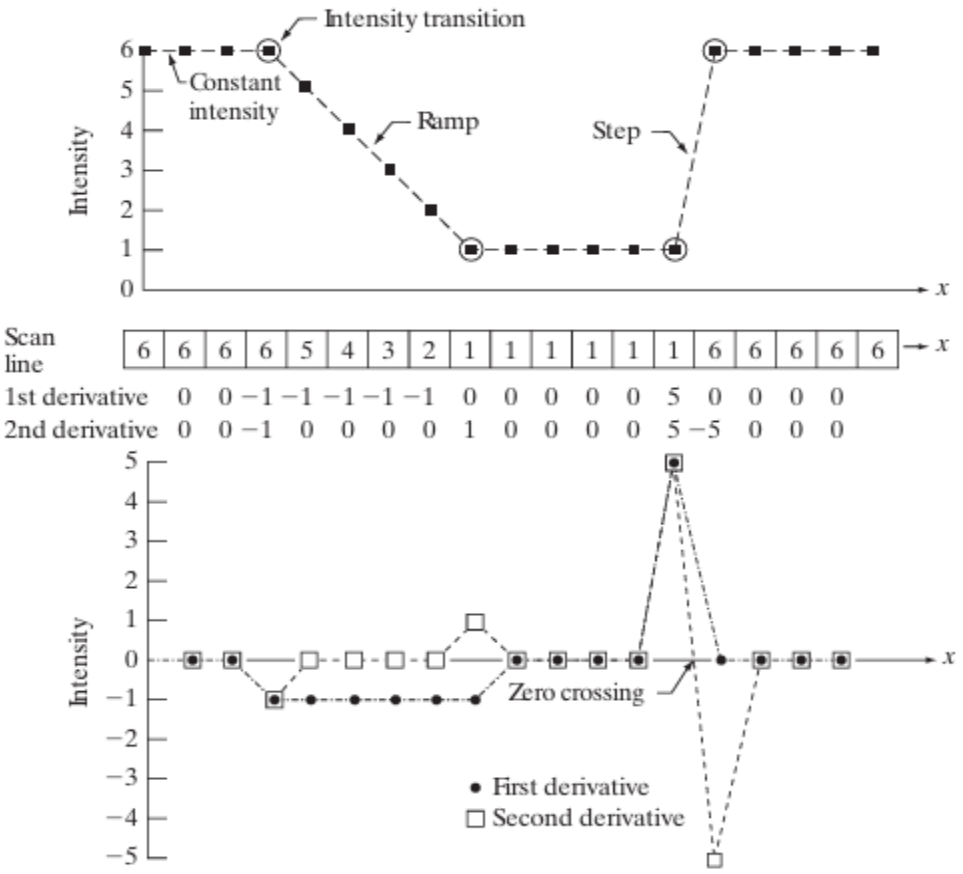




Sharpening Spatial Filters

- Definition for a first derivative must be
  - Zero in flat segments (areas of constant gray level)
  - Nonzero at the onset of a gray-level step or ramp
  - Nonzero along ramps $\frac{\partial f}{\partial x} = f(x+1) - f(x)$
- Definition of a second derivative must be
  - Zero in flat areas
  - Nonzero at the onset and end of a gray level step or ramp
  - Zero along ramps of constant slope

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$$



First and Second Order Derivatives

- Comparing the 1st and 2nd derivatives we can conclude the following:
  - 1st order derivatives generally produce thicker edges
  - 2nd order derivatives have a stronger response to fine detail e.g. thin lines
  - 1st order derivatives have stronger response to a grey level step
  - 2nd order derivatives produce a double response at step changes in grey level

Using the Second Derivative for Image Enhancement

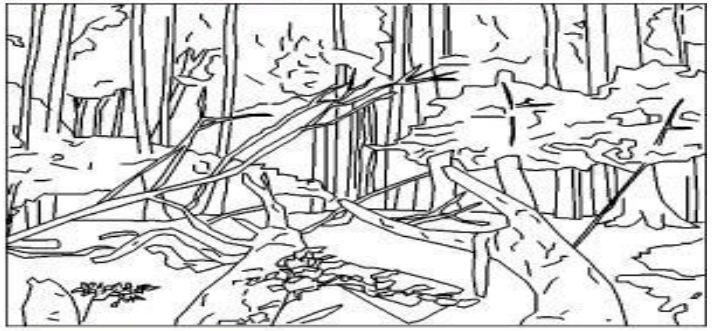
- The 2nd derivative is more useful for image enhancement than the 1st derivative
  - Stronger response to fine detail
  - Simpler implementation
- The 1st derivative used together with the 2nd derivative results in impressive enhancement effect

Various situations encountered for derivatives

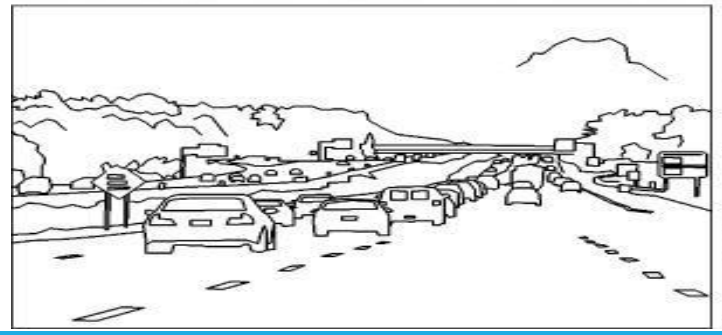
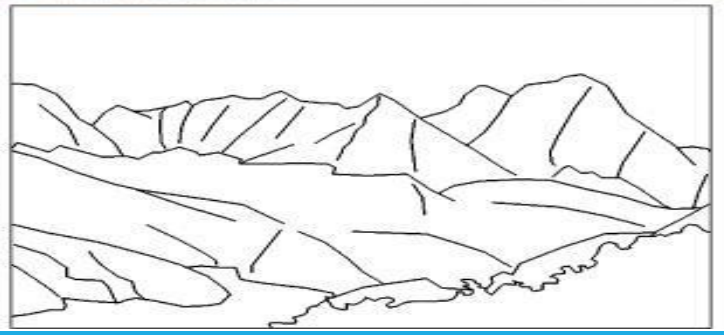
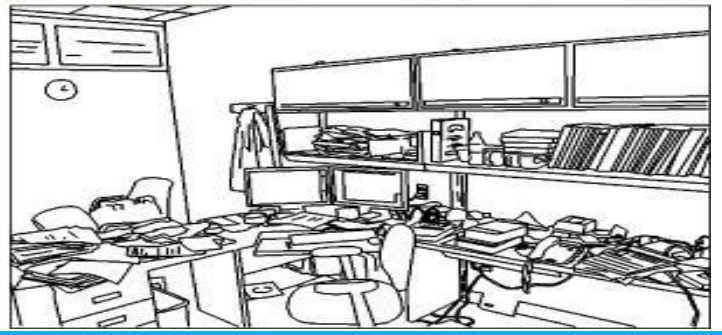
- Ramps or Steps in the 1D profile normally characterize edges in an image
- $f^{"}$is nonzero at the onset and end of the ramp: producing thin (double) edges
- f' is nonzero along the entire ramp producing thick edges





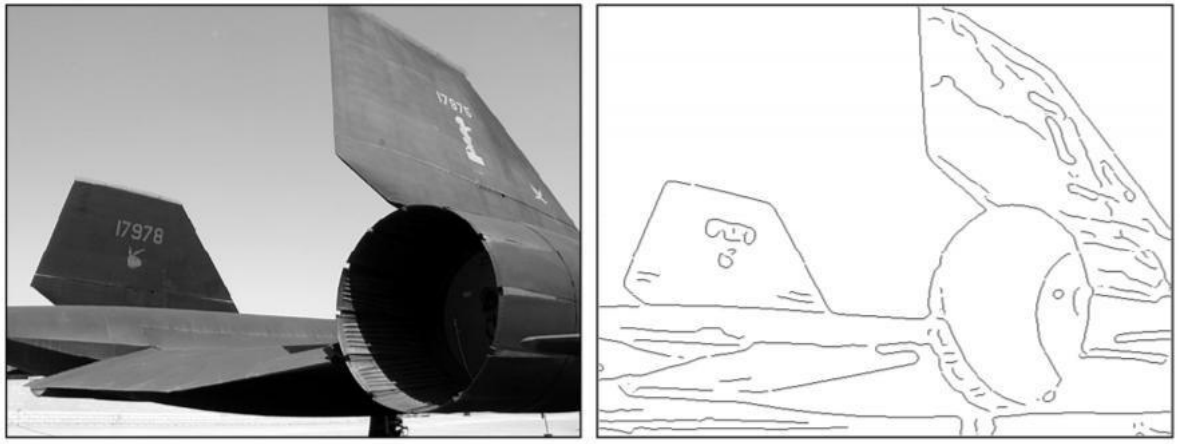





Edge Detection

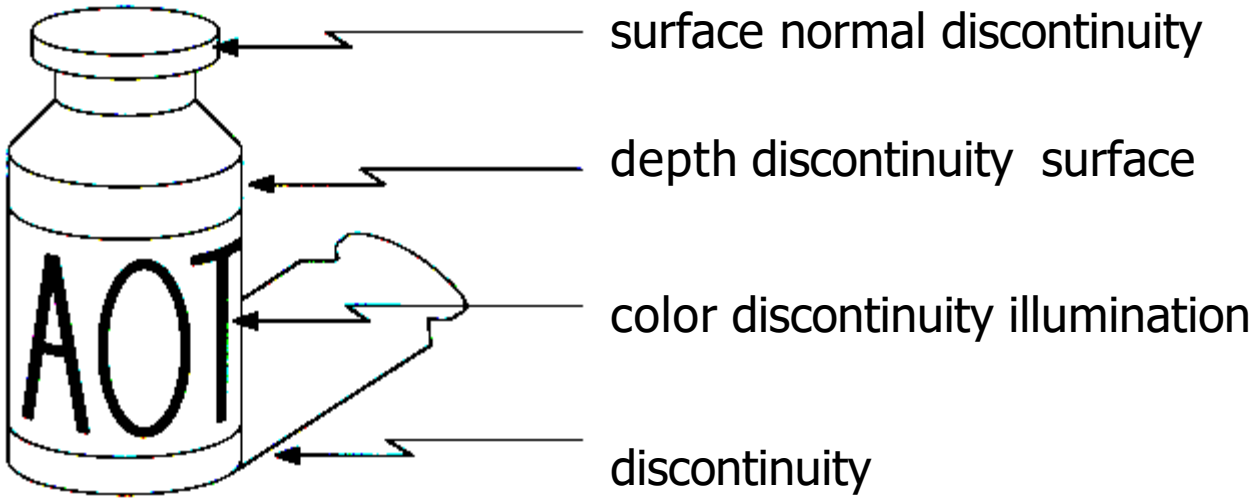
- Edges are sharp change in brightness (discontinuities).
- Goal: Identify sudden changes in an image
  - Intuitively, most semantic and shape information from the image can be encoded in the edges
  - More compact than pixels



(a) (b)

Origin of Edges

- In most cases, edges are the boundaries between objects.



Edge Detection

- Basic idea: look for a neighborhood with strong signs of change.
- Problems:
  - Neighborhood size
  - How to detect change

Differential operators:

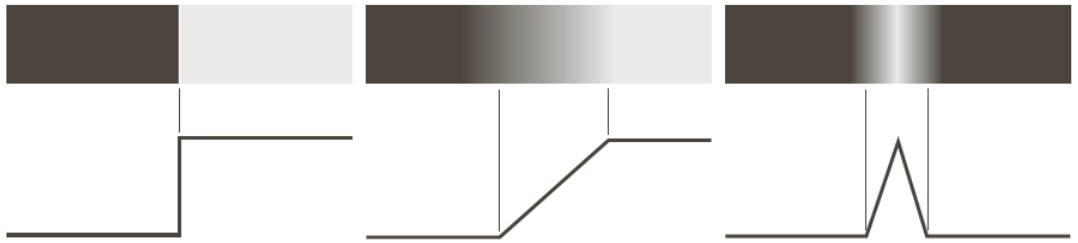
- Attempt to approximate the gradient at a pixel via masks.

  Threshold the gradient to select the edge pixels

81	**82**	26	24
82	33	25	25
81	**82**	26	24

Characteristics of an Edge

- Edge: A sharp change in brightness
- Edge models/types:
  - Step
  - Ramp
  - Roof



Steps for Edge Detection

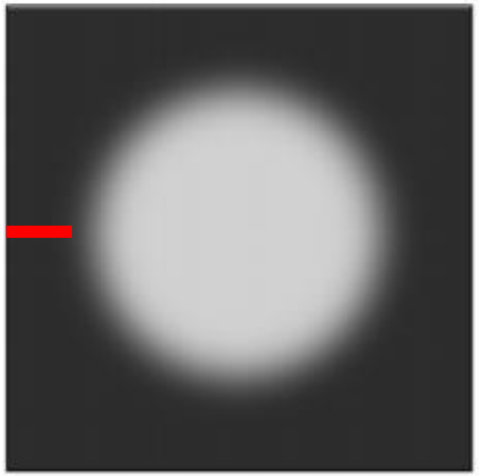
- Main Steps in Edge Detection
  - **Smoothing:** suppress as much noise as possible, without destroying true edges.
  - **Enhancement:** apply differentiation to enhance the quality of edges (i.e., sharpening)
  - **Thresholding:** determine which edge pixels should be discarded as noise and which should be retained (i.e., threshold edge magnitude).
  - **Localization:** determine the exact edge location.

Edge Detection Using Derivatives

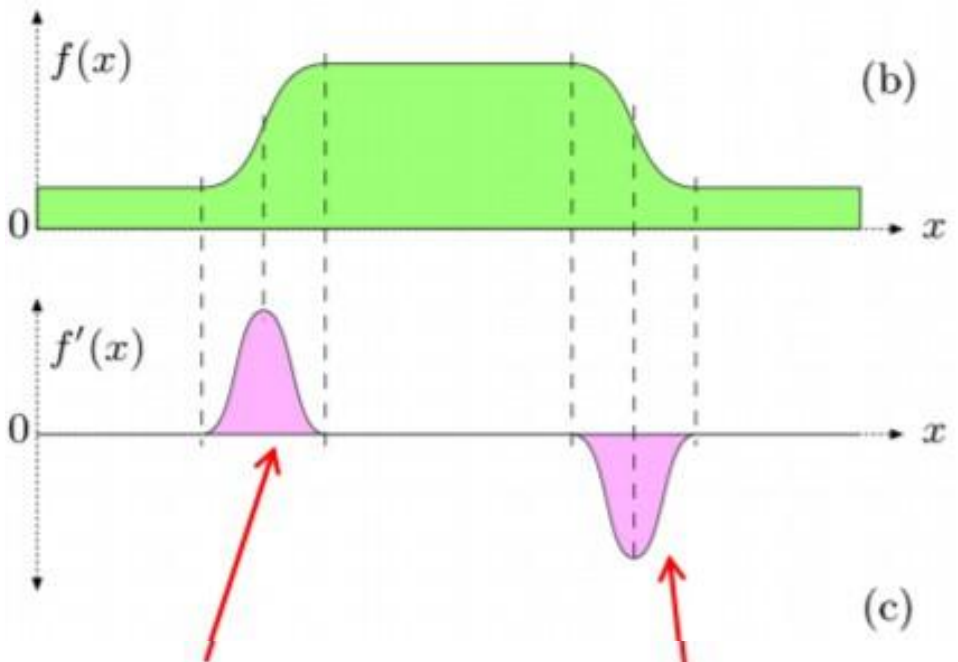
Edge Detection Using Derivatives

- Edge detection using derivatives
  - Calculus describes changes of continuous functions using derivatives
- An image is a 2D function, so operators describing edges are expressed using **partial derivatives**.
- Points which lie on an edge can be detected by either:
    - detecting local maxima or minima of the first derivative
    - detecting the zero-crossing of the second derivative

Interpreting Derivatives



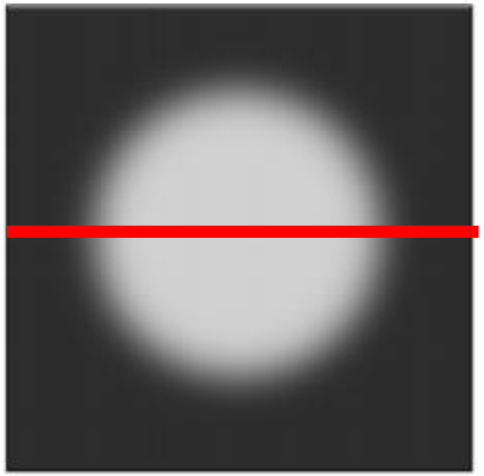
(a)



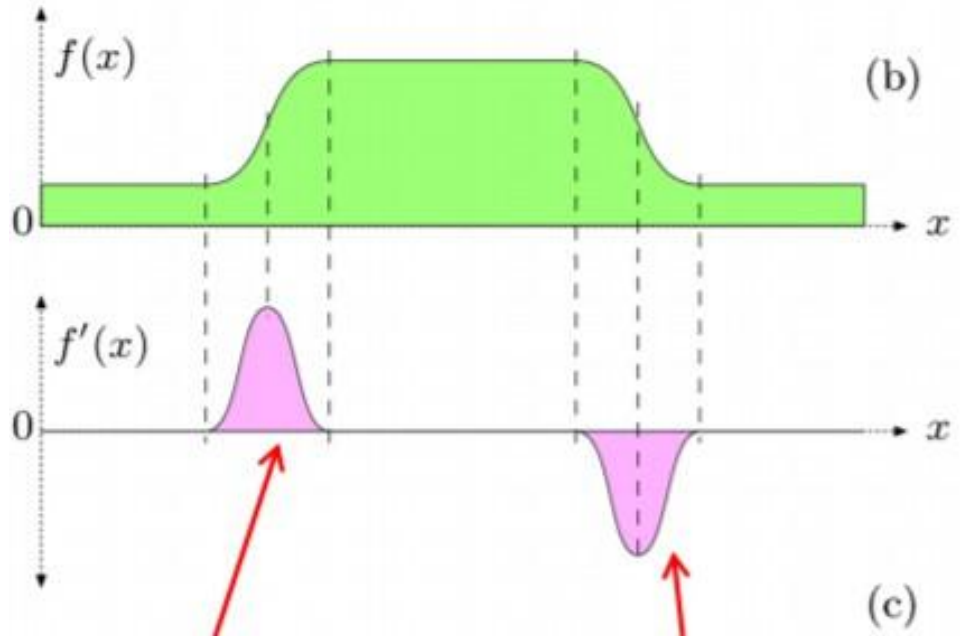
Rising slope causes positive
+ high value first derivative

Falling slope causes negative
+ high value first derivative

Interpreting Derivatives



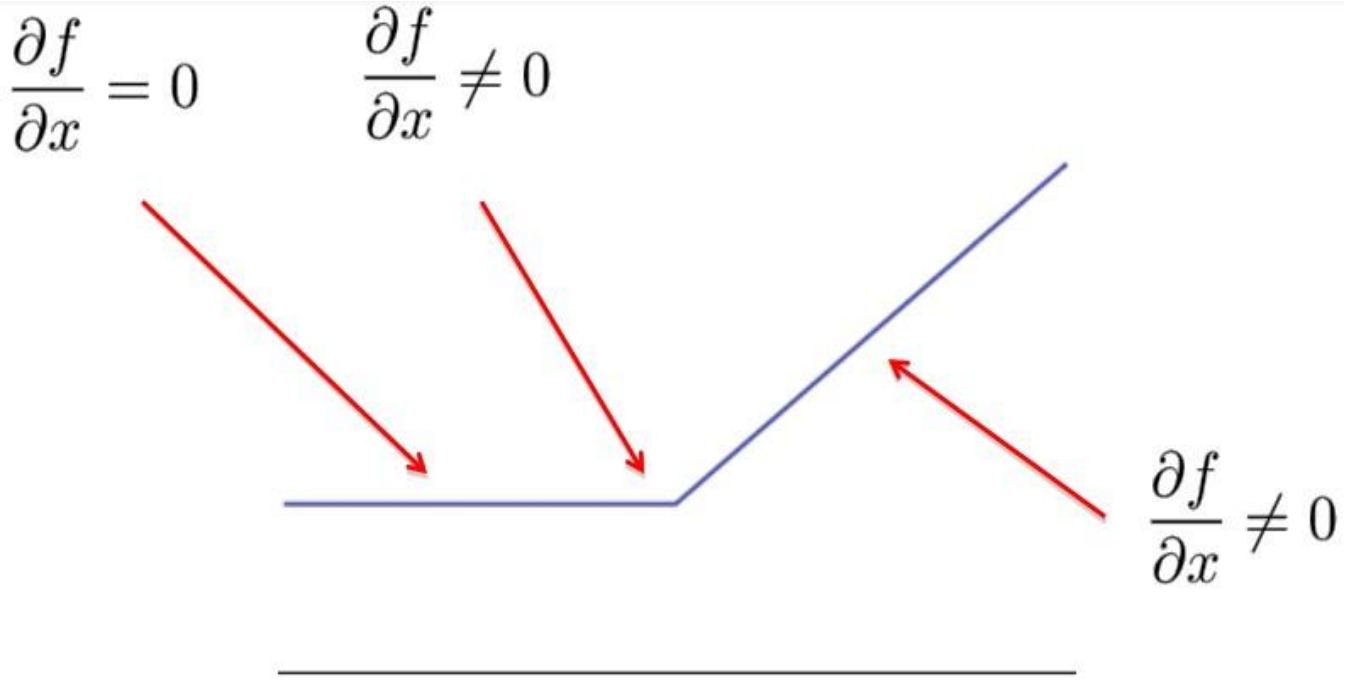
(a)

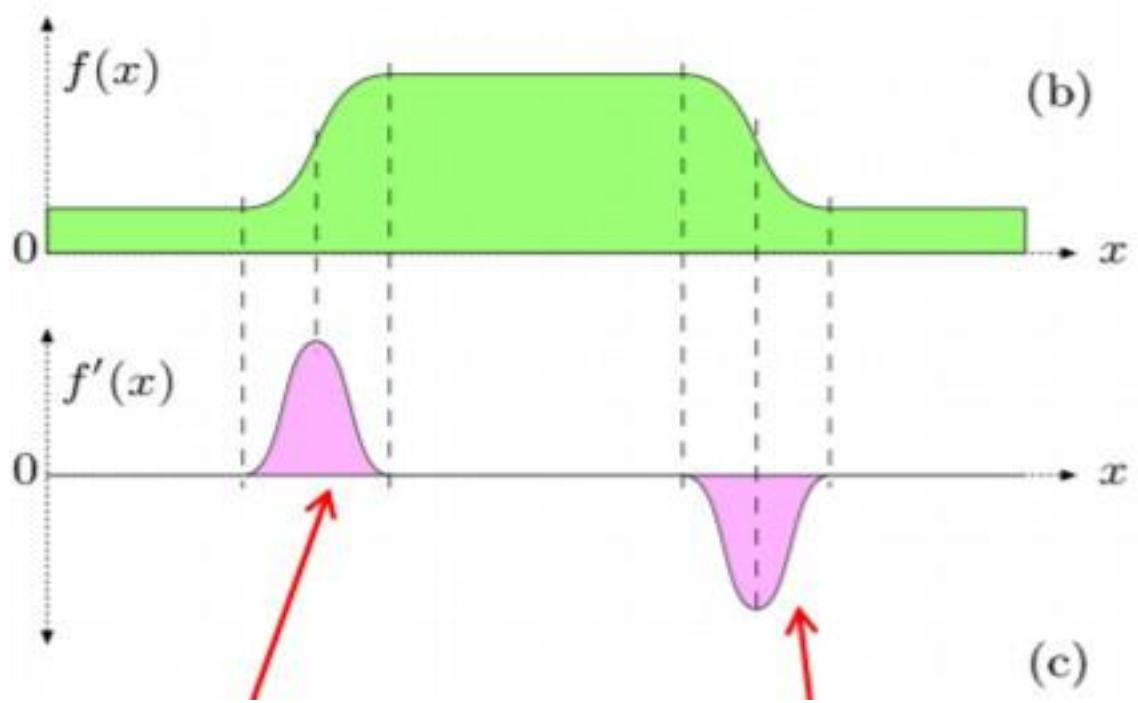


Rising slope causes positive
+ high value first derivative

Falling slope causes negative
+ high value first derivative

Interpreting Derivatives



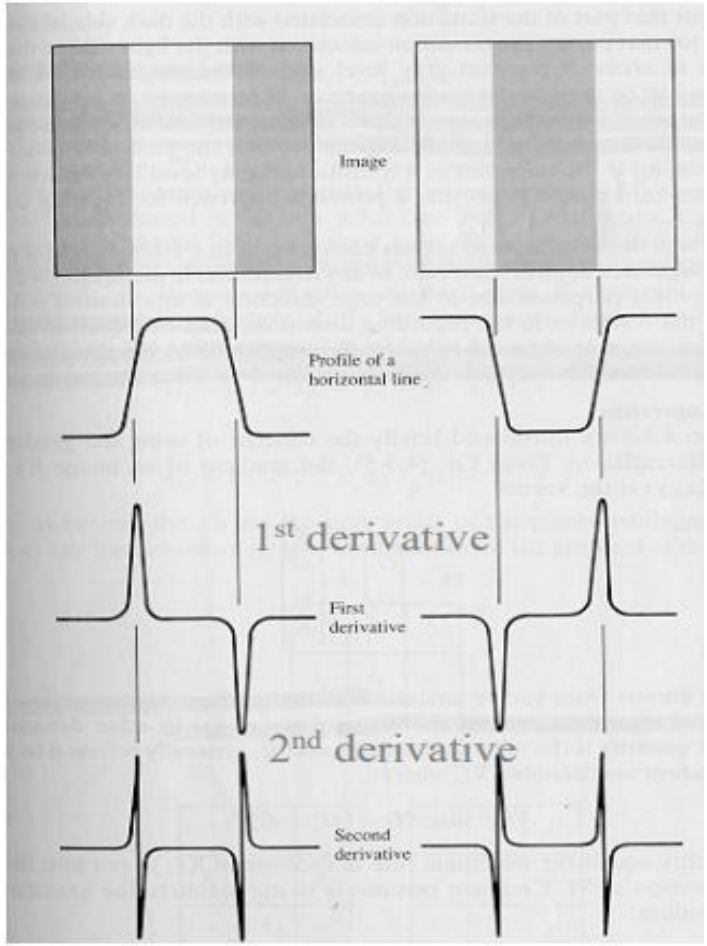


2^{nd} Order Derivative Interpretation

- 2^{nd} order derivative is a derivative of a 1^{st} order derivative
- It is the rate of change of a function w.r.t. to a variable.
- 2^{nd} order derivatives enhance fine detail much better

Edge Detection Using Derivative

- Often, points that lie on an edge are detected by:
  1) Detecting the local maxima or minima of the first derivative.
  2) Detecting the zero-crossings of the second derivative.



Sharpening Filters: Gradient

- Taking the derivative of an image results in sharpening the image
- The derivative of an image (i.e., 2D signal) can be computed using the gradient
- The gradient is a vector which has **magnitude** and **direction**
- We use magnitude of gradient vector as strength indicator of an edge, and its direction as normal to the edge i.e.
  - **Gradient magnitude:** provides information about edge strength.
  - **Gradient direction:** perpendicular to the direction of the edge.

$$\nabla f \equiv grad(f) \equiv \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

$$magnitude\left(grad(f)\right) = \sqrt{{g_x}^2 + {g_y}^2}$$

$$direction\left(grad(f)\right) = \tan^{-1}\left({}^{g_x}/_{g_y}\right)$$

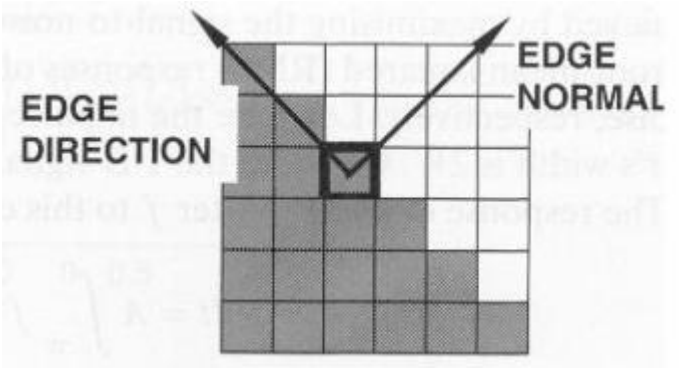
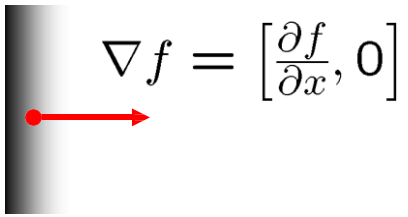
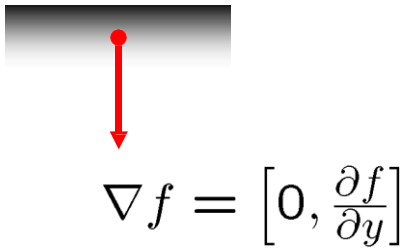


Image Gradient

$$\nabla f = \left[\frac{\partial f}{\partial x}, \frac{\partial f}{\partial y}\right]$$

- Let f(x,y) be a 2D function (ex. image)
- Gradient: Vector whose direction is in direction of maximum rate of change of *f* and whose magnitude is maximum rate of change of *f*





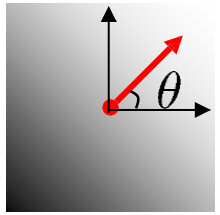


Image Gradient

- Gradient is perpendicular to edge contour
- The gradient direction is given by:

$$\theta = \tan^{-1}\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

- The edge strength is given by the gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

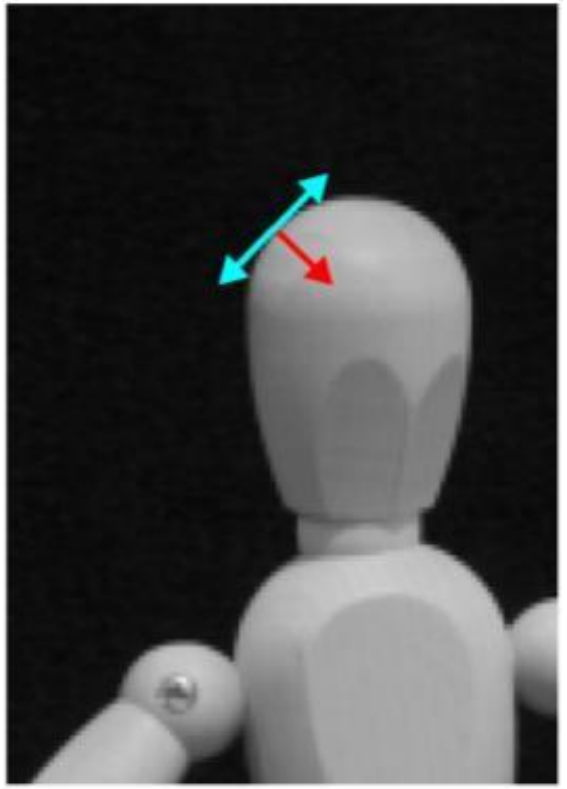
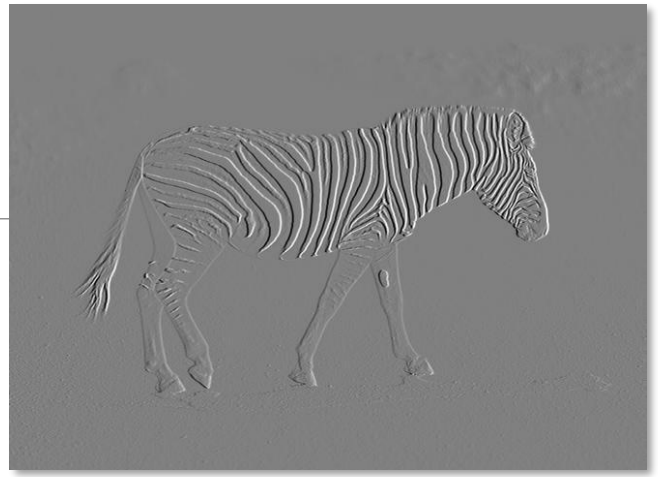


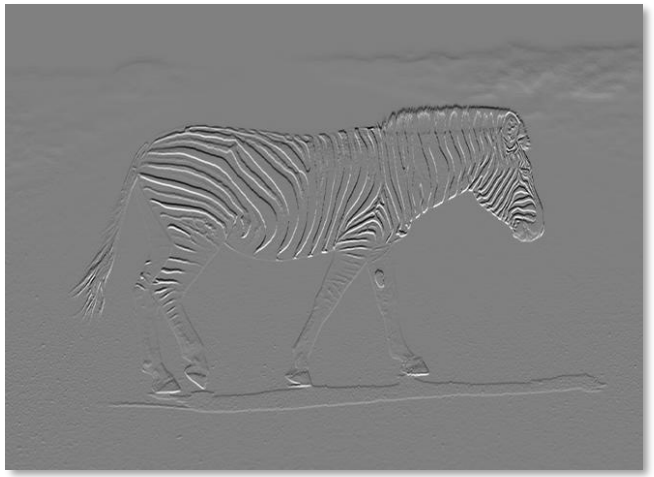
Image Gradient

f



$\frac{\partial f}{\partial x}$

$\|\nabla f\| = \sqrt{(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2}$



$\frac{\partial f}{\partial y}$

Image Derivatives

- How can we differentiate a digital image F[x,y]?
  - Option 1: reconstruct a continuous image, f, then compute the derivative
  - Option 2: take discrete derivative (finite difference)

$$\frac{\partial f}{\partial x}[x, y] \approx F[x+1, y] - F[x, y]$$

Pick Option 2 for Digital Processing

Gradient Computation

Approximate gradient using finite differences:

$$\nabla f \approx \left|(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)\right| + \left|(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)\right|$$

$$g_x = \frac{\partial f}{\partial x} = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

$$g_y = \frac{\partial f}{\partial y} = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

That is based on the following coordinates

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

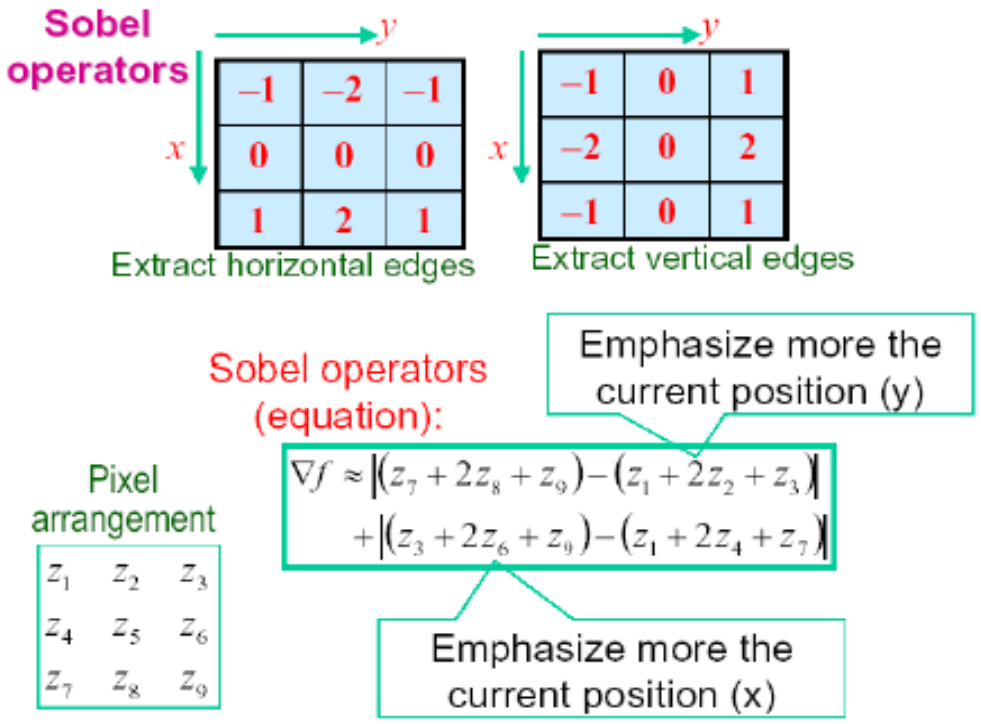
Sobel Operators - Example

Based on the previous equations we can derive the Sobel Operators

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

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

To filter an image, it is filtered using both operators the results of which are added together



Gradient Operators

For a sub-image given in the figure:
The gradient is approximated by

$$\nabla f \approx |z_9 - z_5| + |z_8 - z_6|$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

This equation can be represented in the following masks (Robert cross gradient operators)

-1	0
0	1

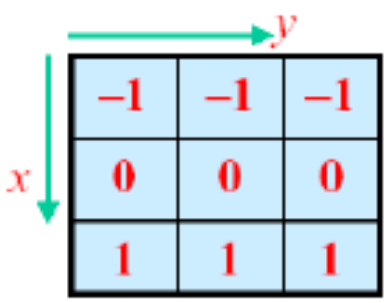
0	-1
1	0

Gradient Operators

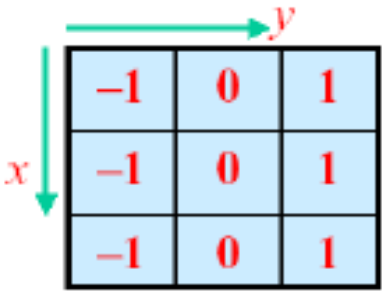
Normally the smallest mask used is of size 3 x 3

Based on the concept of approximating the gradient, several spatial masks have been proposed:

Prewitt operator



Extract horizontal edges



Extract vertical edges

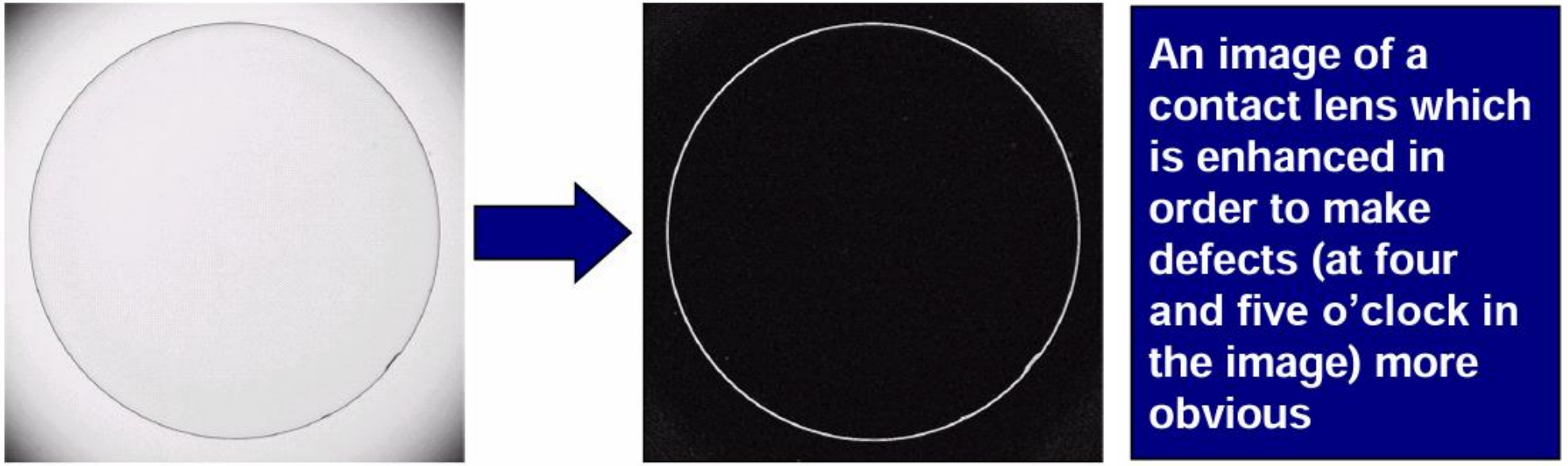
Pixel arrangement

$$\begin{bmatrix} z_1 & z_2 & z_3 \\ z_4 & z_5 & z_6 \\ z_7 & z_8 & z_9 \end{bmatrix}$$

Prewitt operators (equation):

$$\nabla f \approx \left|(z_7 + z_8 + z_9) - (z_1 + z_2 + z_3)\right| + \left|(z_3 + z_6 + z_9) - (z_1 + z_4 + z_7)\right|$$

Sobel Example



Sobel filters are typically used for edge detection

Directional Derivative

- Let **f(x, y) be a function** mapping **two real numbers** to **a real number** (intensity values).
- For the directional derivative of *f* along the x axis, we use notation df/dx.
  - Vertical edges correspond to points in g with high df/dx.
- For the directional derivative of *f* along the y axis, we use notation df/dy.
  - Horizontal edges correspond to points in g with high dg/dy.

Directional Derivative

- Interpreting
  - Results far from zero (positive and negative) correspond to strong vertical edges.
  - Results close to zero correspond to weak vertical edges, or no edges whatsoever.

DERIVATIVES IN 1D

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x) = f_x$$

DERIVATIVES IN 1D - EXAMPLE

$$y = x^2 + x^4$$

$$\frac{dy}{dx} = 2x + 4x^3$$

DERIVATIVES IN 1D - EXAMPLE

$$y = x^2 + x^4 \qquad y = \sin x + e^{-x}$$

$$\frac{dy}{dx} = 2x + 4x^3 \qquad \frac{dy}{dx} = \cos x + (-1)e^{-x}$$

DISCRETE DERIVATIVE IN 1D

- This is the definition of the derivative.
- It measures how much $f(x)$ changes as x changes by a tiny amount Δx

$$\frac{df}{dx} = \lim_{\Delta x \to 0} \frac{f(x) - f(x - \Delta x)}{\Delta x} = f'(x)$$

- In digital images (or discrete signals), we don't have infinitesimal steps. Instead, pixels are sampled at unit intervals.
- So here $\Delta x = 1$

$$\frac{df}{dx} = \frac{f(x) - f(x-1)}{1} = f'(x)$$

- Since dividing by 1 doesn't change the value:

$$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

This is called the **backward difference approximation** of the derivative.

TYPES OF DISCRETE DERIVATIVE IN 1D

Backward $$\frac{df}{dx} = f(x) - f(x-1) = f'(x)$$

Forward $$\frac{df}{dx} = f(x) - f(x+1) = f'(x)$$

Central $$\frac{df}{dx} = f(x+1) - f(x-1) = f'(x)$$

DERIVATIVE FILTERS

- Can we make linear filter that computes central difference?
- Yes, through kernels and convolution

1D DISCRETE DERIVATE FILTERS

- Backward filter: H = [0 1 -1] $f(x) - f(x-1) = f'(x)$
- Forward: H = [-1 1 0] $f(x) - f(x+1) = f'(x)$
- Central: H = [1 0 -1] $f(x+1) - f(x-1) = f'(x)$

Note: Derivative kernels sum to zero

1D DISCRETE DERIVATE EXAMPLE

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

Forward derivative:

$$f'(x) = 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

H = [-1 1 0]
(Kernel)

1D DISCRETE DERIVATE EXAMPLE

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

Forward derivative:

$$f'(x) = 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

H = [-1 1 0]
(Kernel)

1D DISCRETE DERIVATE EXAMPLE

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

Forward derivative:

$$f'(x) = \quad 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

H = [-1 1 0]
(Kernel)

1D DISCRETE DERIVATE EXAMPLE

$$f(x) = 10 \quad 15 \quad 10 \quad 10 \quad 25 \quad 20 \quad 20 \quad 20$$

Forward derivative:

$$f'(x) = 0 \quad 5 \quad -5 \quad 0 \quad 15 \quad -5 \quad 0 \quad 0$$

H = [-1 1 0]
(Kernel)

DIFFERENTIAL OPERATORS

- Differential operators: some operators that when applied to the image return some derivatives.
- Model these "operators" as masks/kernels which when applied to the image yields a new function that is the image gradient function.
- Threshold this gradient function to select the edge pixels.

2D DERIVATIVE FILTERS- EXAMPLE

What happens when we apply this filter?

$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

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

$$I_y = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

2D DERIVATIVE FILTERS- EXAMPLE

What about this filter?

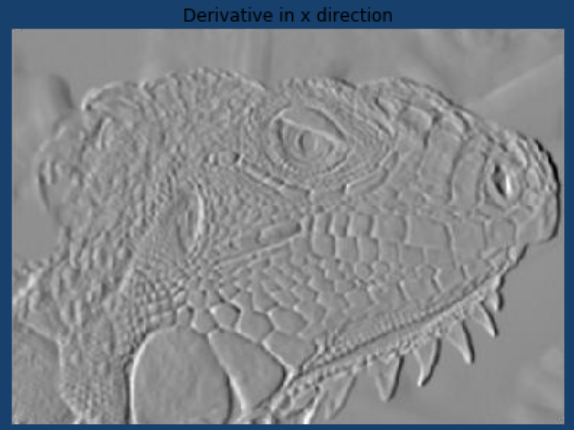
$$I = \begin{bmatrix} 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \\ 10 & 10 & 20 & 20 & 20 \end{bmatrix}$$

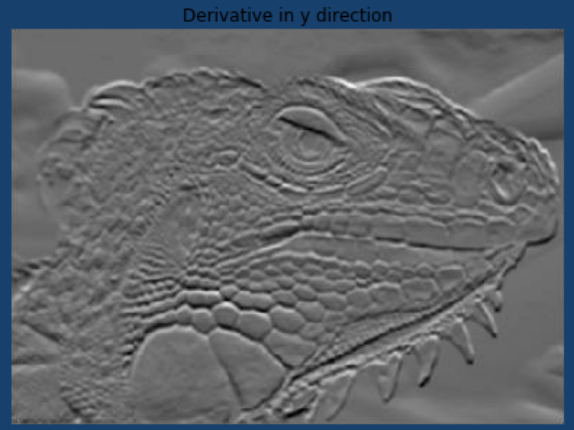
$$\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

$$I_x = \begin{bmatrix} 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \\ 0 & 10 & 10 & 0 & 0 \end{bmatrix}$$

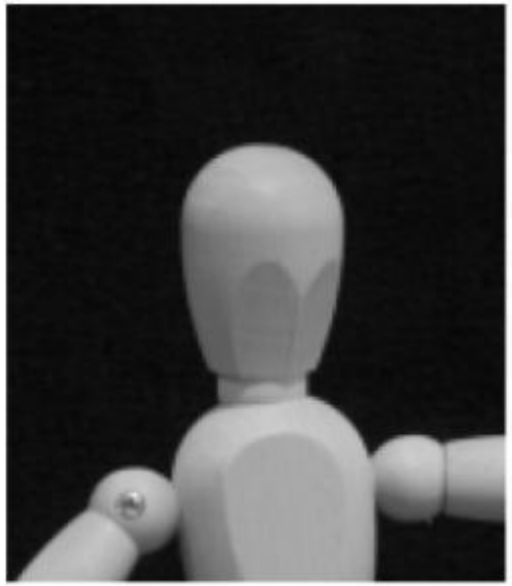
3X3 IMAGE GRADIENT FILTERS

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



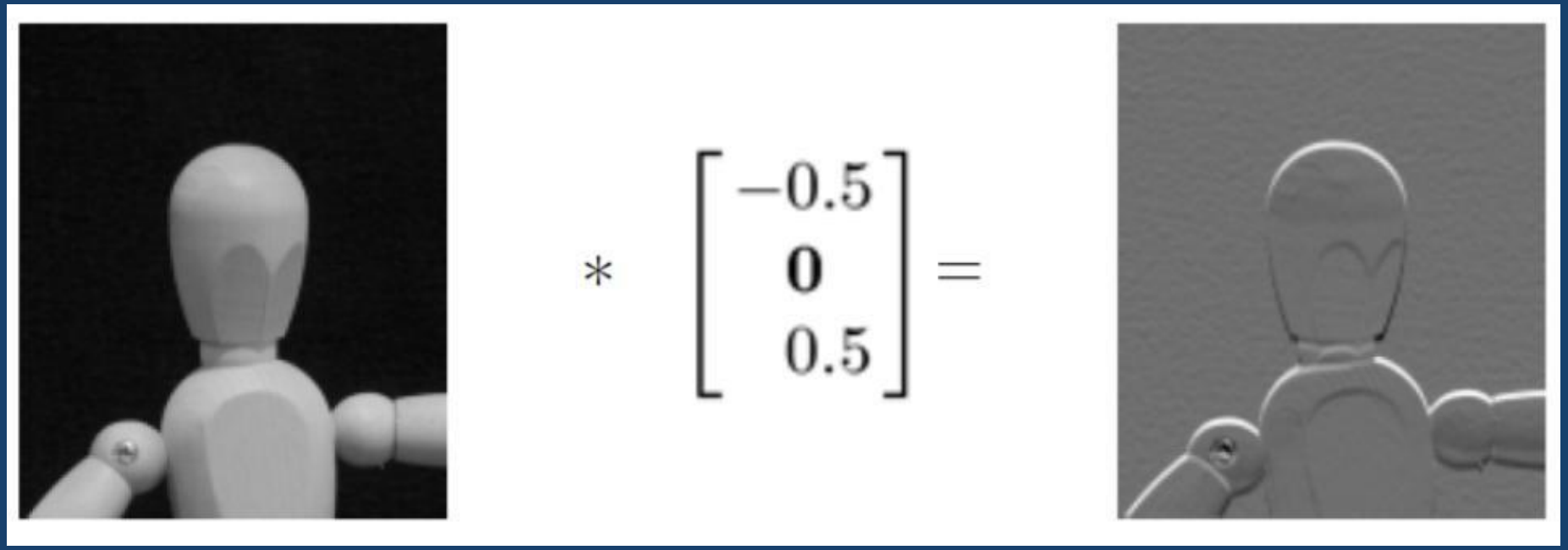


X-DERIVATIVE OF IMAGE USING CENTRAL DIFFERENCE



$$* \begin{bmatrix} -0.5 & 0 & 0.5 \end{bmatrix} =$$

Y-DERIVATIVE OF IMAGE USING CENTRAL DIFFERENCE

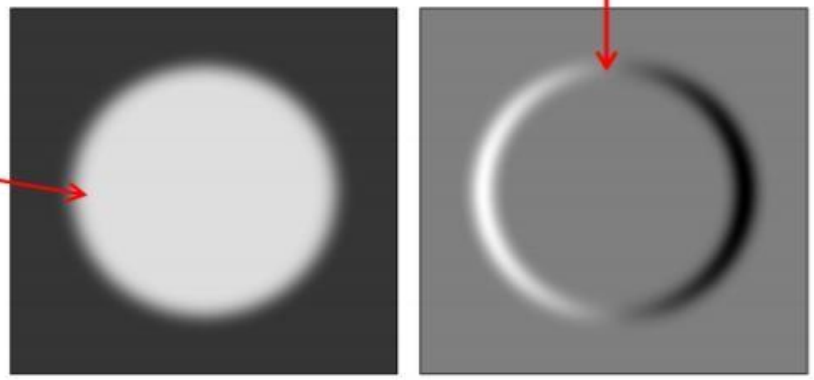


DERIVATIVE FILTERS-EXAMPLE

Gradient slope in horizontal direction

$$H_x^D = [-0.5 \quad \mathbf{0} \quad 0.5] = 0.5 \cdot [-1 \quad \mathbf{0} \quad 1]$$

A synthetic image

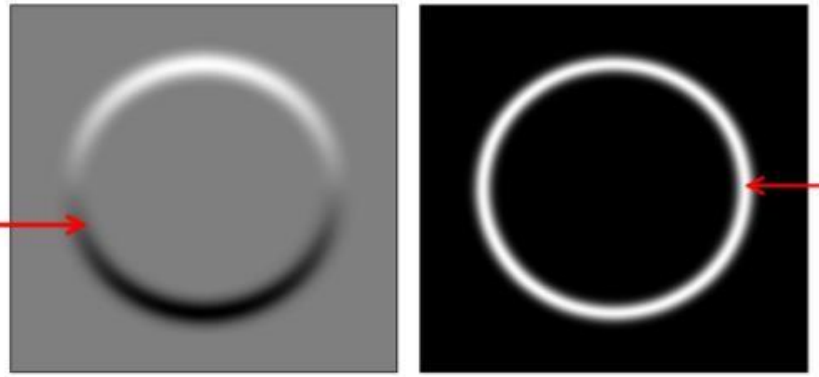


(a)

(b)

$$H_y^D = \begin{bmatrix} -0.5 \\ \mathbf{0} \\ 0.5 \end{bmatrix} = 0.5 \cdot \begin{bmatrix} -1 \\ \mathbf{0} \\ 1 \end{bmatrix}$$

Gradient slope in vertical direction



(c)

(d)

Magnitude of gradient

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Using the Second Derivative for Image Sharpening

- The Laplacian Filter
  - The simplest isotropic filter
  - Rotation invariant
- Laplacian Filter produces an image that has grayish edge lines and other discontinuities on a dark, featureless background.

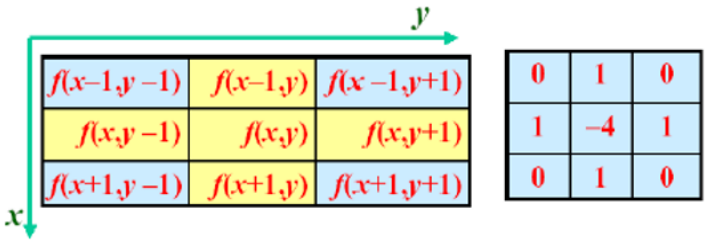
$$\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$$

$$\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$$

$$\nabla^2 f = f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1) - 4f(x, y)$$

We can easilv build a filter based on this



Variant of Laplacian

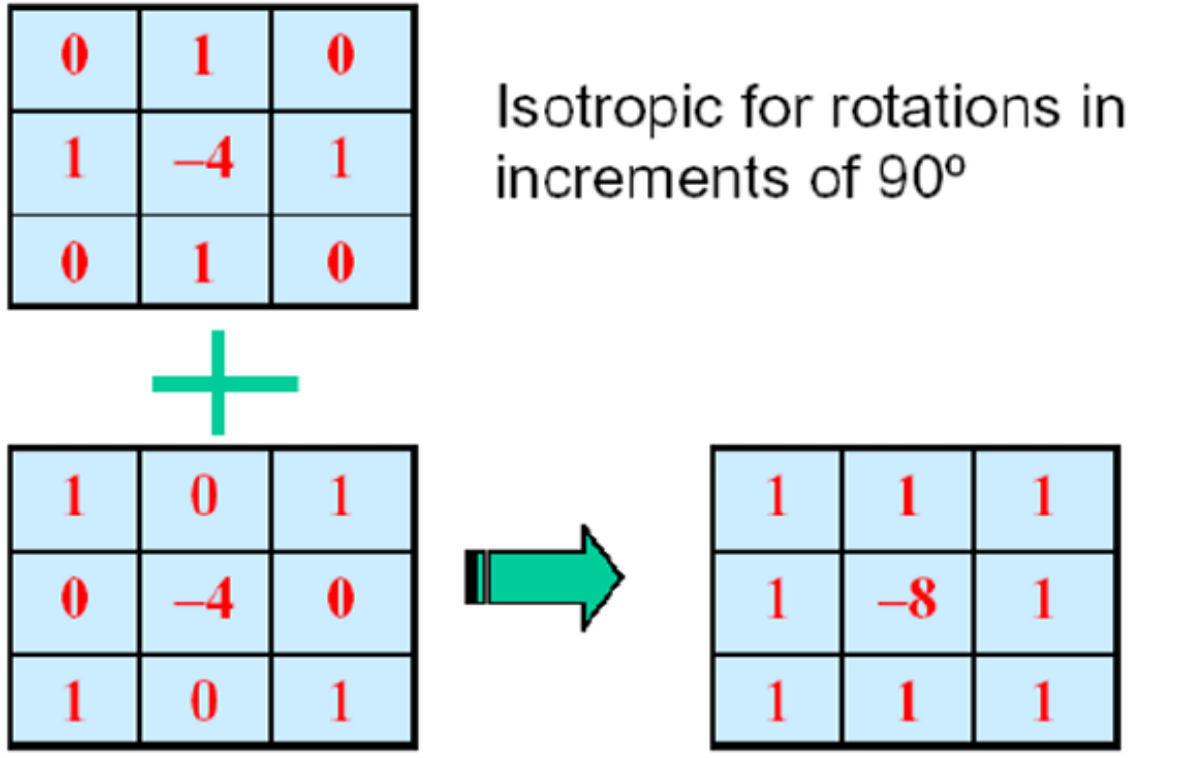
0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

−1	−1	−1
−1	8	−1
−1	−1	−1

Variant of Laplacian



Isotropic for rotations in increments of 90°

Isotropic for rotations in increments of 45°

The Laplacian Filter

- Background Features can be recovered while preserving the sharpening of the image
- Add the Laplacian image to the original image

$$g(x,y) = f(x,y) + c[\nabla^2 f(x,y)]$$

Sharpened Image

Input Image

- If the utilized Laplacian filter has **negative center coefficient**, then use **subtraction** instead of addition.
- c = +1 if the center coefficient of Laplacian Filter is positive
- c = -1 if the center coefficient of Laplacian Filter is negative

The Laplacian Filters

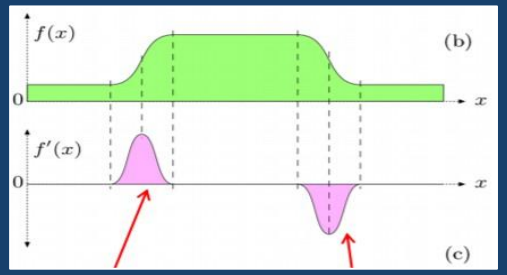
- Positive Laplacian Operator
  - Detects outward edges in an image

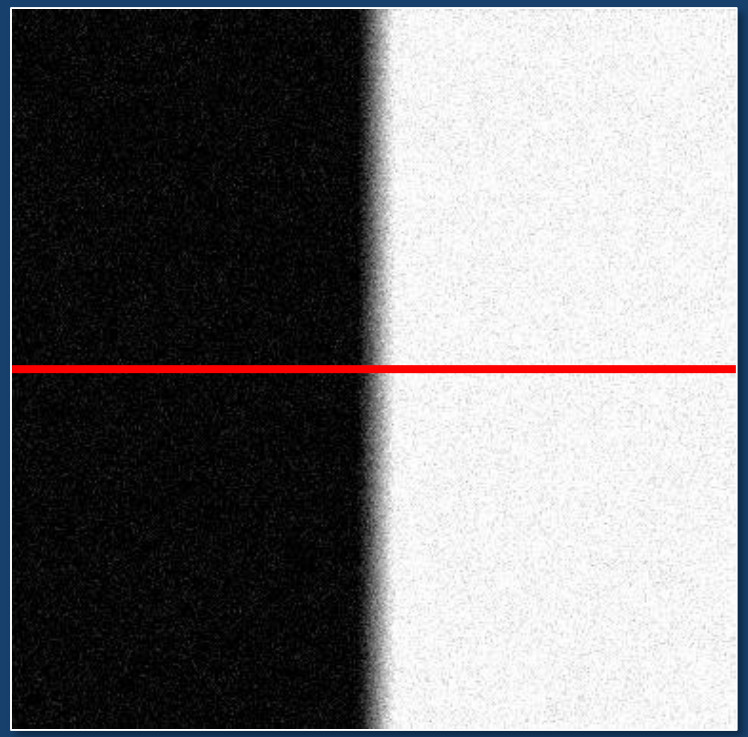
0	1	0
1	−4	1
0	1	0

- Negative Laplacian Operator
  - Detects inward edges in an image

0	−1	0
−1	4	−1
0	−1	0

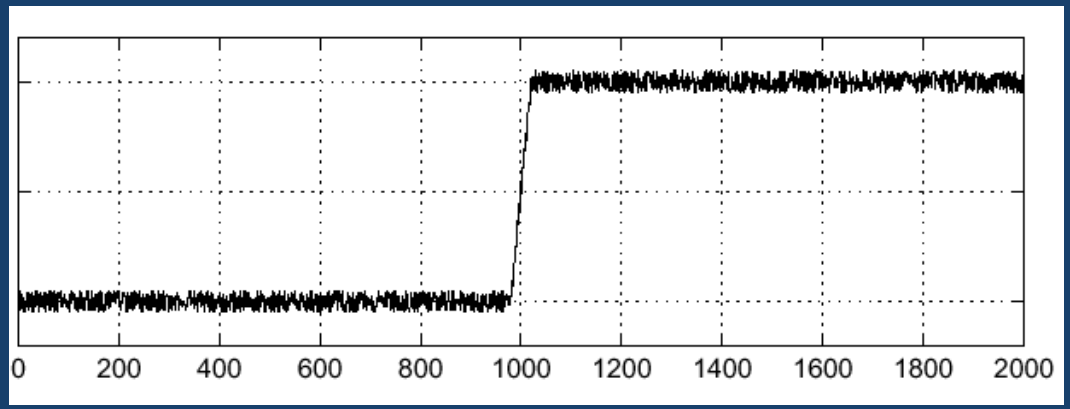
EFFECTS OF NOISE



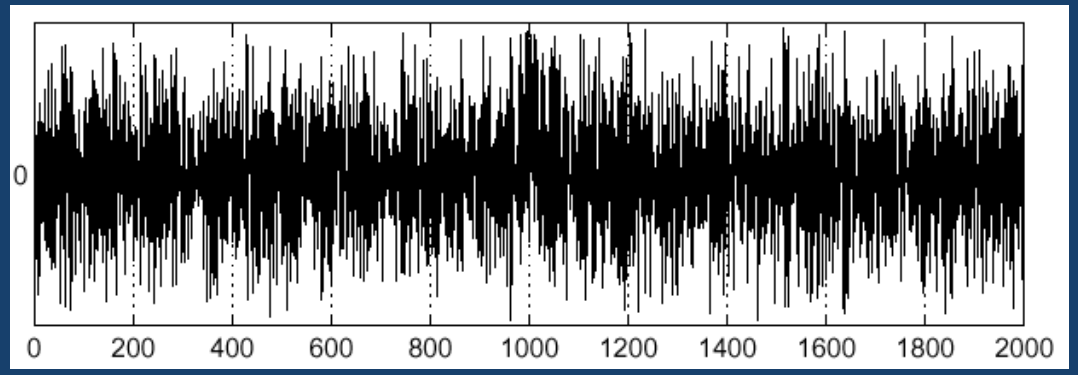


Noisy input image

$f(x)$



$\frac{d}{dx}f(x)$



Where is the edge?

Source: S. Seitz

EFFECTS OF NOISE

- Finite 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 is to be done?

Source: D. Forsyth

EFFECTS OF NOISE

- Finite 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 is to be done?
  - *Smoothing the image should help, by forcing pixels different to their neighbors (=noise pixels?) to look more like neighbors*

Source: D. Forsyth

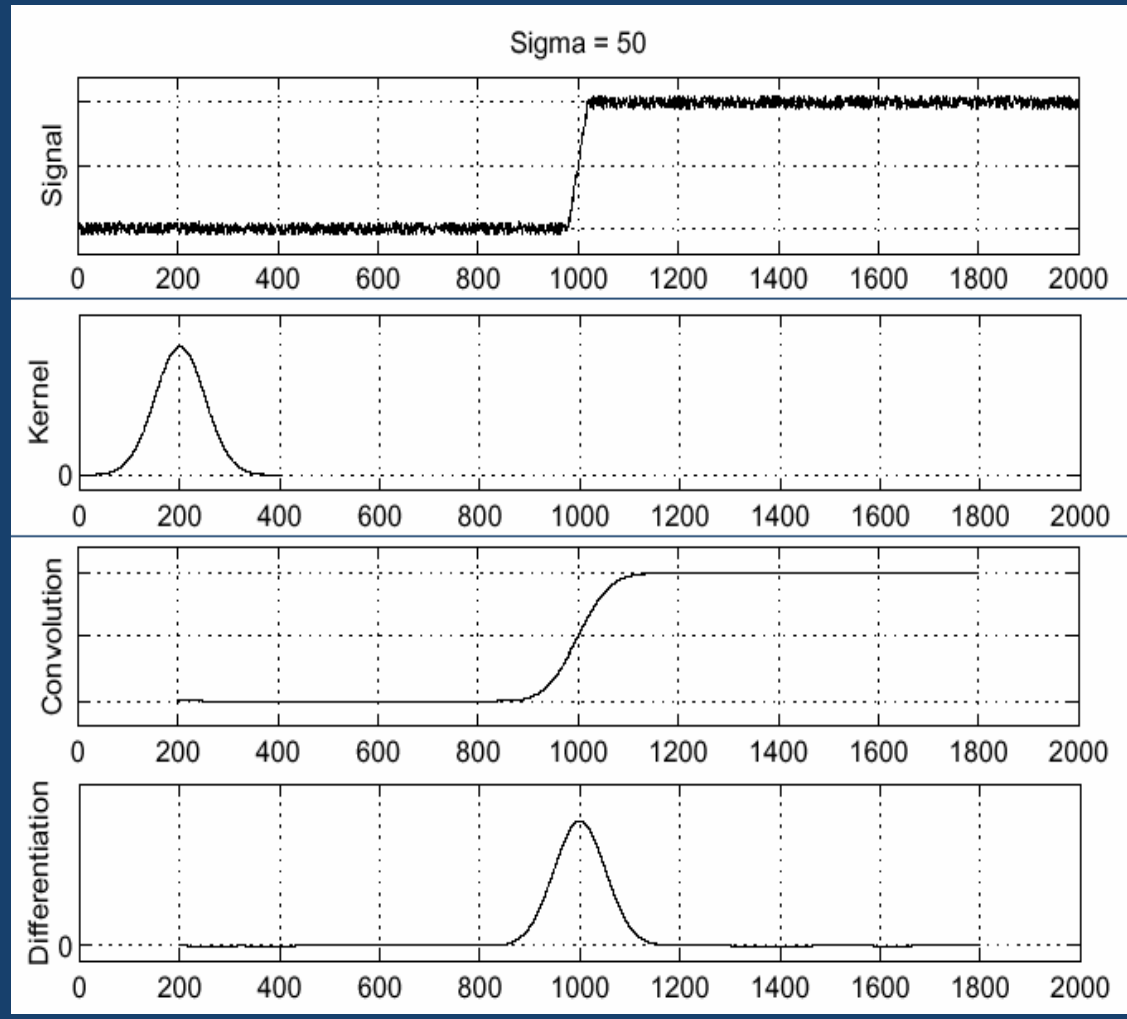
SOLUTION: SMOOTH FIRST

f

g

$f * g$

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



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

Source: S. Seitz

DERIVATIVE THEOREM OF CONVOLUTION

- Differentiation is convolution, and convolution is associative:

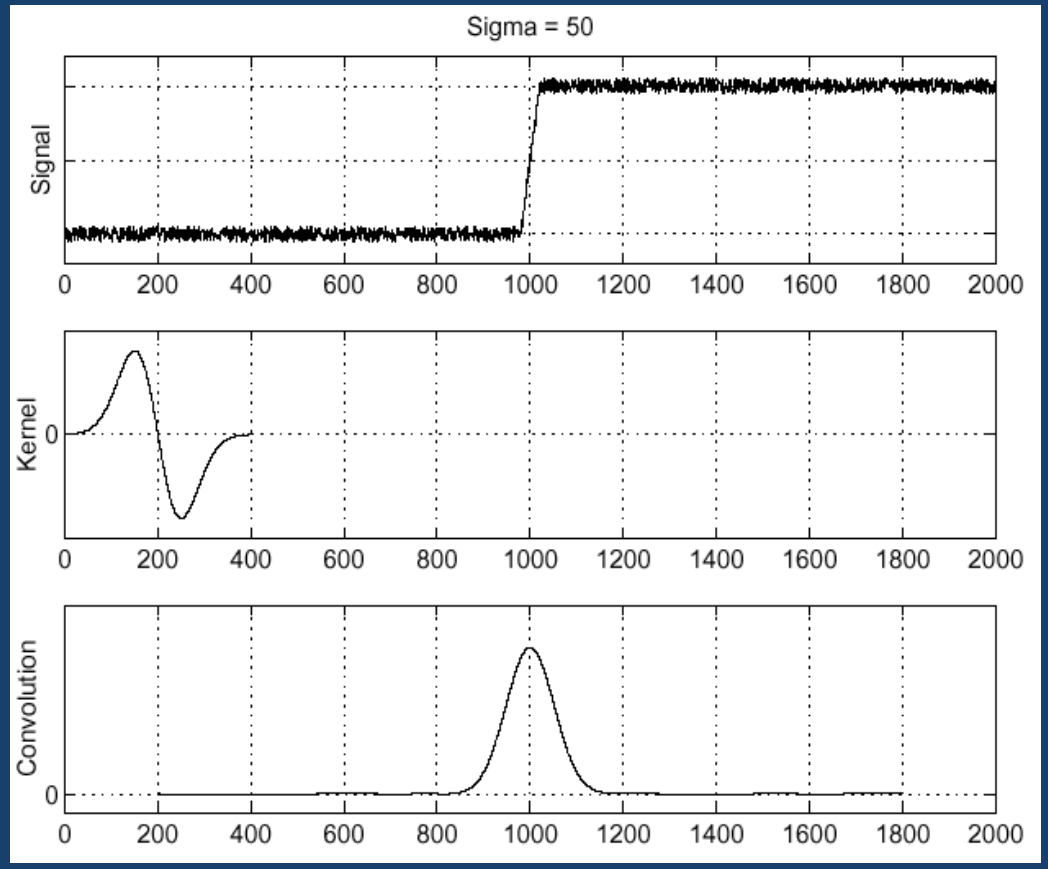
$$\frac{d}{dx}(f * g) = f * \frac{d}{dx} g$$

- This saves us one operation:

f

$\frac{d}{dx} g$

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



Source: S. Seitz

EDGE OPERATORS

- Approximating local gradients in image is the basis of many classical edge-detection operators
- Main differences?
  - *Type of filter used to estimate gradient components*
  - *How gradient components are combined*

SOBEL OPERATOR

- Uses two 3×3 kernels which are convolved with the original image to calculate approximations of the derivatives
- One for horizontal changes, and one for vertical

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} \qquad \mathbf{G}_y = \begin{bmatrix} +1 & +2 & +1 \\ 0 & 0 & 0 \\ -1 & -2 & -1 \end{bmatrix}$$

SOBEL OPERATION

- Smoothing + differentiation

$$\mathbf{G}_x = \begin{bmatrix} +1 & 0 & -1 \\ +2 & 0 & -2 \\ +1 & 0 & -1 \end{bmatrix} = \begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix} \begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$$

Gaussian smoothing

differentiation

This means the Sobel kernel can be expressed as the **outer product** of two smaller 1D filters:

Vertical smoothing filter: $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$

Horizontal difference filter: $\begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$

Instead of convolving the image with the full 3×3 kernel at once, you can convolve it in **two steps**:

1. Smooth vertically with
2. Take horizontal differences with

This reduces computation cost and makes the operation more efficient.

Sobel Kernel

The matrix shown is the Sobel operator in the x-direction.
It detects vertical edges by first smoothing in the vertical direction $\begin{bmatrix} 1 \\ 2 \\ 1 \end{bmatrix}$

and then taking differences in the horizontal direction $\begin{bmatrix} +1 & 0 & -1 \end{bmatrix}$

That’s why the 3×3 kernel can be written as the product of two 1D filters.

SOBEL OPERATION

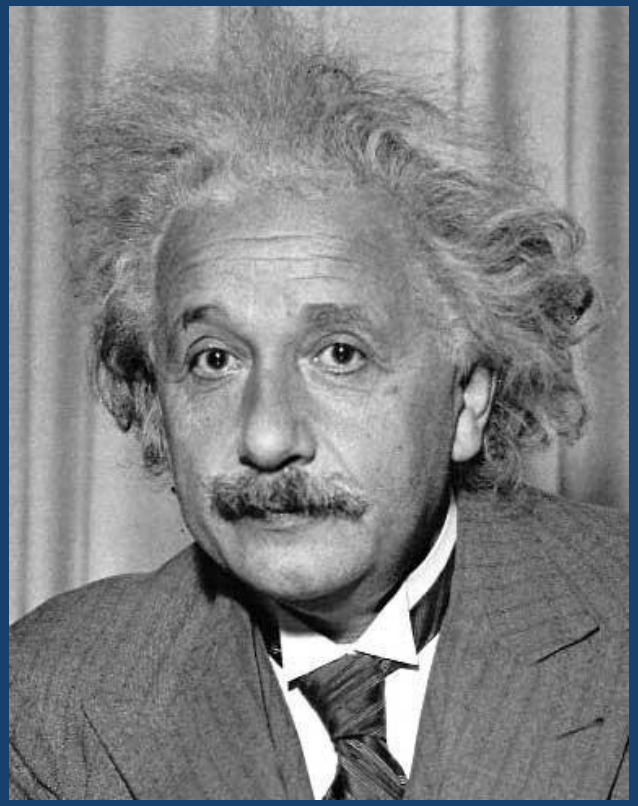
- Magnitude:

$$\mathbf{G} = \sqrt{\mathbf{G}_x{}^2 + \mathbf{G}_y{}^2}$$

- Angle or direction of the gradient:

$$\Theta = \operatorname{atan}\left(\frac{\mathbf{G}_y}{\mathbf{G}_x}\right)$$

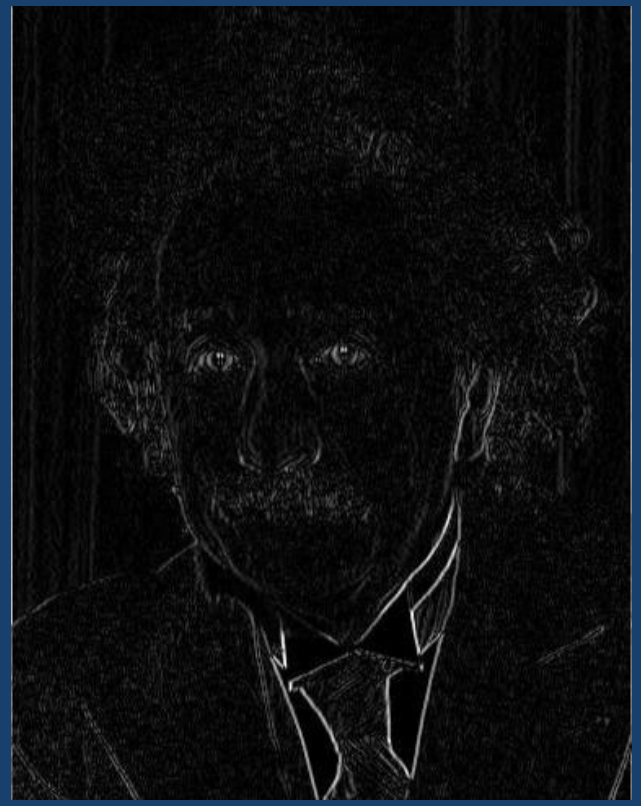
SOBEL OPERATION EXAMPLE



G_x

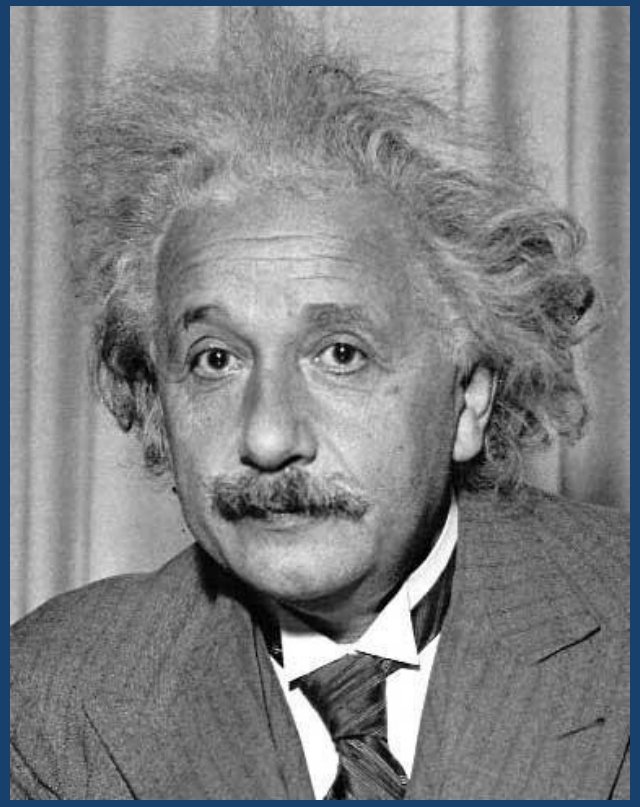
1	0	-1
2	0	-2
1	0	-1

Sobel



Vertical Edge
(absolute value)

SOBEL OPERATION EXAMPLE



Gy

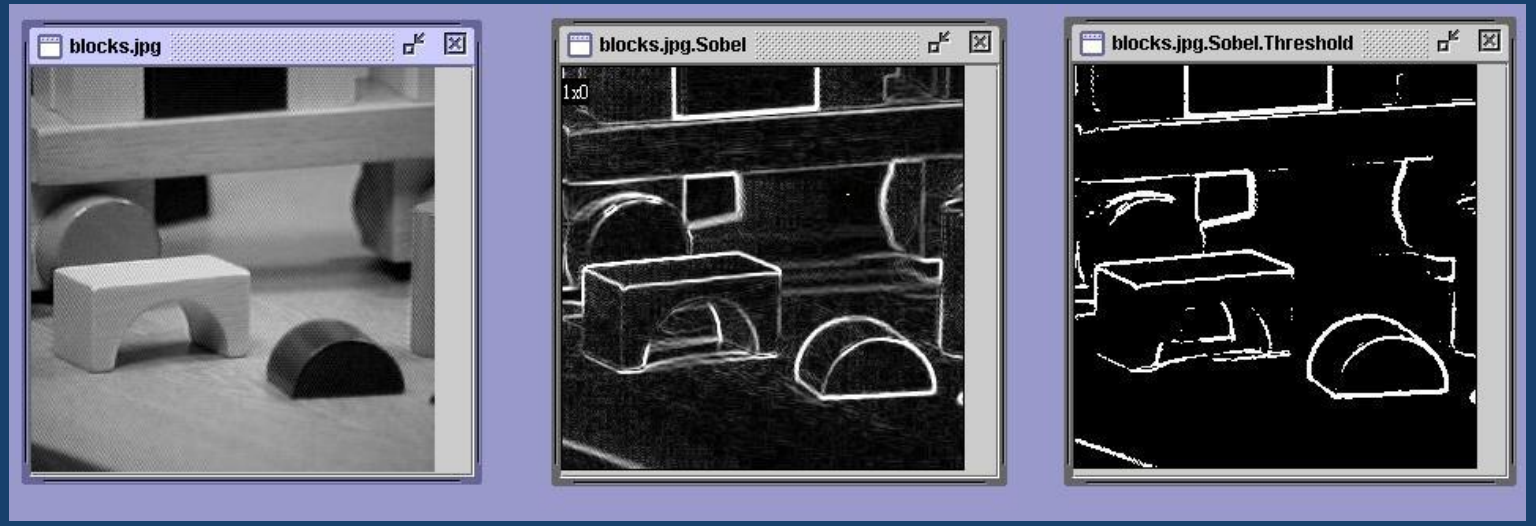
1	2	1
0	0	0
-1	-2	-1

Sobel

Horizontal Edge
(absolute value)

SOBEL OPERATION EXAMPLE

- We can threshold the gradient image, and choose edges to be the pixels above T (threshold value).

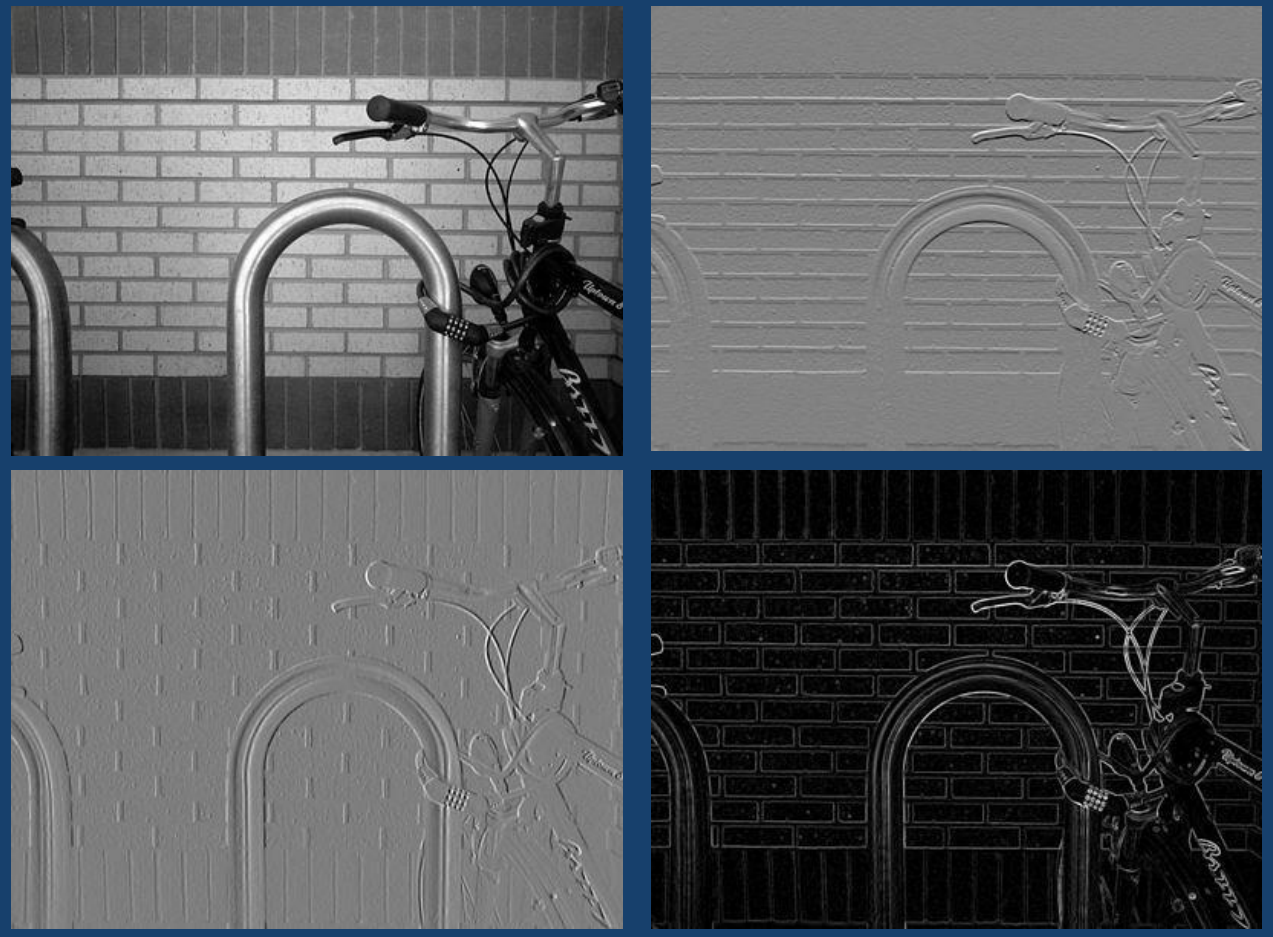


original image

gradient magnitude

thresholded gradient magnitude

SOBEL OPERATION EXAMPLE



Thank you