ISM 204: Introduction to Digital Image Processing

Week 9: Edge Detection: Gradient Operators - Sobel and Prewitt Operator

Christian Sada
(csada@pau.edu.ng)

SCHOOL OF MEDIA AND COMMUNICATION
PAN-ATLANTIC UNIVERSITY

Learning Outlines

1 Introduction to Edge Detection Techniques

2 Exploring Gradient Operators for Edge Detection

3 Understanding Sobel and Prewitt Operators and Their Role in Edge Detection

4 Practical: Implementing Gradient Operators for Edge Detection

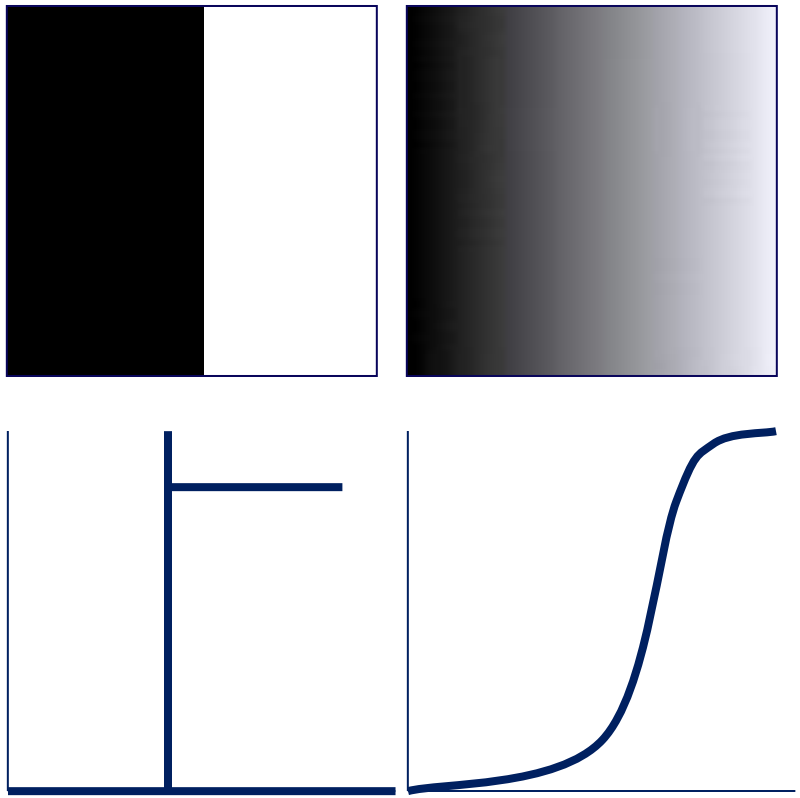
What are edges in an image?

Edges are fundamental features in images that represent boundaries or transitions between different regions of intensity or color.

They typically correspond to significant changes in pixel values, signifying the presence of **object boundaries**, **texture boundaries**, or **surface discontinuities**. Understanding and detecting edges are crucial tasks in image processing and computer vision, as edges often encode important visual information about objects and scenes.

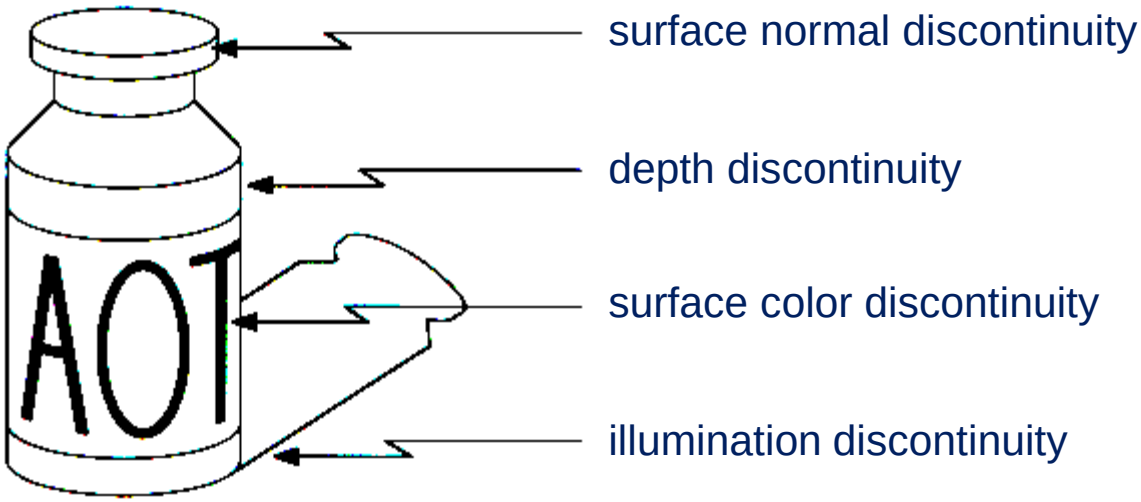
What are edges in an image?

- Edges are those places in an image that correspond to object boundaries.
- Edges are pixels where image brightness changes abruptly.



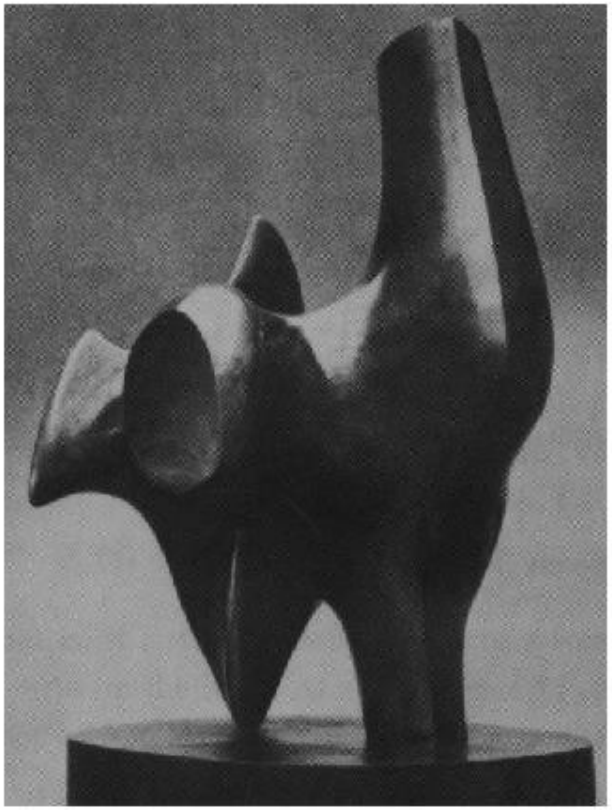
Brightness vs. Spatial Coordinates

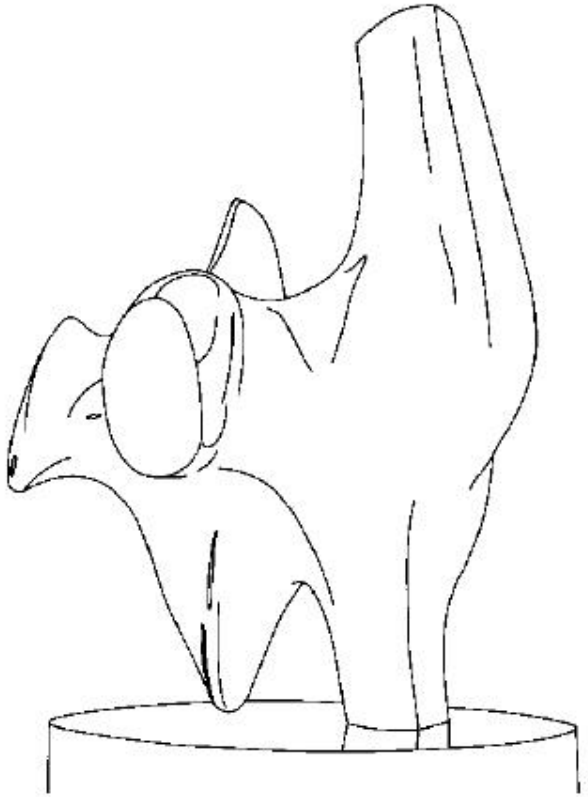
Origin of Edges



- Edges are caused by a variety of factors

How can you tell that a pixel is on an edge?





Convert a 2D image into a set of curves

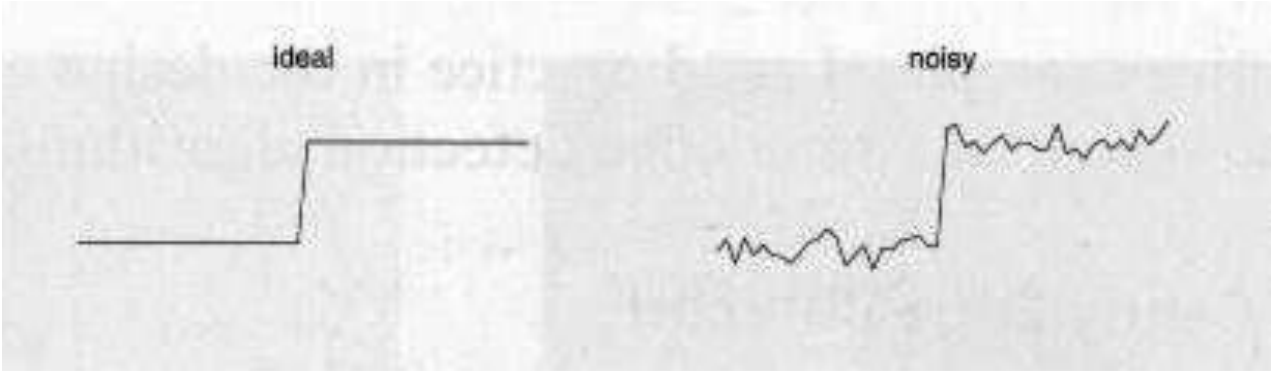
- Extracts features of the scene
- More compact than pixels

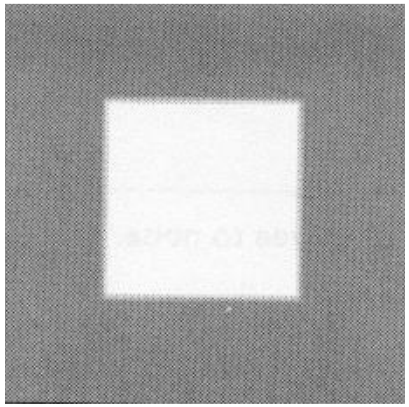
Types of Edges

Step Edges:

- Step edges are characterized by abrupt changes in intensity, where the pixel values transition sharply from one level to another. Also known as abrupt edges or sudden edges.

- The boundary between a dark object and a light background in an image is often represented by a step edge.

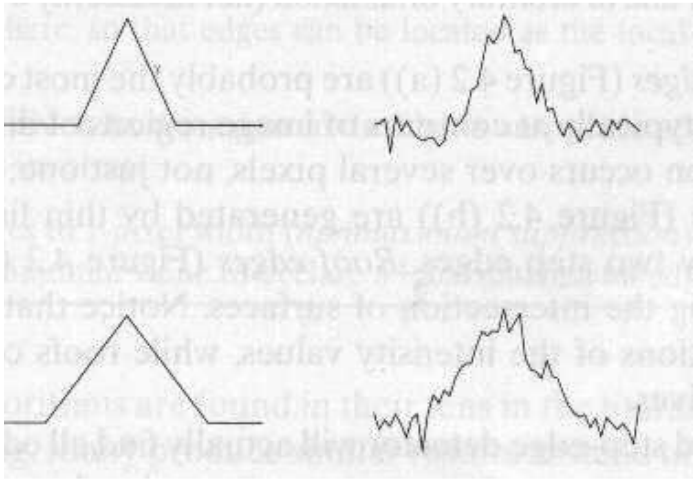




Types of Edges

Roof Edges:

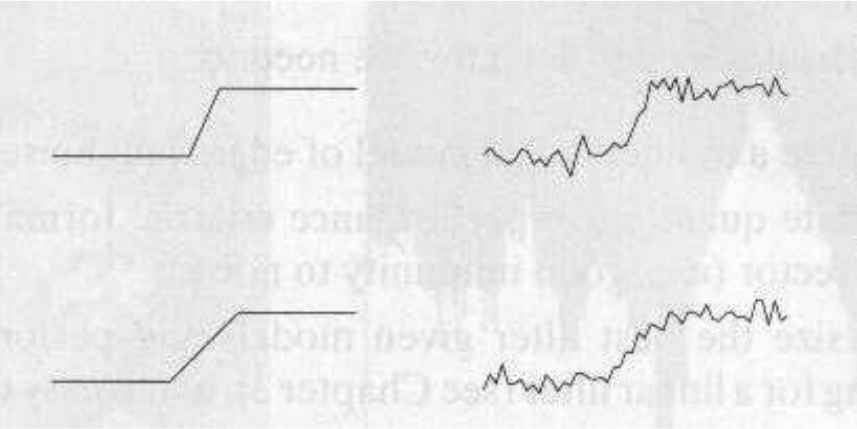
- Roof edges are a type of edge where the intensity changes along a line or curve, resembling the shape of a roof.
- Unlike step edges, roof edges exhibit a gradual transition in intensity, resembling a sloped roofline rather than a sharp change.
- These edges are often encountered in images containing curved surfaces or objects with smooth gradients in intensity.

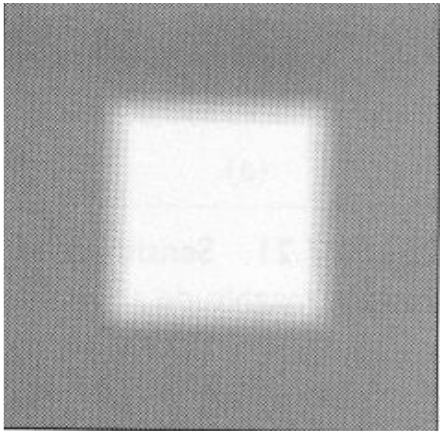


Types of Edges

Ramp Edges:

- Ramp edges exhibit gradual changes in intensity, where the pixel values transition smoothly from one level to another over a certain distance. Also referred to as gradual edges or smooth edges.
- A step edge where the intensity change is not instantaneous but occur over a finite distance.



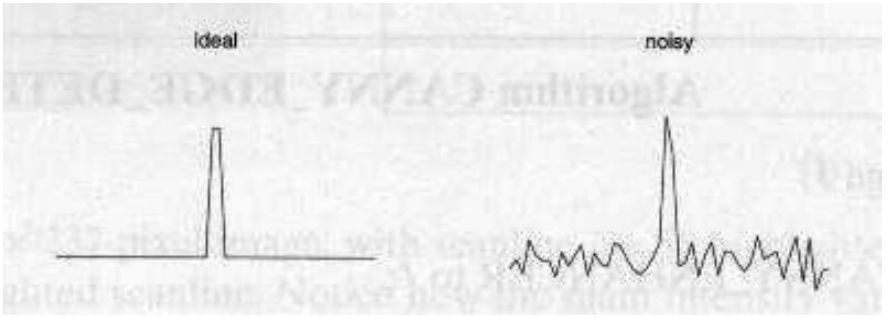


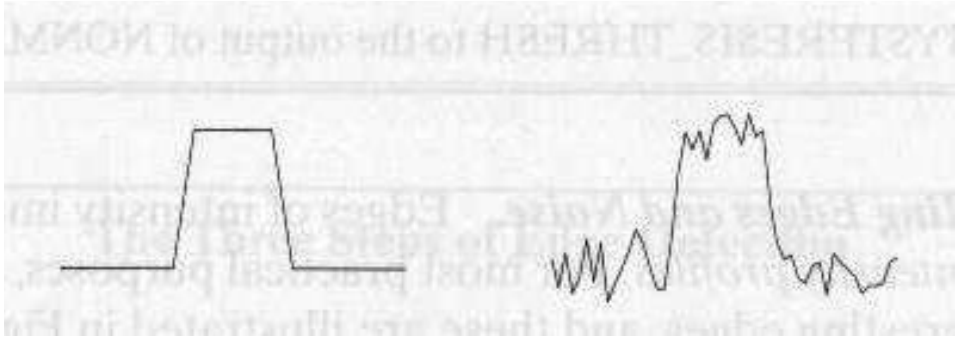
Types of Edges

Line Edges:

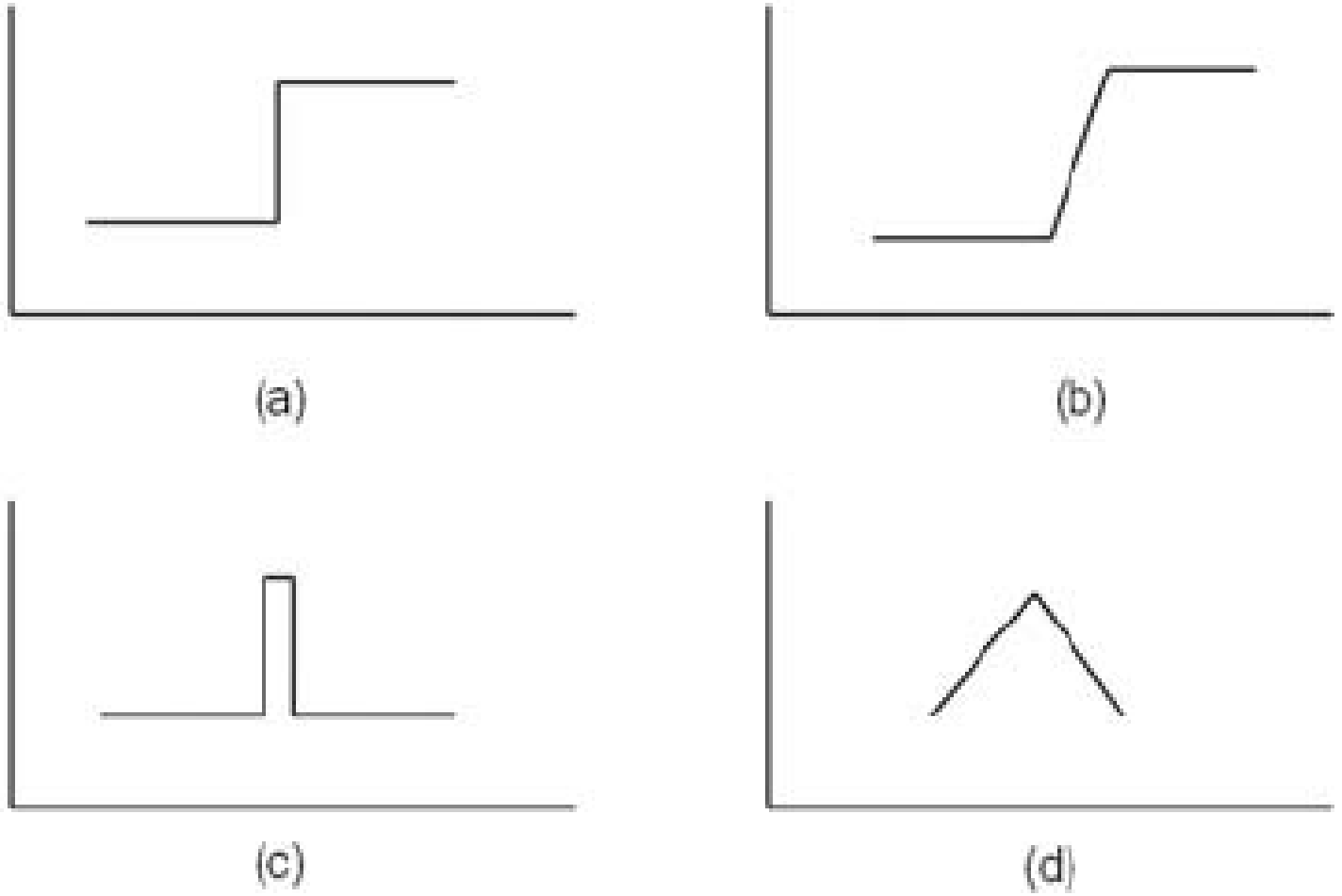
- Line edges are formed by the alignment of pixels along a linear feature, such as an edge of a building, a road, or a straight object in the image. Sometimes known as linear edges or contour edges.

- The boundary between the roof and the sky in an urban skyline image often forms a line edge.



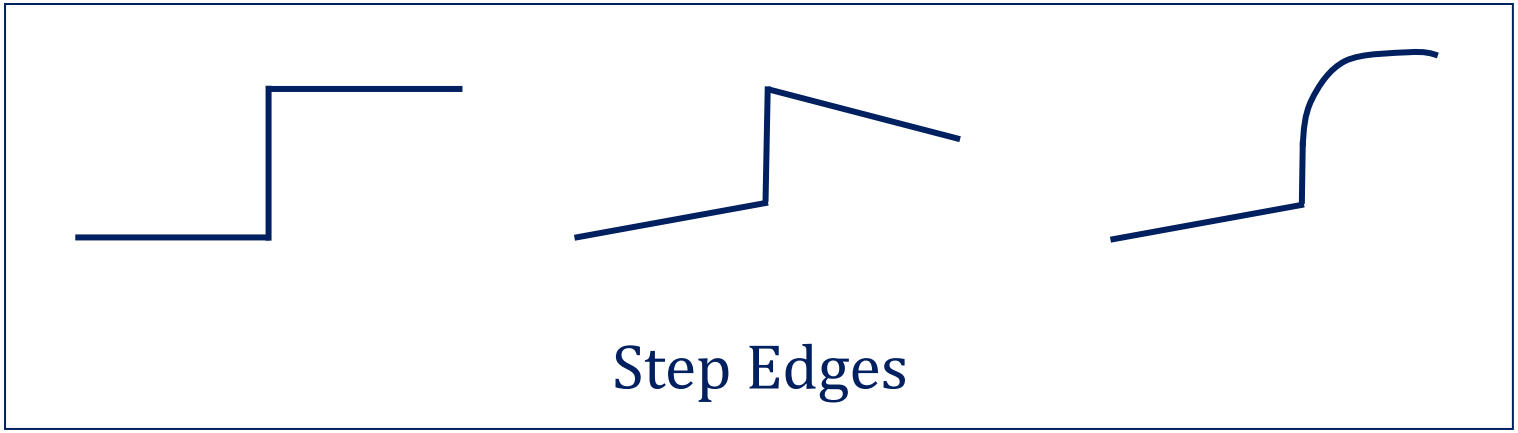


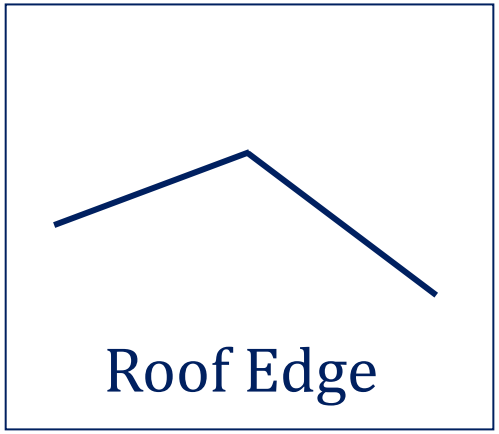
Edge Types

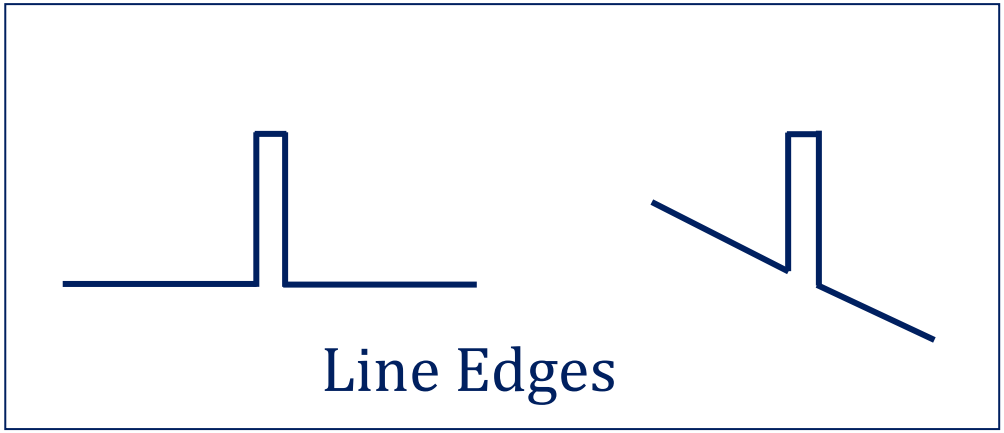


Type of Edges (a) Step Edge (b) Ramp Edge (c) Line Edge (d) Roof Edge

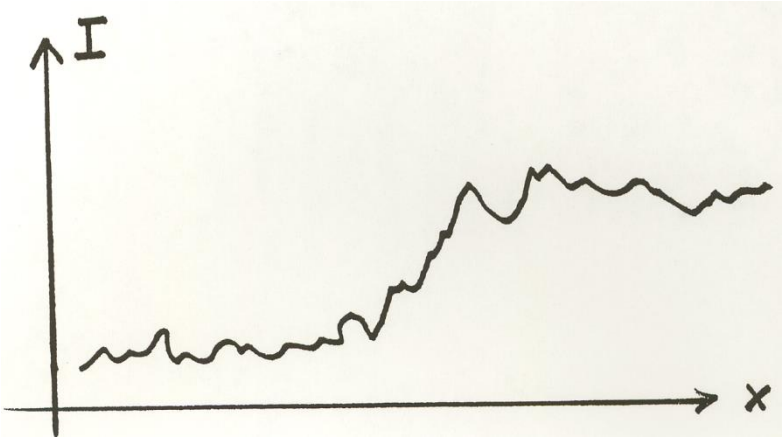
Edge Types







Real Edges

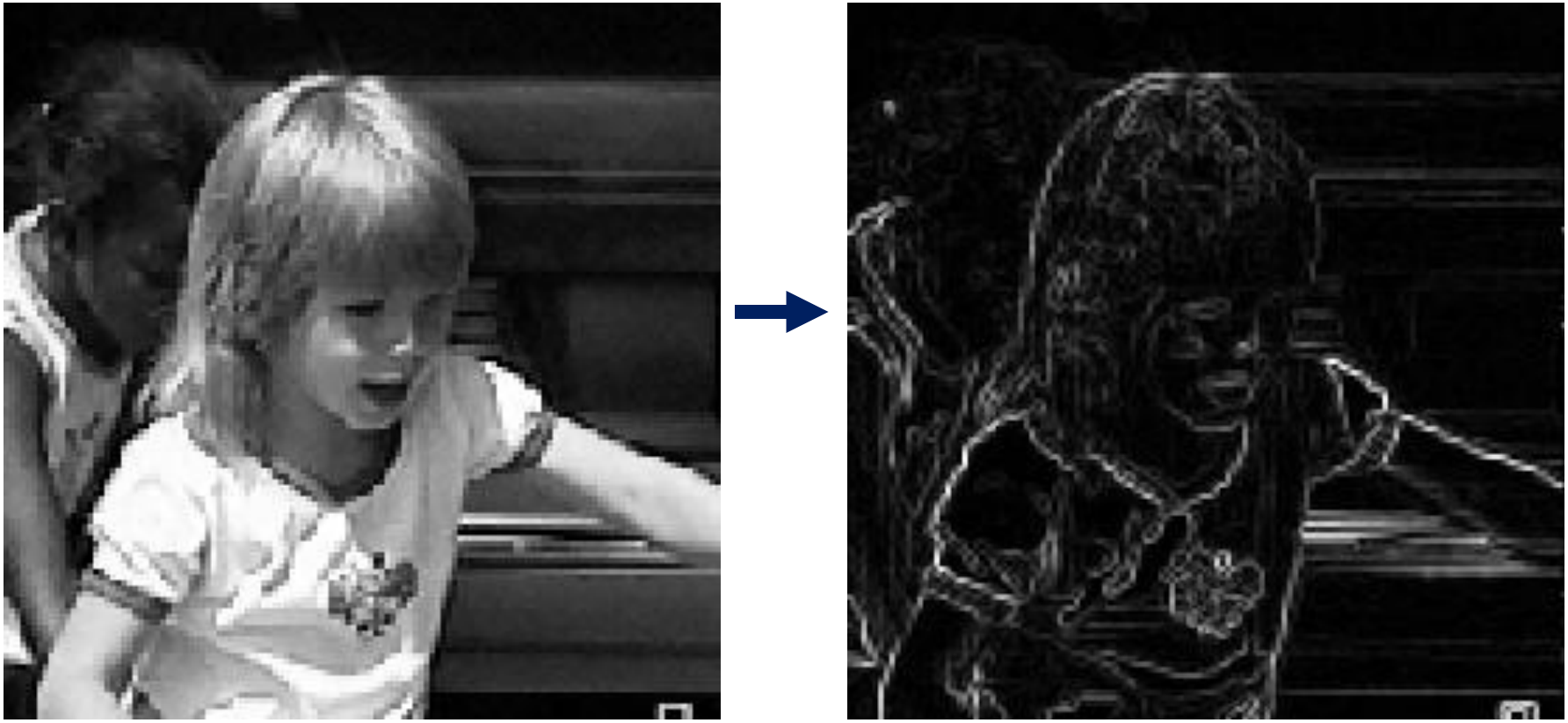


Noisy and Discrete!

We want an **Edge Operator** that produces:

- Edge Magnitude
- Edge Orientation
- High Detection Rate and Good Localization

Image To Edge Map

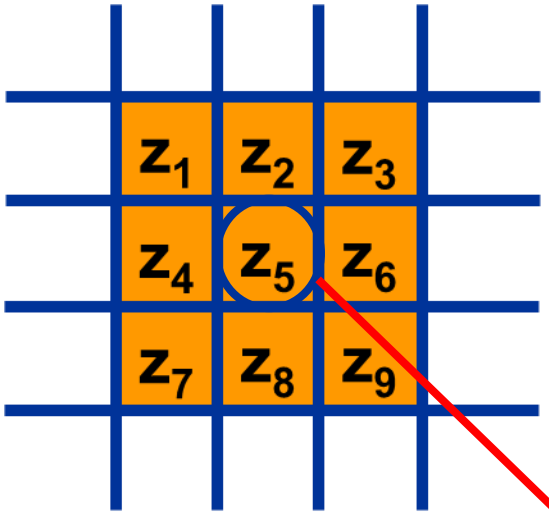


Edge Detection

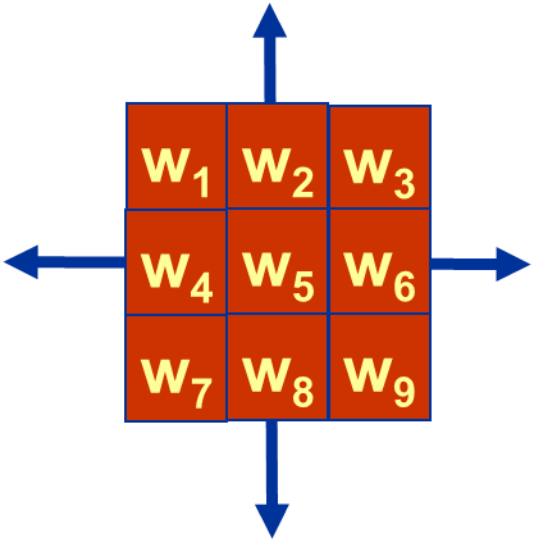
- Edge information in an image is found by looking at the relationship a pixel has with its neighborhoods.
- If a pixel's gray-level value is similar to those around it, there is probably not an edge at that point.
- If a pixel's has neighbors with widely varying gray levels, it may present an edge point.

Spatial Filtering (Masking)

Portion of
a digital image



Mask



Replace
with

$R = w_1z_1 + w_2z_2 + \ldots\ldots + w_9z_9$

https://www.youtube.com/watch?v=-pmUQ6RSejQ

Edge Detection

12	90	89	86	87	82
10	12	88	85	83	84
9	15	12	84	84	88
12	14	10	82	88	89
11	17	16	12	88	90
10	16	15	17	89	88

Edge Detection

Vertical edge detection

10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0
10	10	10	0	0	0

6x6

*

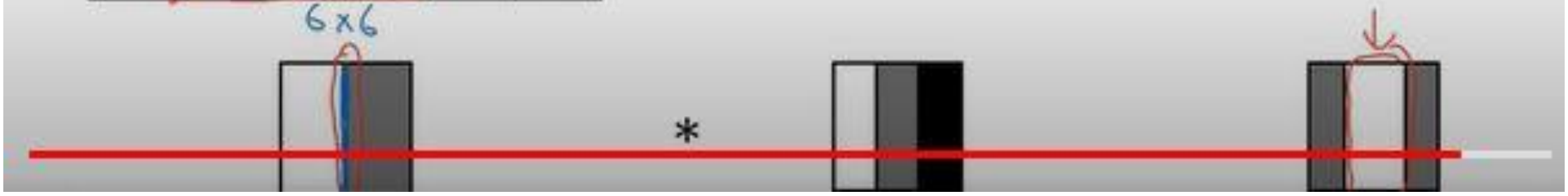
1	0	-1
1	0	-1
1	0	-1

3x3

=

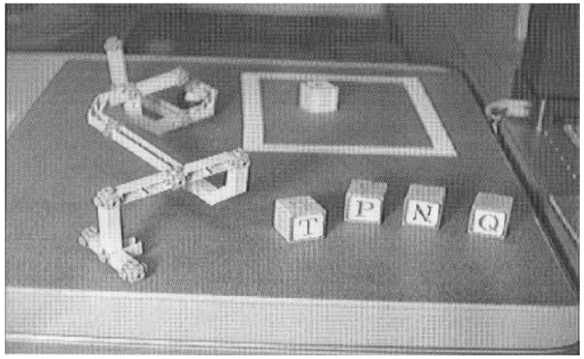
0	30	30	0
0	30	30	0
0	30	30	0
0	30	30	0

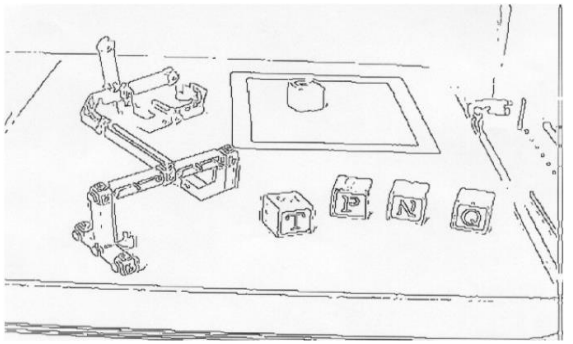
4x4



Advantages Edge Detection

- Produce a line drawing of a scene from an image of that scene.
- Important features can be extracted from the edges of an image (e.g., corners, lines, curves).
- These features are used by higher-level computer vision algorithms (e.g., segmentation, recognition).





Edge Detection Methods

- Many are implemented with convolution mask and based on discrete approximations to differential operators.
- Differential operations measure the rate of change in the image brightness function.
- Some operators return orientation information. Others only return information about the existence of an edge at each point.

Edge detectors

- locate sharp changes in the intensity function
- edges are pixels where brightness changes abruptly.

- Calculus describes changes of continuous functions using derivatives; an image function depends on two variables - partial derivatives.
- A change of the image function can be described by a gradient that points in the direction of the largest growth of the image function.
- An edge is a property attached to an individual pixel and is calculated from the image function behavior in a neighborhood of the pixel.
- It is a **vector variable**:
- **magnitude** of the gradient and **direction**

- The gradient direction gives the direction of maximal growth of the function, e.g., from black (f(i,j)=0) to white (f(i,j)=255).
- This is illustrated below; closed lines are lines of the same brightness.
- Boundary and its parts (edges) are perpendicular to the direction of the gradient.

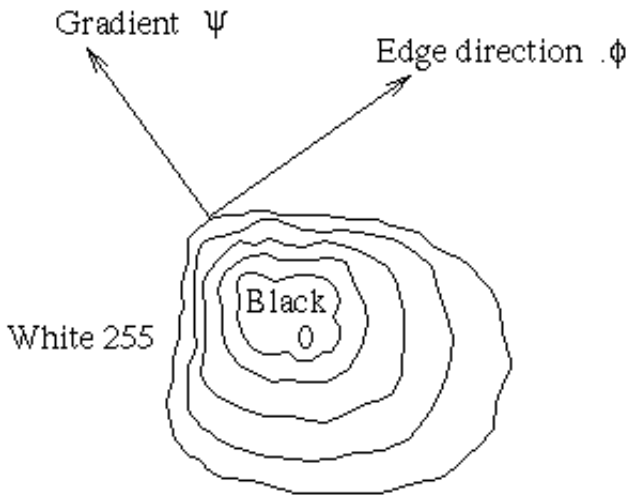


Figure 4.16 Gradient direction and edge direction.

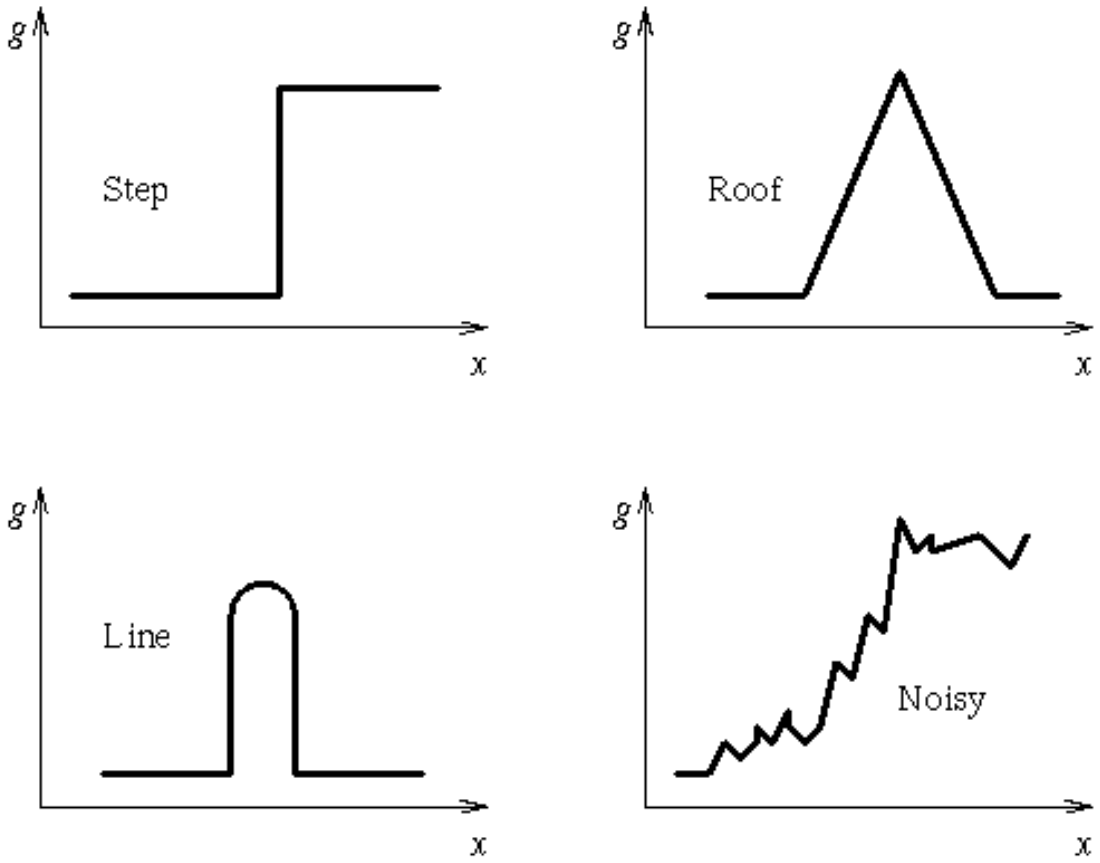


Figure 4.17 Typical edge profiles.

Edge Detection Methods

- Edges in images are areas with strong intensity contrasts; a jump in intensity from one pixel to the next.
- There are many different edge detection methods:
  - Sobel
  - Prewitt
  - Laplacian.

https://www.youtube.com/watch?v=VL8PuOPjVjY

Prewitt Operator

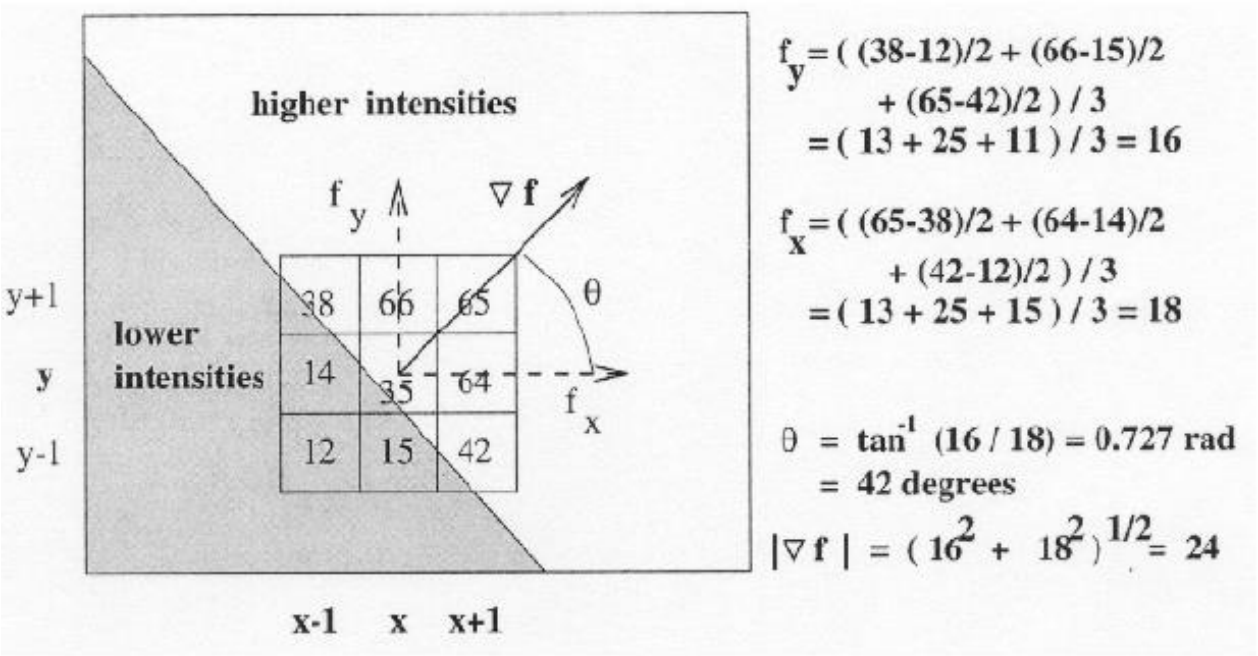
- Looks for edges in both horizontal and vertical directions, then combine the information into a single metric.

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

- Edge Magnitude = $\sqrt{x^2 + y^2}$
- Edge Direction = $\tan^{-1}\left[\frac{y}{x}\right]$

Example (using Prewitt operator)

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



$f_y = ((38-12)/2 + (66-15)/2 + (65-42)/2) / 3$
$= (13 + 25 + 11) / 3 = 16$

$f_x = ((65-38)/2 + (64-14)/2 + (42-12)/2) / 3$
$= (13 + 25 + 15) / 3 = 18$

$\theta = \tan^{-1} (16 / 18) = 0.727$ rad
$= 42$ degrees

$|\nabla f| = (16^2 + 18^2)^{1/2} = 24$

Note: in this example, the divisions by 2 and 3 in the computation of f_x and f_y are done for normalization purposes only

Sobel Operator

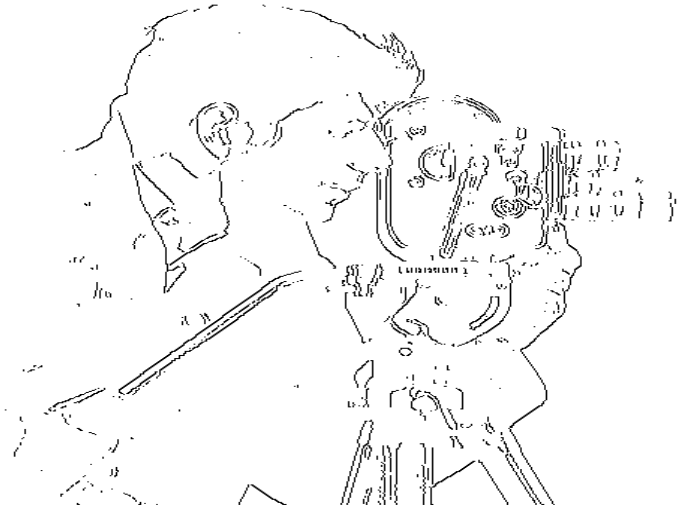
- Similar to the Prewitt, with different mask coefficients:

$$y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \qquad x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix}$$

- Edge Magnitude = $\sqrt{x^2 + y^2}$
- Edge Direction = $\tan^{-1}\left[\frac{y}{x}\right]$

Directional Edge Detection

- (Mx) finds vertical edges
- (My) horizontal edges



(Mx)

(My)

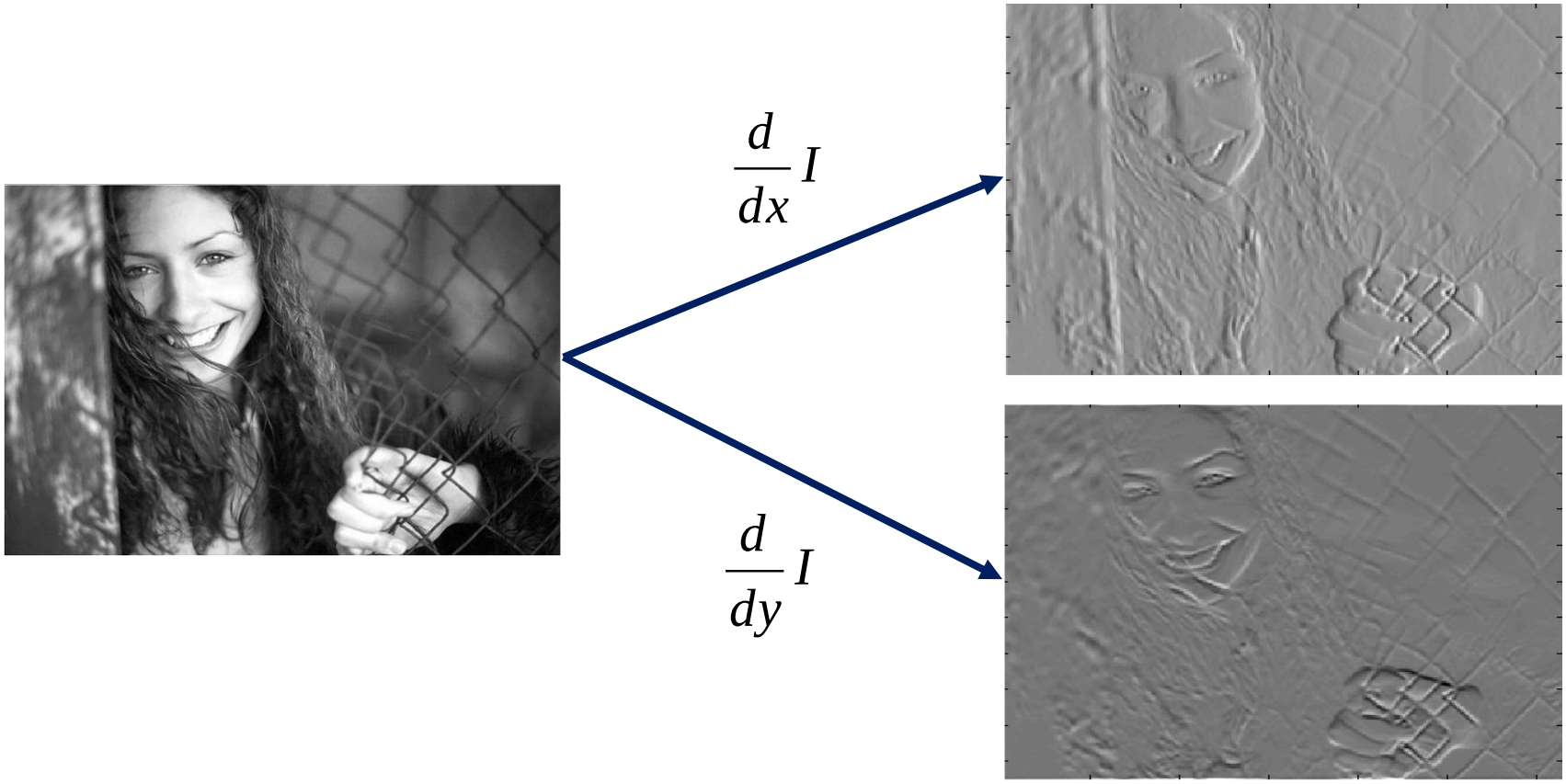
https://www.youtube.com/watch?v=uihBwtPIBxM

Prewitt and Sobel Operators

- We shall experiment with the Prewitt, Sobel and Laplacian edge detectors, in class, using the link below.

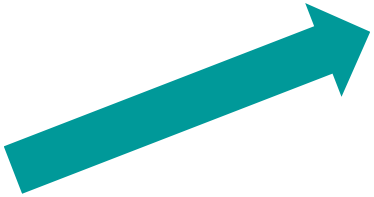
- http://setosa.io/ev/image-kernels/

Another Example



Another Example (cont'd)

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

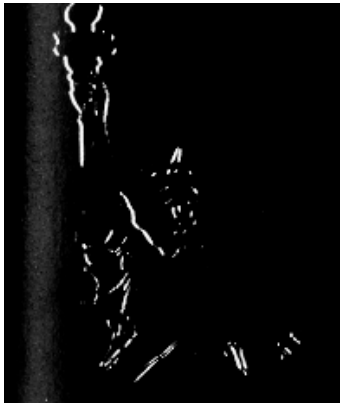


$$\nabla \geq Threshold = 100$$

Isotropic property of gradient magnitude

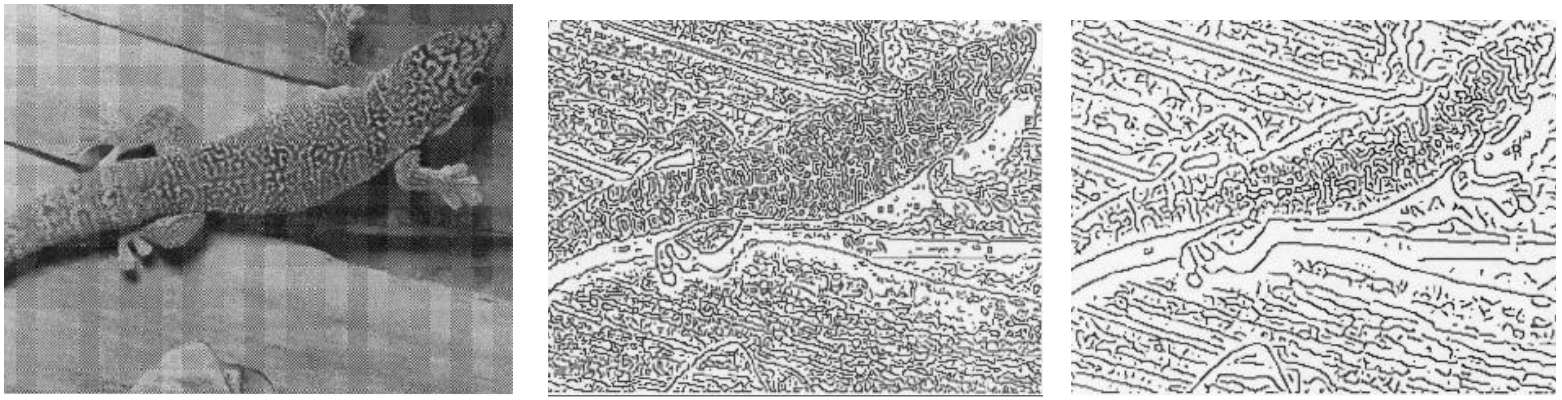
- The magnitude of the gradient detects edges in all directions.

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

Practical Issues

- Noise suppression-localization tradeoff.
- Smoothing depends on mask size (e.g., depends on σ for Gaussian filters).
- Larger mask sizes reduce noise, but worsen localization (i.e., add uncertainty to the location of the edge) and vice versa.



Assignment

- Using your desired Image apply the Sobel and Prewitt Operators for Edge Detection.

Practical Session: Applying Prewitt Operators

Input Image

Read the color image:

```
color_img = imread('lion_cub.jpg');
```

Convert the color image to grayscale:

```
gray_img = rgb2gray(color_img);
```

Define the Prewitt operators:

horizontal_prewitt = [-1, 0, 1; -1, 0, 1; -1, 0, 1];
vertical_prewitt = [-1, -1, -1; 0, 0, 0; 1, 1, 1];

Apply the horizontal Prewitt operator:

```
horizontal_edges = conv2(double(gray_img), horizontal_prewitt, 'same');
```

This line applies the horizontal Prewitt operator to the grayscale image gray_img using the conv2() function with 'same' option to keep the output size the same as the input size. The result is stored in the variable horizontal_edges.

Apply the vertical Prewitt operator:

```
vertical_edges = conv2(double(gray_img),
vertical_prewitt, 'same');
```

Similar to the previous step, this line applies the vertical Prewitt operator to the grayscale image gray_img and stores the result in the variable vertical_edges.

Compute the magnitude of the edges:

```
edge_magnitude = sqrt(horizontal_edges.^2
+ vertical_edges.^2);
```

This line computes the magnitude of the edges by taking the square root of the sum of squares of horizontal and vertical edges. The result is stored in the variable edge_magnitude.

Display the grayscale image and its edges:

```
subplot(2, 2, 1);
imshow(gray_img);
title('Grayscale Image');

subplot(2, 2, 2);
imshow(uint8(horizontal_edges));
title('Horizontal Edges');
```

Display the grayscale image and its edges:

```
subplot(2, 2, 3);
imshow(uint8(vertical_edges));
title('Vertical Edges');

subplot(2, 2, 4);
imshow(uint8(edge_magnitude));
title('Edges using Prewitt Operator');
```

Practical Session: Applying Sobel Operators

Input Image

Read the color image:

```
color_img = imread('lion_cub.jpg');
```

Convert the color image to grayscale:

```
gray_img = rgb2gray(color_img);
```

Define the Sobel operators:

horizontal_sobel = [-1, 0, 1; -2, 0, 2; -1, 0, 1];
vertical_sobel = [-1, -2, -1; 0, 0, 0; 1, 2, 1];

Apply the horizontal Sobel operator:

```
horizontal_edges_sobel = conv2(double(gray_img), horizontal_sobel, 'same');
```

This line applies the horizontal Sobel operator to the grayscale image gray_img using the conv2() function with 'same' option to keep the output size the same as the input size. The result is stored in the variable horizontal_edges_sobel.

Apply the vertical Sobel operator:

```
vertical_edges_sobel = conv2(double(gray_img), vertical_sobel, 'same');
```

Similar to the previous step, this line applies the vertical Sobel operator to the grayscale image gray_img and stores the result in the variable vertical_edges_sobel.

Compute the magnitude of the edges:

```
edge_magnitude_sobel                    =
sqrt(horizontal_edges_sobel.^2          +
vertical_edges_sobel.^2);
```

This line computes the magnitude of the edges by taking the square root of the sum of squares of horizontal and vertical edges. The result is stored in the variable edge_magnitude_sobel.

Display the grayscale image and its edges:

```
subplot(2, 2, 1);
imshow(gray_img);
title('Grayscale Image');

subplot(2, 2, 2);
imshow(uint8(horizontal_edges_sobel));
title('Horizontal Edges (Sobel)');
```

Display the grayscale image and its edges:

```
subplot(2, 2, 3);
imshow(uint8(vertical_edges_sobel));
title('Vertical Edges (Sobel)');

subplot(2, 2, 4);
imshow(uint8(edge_magnitude_sobel));
title('Edges using Sobel Operator');
```