M30242 Graphics and Computer Vision

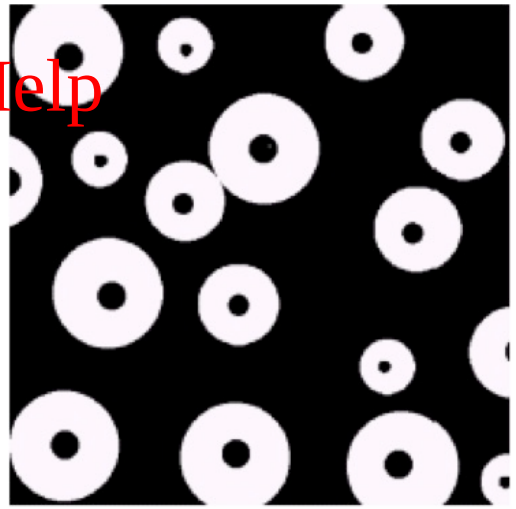
Lecture 3 Morphological Operation and Edge Detection

Overview

- Binary image operations (cont'd)
  - Binary image morphology
- Grayscale image processing - edge detection
  - Image function and edges
  - Detecting edges by image gradients (first derivatives)
    - Prewitt and Sobel detectors

Review: An Example of Binary Image Application

- Scenario:
  - You are given images of conveying belt
  - Problem: How many washers in this image?
- What to do
  - Binary images is obtained by thresholding the intensity images (e.g., using im2bw or imbinarizer)
  - Indentify washers by labelling (bwlabel)
  - Count the number of washers in the image



Limitations with Thresholding

- Real applications are often more trickier than our example:
  - Image noises;
  - Lighting variations.
  - E.g., thresholding would not remove the "pepper" noise in the fingerprint image.
- More sophisticated processing is needed to obtain good binary images.

Binary Image Morphology

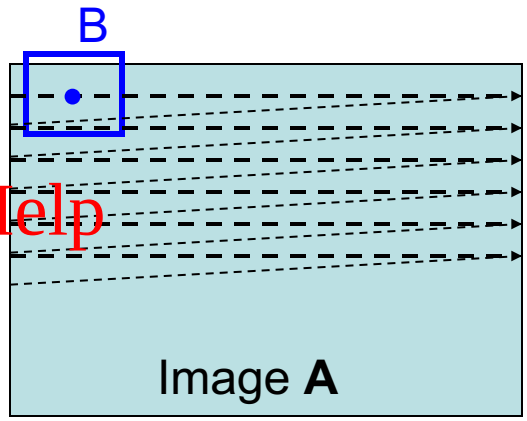
- Morphology means changing the shape of features.
- Morphological operations on ***binary*** images:
  - *dilation* and *erosion* for expanding and shrinking feature regions.
  - *opening* and *closing* for eliminating small isolated regions, protrusion or holes.

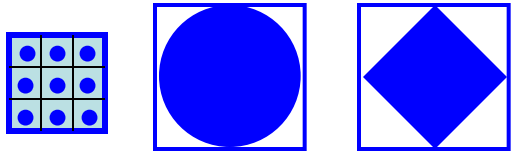
The original

After treating with erosion

Dilation

- The operation that "thickens" objects in a binary image.
- How it works:
  - Scan an image **A** with a small "window" or "mask" **B;**
  - **B** is a small **binary image** and is called a structural element. B (its foreground) can have different shapes (e.g., square, circular, diamond, etc) and sizes (e.g., 2-pixel by 2-pixel, 3x3, …);
  - While scanning, the centre of **B** goes through ***all*** pixels of image **A (**pixel-by-pixel, from left to right and from top to bottom**)**
- Scanning an image with a small mask is an idea common to many image processing operations.

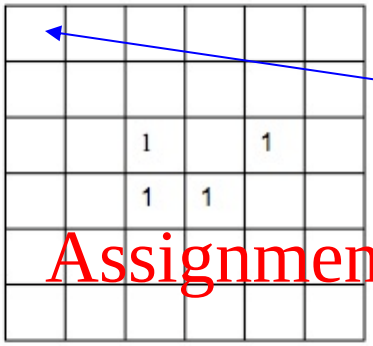


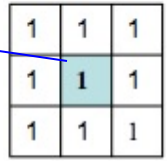


Shapes and sizes of B

Dilation

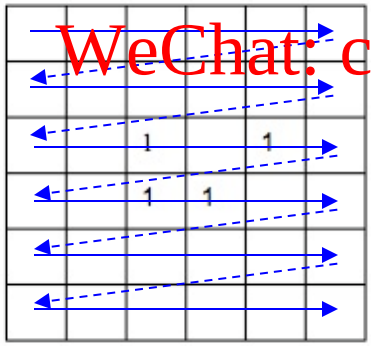
A

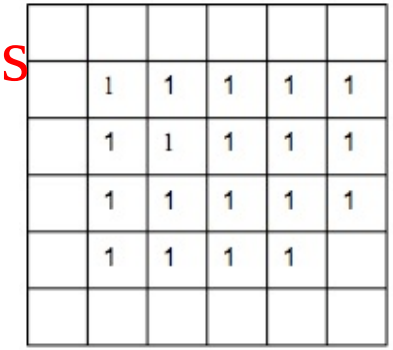




B: structural element

$A \oplus B$





Matlab function: imdilate(A, B)

Dilation

- As the structural element B runs through each of the pixels of image A, the following operations are carried out:
  - Check the overlaps between the elements of B and the corresponding pixels of A (when any pixel beneath the mask is a foreground pixel, i.e.. have a value 1, the mask B and image A are considered overlapped at that location)
  - Then the current pixel (the pixel of A that corresponds to the ***centre*** of B) is recorded as a foreground pixel (e.g., “1”), Otherwise record as a background pixel (e.g., “0”)
- Dilation of A by B is noted as

$$A \oplus B$$

Erosion

- An operation that "thins" objects in a binary image.
- How it work:
  - Scan an image **A** with a small "window" or "mask" **B** - The same as dilation
  - Operation
    - As mask B runs through each pixel of image A, check the elements of B that are overlapped with and the corresponding pixels of A
    - If at the current pixel of A, B completely overlap A (i.e., all the pixels corresponding to B are foreground pixels, record the current pixel as foreground (e.g., value “1”). Otherwise record it as background (“0”)
- Erosion of A by B is noted as $A \ominus B$

Erosion

A

	1	1	1	1	1
	1	1	1	1	1
	1	1	1	1	1
	1	1	1	1	

Matlab function: imerode(A, B)

B

1	1	1
1	1	1
1	1	1

$A \ominus B$

		1	1	1	
		1	1		

Original noisy image of fingerprint

After erosion

Dilation and Erosion

- Dilation "thickens" objects, whereas erosion "thins".
- However, they do **NOT** cancel each other, i.e., erosion followed by dilation does not give the original image, i.e.,

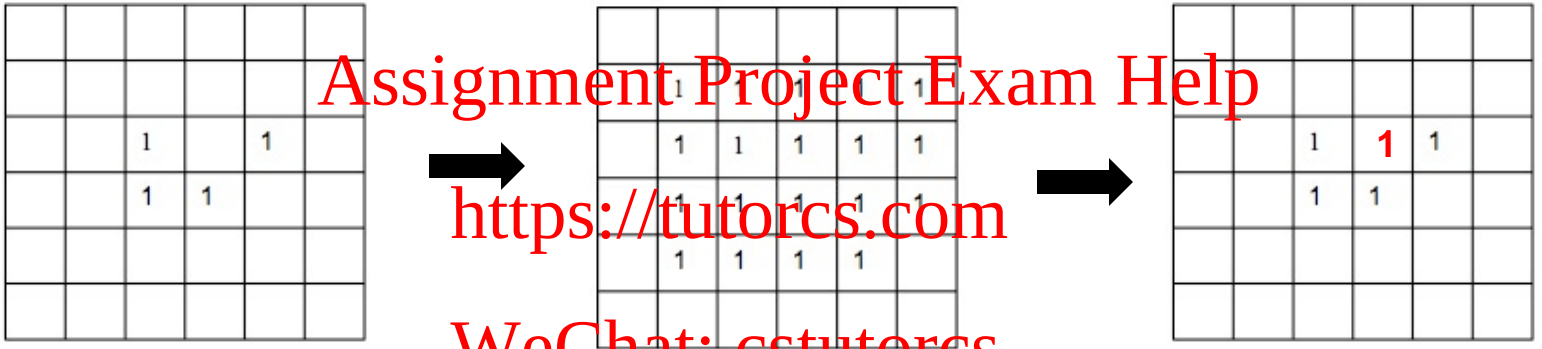
$$A \neq (A \ominus B) \oplus B$$

$$A \neq (A \ominus B) \oplus B$$

- Be careful to use them if keeping the original shapes of the foreground features is critical.

Dilation and Erosion

$$A \neq (A \oplus B) \ominus B$$



Original

After dilation

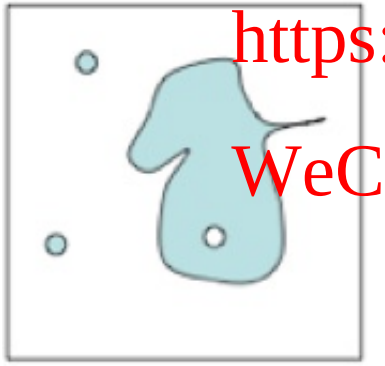
After erosion

- Be careful to use them if keeping the original shapes of the foreground features is critical.

Example

- Consider applying dilation and erosion, but in different order, to the following image:

$$(A \ominus B) \oplus B$$

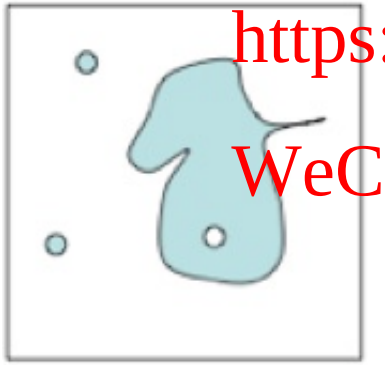


$$(A \oplus B) \ominus B$$

- How would the processed image look like (roughly)?

Cont'd

- Consider applying dilation and erosion, but in different order, to the following image:



- How would the processed image look like (roughly)?

$(A \ominus B) \oplus B$

$(A \oplus B) \ominus B$

Opening and Closing

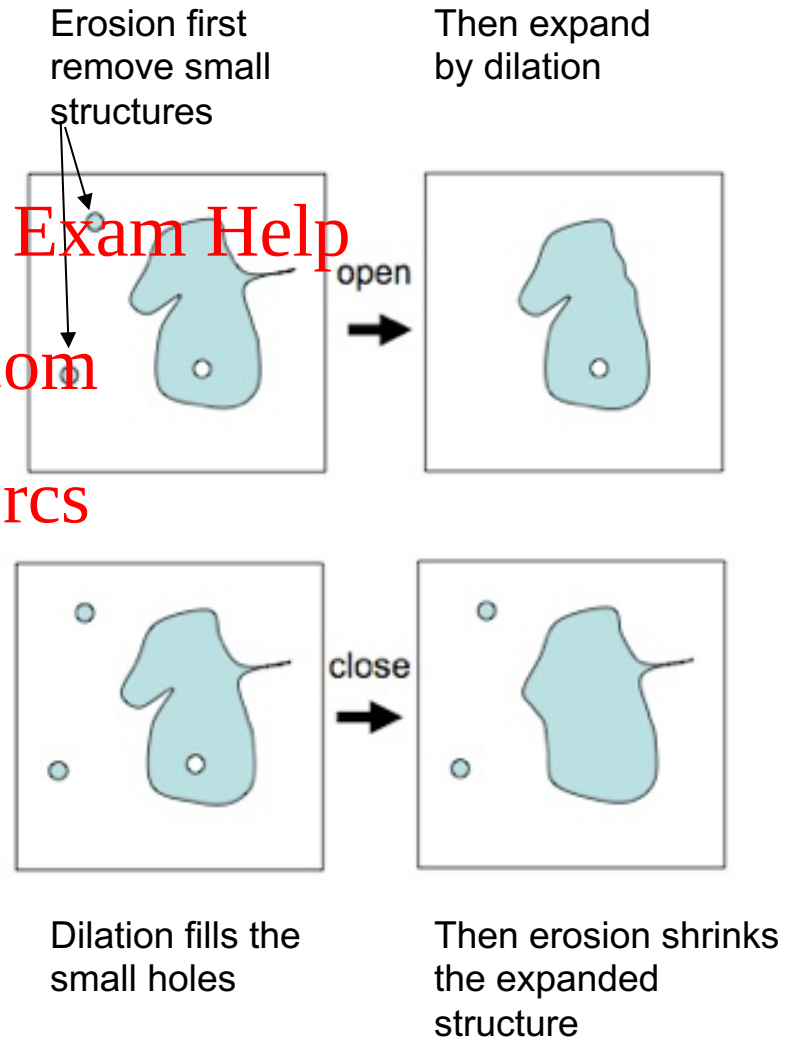
- In fact, these two set of operations define two new operations.
- Opening: Erosion followed by dilation

$$A \circ B = (A \ominus B) \oplus B$$

  - *Eliminates small regions and projections*
- Closing: Dilation followed by erosion

$$A \bullet B = (A \oplus B) \ominus B$$

  - *Fills in small holes and gaps*



Openings and Closings

- Like dilation and erosion, opening and closing do not ***exactly*** cancel each other.
- Better result can be obtained if they are used in pairs

Original noisy image

$A \circ B = (A \ominus B) \oplus B$

Opening remove noise but also create small gaps

$(A \circ B) \cdot B$

Close the small gaps by ***closing*** operation

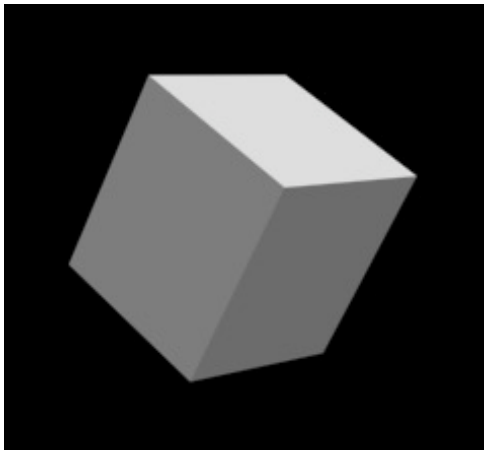
Grayscale Image Processing

- Binary images are restricted to 2D and planar applications, e.g., recognise objects on conveyer belt, text recognition etc.

- Grayscale images contain lot more information than binary images do.
  - E.g., Shading contains valuable 3D information

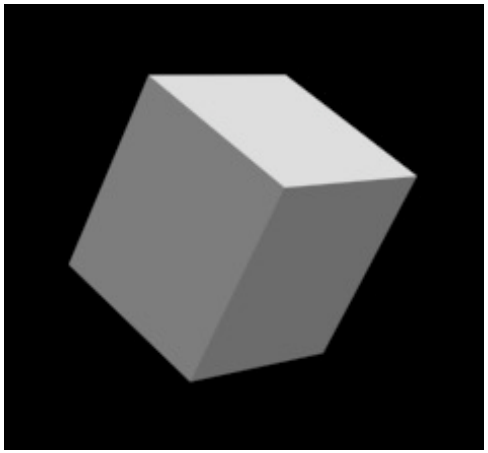
A Question

- What information is most significant when you work out the 3D structure?



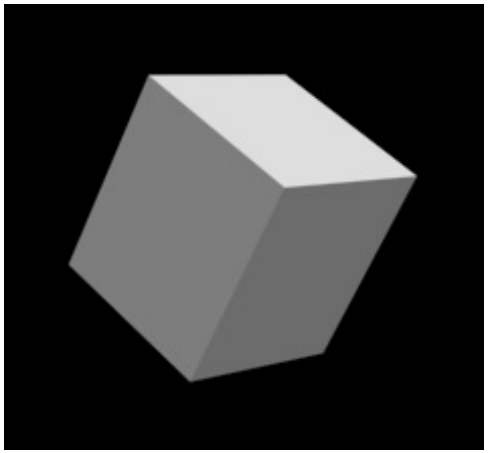
A Question

- What information is most significant when you work out the 3D structure?
  - Shading information (change of the intensity level of pixels)?
  - Edge?



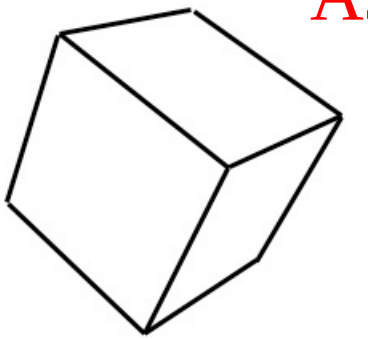
An Open Question

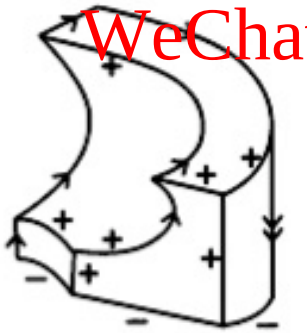
- Cognitive vision research shows both are at working.



Edges v.s Shading

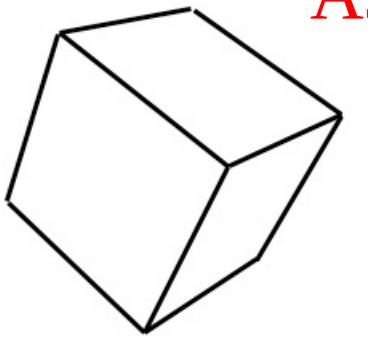
- Edges define shapes.

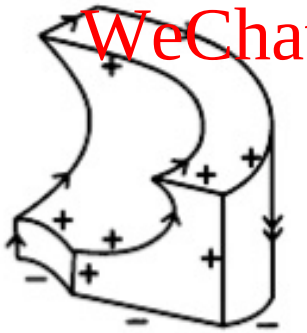


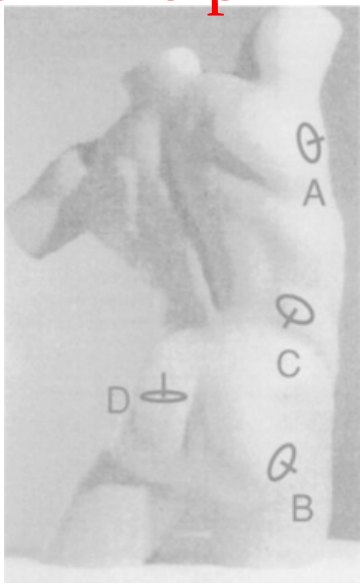


Edges v.s Shading

- Edges define shapes.
- Shading also define shapes.







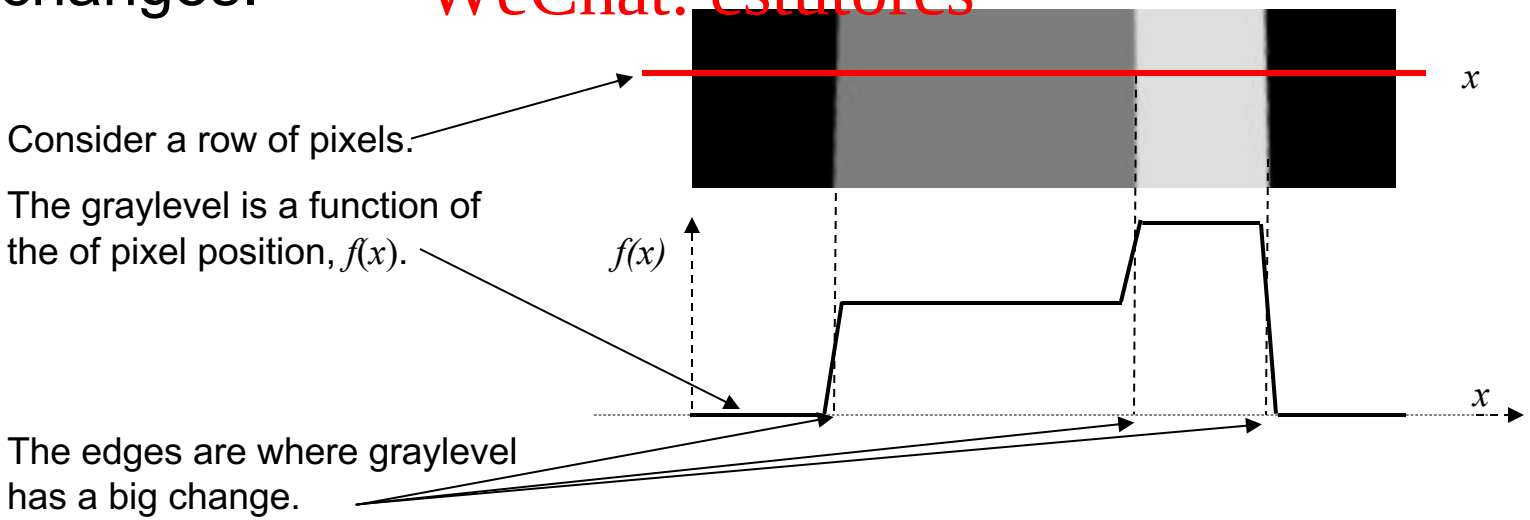
Both have been rigorously studied. In this lecture, we shall focus on the extraction of edge information.

Edges & Image Function

- The issue of edge finding is to find the (abrupt) changes in image gray level/intensity.
- If we view the gray level as a function (image function) of pixel position, i.e.,

$$f(x) = graylevel\ of\ pixel\ at\ x$$

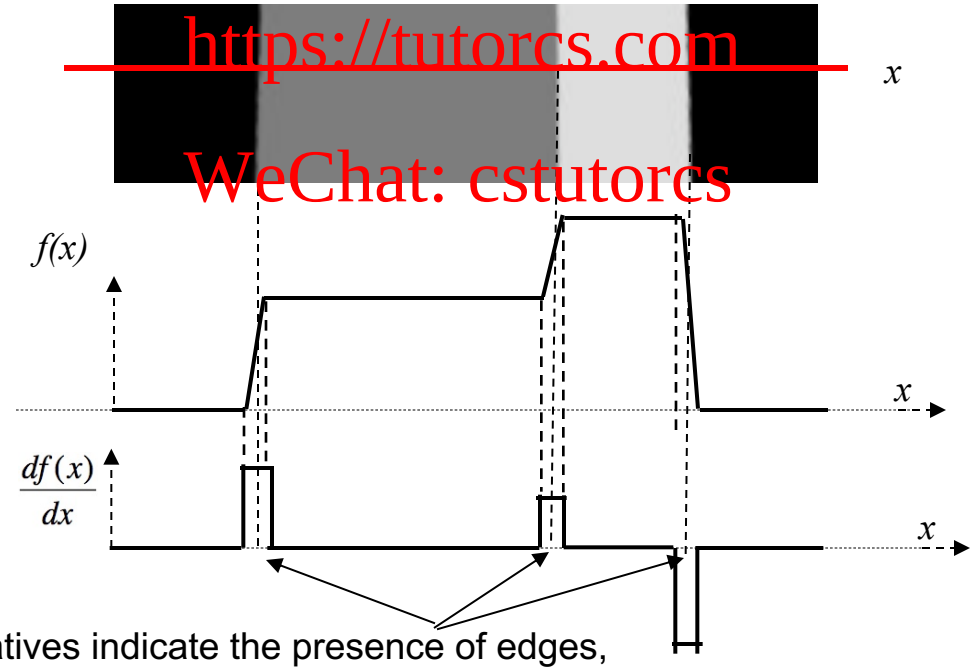
then edges correspond to where the function value changes.



Edges & Derivatives of Image Functions

- Mathematically, it means that an edge is where the *first derivative* (gradient) has a non-zero value, i.e.,

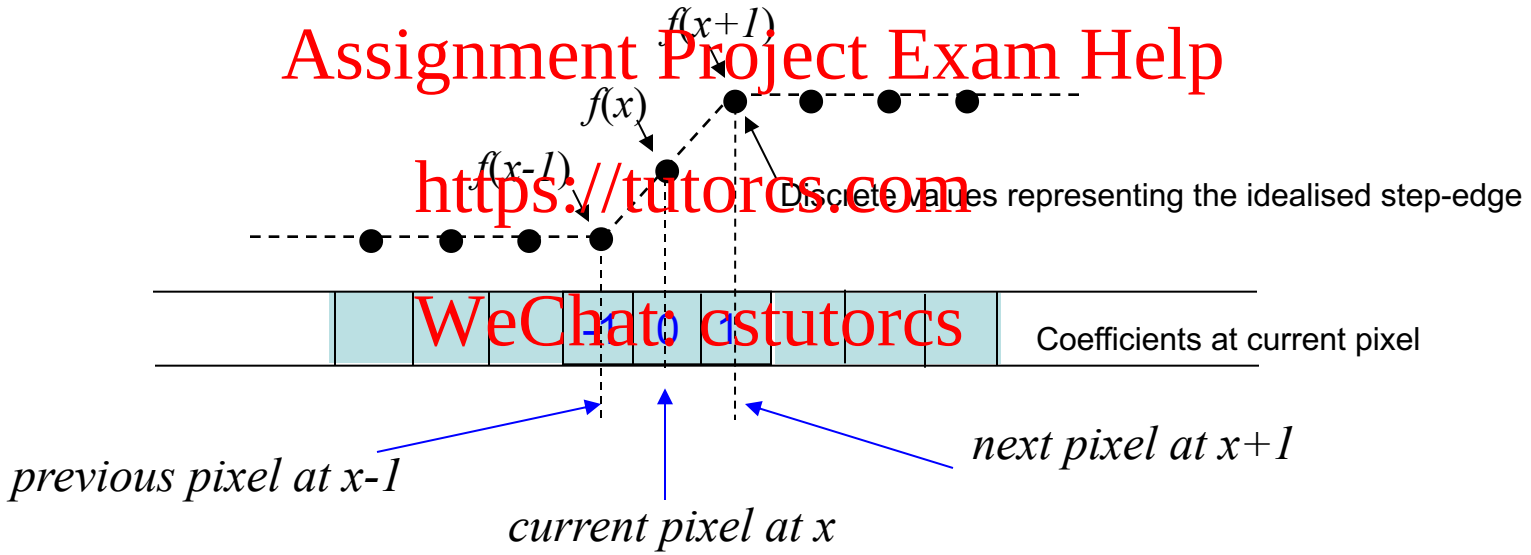
$$\frac{df(x)}{dx} = some\_value \qquad \frac{df(x)}{dx} = \lim_{\Delta x \to 0} \frac{\Delta f(x)}{\Delta x}$$



Non-zero first derivatives indicate the presence of edges, although the exact locations of the edges are to be determined.

Compute Derivatives: 1D

- Discrete (i.e., digital approximation of) first derivative

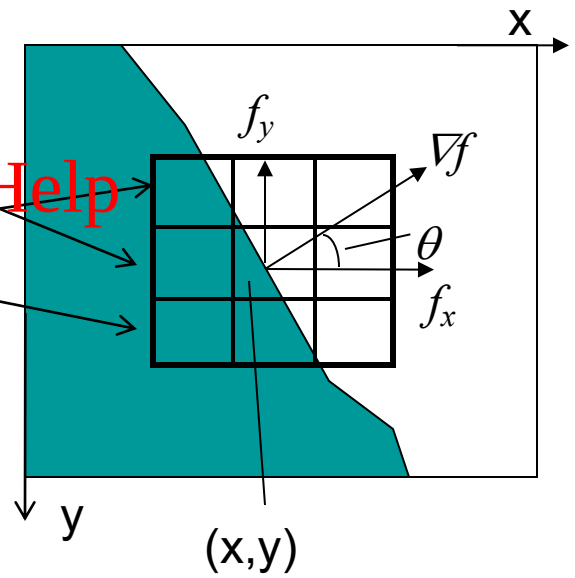


$$\frac{df}{dx} = \frac{f(x+1) - f(x-1)}{2} \quad \Rightarrow \quad \frac{df}{dx} = \frac{(+1) \bullet f(x+1) + (-1) \bullet f(x-1)}{2}$$

Partial Derivatives: 2D

- In 2D, we need to know the gradients/derivatives in two directions. These are called partial derivatives.

$$\partial f/\partial x \equiv f_x \approx \tfrac{1}{3}[(I[x+1,y]-I[x-1,y])/2$$
$$+(I[x+1,y-1]-I[x-1,y-1])/2$$
$$+(I[x+1,y+1]-I[x-1,y+1])/2)]$$
$$\partial f/\partial y \equiv f_y \approx \tfrac{1}{3}[(I[x,y+1]-I[x,y-1])/2$$
$$+(I[x-1,y+1]-I[x-1,y-1])/2$$
$$+(I[x+1,y+1]-I[x+1,y-1])/2)]$$



- The two partial derivatives define a directional derivative that has both a magnitude and a direction:

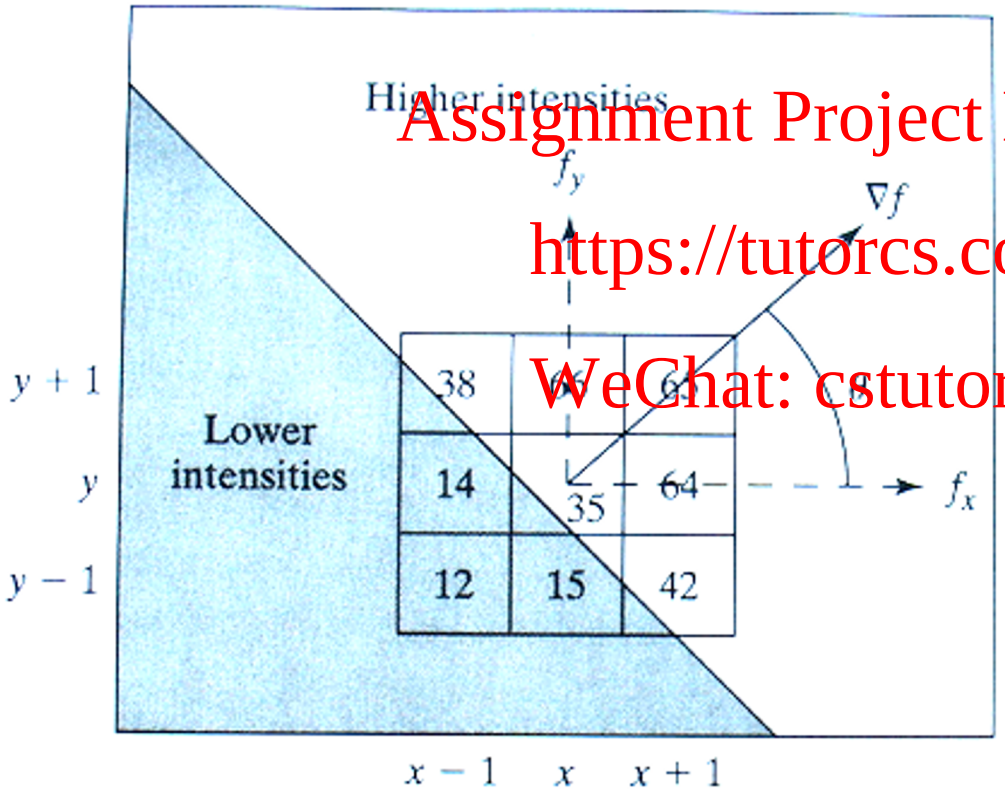
$$|\nabla f| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}$$

Edge magnitude

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

Edge direction angle

An Example



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

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

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

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

Masks & Filtering

- From the computation, the calculation of partial derivatives at a pixel is equivalent to applying a mask (a small image window containing coefficients) to the pixel and its neighbours.

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

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

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

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

- i.e.,
  - 1. multiply the pixel values with the corresponding values in the masks;
  - 2. find the sum of the products;
  - 3. divide the sum by 6 (i.e., 2*3)

$$\partial f/\partial x \equiv f_x \approx \tfrac{1}{3}[(I[x+1,y]-I[x-1,y])/2 + (I[x+1,y-1]-I[x-1,y-1])/2 + (I[x+1,y+1]-I[x-1,y+1])/2)]$$

Cont'd

- When the masks (operations) are applied to every pixel of an image, we say the image is filtered by the masks.
- When filtering images, step 3 (i.e., dividing by a constant) can be omitted, because it is just a scale factor and the same for every pixel.

- For different operations, we will have different masks. But the concept of filtering is the same.

Edge-Detection Masks

- In fact, the two masks we have discussed are edge-detection masks called *Prewitt masks.*

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

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

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

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

- When more importance (by doubling the weight) is given to the centre column/row, we have the *Sobel masks.*

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

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

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

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

Matlab Examples

- Read a (grayscale) image

```
i = imread('building.tif');
```


- Construct Sobel masks

```
sx=[-1   0   1;  -2   0   2;  -1   0   1] % construct the mask as an matrix
sy =sx'    %  x' is the transpose of x
```


- Filter the image using sx and sy

```
edge_x = filter2(sx, i);    %where edge_x is the derivatives
                            %at every pixels
edge_y = filter2(sy, i);    %Do the same with sy
```

Where filter2() is the filter function that applies a mask to an image.

Cont'd

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

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

Find vertical edges

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

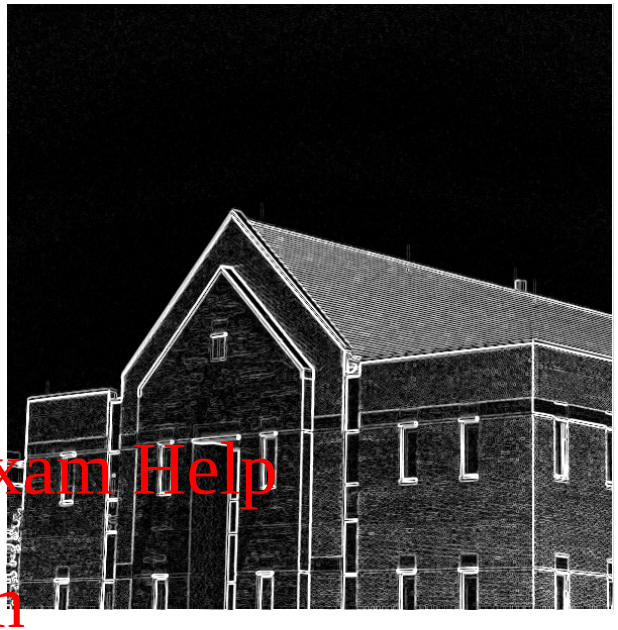
1	2	1
0	0	0
-1	-2	-1

Find horizontal edges

Directional derivative/gradient

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

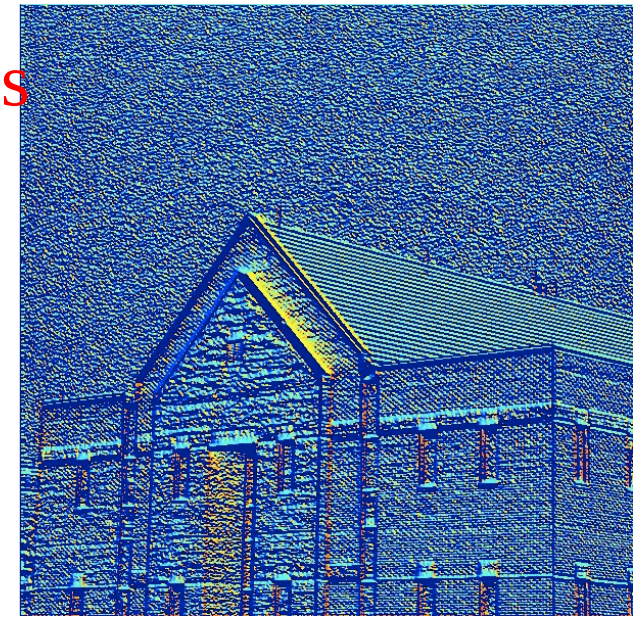
bd_edge = sqrt(bx. ^2+by. ^2);



Calculate gradient angles (and show them in random colours)

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

ang=atan2(by, bx);



Properties of Good Edge Detectors

- Isotropic
  - Work well for edges of different directions.
- Good localisation
  - Give accurate edge location and orientation.
- Good signal-to-noise characteristics
  - Robust to noise.

How Good Are Prewitt & Sobel?

- Isotropic
  - Not so good. Each work in one direction only (vertical or horizontal)
- Good localisation & orientation
  - No
- Good signal-to-noise characteristics
  - Noisy
- But they are simple and efficient, and provide good results when appropriately used.

- We will introduce better edge detectors.

Readings

- Shapiro, L.G., Stockman, G.C., Computer Vision, Prentice-Hall, 2001, ISBN 0-13-030796-3
  - Section 5.1 to 5.8