

# EE6222: Machine Vision

Lecturer :

Dr. P. N. Suganthan

Office: S2-B2a-21

Tel : 67905404

E-mail: [epnsugan@ntu.edu.sg](mailto:epnsugan@ntu.edu.sg)



Slides can be downloaded from NTUlearn after enrolling in this subject.

This set is available from: <https://github.com/P-N-Suganthan>

# Course Schedule

- Weeks 1-7: Fundamentals of Computer Vision, Machine Intelligence and Applications
  - A/Prof P N Suganthan
- Weeks 8-13: 3D Vision
  - A/Prof Wang Han
- I have an assignment for the first module.
- **Assessments**
  - Final Exam – 80% (You can see the past exams online from NTU Library web pages)
  - Two take-home CAs – 10% + 10%.

# Outline of Topics

1. Overview of image analysis (slides 10-11)
2. Image capturing (slides 12-20)
  - sensors, video camera, lighting
3. Image formation (slides 21-27):
  - Resolution, gray levels

# Outline of Topics

## Preprocessing of images (28-84)

### 4. Image enhancement (slides 28-46):

- Look-up Table (LUT) operations, contrast stretching, histogram equalization;

### 5. Image filtering (slides 47-57):

- Fourier Transform, Convolution and Correlation, low-pass, band-pass, high-pass filters.

# Outline of Topics

## 6. Neighborhood Operations (slides 58-79):

- Lowpass smoothing :Block Averaging, Weighted Averaging, Gaussian Smoothing,
- Highpass sharpening;
  - Noise Removal by temporal averaging
  - Nonlinear filtering: Rank value filters, median filtering

# Outline of Topics

## 7. Image Segmentation (slides 80-104):

- Thresholding: Otsu's threshold selection.
- Edge extraction
  - spatial domain :
    - Gaussian Smoothing + differentiation, Roberts, Sobel, Laplacian, Laplacian of Gaussian, Canny edge detection scheme;
  - spatial-frequency domain :
    - Difference of Gaussians;
- Region segmentation:
  - region growing, split & merge

# Outline of Topics

## 8. Feature Extraction-I (slides 105-141):

- Line extraction : Thinning, Hough Transform
- Curve extraction : functional fit, Hough Transform
- Corner extraction :
  - Gray-level corner : Plessey
  - Geometric corner : Curvature extreme points
- Pattern (Model) Extraction : Template matching

## • 9. Morphological Operations (slides 142-153):

- Erosion, Dilation, Opening, Closing,  
Skeletonization, Morphological edge detection.

# Outline of Topics

## 10. Textures Analysis (slides 154-166):

- Overview, Gray level co-occurrence matrix, Texture features, Multi-channel features

## 11. Feature extraction-II (slides 167-204):

- Projection profile, chain codes, polygonal approximation, Fourier descriptors, etc.
- Color Images

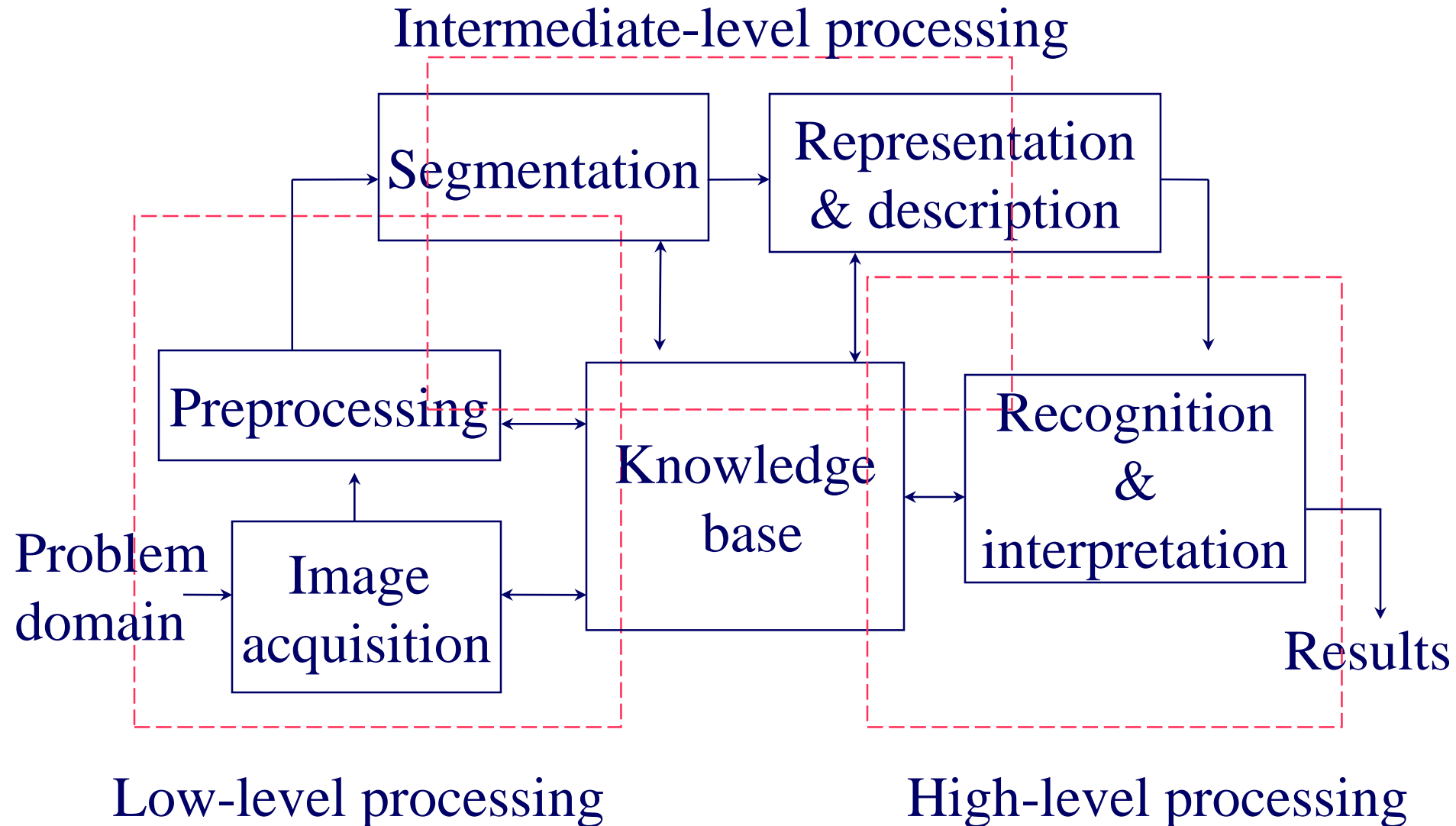
## 12. Pattern Recognition: Lecture Set II for Weeks 5-7.



# Books

1. R. M. Haralick & L.G. Shapiro, *Computer and robot vision* . Addison-Wesley Pub. Co., 1992 g TA1632.H254
2. R.C. Gonzalez & R. E. Woods, *Digital image processing*, Addison-Wesley Pub. Co., g TA1632.G643. 2002 or 2008 editions Prentice Hall.
3. R. O. Duda, P. E. Hart, D. G. Stork, *Pattern Classification*, Wiley, 2001, **Q327.D844 2001**.
4. **W. K. Pratt, Digital image processing, John Wiley & Sons, 2001, also TA1632.P917 2007.**
5. Bow ST, *Pattern recognition and image preprocessing*, M. Dekker, TK7882.P3B784P 2002.
6. Louis J Galbiati Jr., *Machine Vision and Digital Image Processing Fundamentals*, Prentice Hall, 1990. TA1632.G148

# 1. Overview of image analysis



# Low, intermediate & High Level Processing

- Low level Processing: If the output of the processing is another image
- Intermediate Level Processing: If the output is intermediate representation of the input.
- High Level Processing: If the output is high level interpretation / recognition of the image.

## 2. Vision System Hardware

A typical computer vision system consists of

- image sensor (camera + lens)
  - image digitizer
  - image store
  - computer
  - appropriate illumination of the scene.
- 
- There are numerous other imaging devices such as X-ray, thermal, MRI, ultrasound, nuclear, ultra-violet, infrared, etc.

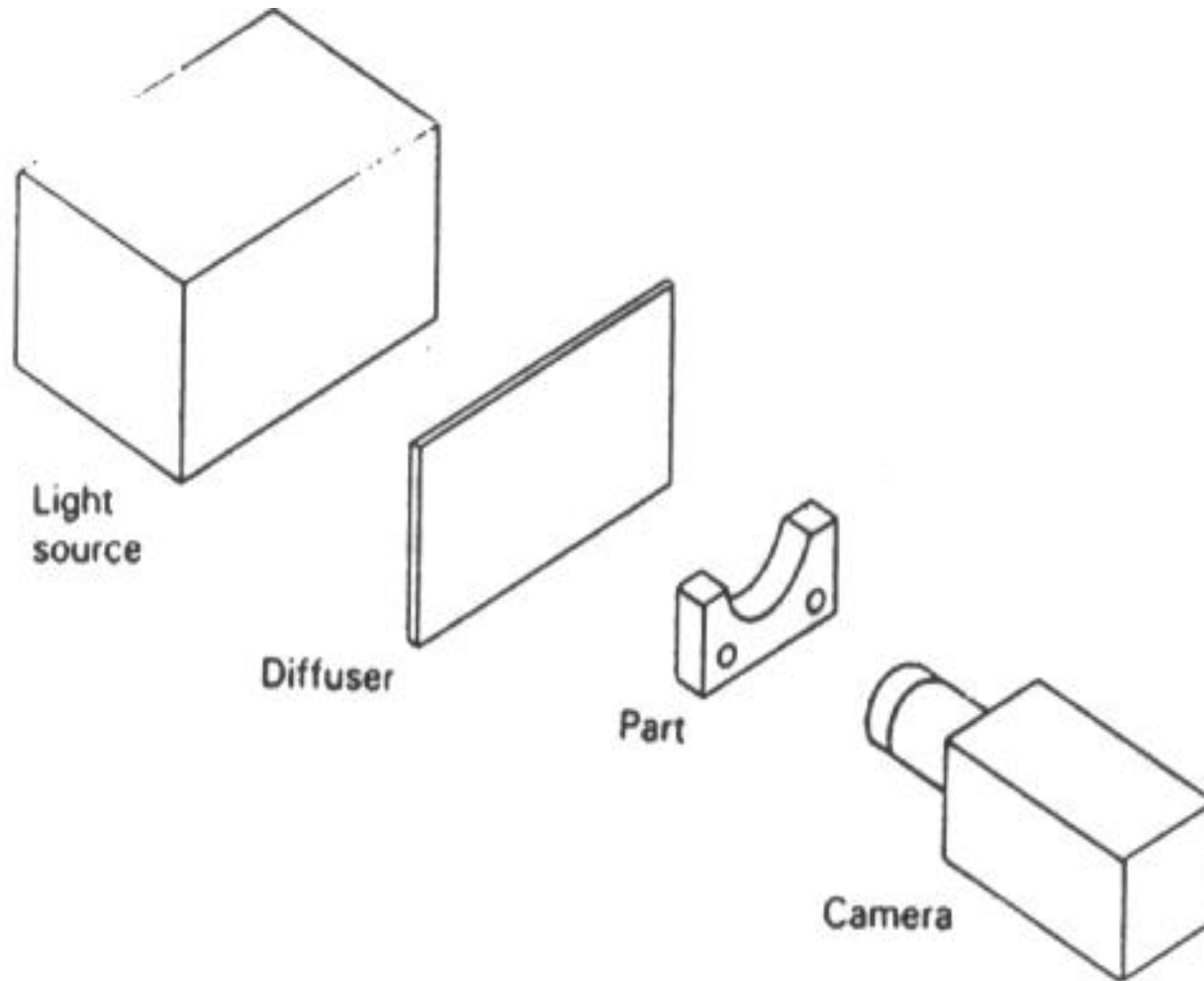
# Solid-state Video Camera

- Separate photodiode located at each location.
- Photodiodes are arranged in either a linear array or a rectangular array.
- Resistivity of the photodiode decreases proportionally to the amount of light falling on it.
- The output signal voltage is dependent on the resistance of the photo-conductor.
- Output video is obtained by scanning each diode in ordered sequence to produce a series of voltage pulses representing the pixel value at the respective location.
- Most common type of solid-state camera : CCD (or charge-coupled device).

## Lighting (Galbiati chapter 2)

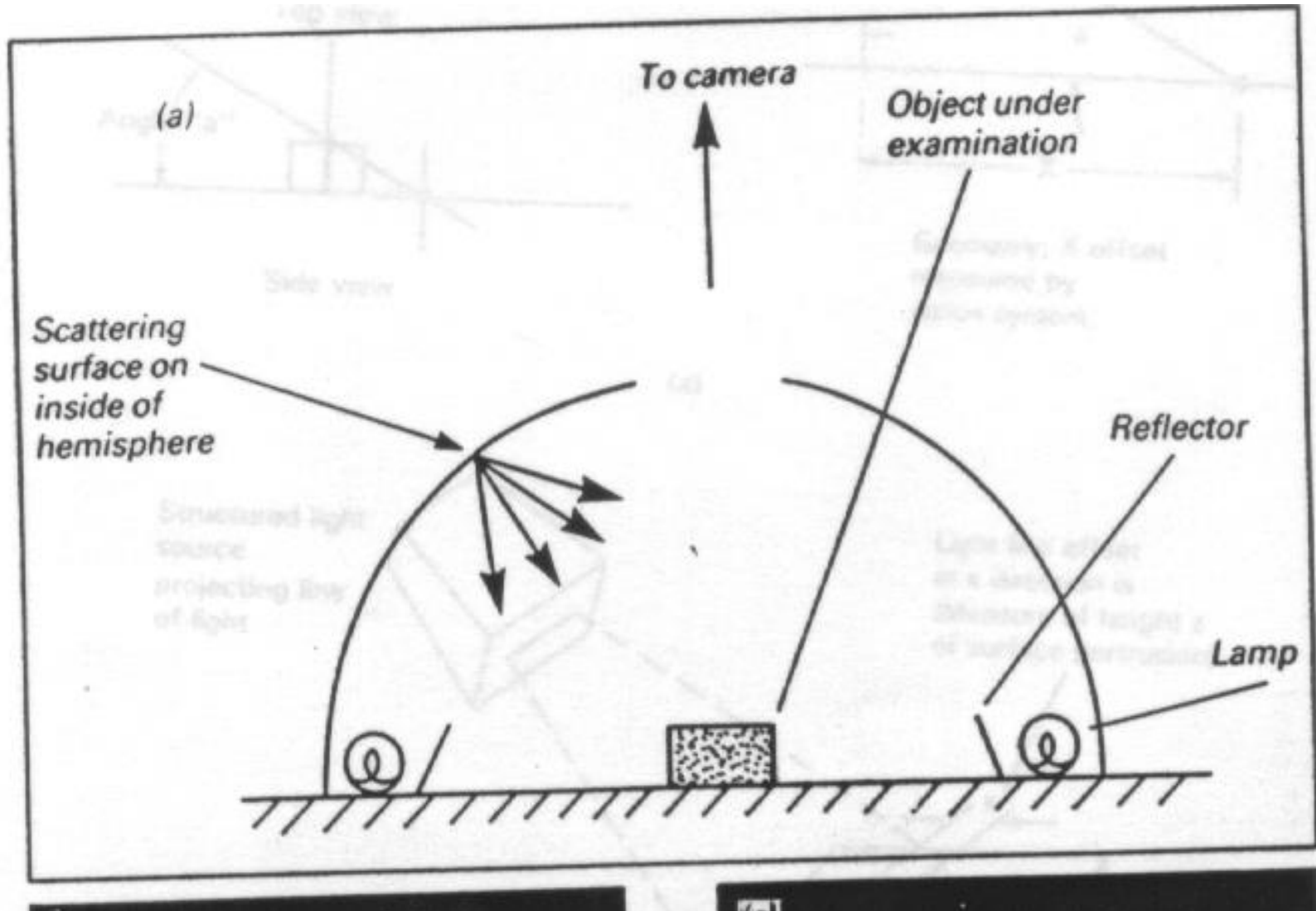
- Very important parameter in computer vision.
- Use controlled lighting whenever possible.
- No general, standard lighting equipment available, it is application dependent.
- Choice of illumination method affects the amount of processing and achievable results.
- main categories of lighting techniques:
  - Back lighting
  - Front lighting : uniform and directional
  - Structured lighting
  - Polarized lighting

## Back lighting



**Figure 2-3** Back lighting: the light source is on the opposite side of the object from the camera.

## Front lighting: uniform





## Front lighting (directional)

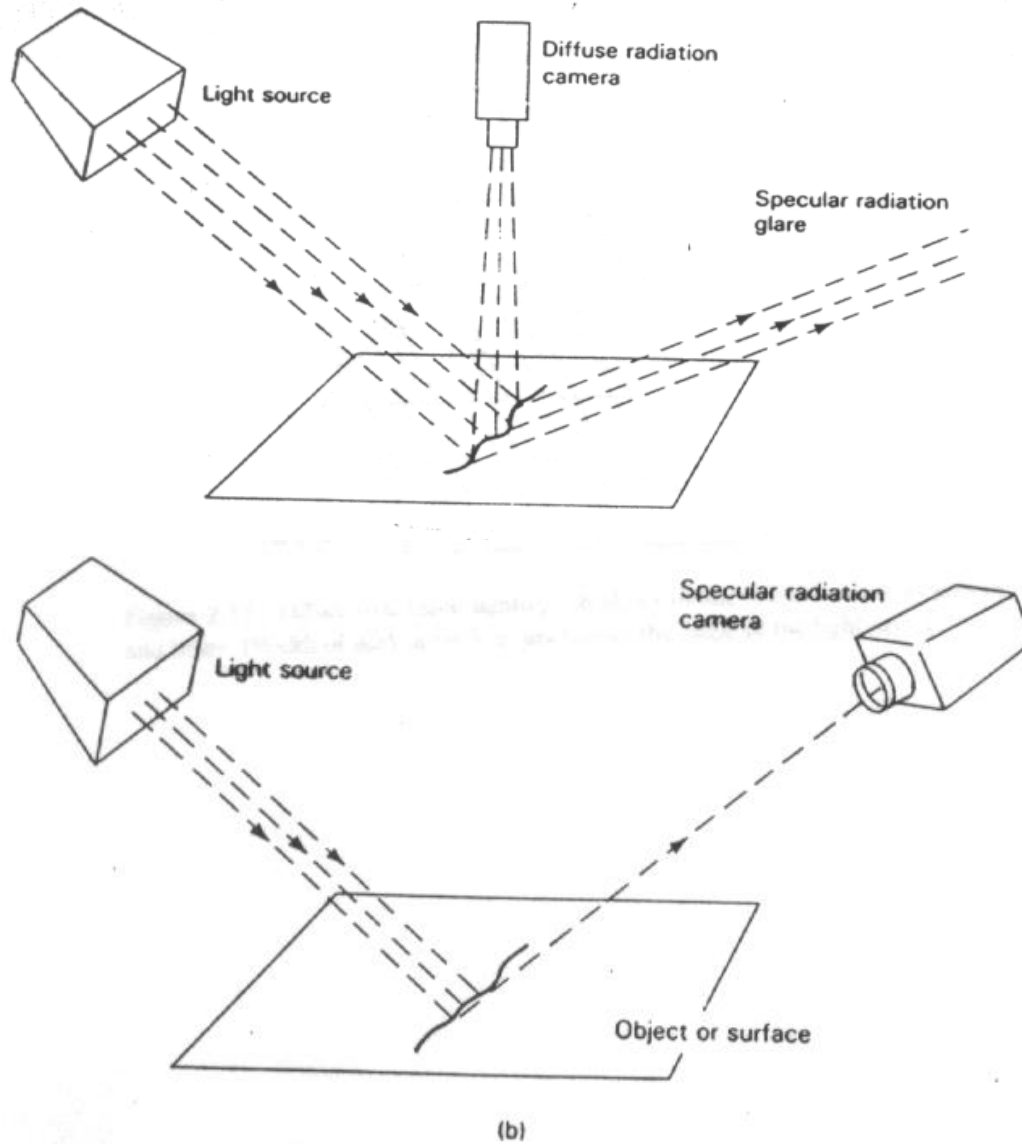
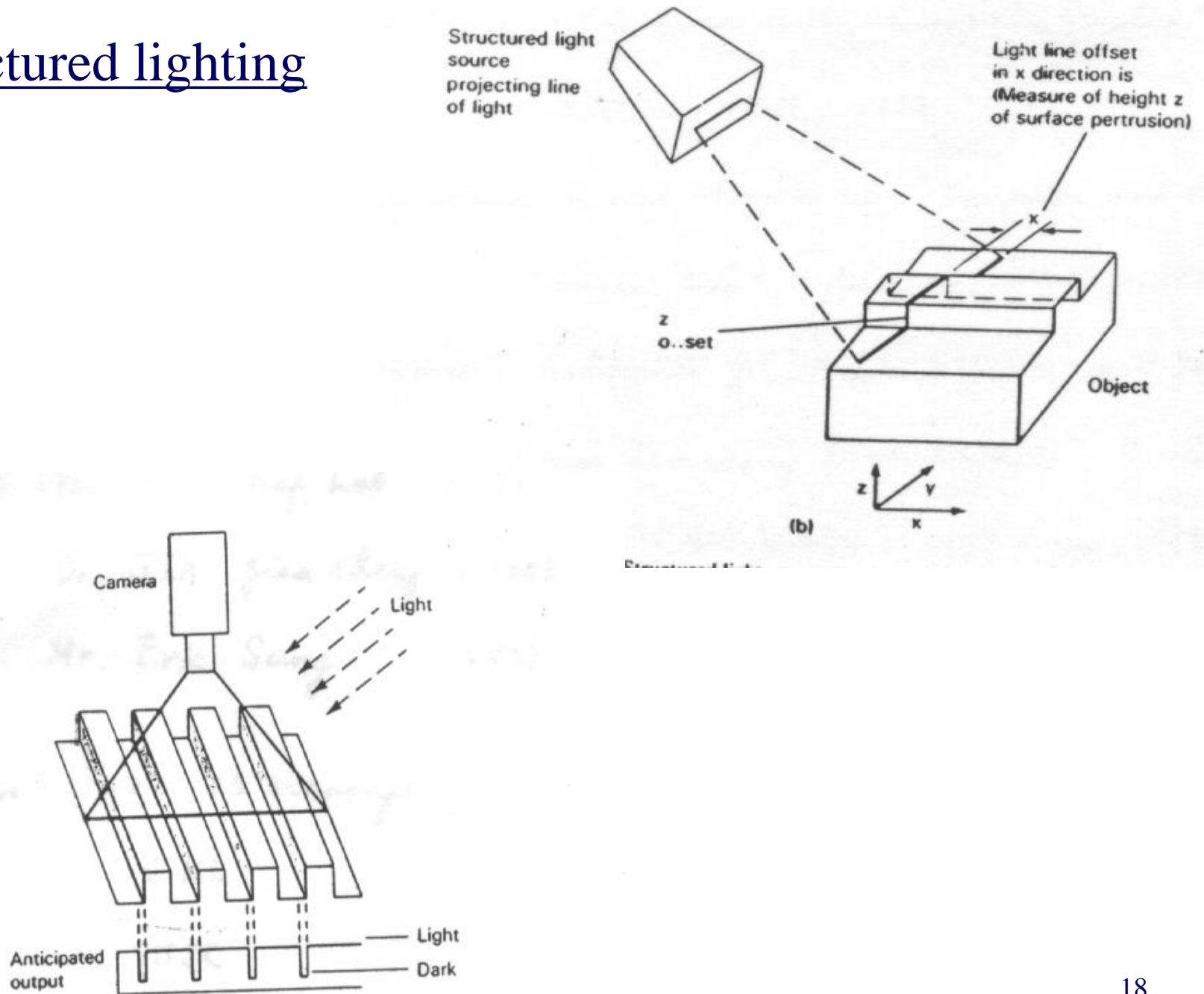


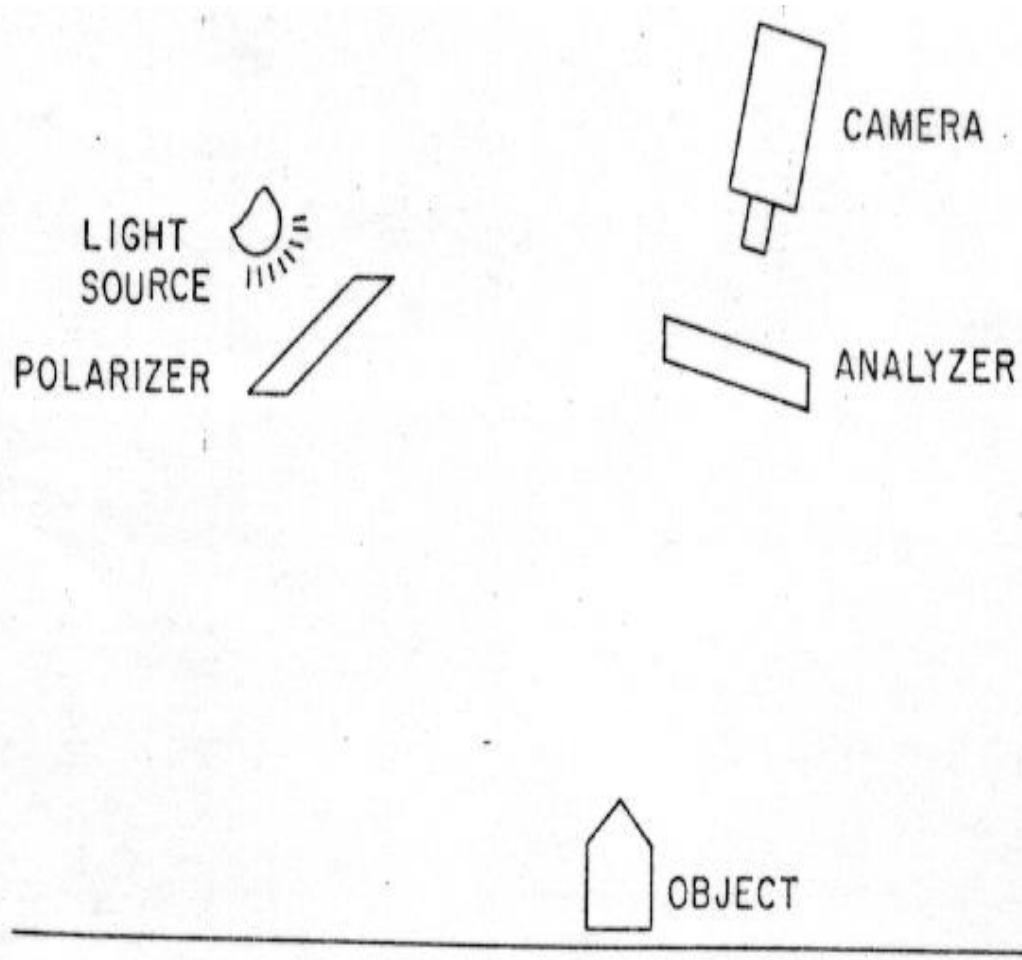
Figure 2-6 Specular and diffuse illumination. Defects show up as bright spots in

# Structured lighting

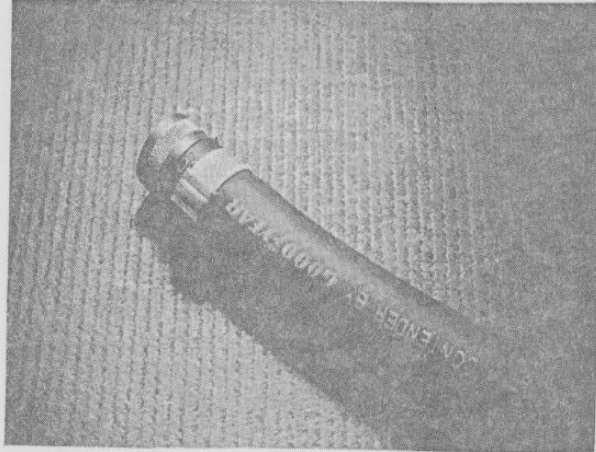


**Figure 2-11** Offset structured lighting: shadows define features such as grooves

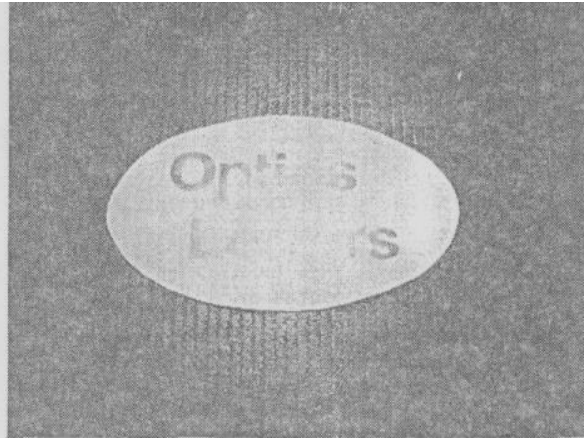
# Polarized lighting



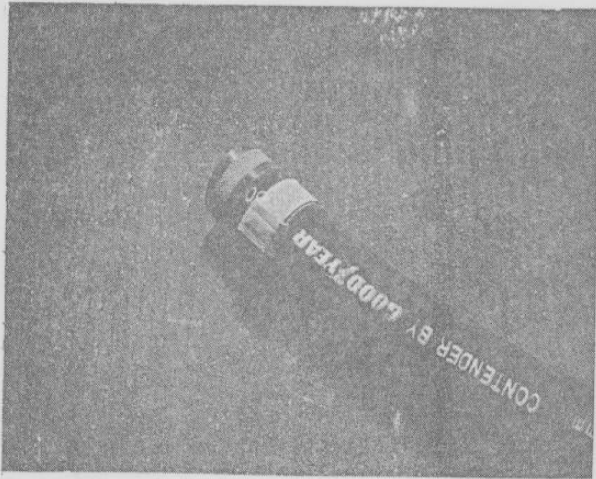
# Polarized lighting



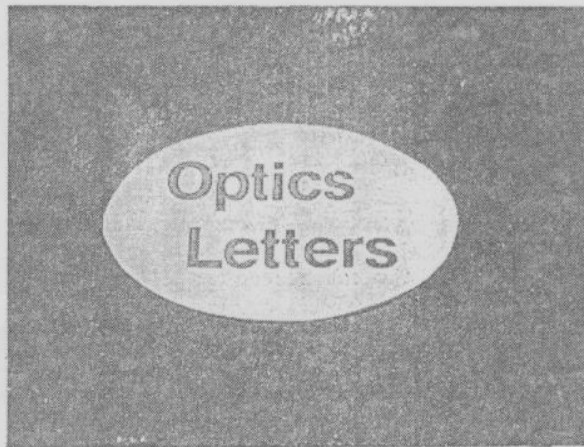
5a. No PAC



7a. No PAC



5b. PAC in place



7b. PAC in place

Figure 7. Elimination of glare from a label

Without polarized lighting. (specular objects)

With polarized lighting.

# 3. Image Formation

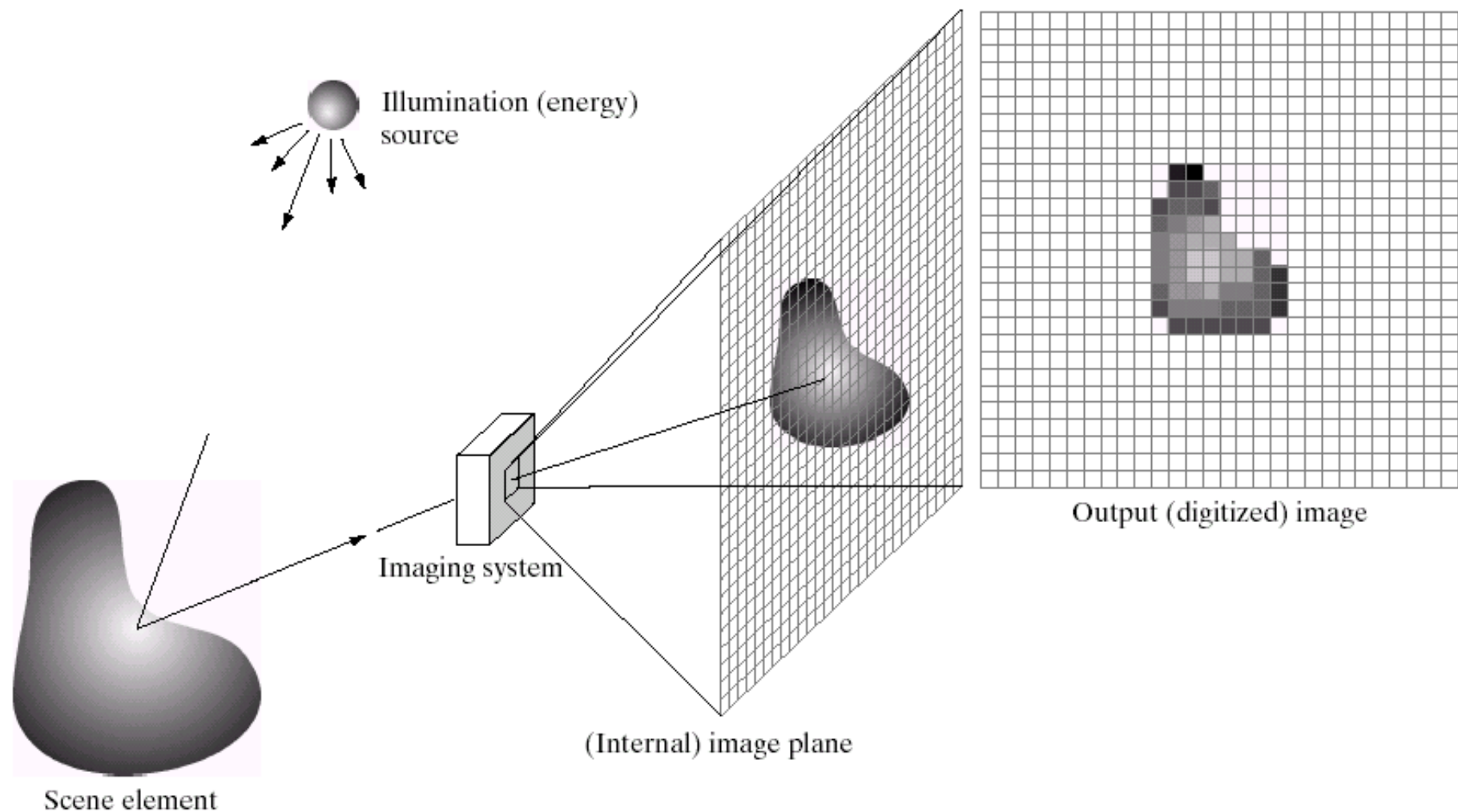
## Lens and geometric optics

- The image of a scene is focused on the image sensor with a lens.
- parameters associated with lens:
  - Magnification
  - Focal length
  - Depth of field
- The simple lens formula is often used:

$$\frac{1}{f} = \frac{1}{u} + \frac{1}{v}$$

- $f$  is the focal length of the lens.
  - $u$  is the object to lens distance.
  - $v$  is the image to lens distance.
- 
- The image formed is always inverted.
  - $u + v$  is always greater or equal to  $4f$ 

$$u + v \geq 4f$$
  - Choose  $f$  to obtain the required magnification:
 
$$\text{magnification, } m = \frac{v}{u}$$
  - Shorter focal length lens reduces the image size.



a b c d e

**FIGURE 2.15** An example of the digital image acquisition process. (a) Energy (“illumination”) source. (b) An element of a scene. (c) Imaging system. (d) Projection of the scene onto the image plane. (e) Digitized image.



## Image attributes

- An image is a 2D light-intensity function denoted by  $f(x,y)$ .
- The value or amplitude of  $f$  at spatial coordinates  $(x,y)$  gives the intensity (brightness) of the image at that point.
- $f(x,y)$  is non-negative and finite.
- $f(x,y)$  must be digitized both spatially and in amplitude for computer processing.
- Digitization of spatial coordinates  $(x,y)$  is called spatial quantization or discretization or sampling.
- Amplitude digitization is called gray-level quantization



## Image attributes

- The resolution of the image refers to the  $N \times M$  array to which an image is sampled.

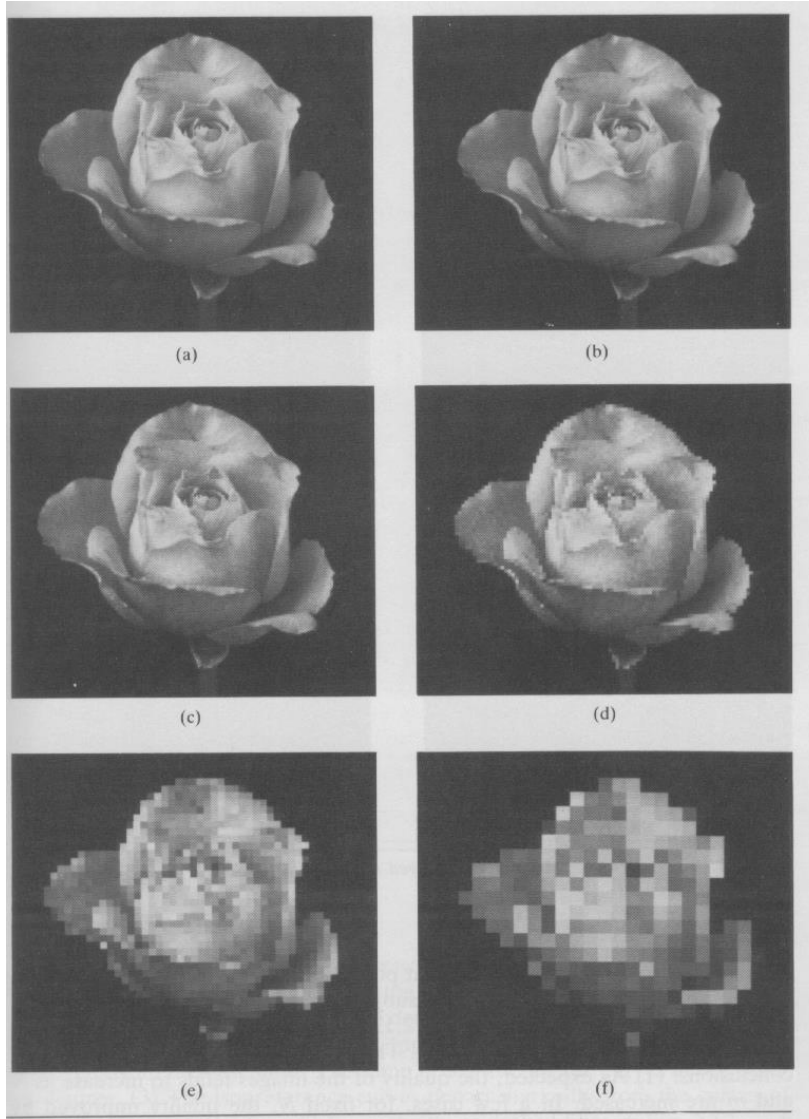
$$f(x, y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & & & \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix}$$

- Common image resolution has the following characteristics:  $N=2^n$ ;  $M=2^k$ , where  $n$  and  $k$  are integers.
- gray level  $G=2^m$ , where  $m$  is the no. of bits.

# Examples (spatial resolution varies with a fixed number of grey levels )

Grey level 0 – Black  
Grey level 255 – White

1024x1024



512x512

256x256

128x128

Image size is fixed  
Pixel size is increasing.

64x64

Instead, we can keep pixel size fixed while down sizing images!

32x32

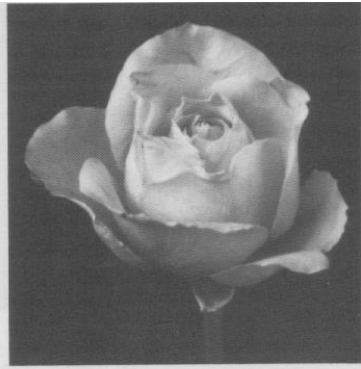
# Examples (gray level resolution varies with a fixed spatial resolution)

256 level



(a)

128 level



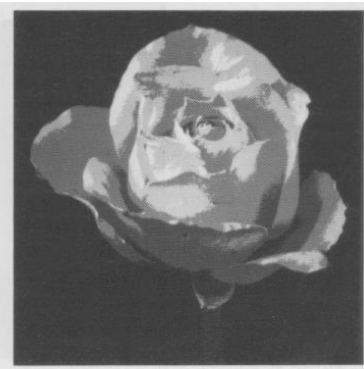
(b)

16 level



(e)

8 level



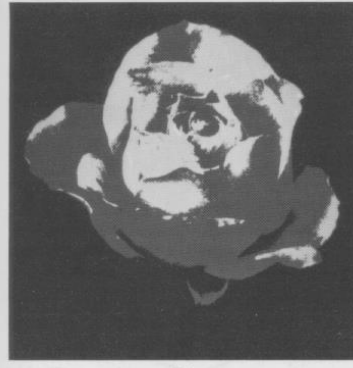
(f)



(c)



(d)



(g)



(h)

64 level

32 level

4 level

2 level

# Preprocessing of Images

## 4. Image Enhancement

- spatial domain methods:
  - pixel by pixel transformations (point processing).
  - neighborhood operations (area processing).
  - algorithms operate directly on pixel values in an image in spatial domain.
- spatial frequency domain methods:
  - image filtering in spatial frequency domain
  - image must be transformed from spatial domain to spatial frequency domain by Fourier transform before filtering
  - Filters can be designed in either spatial or spatial frequency domains, but used only in spatial frequency domain.
  - Inverse Fourier transform is needed to convert the results back to spatial domain.

- Model for spatial domain methods:

$$g(x, y) = T[f(x, y)]$$

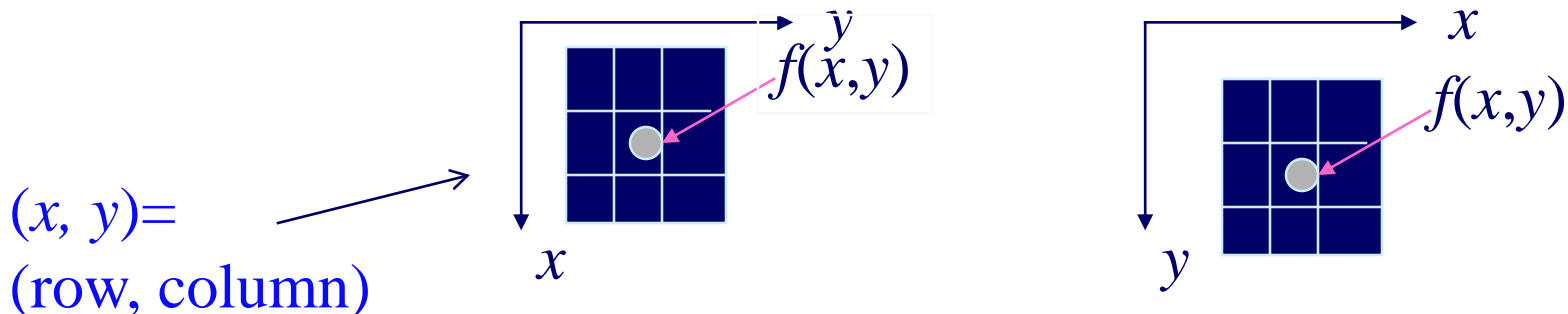
where  $f(x, y)$  is the input image

$g(x, y)$  is the processed (output) image

$T$  is an operator on  $f$  defined over some

**neighborhood  $N$**

- $T$  can operate on a set of images (**i.e. temporal averaging**)
- $N$  can be  $1 \times 1$ ,  $2 \times 3$ ,  $3 \times 3$ , ...  $N \times N$  (**usually square**)
- Image coordinates can be :



- Point processing ( $N=1$ ):
  - $T$  becomes a gray level transformation function

$$s = T(r) \quad \leftarrow x,y \text{ coordinates are removed!}$$

where  $r, s$  are variables denoting the gray level of  $f(x,y)$  and  $g(x,y)$  at  $(x,y)$ .

- Look-up-table (LUT) can be used to implement the transformation function.
- By defining the LUT, point processing can be performed by indexing the LUT.

## Point processing in C by Using the LUT:

```
void point_proc (BYTE *in_image, BYTE
    *out_image,BMPINFOHDR file_info, int LUT[])
{
    int image_size, i;

    image_size = file_info.biWidth * file_info.biHeight;
    for (i = 0; i < image_size; i++)
        out_image[i] = LUT[in_image[i]];
}
```

Before using the LUT, it has to be defined as shown in the next two slides.

## Defining the LUT in C:

```
void invert_LUT(int LUT[])
```

```
{  int i;  
    for (i = 0; i < 256; i ++)  
        LUT[i] = 255 - i;  
}
```

```
void point_threshold_LUT(int t, int LUT[])
```

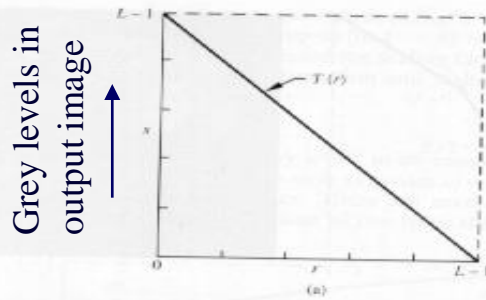
```
{  int i;  
    for (i = 0; i < t; i++)    LUT[i] = 0;  
    for (i = t; i < 256; i++)  LUT[i] = 255;  
}
```



## Defining the LUT in C:

```
void log_transform (int LUT[])
{
    int i;    double scale;
    scale = log(255.0);  LUT[0] = 0;
    for (i = 1 ; i < 256; i++)
        LUT[i] = (int)(255*log((double)i)/scale);
}

void exp_transform (int LUT[])
{
    int i;    double scale;
    scale = exp(2.55);  for (i = 0 ; i < 256; i++)
        LUT[i] = (int)(255*exp((double)i/100)/scale);
}
```



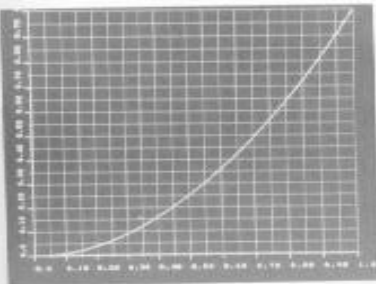
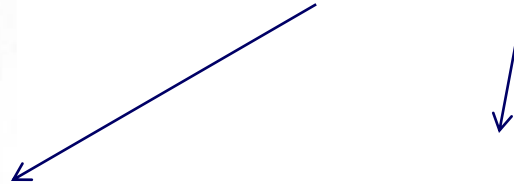
(b)



(c)

Figure 4.4 Obtaining the negative of an image: (a) gray-level transformation function; (b) an image; and (c) its negative. In (a),  $r$  and  $s$  denote the input and output gray levels, respectively.

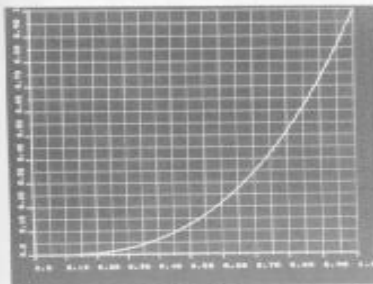
Original plane image is given in slide ~39. These 4 pictures show results after applying the transformations.



(a) Square function



(b) Square output

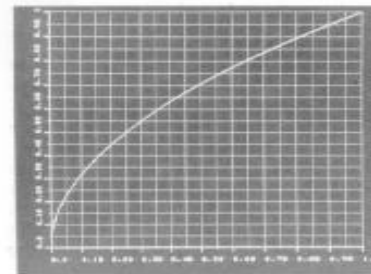


(c) Cube function



(d) Cube output

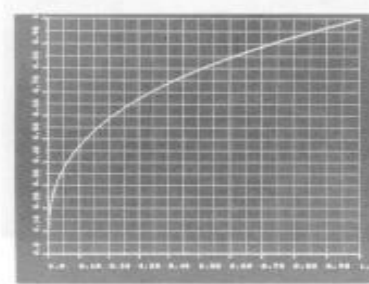
FIGURE 10.1-6. Example of square and cube contrast modification of the jet image.



(a) Square root function



(b) Square root output



(c) Cube root function



(d) Cube root output

FIGURE 10.1-7. Examples of square root and cube root contrast modification of the jet image.

## Point processing based on **histogram**:

- a histogram for an image with gray level range  $[0, L-1]$  is a discrete function

$$p(r_k) = \frac{n_k}{n}$$

where  $r_k$  is the  $k^{\text{th}}$  gray level;  $k = 0, 1, 2, \dots, L-1$

$n_k$  is the no. of pixels with  $k^{\text{th}}$  gray level

$n$  is the total no. of pixels in the region of the image being processed

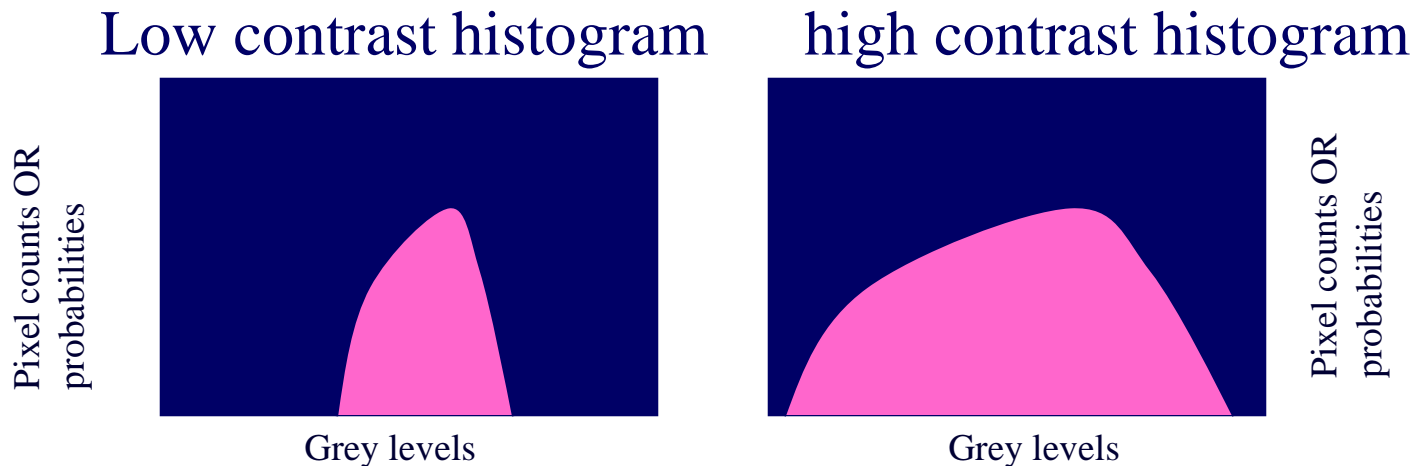
- $p(r_k)$  is an estimate of the probability of occurrence of gray-level  $k$ .

- Histogram shows the distribution of pixel values.
- Probability density function (pdf) can be obtained by normalizing the histogram with the total number of pixels.
- Cumulative pdf can be derived from pdf

```
void histogram(BYTE *img, BMPINFOHDR file_info, float hist[])
{
    int i, size;
    for (i = 0; i < 256; i++)    hist[i] = 0;
    size = file_info.biHeight*file_info.biWidth;
    for (i = 0; i < size; i++)
        ++hist[img[i]];
    for (i = 0; i < 256; i++)
        hist[i] = hist[i]/size;
}
```

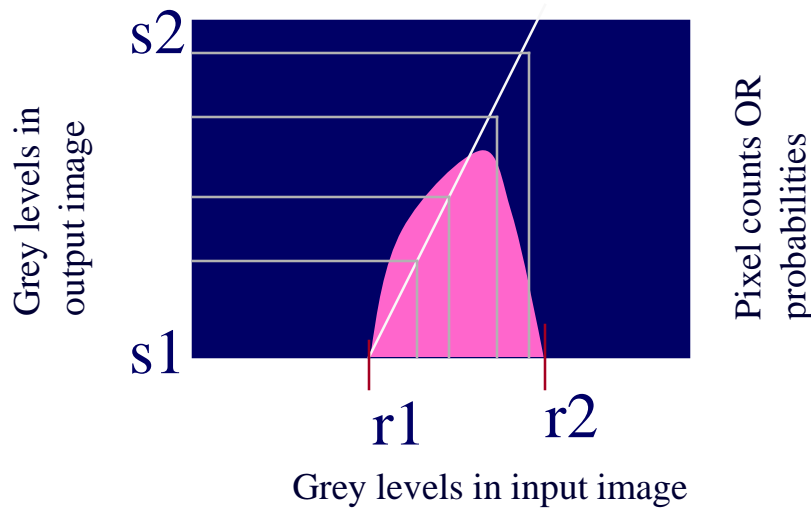
# Contrast-stretching:

- Low-contrast image may be due to :
  - poor illumination,
  - lack of dynamic range in the image sensor
  - wrong setting of the aperture
- Contrast stretching aims to increase the dynamic range of the gray levels in the image.
- Image contrast can be assessed from image histogram.

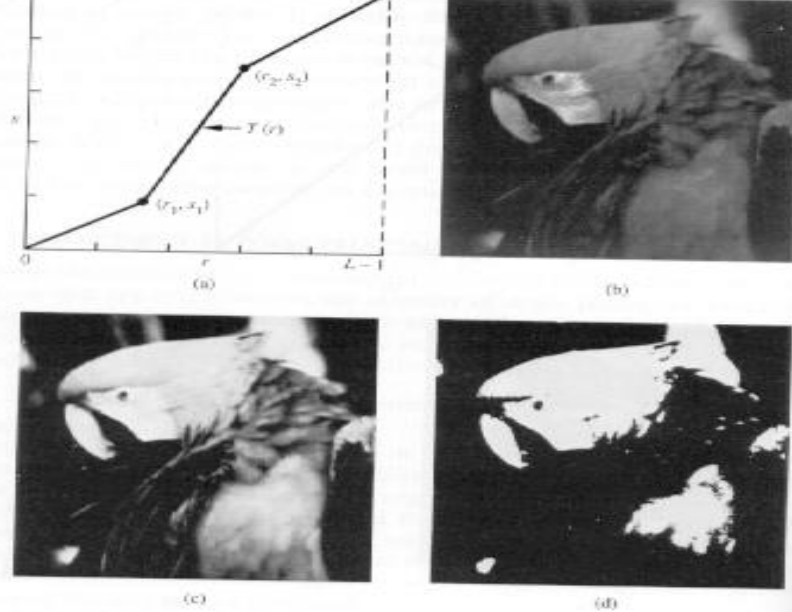


## Contrast-stretching:

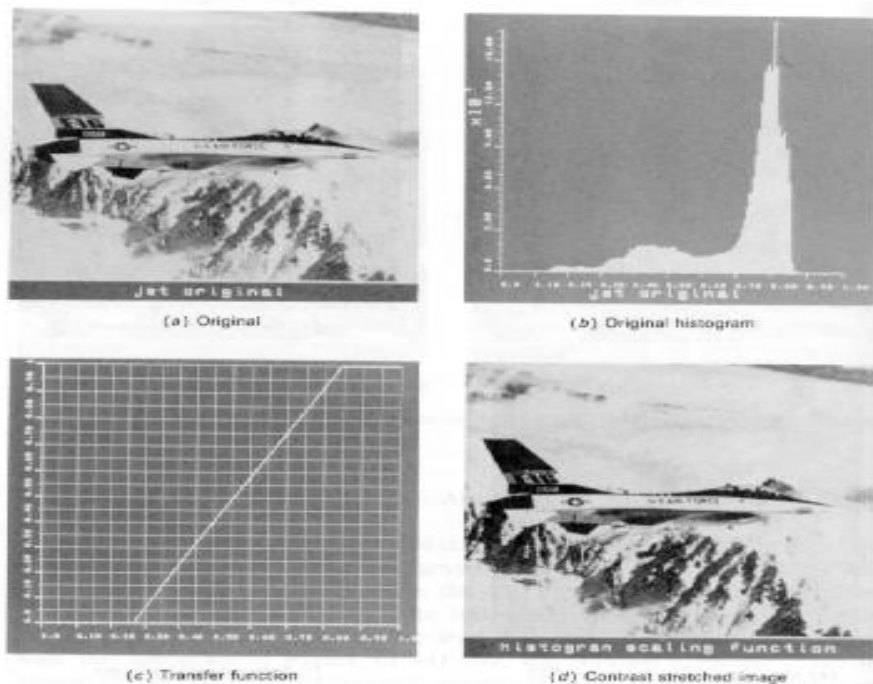
- Select intensity range  $r1$  to  $r2$  and map the range  $[r1, r2]$  to the range of  $[s1, s2]$  linearly



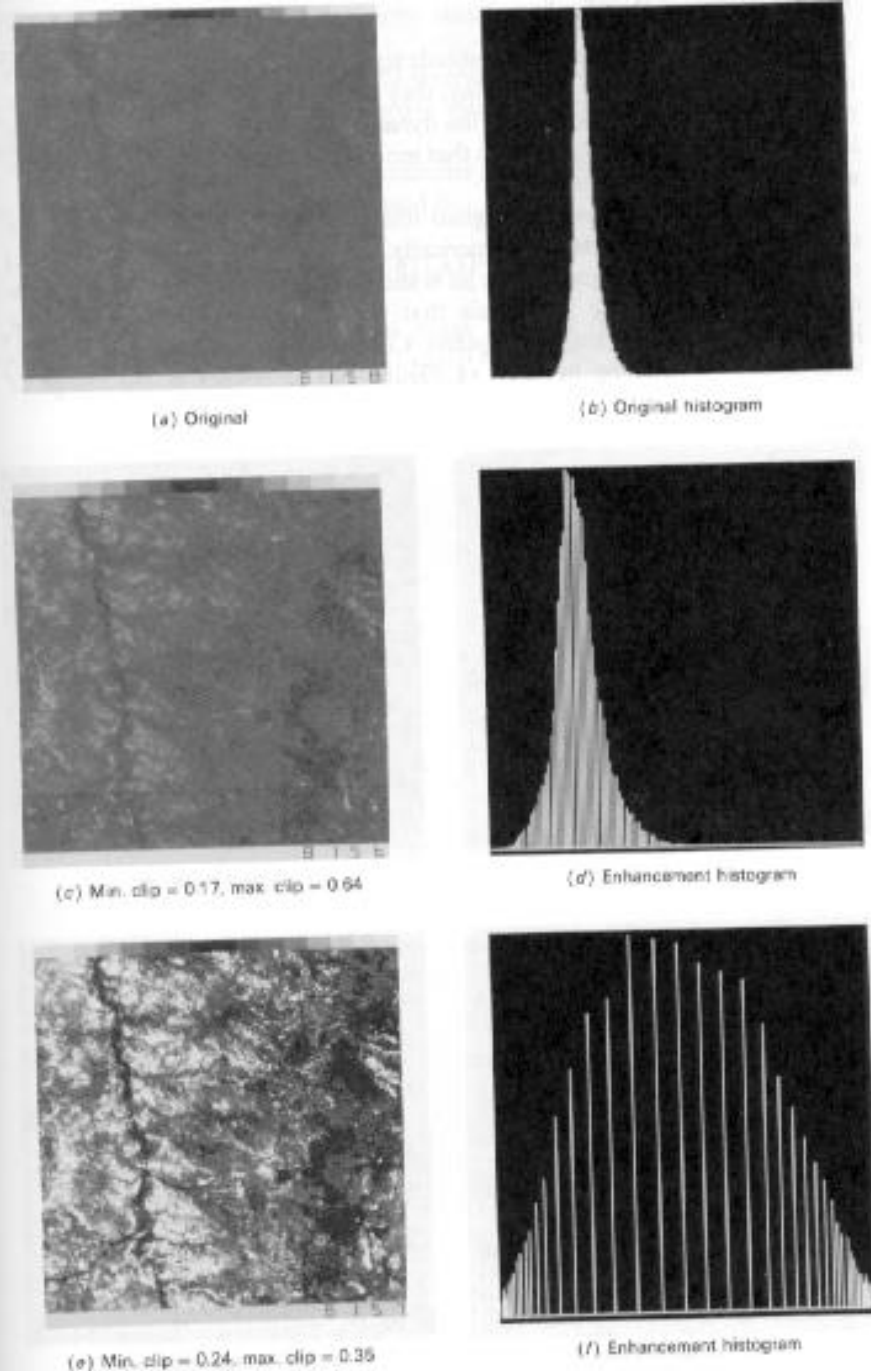
- $r1$  and  $r2$  can be selected by absolute values of greys or by skipping a percentage of pixels from low and high ends of the histogram (as in Fig 10-1-4 & -5).
- Define LUT using the step interval  $(r2-r1)/(s2-s1)^{38}$



**Figure 4.5** Contrast stretching: (a) form of transformation function; (b) a low-contrast image; (c) result of contrast stretching; (d) result of thresholding.

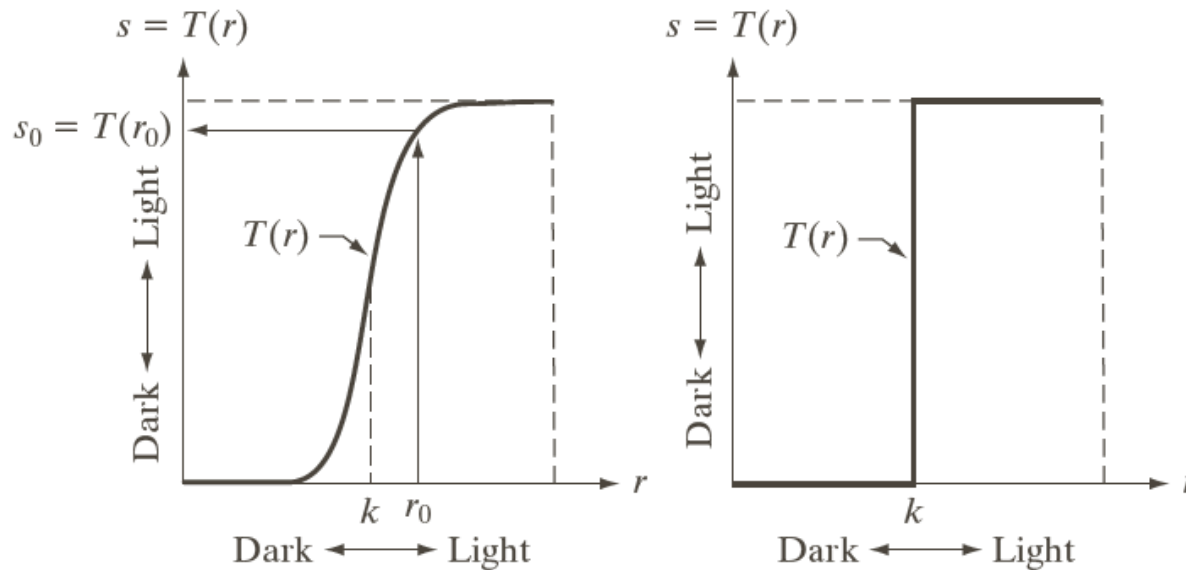


**FIGURE 10.1-5.** Example of window-level contrast stretching of the jet image.



**FIGURE 10.1-4.** Example of window-level contrast stretching of an Earth satellite image.

# Contrast-stretching:

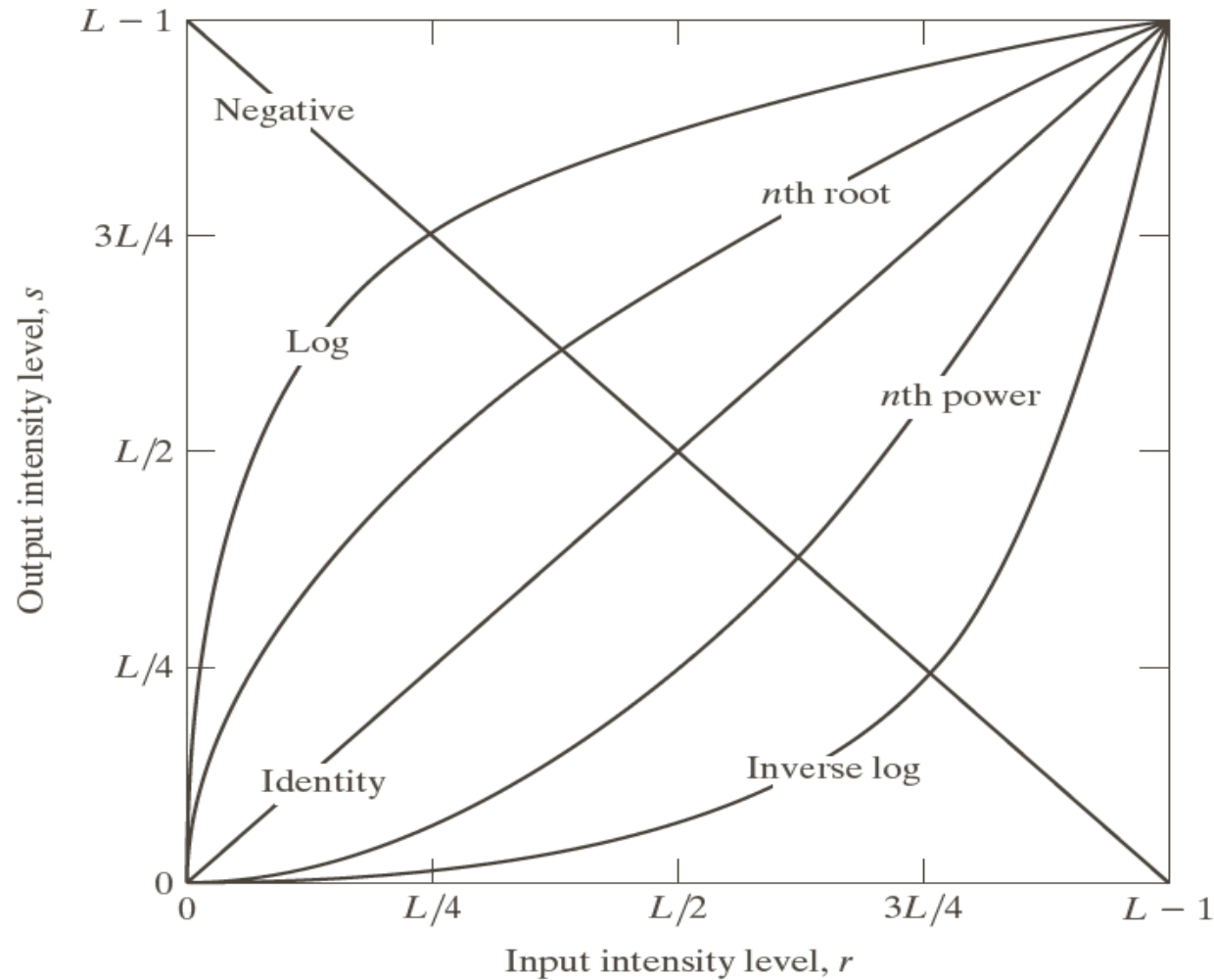


a b

**FIGURE 3.2**  
Intensity transformation functions.  
(a) Contrast-stretching function.  
(b) Thresholding function.



# Contrast-stretching:



**FIGURE 3.3** Some basic intensity transformation functions. All curves were scaled to fit in the range shown.

## Histogram equalization (in continuous domain):

- Let  $r$  be the gray levels normalized to  $[0,1]$
- Since  $s=T(r)$ ,  $T(r)$  is single-valued, monotonically increasing in  $0 \leq r \leq 1$  s.t.  $0 \leq T(r) \leq 1$  for  $0 \leq r \leq 1$ .
- The inverse transform is  $r=T^{-1}(s)$
- $T^{-1}(s)$  should also be single-valued, monotonically increasing and  $0 \leq T^{-1}(s) \leq 1$  for  $0 \leq s \leq 1$ .
- From probability theory, if original gray level pdf  $p_r(r)$  and  $T(r)$  are known &  $T^{-1}(s)$  satisfies the above condition, then the transformed gray level's pdf  $p_s(s)$  is

$$p_s(s) = \left[ p_r(r) \frac{dr}{ds} \right]_{r=T^{-1}(s)}$$

# Histogram equalization (in continuous domain):

- Consider the following input/output transformation

$$s = T(r) = \int_0^r p_r(w)dw \quad 0 \leq r \leq 1$$

- The above function is the **cumulative distribution function (cdf)** of  $r$ .
- cdf is single-valued and monotonically increasing
- Therefore, by differentiating the integration above

$$\frac{ds}{dr} = p_r(r) \Rightarrow \frac{dr}{ds} = \frac{1}{p_r(r)} \leftarrow \text{And combining with the eqn on the previous slide}$$

$$p_s(s) = \left[ p_r(r) \frac{1}{p_r(r)} \right]_{r=T^{-1}(s)}$$

$$\therefore p_s(s) = 1 \quad 0 \leq s \leq 1$$

## Histogram equalization (in discrete domain):

- Therefore the histogram after equalization is flat for the continuous formulation.

- Thus, in discrete form,

$$p(r_k) = \frac{n_k}{n} \quad 0 \leq r_k \leq 1 \quad \text{and } k = 0, 1, \dots, L-1$$

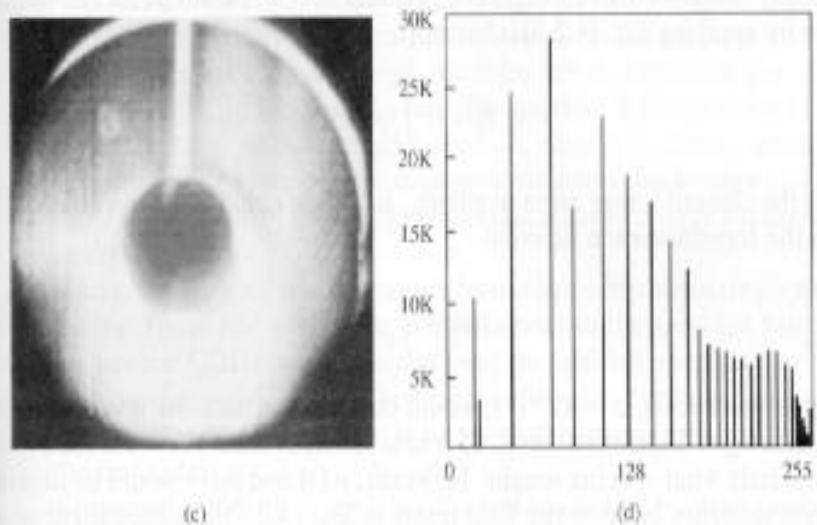
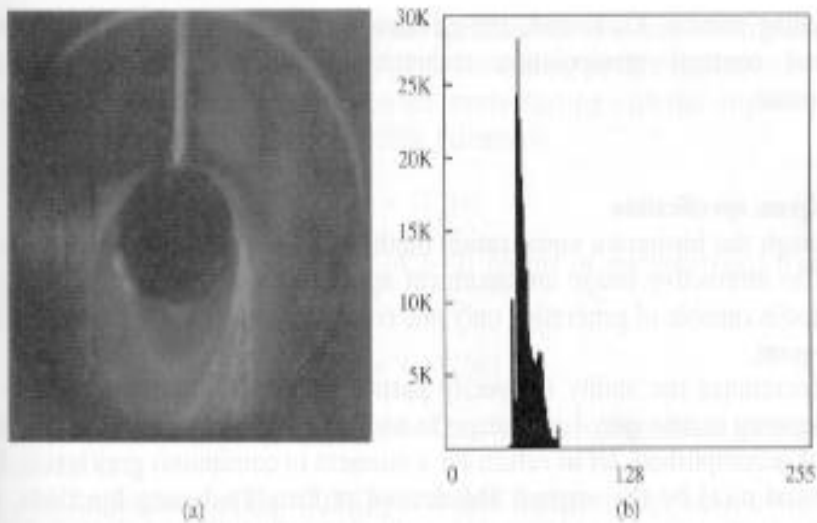
$$s_k = T(r_k) = \sum_{j=0}^k \frac{n_j}{n} = \sum_{j=0}^k p_r(r_j) \quad 0 \leq r_k \leq 1 \text{ and } k = 0, 1, \dots, L-1$$

Would the discrete formulation yield a flat histogram? **NO.**

## Histogram equalization in C:

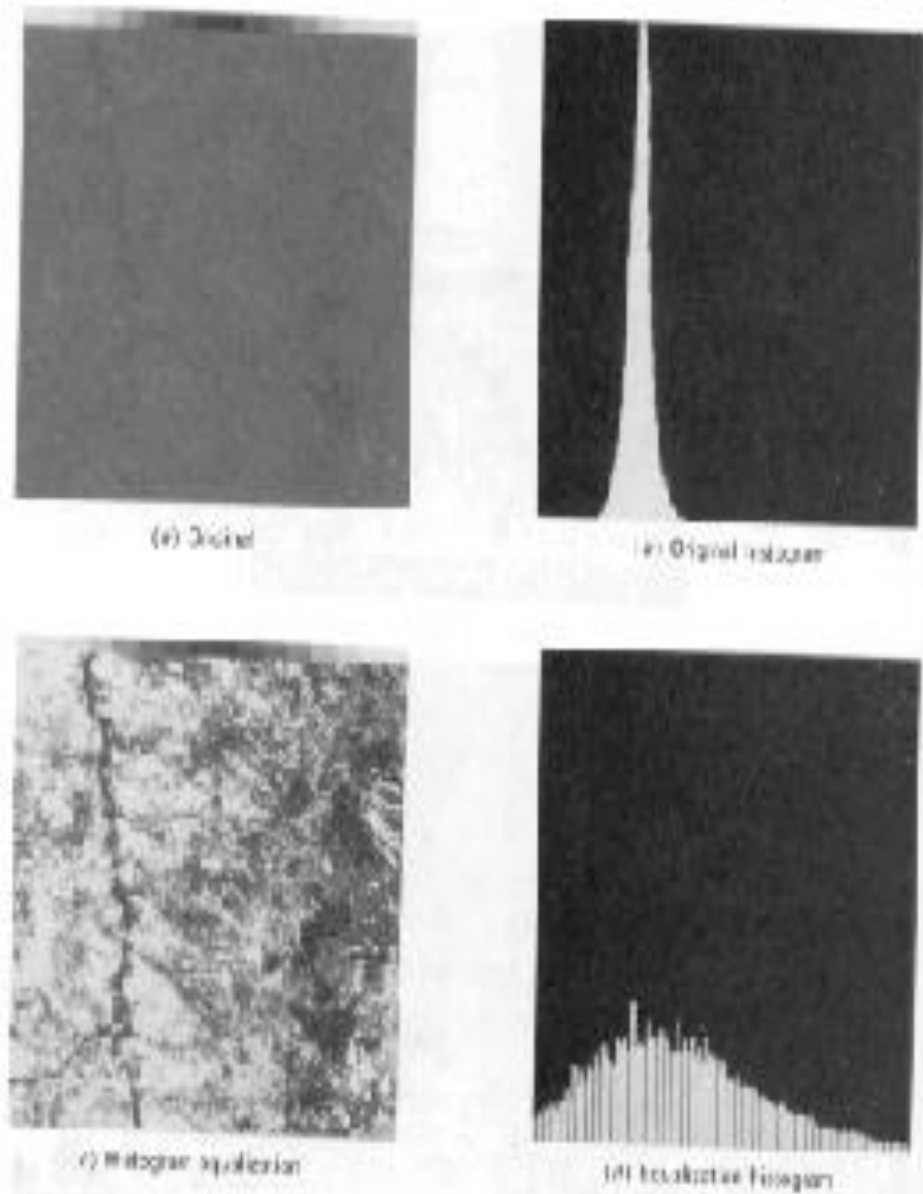
```
void histogram_equalise (float hist[], int LUT[])
{
    int i, j;  float cdf[256];

    cdf[0] = hist[0];    \\      normalized histogram
    for (i = 1; i < 256; i++)
        cdf[i] = hist[i] + cdf[i-1];
    for (i = 0; i < 256; i++)
        LUT[i] = (int)(cdf[i]*255.0);
}
```



**Figure 4.13** (a) Original image and (b) its histogram; (c) image subjected to histogram equalization and (d) its histogram.

Standard histogram  
equalization.



Histogram equalization  
with pixel re-distribution.

## 5. Image Filtering

- Image filtering can be performed in spatial domain or in spatial frequency domain.
- A filter with desired spatial frequency response is designed to enhance the corresponding spatial frequency components in the image.
- The filter can be designed in the spatial domain or the spatial frequency domain.
- Image filtering in spatial domain is achieved by convolution.
- Image filtering in spatial frequency domain is achieved by spectral multiplication.

## Fourier Transform (FT) (Based on G & W)

- The two dimensional FT and inverse FT are:

$$F(u, v) = \mathfrak{F}\{f(x, y)\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) \exp[-j2\pi(ux + vy)] dx dy$$

$$f(x, y) = \mathfrak{F}^{-1}\{F(u, v)\} = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} F(u, v) \exp[j2\pi(ux + vy)] du dv$$

- $u$  and  $v$  are frequency variables.
- The **magnitude**, **phase** and **power** of the spatial frequency spectrum are

$$|F(u, v)| = [R^2(u, v) + I^2(u, v)]^{1/2}$$

$$\phi(u, v) = \tan^{-1}[I(u, v) / R(u, v)]$$

$$P(u, v) = |F(u, v)|^2$$

where  $R(u, v)$  is the real part and  $I(u, v)$  is the imaginary part.



## Discrete Fourier Transform (DFT)

- The 2 dimensional DFT and inverse DFT are for non-square image:

$$F(u, v) = \frac{1}{MN} \sum_{x=0}^{M-1} \sum_{y=0}^{N-1} f(x, y) \exp[-j2\pi(ux/M + vy/N)]$$

$$f(x, y) = \sum_{u=0}^{M-1} \sum_{v=0}^{N-1} F(u, v) \exp[j2\pi(ux/M + vy/N)]$$

*where  $N$  and  $M$  are the dimensions of the 2D matrix*

$$\Delta u = 1/(M\Delta x); \quad \Delta v = 1/(N\Delta y) \quad P(u, v) = |F(u, v)|^2$$

- For square image:

$$F(u, v) = \frac{1}{N} \sum_{x=0}^{N-1} \sum_{y=0}^{N-1} f(x, y) \exp\left[-\frac{j2\pi(ux + vy)}{N}\right]$$

$$f(x, y) = \frac{1}{N} \sum_{u=0}^{N-1} \sum_{v=0}^{N-1} F(u, v) \exp\left[\frac{j2\pi(ux + vy)}{N}\right]$$

# Convolution and Correlation

- The 1D convolution is :

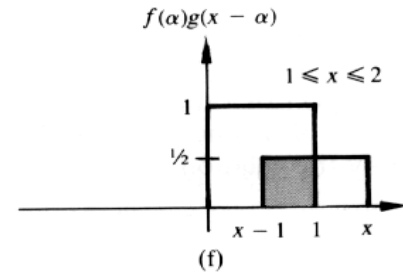
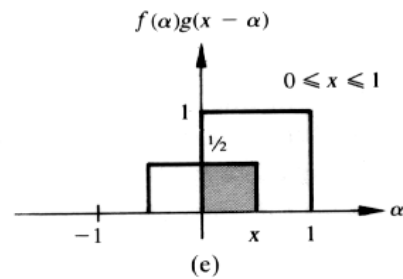
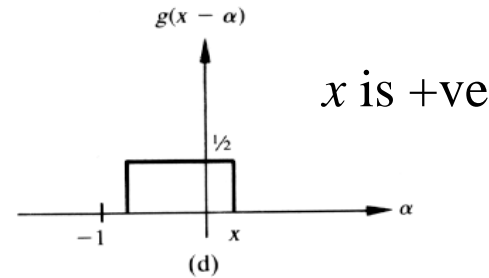
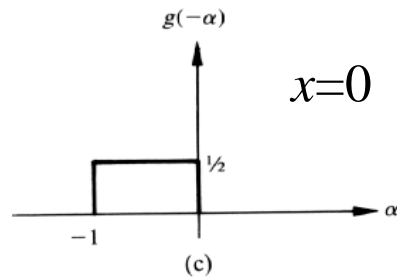
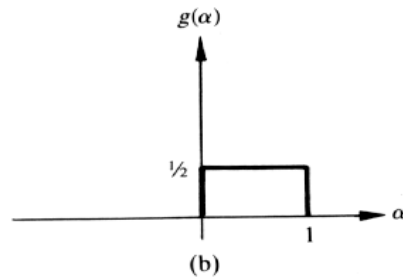
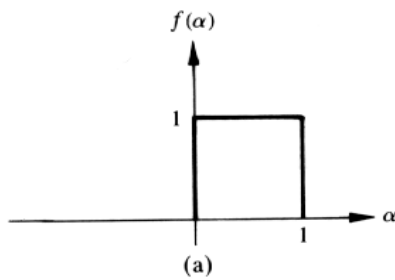
$$f(x) \otimes g(x) = \int_{-\infty}^{\infty} f(\alpha) g(x - \alpha) d\alpha$$

- Convolution in time / spatial domain is equivalent to multiplication in frequency domain **and vice versa**.

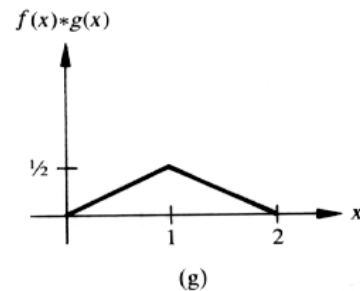
$$f(x) \otimes g(x) \Leftrightarrow F(u)G(u)$$

$$f(x)g(x) \Leftrightarrow F(u) \otimes G(u)$$

# Graphical illustration of Convolution



In shaded areas the product is Non-zero.



## Convolution and Correlation

- The 2D convolution is :

$$f(x, y) \otimes g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(\alpha, \beta) g(x - \alpha, y - \beta) d\alpha d\beta$$

- Convolution in spatial domain is equivalent to multiplication in spatial frequency domain **and vice versa**.

$$f(x, y) \otimes g(x, y) \Leftrightarrow F(u, v) G(u, v)$$

$$f(x, y) g(x, y) \Leftrightarrow F(u, v) \otimes G(u, v)$$

# Convolution and Correlation

- The 1D correlation is :

$$f(x) \circ g(x) = \int_{-\infty}^{\infty} f^*(\alpha) g(x + \alpha) d\alpha$$

- Correlation in time domain is equivalent to multiplication in frequency domain **and vice versa** as follows.

$$f(x) \circ g(x) \Leftrightarrow F^*(u)G(u)$$

$$f^*(x)g(x) \Leftrightarrow F(u) \circ G(u)$$

## Convolution and Correlation

- The 2D correlation is :

$$f(x, y) \circ g(x, y) = \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f^*(\alpha, \beta) g(x + \alpha, y + \beta) d\alpha d\beta$$

- Correlation in spatial domain is equivalent to multiplication in spatial frequency domain **and vice versa**:

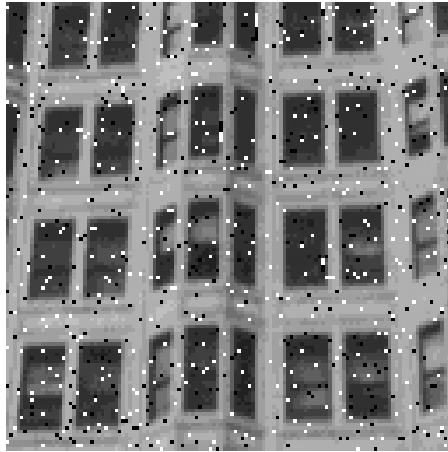
$$f(x, y) \circ g(x, y) \Leftrightarrow F^*(u, v) G(u, v)$$

$$f^*(x, y) g(x, y) \Leftrightarrow F(u, v) \circ G(u, v)$$

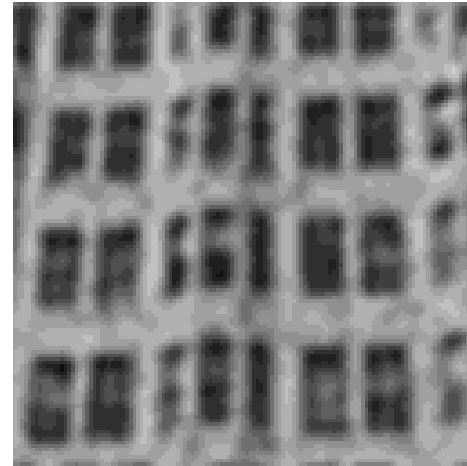
# Spatial Frequency Filtering

- Filtering an image involves the following steps:
  - Transform image into spatial frequency domain using FT.
  - Filter the spatial frequency domain with pre-designed filters.
  - Transform from spatial frequency domain into image domain using Inverse FT.

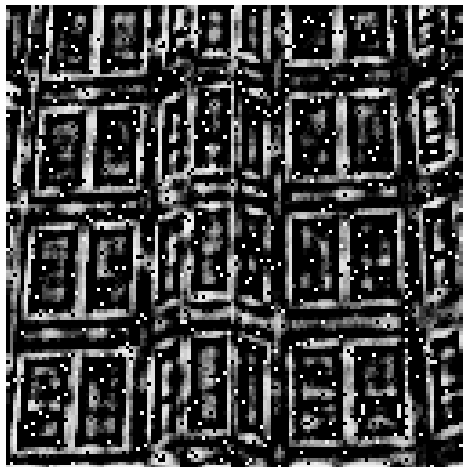
# Image Filtering



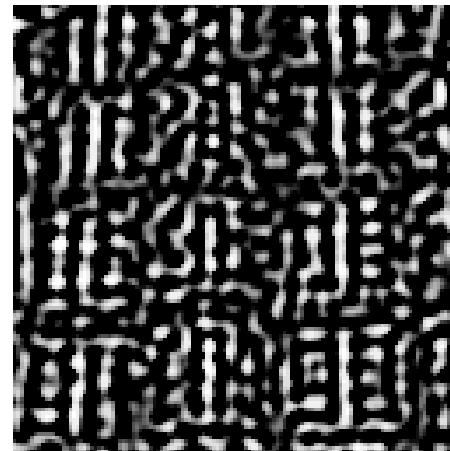
original



Low-passed



High-passed



Band-passed



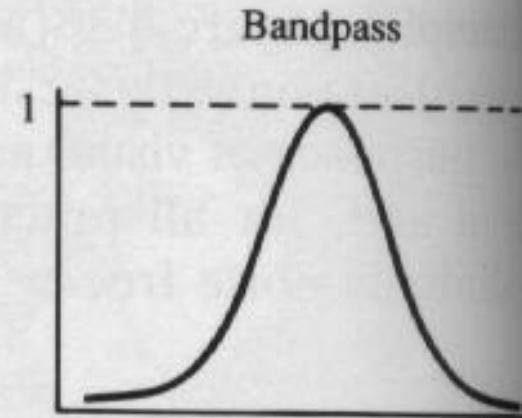
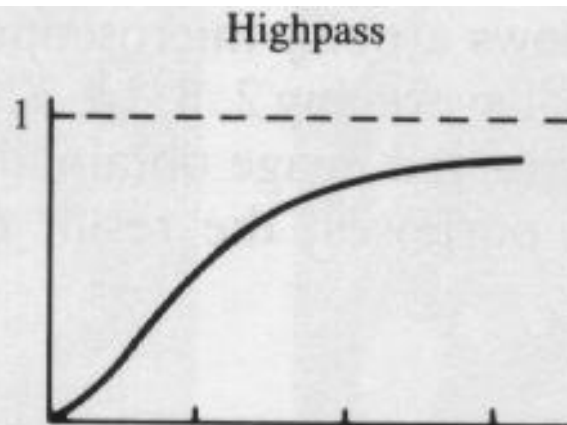
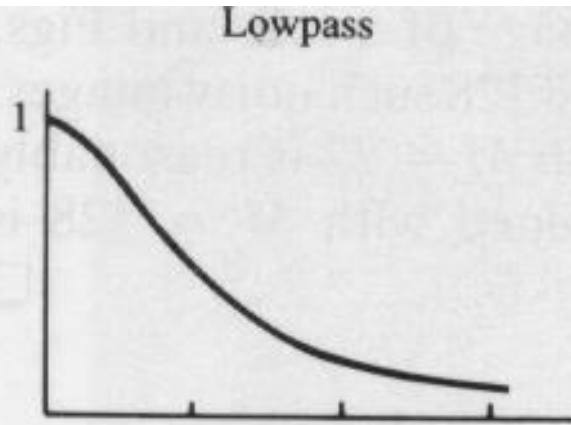
# Band-pass filtering

- Band-pass filtering enhances edges and suppresses image noise (in frequency domain).
- Equivalent band-pass action in spatial domain (more details later):
  - Gaussian + First Derivative Operator :
    - First derivative of Gaussian
  - Gaussian + Second Derivative Operator :
    - Laplacian of Gaussian

## 6. Neighborhood Operations (Spatial Domain)

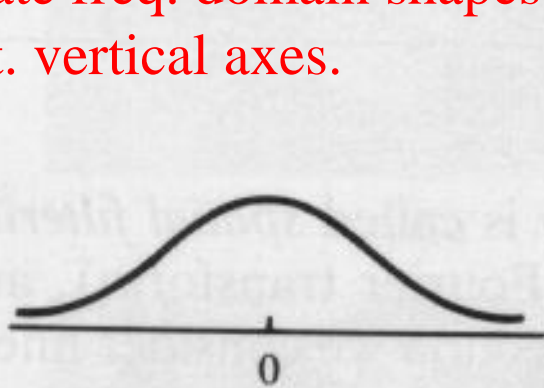
- Most neighborhood operations can be considered as image filtering by convolution with a convolution kernel
- The operation is performed in spatial domain.
- Low-pass, high-pass and band-pass filters are designed and used in spatial domain.
- Block averaging is considered as a low-pass spatial filter
- Weighted averaging uses a Gaussian kernel.

# Equivalent **Cross-Sectional shapes** in spatial and frequency domains



## Frequency domain

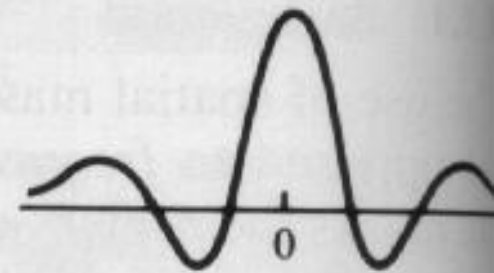
Rotate freq. domain shapes by 360 degrees and spatial domain shapes by 180 w.r.t. vertical axes.



(a)



(b)



(c)

## Spatial domain

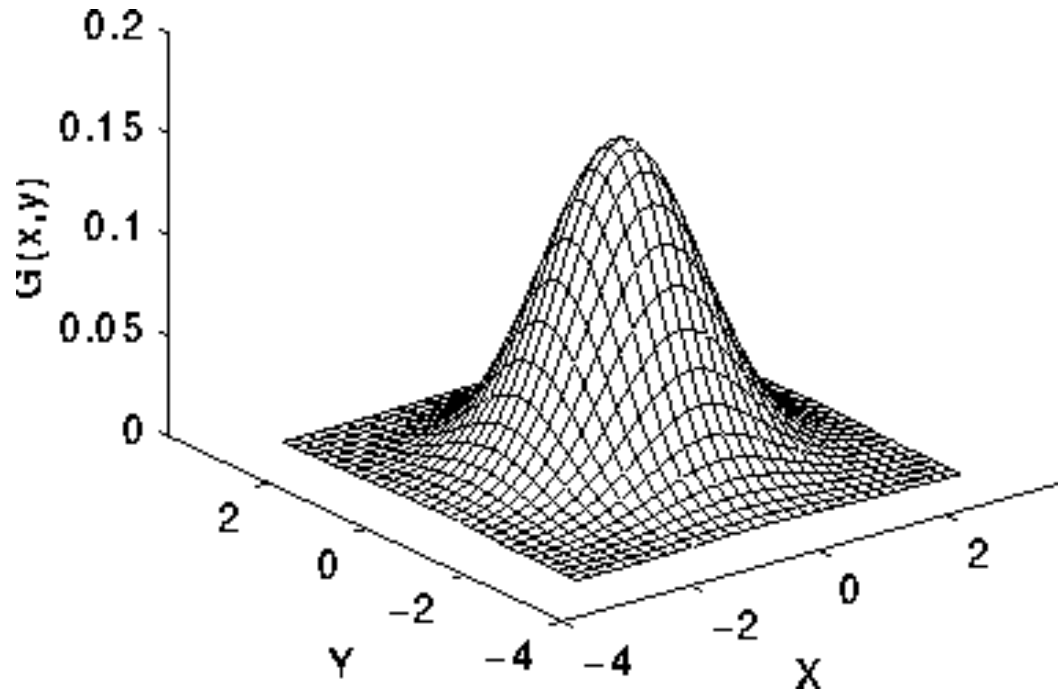
**Circular symmetry frequency domain filters** and their spatial domain counter parts. The pictures show cross-sectional profiles.

## Gaussian Smoothing

- The Gaussian smoothing operator is a 2-D convolution operator that is used to 'blur' images and remove noise (and details).
- Although the objective is removing noise, unwanted side effects are blurring and removal of important information.
- The Gaussian function in 2-D has the form:

$$G(x, y) = \frac{1}{2\pi\sigma^2} e^{-\frac{x^2 + y^2}{2\sigma^2}}$$

## 2-D Gaussian function with mean (0,0) and $\sigma = 1$



The idea of Gaussian smoothing is to use this 2-D distribution as a 'point-spread' function, and this is achieved by convolution

## Discrete approximation to Gaussian function with $\sigma=1.4$

$$\text{Gaussian} = \frac{1}{159}$$

2	4	5	4	2
4	9	12	9	4
5	12	15	12	5
4	9	12	9	4
2	4	5	4	2

Origin  
x=0  
y=0

```
convolve(img, oimg, xsize, ysize, gaussian,5,5,115.0);
```

# Implementation of convolution

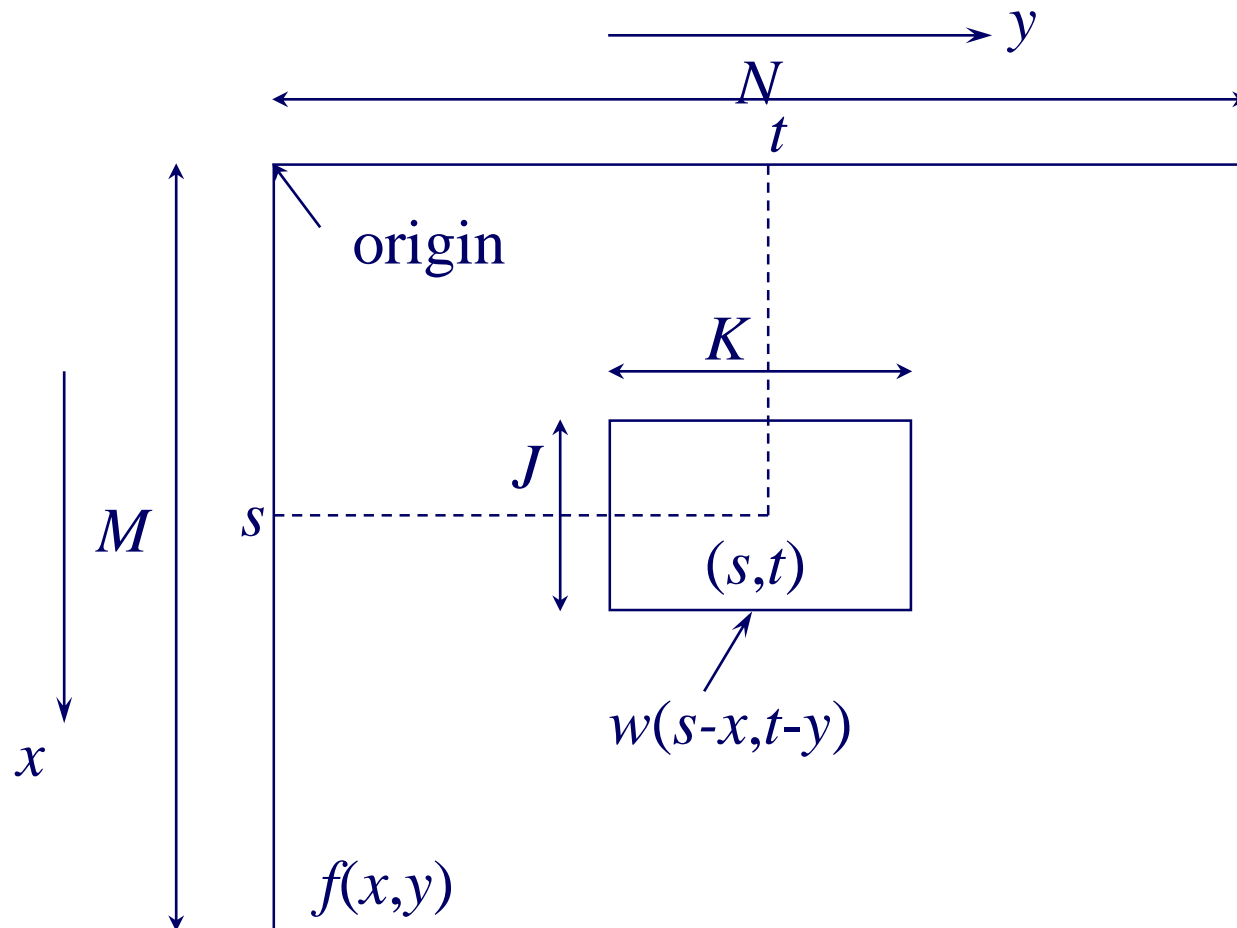
- One of the input arrays is a gray level image.
- The second array is convolution kernel.
- The convolution is performed by sliding the kernel over the image row by row, starting at the top left corner.
- For an image of  $M \times N$ , and a kernel of  $J \times K$  the convolution at an image point  $(s,t)$  is :

$$c(s,t) = \sum_x \sum_y w(x,y) f(s-x, t-y) \quad \leftarrow \text{(Used in the Example 2 slides in front)}$$
$$c(s,t) = \sum_x \sum_y w(s-x, t-y) f(x,y)$$

where  $s = 0, 1, 2, \dots, M-1$ ,  $t = 0, 1, 2, \dots, N-1$ .

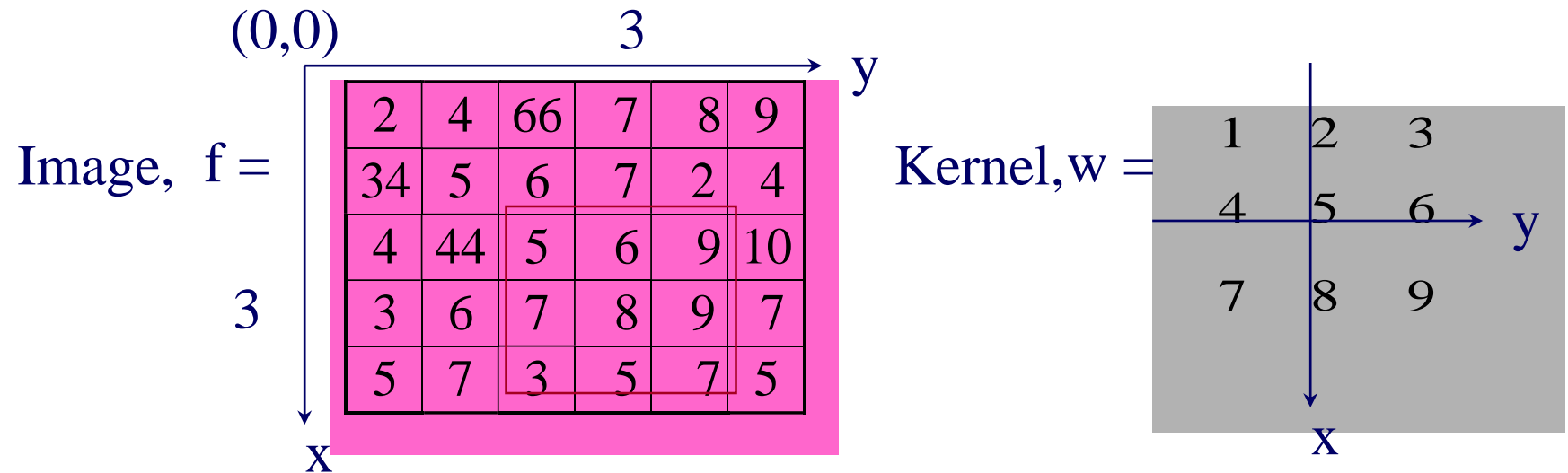
- summation is taken over the image region where  $w$  and  $f$  overlap.

# Illustration of Convolution





## Numerical example illustrating convolution:



Note that  $x$  &  $y$  are used within image and kernel

e.g. convolve at (3,3) image coordinates:  $x = s = 3, y = t = 3$ .

$x, y$  range of  $3 \times 3$  kernel :  $-1 \leq x, y \leq 1$  (USED IN SUMMATION)

centre of kernel moved to  $f(3,3)$  then  $w(0,0) = w(3-3, 3-3)$ ;

overlapped region within the image  $f$ :  $2 \leq x \leq 4, 2 \leq y \leq 4$  ;

$$c(3,3) = (7 \cdot \underline{1} + 5 \cdot \underline{2} + 3 \cdot \underline{3} + 9 \cdot \underline{4} + 8 \cdot \underline{5} + 7 \cdot \underline{6} + 9 \cdot \underline{7} + 6 \cdot \underline{8} + 5 \cdot \underline{9})$$

## Convolution in C:

```
void convolve (BYTE *in_image, BYTE *out_image, int xsize, int
ysize, int *mask, int mxs, int mys, double normalizer)
{
    int i,j,k,l,sum,halfwx,halfwy,mwxy;

    halfwx = mxs/2;  halfwy = mys/2;  mwxy = mxs*mys -1;
    for (i = halfwy; i < ysize - halfwy; i++)
        for (j = halfwx; j < xsize - halfwx; j++) {
            sum = 0;
            for (k = 0; k < mys; k++)
                for (l = 0; l < mxs; l++)
                    sum += (int)in_image[(i+k-halfwy)*
                    xsize+j+l-halfwx]*(int)mask[mwxy-k*mxs - l];
            sum = (int)fabs(sum/normalizer);
            out_image[i*xsize+j] = (sum > 255) ? 255 : (BYTE)sum;
        }
}
```

Decompose 2D Gaussian into two separate 1D filters :

=> **Significant reduction in computation cost**

$$\begin{aligned} G(x, y) &= \frac{1}{2\pi\sigma^2} e^{-\frac{x^2+y^2}{2\sigma^2}} = \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \cdot \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} \\ &= \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} \delta(y) * \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{y^2}{2\sigma^2}} \delta(x) \end{aligned}$$

$$G(x, y) = g_x * g_y$$

where  $*$  is the convolution operator

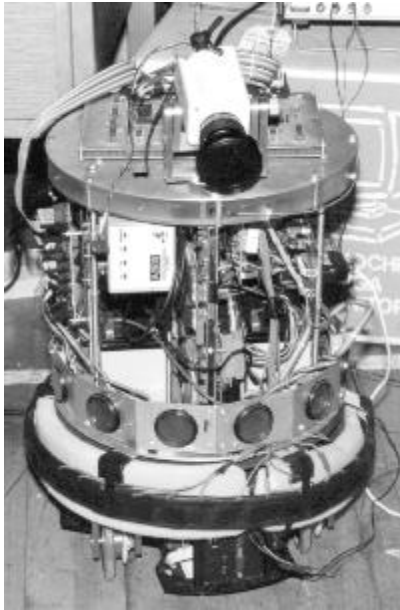
$$\therefore I(x, y) * G(x, y) = I(x, y) * g_x * g_y = [I(x, y) * g_x] * g_y$$

- 2-D Gaussian is separable into  $x$  and  $y$  components.
- Thus 2-D convolution can be performed by first convolving with a 1-D Gaussian in the  $x$  direction, and then convolving with another 1-D Gaussian in the  $y$  direction.
- Therefore,

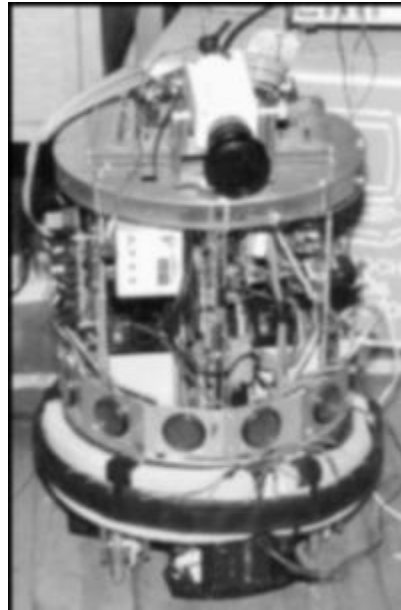
$$\text{Gaussian} = \frac{1}{12} \begin{array}{|c|c|c|c|c|} \hline 1 & 3 & 4 & 3 & 1 \\ \hline \end{array}$$

```
convolve(img, tmp, xsize, ysize, gaussian, 1, 5, 10.7);  
convolve(tmp, oimg, xsize, ysize, gaussian, 5, 1, 10.7);
```

## Examples of Gaussian Smoothing with different $\sigma$



original  
image



$\sigma = 1.0$   
( $5 \times 5$  kernel)



$\sigma = 2.0$   
( $9 \times 9$  kernel)



$\sigma = 4.0$   
( $15 \times 15$  kernel)

# Attractive Properties of Gaussian Smoothing

- Rotationally symmetric. Good as we want to smooth the image to the same degree in all directions.
- It has a single lobe. This implies that a weighted average is used with decreasing weights for pixels away from the centre pixel.
- The degree of smoothing is controlled by just a single parameter, sigma. Large sigma means a higher degree of smoothing
- Gaussian filter is separable in  $x$  and  $y$  directions → significant computational savings.
- Freq. Response of the filter has Gaussian shape. Hence, very high freq. Components are eliminated, i.e. Noise cleaning.

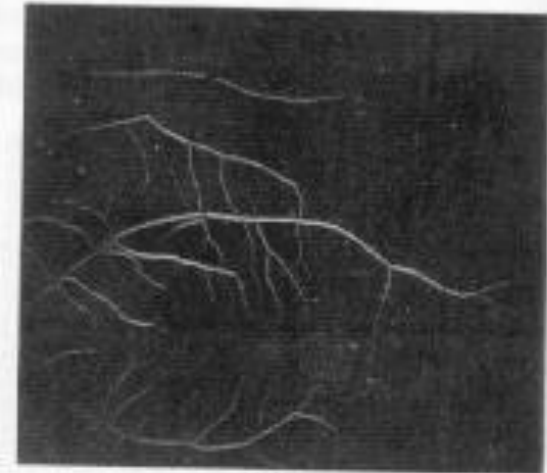
# High-pass spatial filter

- a high-pass spatial filter can be designed as follows.

-1	-1	-1
-1	8	-1
-1	-1	-1



(a)



(b)

*Figure 4.25 (a) Image of a human retina; (b) highpass filtered result using the mask in Fig. 4.24.*

{Refer to slide 59  
Showing Low, high  
band pass to see the  
spatial shape of high pass filter}

Image of human retina and its high pass filtered version

# High-boost filtering

High-pass = original - Lowpass

High boost =  $A(\text{original}) - \text{Lowpass}$

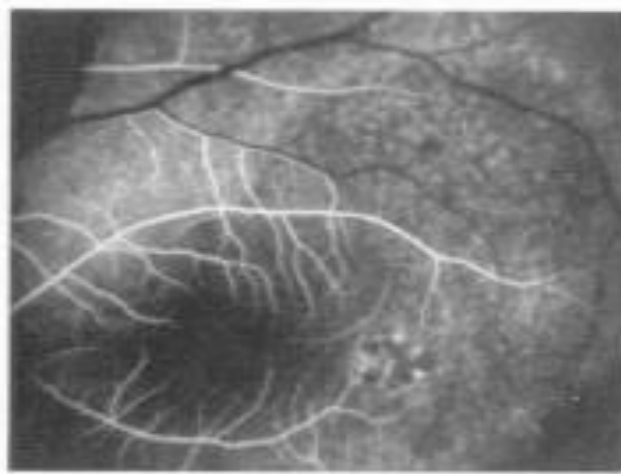
$= (A-1)\text{original} + \text{original} - \text{Lowpass}$

$= (A-1)\text{original} + \text{high-pass}$

$A=1$  yields highpass result

$A > 1$  restores partially the low frequency components lost in the highpass filtering, and the result is a high boosted image.

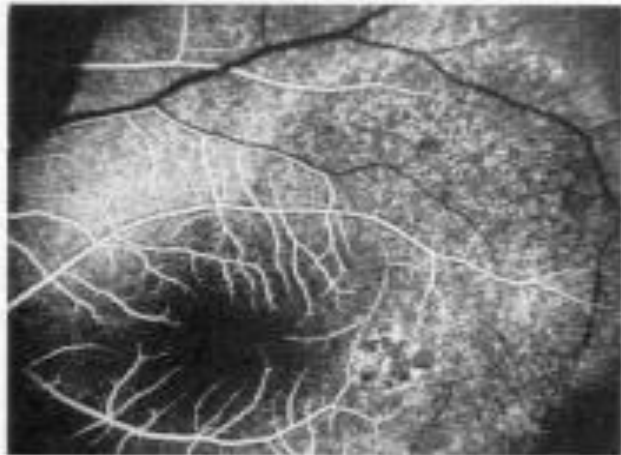




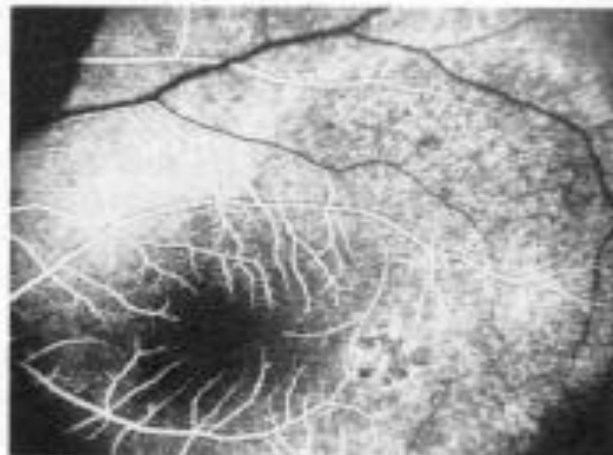
(a)



(b)



(c)



(d)

**Figure 4.27** (a) Original image; (b)–(d) result of high-boost filtering using the mask in Fig. 4.26, with  $A = 1.1$ ,  $1.15$ , and  $1.2$ , respectively. Compare these results with those shown in Fig. 4.25.

Original image & results of applying high boost filters with  $A = 1.1, 1.15$  &  $1.2$  respectively.

# Noise Removal

- Image noise can be thermal white noise or impulse noise.
- Lowpass filtering suppresses thermal noise well at the expense of image resolution, *i.e.* Causing image blur. E.g., Gaussian smoothing which we've already looked at.
- Temporal Averaging reduces image noise without loss of resolution (in the next few slides).
- Median filter reduces impulse noise without loss of resolution (in the next few slides).

# Temporal Averaging

- Consider that the noise is additive, has zero mean and is uncorrelated,

$$g(x, y) = f(x, y) + \eta(x, y)$$

- By taking average of M images,

$$\bar{g}(x, y) = \frac{1}{M} \sum_{i=1}^M g_i(x, y)$$

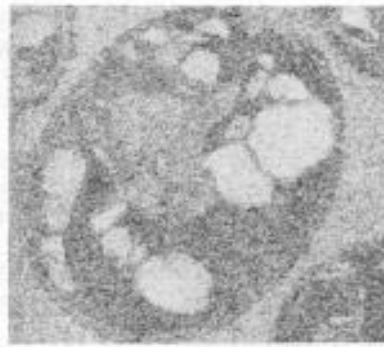
- then

$$E\{\bar{g}(x, y)\} = f(x, y)$$

$$\sigma_{\bar{g}(x, y)}^2 = \frac{1}{M} \sigma_{\eta(x, y)}^2$$

- Hence, noise level is reduced by taking more images.
- When can we apply temporal averaging?

Noisy  
image



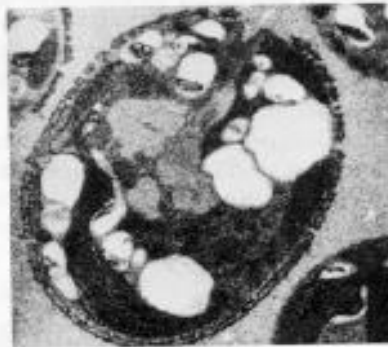
(a)



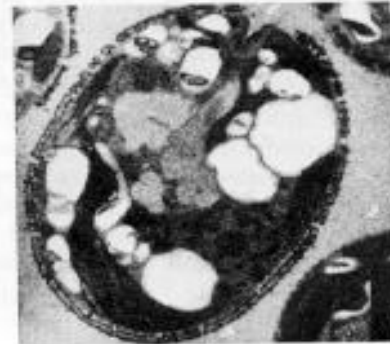
(b)

2 frames  
averaged

8 frames averaged



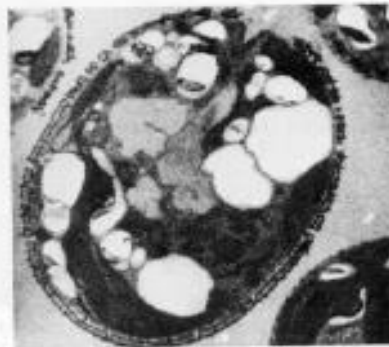
(c)



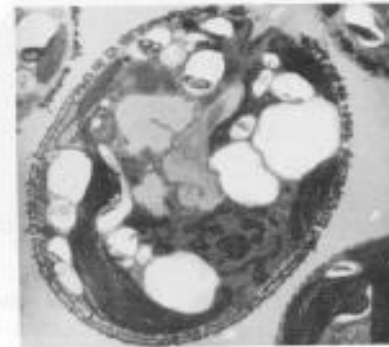
(d)

16 frames

32 frames



(e)



(f)

128 frames  
averaged

*Figure 4.18 Example of noise reduction by averaging: (a) a typical noisy image; (b)–(f) results of averaging 2, 8, 16, 32, and 128 noisy images.*

Noise reduction by averaging

## Rank Value filter (General Case)

- The output at  $\mathbf{x}=(x,y)^T$  is the relative rank of intensity of pixels in the neighborhood of  $\mathbf{x}$ .

Given a set of  $N$  pixel values obtained from a local neighborhood,  $s$ , denoted as  $f_i, i=1,2,3,\dots, N$ , an ordering of these values in ascending order of magnitude results in

$$R(\mathbf{x})=\{f_1, f_2, \dots, f_N\} \text{ where } f_i \leq f_{i+1}$$

$$g(\mathbf{x})= \text{Rank}_j R(\mathbf{x})$$

where  $\text{Rank}_j$  is the intensity of the output at position  $j$ .

$$j = 1, g(\mathbf{x})= \min R(\mathbf{x})$$

$$j = N, g(\mathbf{x})= \max R(\mathbf{x})$$

## Median filter (Special and Important Case)

- For  $N$  being odd, median  $m = (N+1)/2$   
 $\text{med}(R(\mathbf{x})) = \text{Rank}_m(R(\mathbf{x}))$
- The filter is non-linear  
 $\text{med}(R_1(\mathbf{x}) + R_2(\mathbf{x})) \neq \text{med}((R_1(\mathbf{x}) + \text{med}(R_2(\mathbf{x})))$
- median filter does not introduce new intensity values.
- Median filter preserves the location of edges.





(a) 3 x 3 median filter



(b) cascaded 3 x 3 median filter



(c) Original with uniform noise



(d) Original with impulse noise

FIGURE 10.3-1. Test images for noise cleaning evaluation of the peppers image.



(c) 5 x 5 median filter



(d) 7 x 7 median filter

FIGURE 10.3-14. Examples of median filtering on the peppers image with impulse noise.

## Median filtering with impulse noise

# 7. Image Segmentation

- Segmentation is generally used in image analysis.
- The objective of segmentation is to divide an image into meaningful regions such as object, background, outlines, etc.
- Segmentation should stop when the objects of interests have been isolated.
- Segmentation algorithms are based on either:
  - discontinuity of image values over a neighborhood.
    - E.g. isolated points, line, edges detection
  - Similarity of image values over a neighborhood.
    - E.g. region thresholding, region growing, split & merge



## Thresholding for segmentation

- Use threshold values to group pixels.

$$f(x,y) > T \text{ is object}$$

$$f(x,y) \leq T \text{ is background}$$

or

$$f(x,y) < T_1 \text{ is background}$$

$$T_1 \leq f(x,y) < T_2 \text{ object A}$$

$$f(x,y) > T_2 \text{ object B}$$

How to select threshold ? CONSIDER A BIMODAL  
HISTOGRAM (i.e. background & 1-object of interest)

## Threshold selection (Otsu's method, H & S Vol. 1 p. 20)

- Given an image of  $L$  gray levels and  $N$  pixels. Let  $n_i$  be the no. of pixels at the  $i$ th gray level,

$$N = \sum_{i=1}^L n_i \quad p_i = \frac{n_i}{N} \quad \sum_{i=1}^L p_i = 1$$

- Assume pixels can be grouped into  $C_0$  and  $C_1$ , for a threshold value of  $k$ ,

$$\omega_0 = P(C_0) = \sum_{i=1}^k p_i = \omega(k) \quad \omega_1 = P(C_1) = \sum_{i=k+1}^L p_i = 1 - \omega(k)$$

$$\mu_0 = \frac{\sum_{i=1}^k i P(i | C_0)}{\sum_{i=1}^k p_i} = \frac{\sum_{i=1}^k i p_i}{\omega_0} = \frac{\mu(k)}{\omega(k)} \quad (\text{Mean of group } C_0)$$

$$\mu_1 = \frac{\sum_{i=k+1}^L i P(i | C_1)}{\sum_{i=k+1}^L p_i} = \frac{\sum_{i=k+1}^L i p_i}{\omega_1} = \frac{\mu_T - \mu(k)}{1 - \omega(k)} \quad (\text{Mean of group } C_1)$$

$$\mu_T = \sum_{i=1}^L ip_i$$

$$\omega_0\mu_0 + \omega_1\mu_1 = \mu_T$$

$$\omega_0 + \omega_1 = 1$$

$$\sigma_0^2 = \sum_{i=1}^k (i - \mu_0)^2 P(i | C_0) = \sum_{i=1}^k \frac{(i - \mu_0)^2 p_i}{\omega_0}$$

$$\sigma_1^2 = \sum_{i=k+1}^L (i - \mu_1)^2 P(i | C_1) = \sum_{i=k+1}^L \frac{(i - \mu_1)^2 p_i}{\omega_1}$$

w - within; B - between; T -- total/whole image;  
subscript 0 - first cluster (either object or background)  
subscript 1 - 2nd cluster (either background or object )

subscripts,  $B$ ,  $w$ ,  $T$ ,  $0,1$  – defined  
in the previous slide

$$\sigma_w^2 = \omega_0 \sigma_0^2 + \omega_1 \sigma_1^2$$

$$\sigma_B^2 = \omega_0 (\mu_0 - \mu_T)^2 + \omega_1 (\mu_1 - \mu_T)^2 = \omega_0 \omega_1 (\mu_1 - \mu_0)^2$$

$$\sigma_T^2 = \sum_{i=1}^L (i - \mu_T)^2 p_i$$

- high separability can be obtained by either:
  - maximizing between-class variance,  $\sigma_B^2$
  - minimizing within-class variance,  $\sigma_w^2$

OR

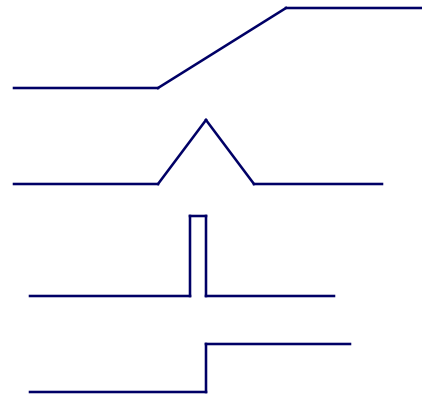
hence,

$$k^* = \arg \max_{1 \leq k < L} \{ \sigma_B^2(k) \}$$

# Edge Extraction

## Edge Models

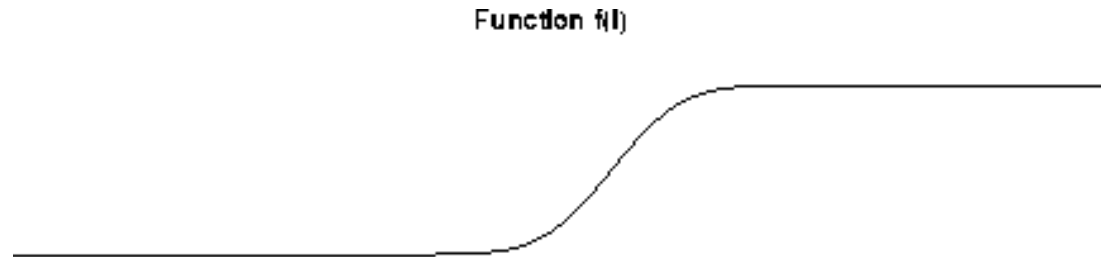
- Edges are places in the image with strong intensity contrast.
- Edges often occur at image locations representing object boundaries.
- Types of edges :
  - Ramp (1D profile)
  - Roof (1D profile)
  - Line (1D profile)
  - Step (1D profile)



# The Step Edge Model

- Step edge exist for artificially generated images.
- Real images **DO NOT** have step edges because anti-aliasing filter used in the imaging system.

Real edge →



1st derivative →

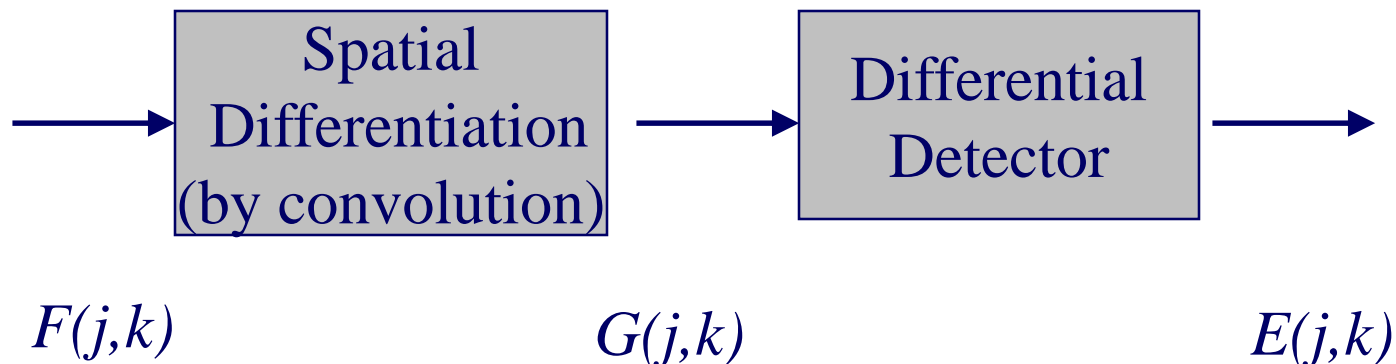


2nd derivative →



# The Step Edge Model

- Edge can therefore be located at
  - local gradient maximum in the 1st derivative ;
  - zero-crossing in the 2nd derivative.
- Edge can therefore be detected using differential operators.

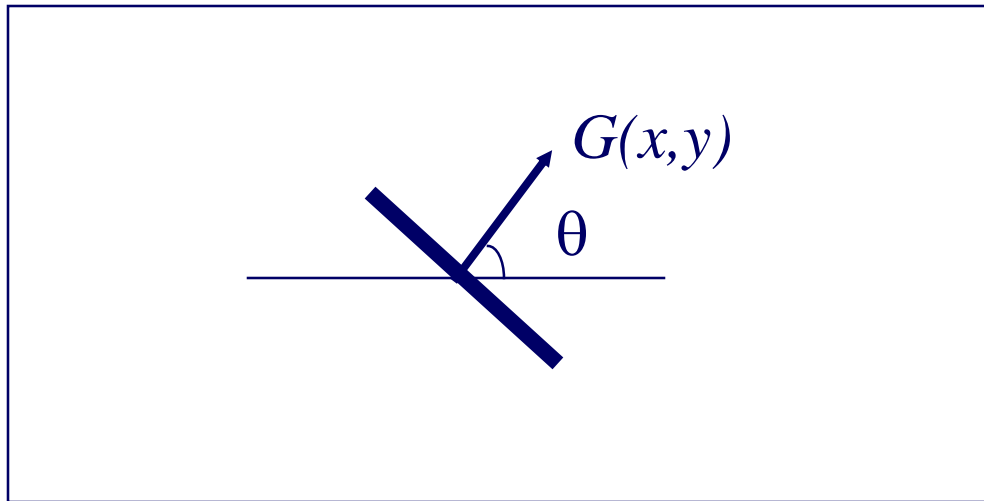


# First derivative operator

- Also known as Gradient operator.
- For an edge in a 2D image,

$$G(x, y) = \frac{\partial F(x, y)}{\partial x} \cos(\theta) + \frac{\partial F(x, y)}{\partial y} \sin(\theta)$$

where  $G(x, y)$  is the gradient normal to the edge.





# First derivative operator

– Usually, we compute

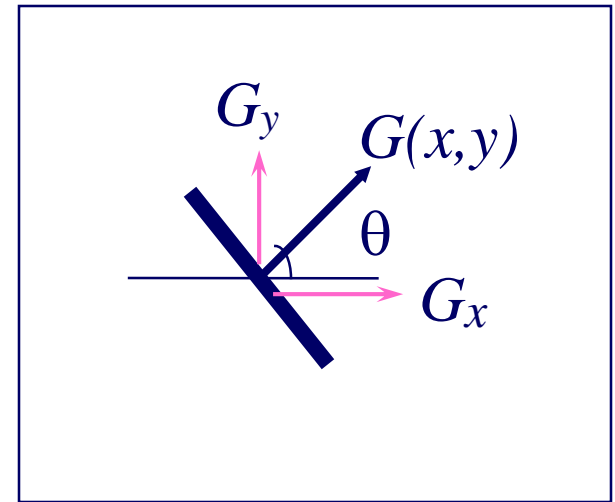
$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \partial f / \partial x \\ \partial f / \partial y \end{bmatrix}$$

where  $G_x$  is the horizontal gradient,

$G_y$  is the vertical gradient.

$$|\nabla f| = \left[ G_x^2 + G_y^2 \right]^{1/2}$$

$$|\nabla f| \approx |G_x| + |G_y| \quad ; \quad \theta(x, y) = \tan^{-1} \left( \frac{G_y}{G_x} \right)$$



# Sobel's gradient operator

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

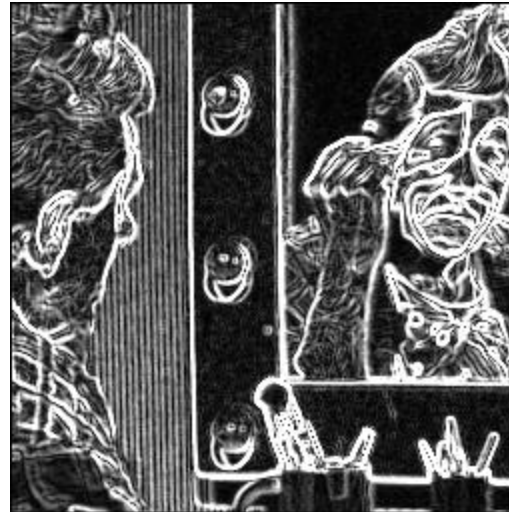
Gx

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

Gy



original



Convolved with  
Sobel's operator

# Gaussian smoothing + gradient operator

Gaussian smoothing is performed to suppress image noise before the image is differentiated.

$$\begin{aligned} G(x, y) &= \nabla[g(x, y) * I(x, y)] && \leftarrow \text{Convolution first} \\ &= \nabla g(x, y) * I(x, y) && \leftarrow \text{Differentiation first} \end{aligned}$$

- Therefore, edge can be detected by convolving the image with the first derivative of Gaussian, as the order of linear operations interchangeable.
- The 2D first derivative of Gaussian is also separable into 1D filters. (Steps are similar to those in slide 67)
- Combining Gaussian smoothing with differentiation reduces computation.
- By what factor?

## Second derivative operator

- Laplacian operator:
  - The Laplacian  $L(x,y)$  of an image at  $I(x,y)$  is:

$$L(x, y) = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

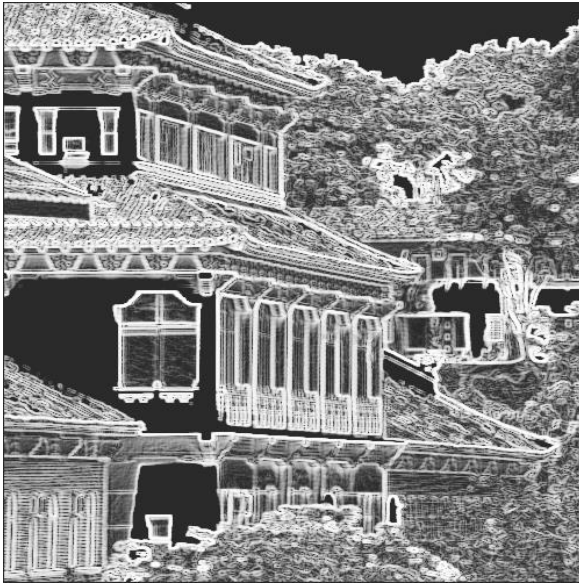
- Laplacian kernels: (basic kernel [1 -2 1])

0	1	0		1	1	1
1	-4	1		1	-8	1
0	1	0		1	1	1

- Laplacian operator usually produces closed contour



original



Sobel output



Laplacian output

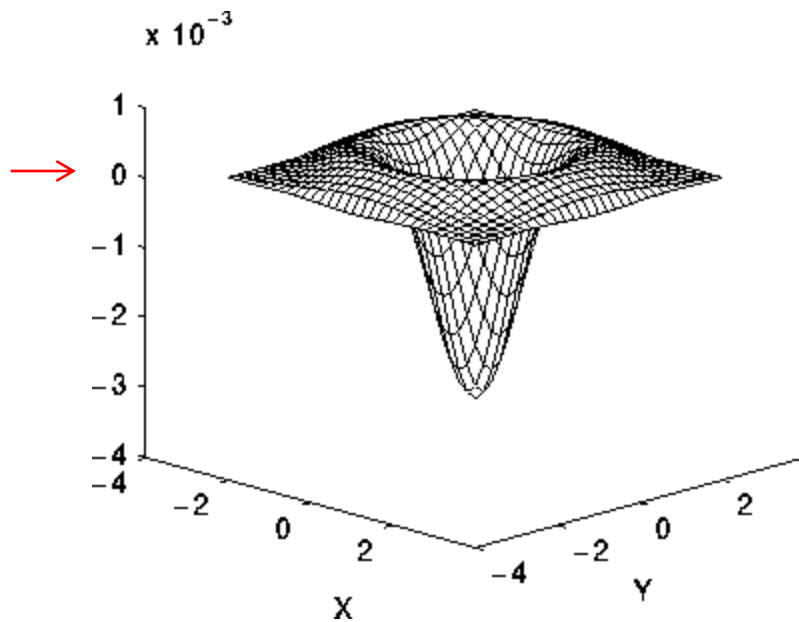
## Second derivative operator

- Laplacian of Gaussian (LoG) operator:
  - This is equivalent to Gaussian smoothing followed by Laplacian.

$$\begin{aligned} G(x, y) &= \nabla^2 [g(x, y) * I(x, y)] \\ &= \nabla^2 g(x, y) * I(x, y) \end{aligned}$$

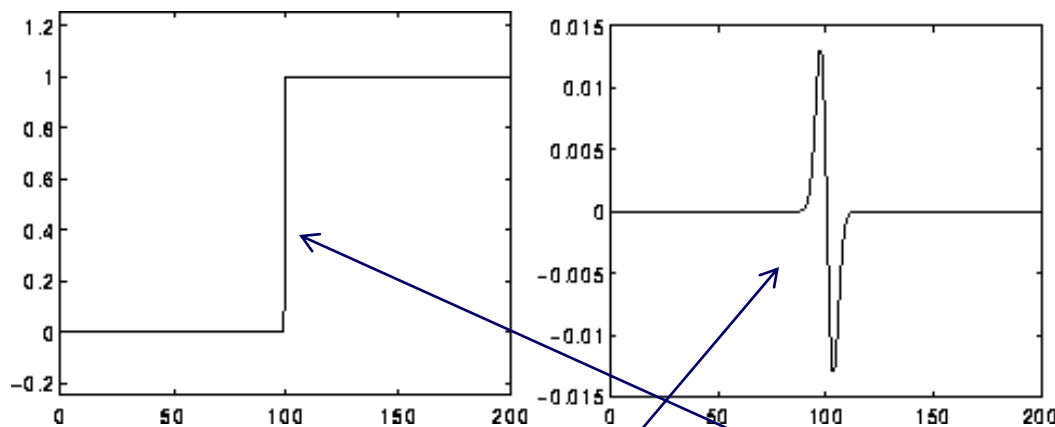
$$\nabla^2 g(x, y) = -\frac{1}{2\pi\sigma^4} \left( 2 - \frac{x^2 + y^2}{\sigma^2} \right) \exp \left[ -\frac{x^2 + y^2}{2\sigma^2} \right]$$

# Continuous 2D LoG function



0	0	9	2	2	2	9	0	0
0	2	9	5	5	5	9	2	0
9	9	5	9	0	9	5	9	9
2	5	9	-12	-29	-12	9	5	2
2	5	0	-29	-40	-29	0	5	2
2	5	9	-12	-29	-12	9	5	2
9	9	5	9	0	9	5	9	9
0	2	9	5	5	5	9	2	0
0	0	9	2	2	2	9	0	0

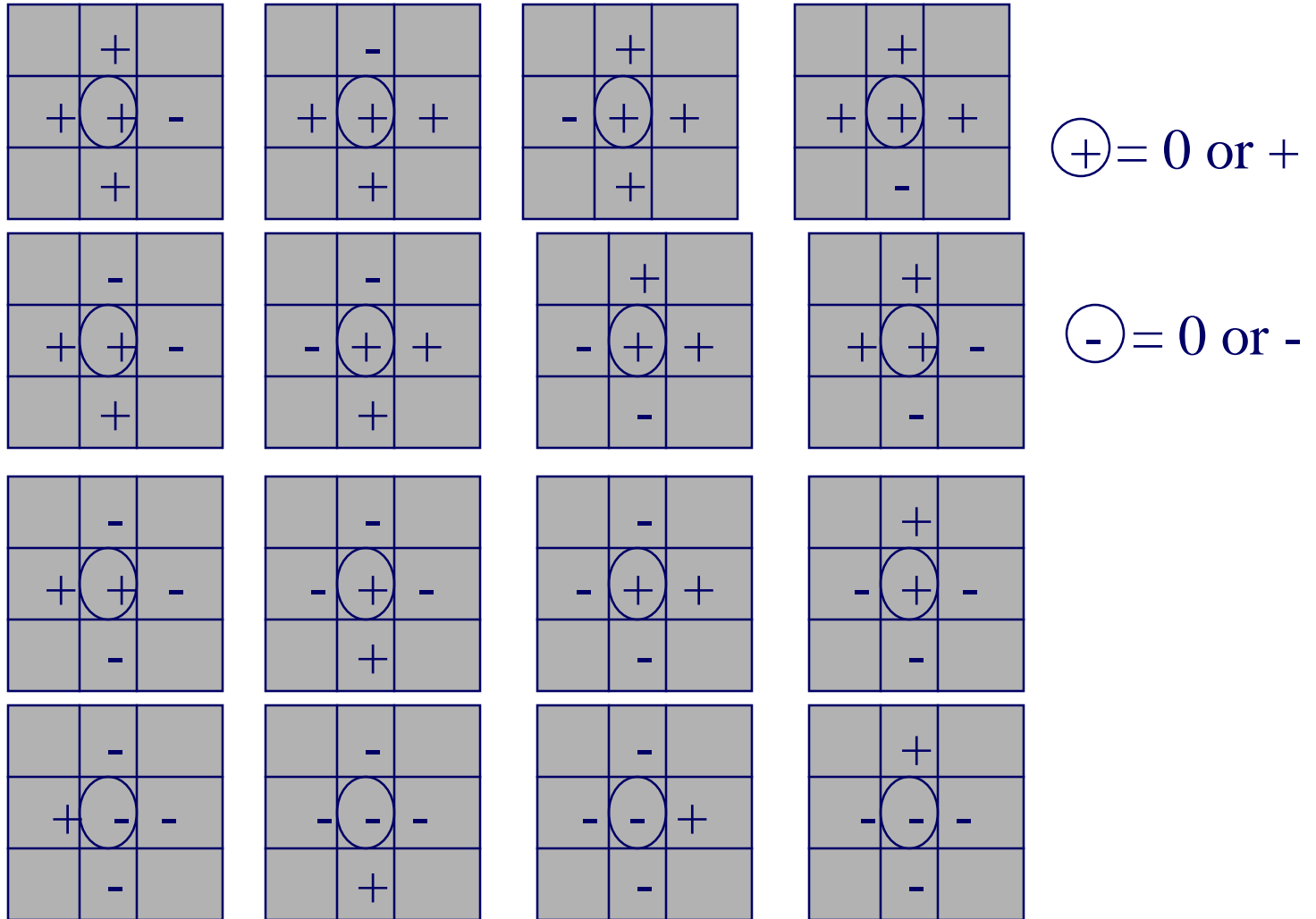
2D LoG kernel



Response of the LoG to a step edge

# Detection of zero-crossings:

- zero-crossing pattern



And more windows for  $\ominus$



# Canny Edge Detection Scheme

- The Canny operator was designed to be an optimal edge detector.
- The first derivative of Gaussian found to be a close approximation of his (Canny's) optimal edge detector.
- Edge is found at the local peak in the first derivative (which also corresponds to the zero-crossings in the second derivative.)
- Outline of the edge detection algorithm (**3 Main Steps**):
  - Convolve image with 1st derivative of Gaussian (already covered)
  - Suppress non-maxima gradients
  - Hysteresis Threshold with two thresholds

# Canny Edge Detection Scheme

- The non-maximal suppression algorithm tracks along the top of gradient ridges and sets to zero all pixels that are not actually on the ridge top so as to give a thin line in the output.
  - A neighborhood of 3 by 3 is used.
  - The gradient value at the center must be the largest in the direction perpendicular to the edge. If not the center pixel is removed.
- hysteresis is controlled by two thresholds:  $T_1$  and  $T_2$ , with  $T_1 > T_2$ . Tracking can only begin at a point on a ridge higher than  $T_1$ . Tracking then continues in both directions out from the start point until the height of the ridge falls below  $T_2$ .
- This hysteresis helps to ensure that noisy edges are not broken up into multiple edge fragments

- Sub-pixel accuracy for edge localization (before non-maximum suppression)
  - A neighborhood of 3 by 3 is defined around local maxima.
  - Linear interpolation of the gradients is carried out along the direction normal to the edge.
  - A quadratic function is fitted, and the position of the edge location can be found by finding the turning point of the function.



Original



Output of  
Canny Edge Detection

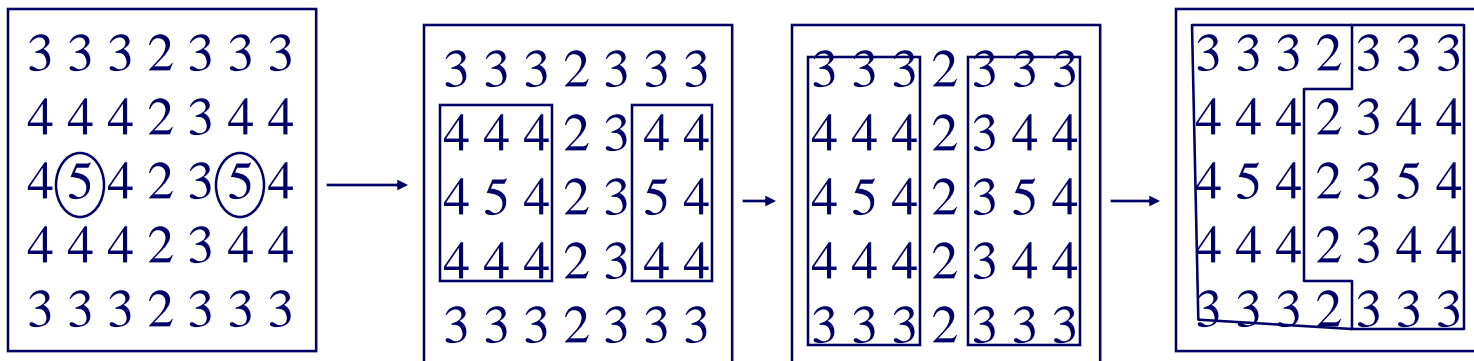
You can compare this result with that of Sobel edge detector  
(slide 91).

# Region Segmentation

- Let  $R$  be the entire image region.
- Segmentation aims to partition  $R$  into  $n$  sub-regions, such that
  - (a)  $R$  consists of all  $n$  subregions,  $\bigcup_{i=1}^n R_i = R$  , so that segmentation is complete.
  - (b)  $R_i$  is a connected region,  $i=1,2, \dots ,n$ . Points in a region are connected.
  - (c) Regions do not overlap,  $R_i \cap R_j = \phi$
  - (d) points in the same segmented region must have the similar intensity/property
  - (e) Different regions have different intensity/property.

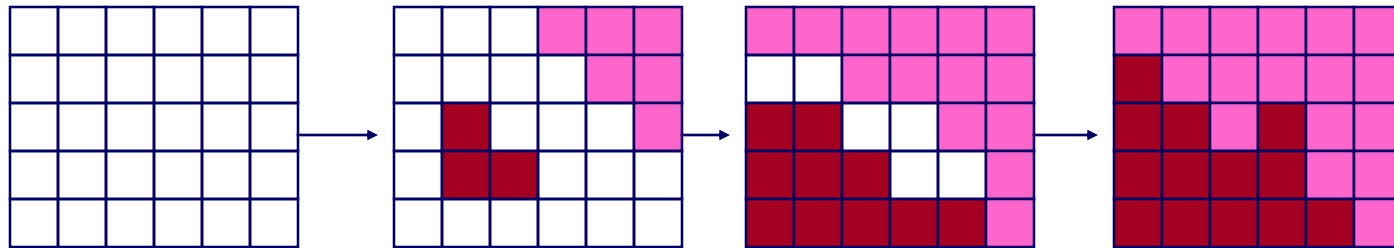
# Region Growing

- Regions are grown from “seed” points.
- A “seed” point is selected for each potential region.  
Append to each point those neighbors that have similar properties (e.g. Gray level, color, texture, etc.)
- problems of this method
  - selection of seed
  - selection of similarity measure
  - termination condition



# Split-merge

- Image is systematically subdivided into regions.
- Merge neighboring regions if they are similar.
- Problem with this method is that the region boundary is jagged.



- **Thresholding** is also a region based method.
- Watershed transform based image segmentation
- Active contours based image segmentation (edge discontinuity)
- And more

## 8. Feature Extraction I

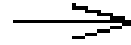
- Thinning is a process to obtain the skeleton of a **binary object (0-1 image)** region.
- The skeleton of a region can be defined by the medial axis transform (MAT):
  - For each point  $p$  in  $R$  (*region*), we find its closest neighbor in  $B$  (*boundary*).
  - If  $p$  has more than one such neighbors, it is said to belong to the medial axis of  $R$ .
- $\text{MAT} \Leftrightarrow \text{Thinning}$
- Thinning can be performed using the Distance Transform.



- Distance Transform (DT)

- DT produces a gray level image in which the gray level intensities of points inside foreground regions are the distance to the closest boundary (between object & background) from each object pixel in vertical or horizontal direction.
- Object region represented by: 1 & Background by: 0

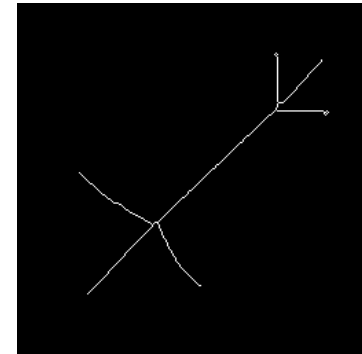
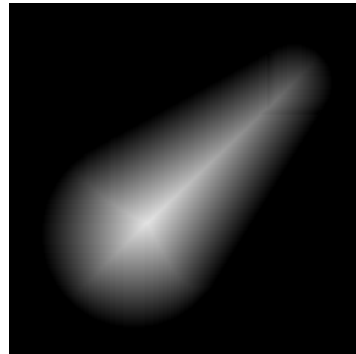
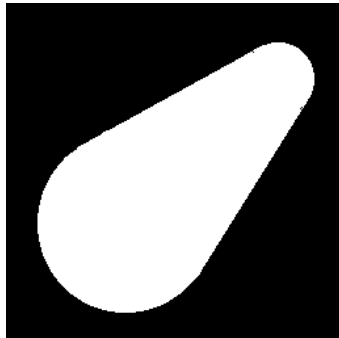
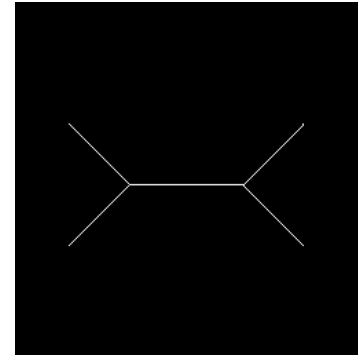
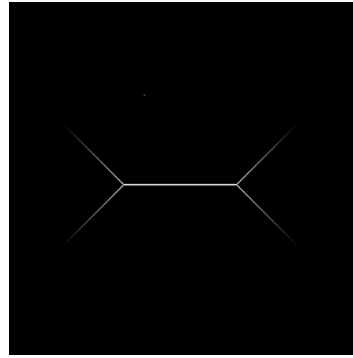
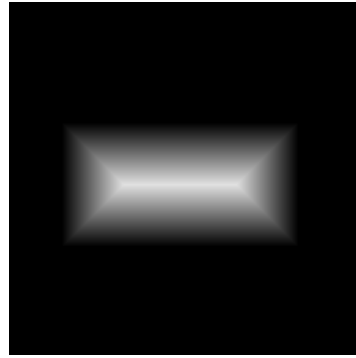
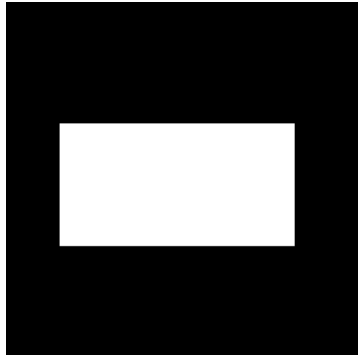
0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0



0	0	0	0	0	0	0	0
0	1	1	1	1	1	1	0
0	1	2	2	2	2	1	0
0	1	2	3	3	2	1	0
0	1	2	2	2	2	1	0
0	1	1	1	1	1	1	0
0	0	0	0	0	0	0	0

———— The Medial Axis

- Examples of Thinning via DT & MAT



original

DT (grey)

MAT (grey)

Skeleton (binary)

## A two-step thinning algorithm

- The thinning algorithm has to satisfy the following constraints:
  - does not remove end points
  - does not break connectedness
  - does not cause excessive erosion of the region
- The 8-neighbors of a boundary pixel belong to the object region are checked against a predefined set of patterns.
- If all the criteria are met, the pixel is flagged to be removed, otherwise, the pixel is not flagged.

- The thinning algorithm performs successive passes of a two step operation. Can we remove p1 (i.e. if p1 is a boundary pixel)?

– Step1:

- a boundary pixel ( $p1=1$ ) is flagged for removal if:

(a)  $2 \leq N(p1) \leq 6$

(b)  $S(p1) = 1$

(c)  $p2 \& p4 \& p6 = 0$

(d)  $p4 \& p6 \& p8 = 0$

p9	p2	p3
p8	p1	p4
p7	p6	p5

where  $N(p1)$  is the number of non-zero neighbors of p1.

- $S(p1)$  is the number of 0-1 transitions in the ordered sequence of p2, p3, ..., p8, p9.

- Step2:
  - a boundary pixel is flagged for removal if:
    - (a)  $2 \leq N(p1) \leq 6$
    - (b)  $S(p1) = 1$
    - (c')  $p2 \ \& \ p4 \ \& \ p8 = 0$
    - (d')  $p2 \ \& \ p6 \ \& \ p8 = 0$
- Flagged pixels are removed after each step.
- The two steps are repeated until no more pixels are deleted after the two steps.
- Example

- $N(p1)=4, S(p1) = 3$

0	0	1
1	p1	0
1	0	1

- What is happening here ?
  - $N(p1) = 1$  means end of skeleton. Deleting  $p1$  when  $N(p1)=7$  causes erosion into the region.
  - $S(p1) > 1$  means the region is already 1-pixel wide.
  - Conditions (c) and (d) are satisfied by the minimum set of values:
    - $(p4=0 \text{ OR } p6=0) \text{ or } (p2=0 \text{ AND } p8=0)$
  - A point that satisfies these conditions is an **S or E boundary point (BP)** or a **NW-NE corner point (CP)**.
  - When satisfying conditions (c'),(d'), the point is a **N or W BP** or **SE-SW CP**.
- Public domain software:

[https://rosettacode.org/wiki/Zhang-Suen\\_thinning\\_algorithm](https://rosettacode.org/wiki/Zhang-Suen_thinning_algorithm)

# Curve extraction from function fit

- Use analytical functions to represent lines, i.e. straight line, curve, circle/arc.
- Analytical function for a straight line:

$$\alpha x + \beta y + \gamma = 0$$

- The parameters to be found are  $\alpha$ ,  $\beta$  and  $\gamma$ .
  - Edge points with x and y coordinates are used in a least square fit.
- Analytical function for a curve can be circle, ellipse, a quadratic, a cubic or even higher order polynomial can be used for function fit.

- It is important to know what is the type of function to be fitted.
- Fitting a straight line to a list of edge points result only a straight line representation of the edge points.
- Fitting a quadratic function to a list of edge points result only a quadratic representation of the edge points, regardless of whether the edge points form a straight line or a curve.
- The least square residue gives an indication of how good a function fit to a group of edge points.



# Hough Transform for straight line detection

- The Hough transform (HT) (G & W or Pratt) :
  - a technique used to detect line features such as straight line, circle, ellipse, arc, etc.
  - Classical HT requires that the desired features be specified in some parametric form.
  - A generalized HT can be employed when a simple analytic description of a feature(s) is not possible.
  - Due to the computational complexity of the generalized HT, we restrict our study to the classical HT.

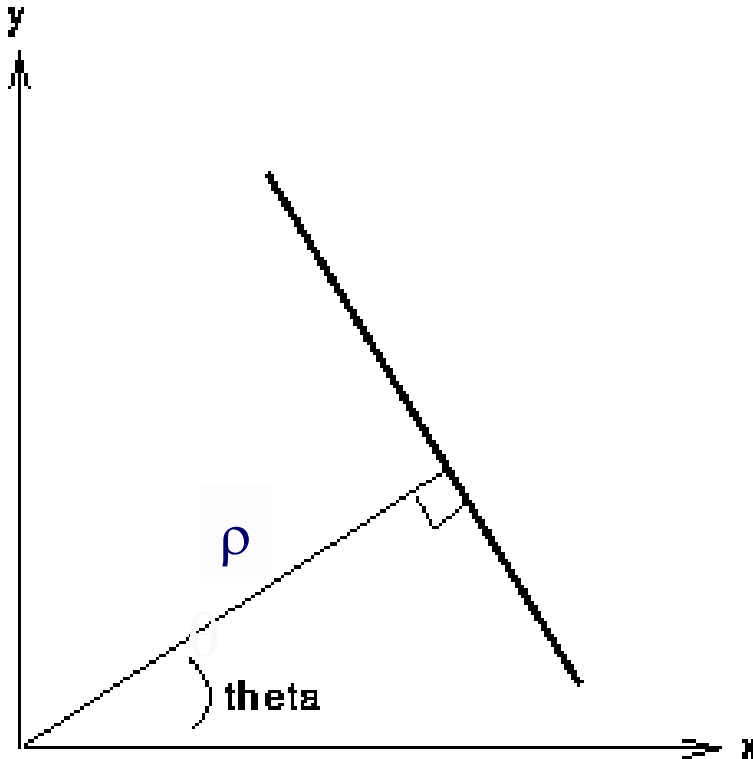
- Main advantage of HT:
  - tolerant to gaps in feature boundary descriptions.  
**This is also a disadvantage if the image actually has broken lines!**
  - relatively unaffected by image noise.
  - useful for computing a global description of a feature(s) (where the number of solution classes need not be known a priori), given (possibly noisy) local measurements.

- HT for straight line use the following parametric equation:

$$\rho = x \cos \theta + y \sin \theta$$

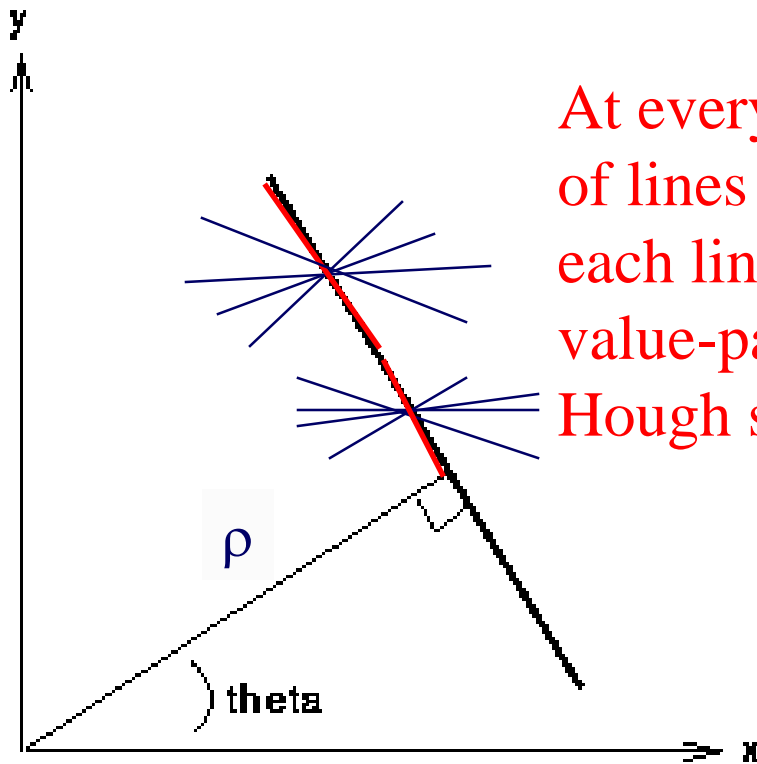
where  $\rho$  is the length of a normal from the origin to the line and  $\theta$  is the orientation of w.r.t. the X-axis.

- For any point on this line,  $\rho$  and  $\theta$  are constant.

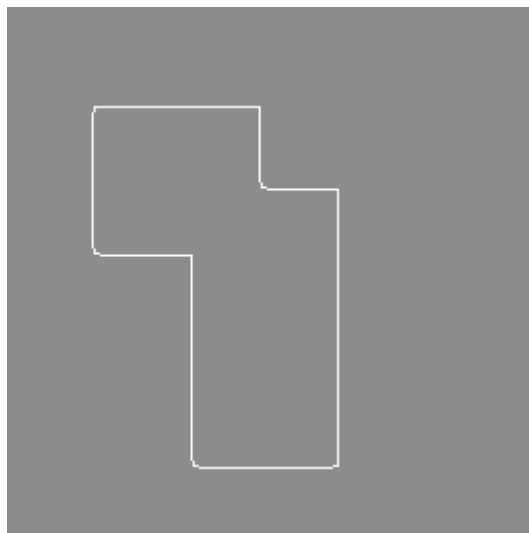


- The transform is implemented by quantizing the Hough parameter space into finite intervals or accumulator cells (2D for straight line).
- As the algorithm runs, each edge pixel is transformed into a discretized curve (equivalent to drawing an infinite number of lines thro' the point) and the accumulator cells which lie along this curve are incremented.
- Resulting peaks in the accumulator array represent strong evidence that a corresponding straight line exists in the image.
- Hough space quantization is a difficult problem as it affects final results and line precision.
- HT is computation intensive, but dedicated hardware is available.

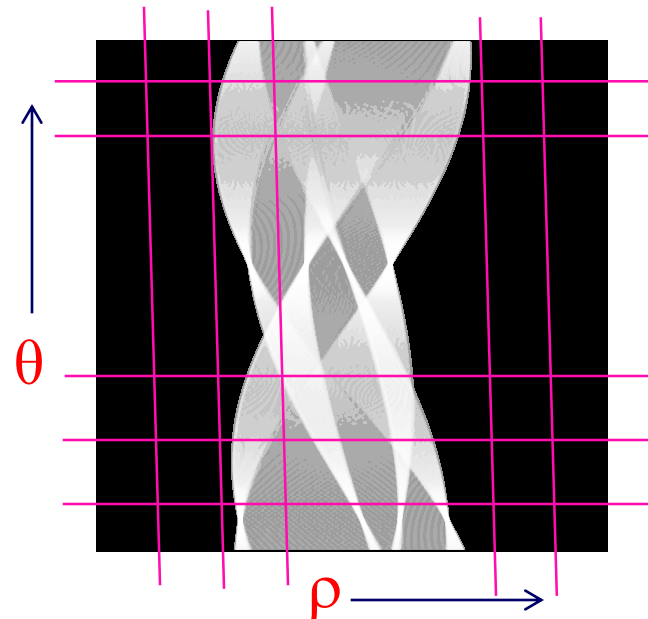
At every edge pixel, we can draw a large number of lines with different orientations / slopes. For each line, we can obtain the corresponding  $(\rho, \theta)$  value-pairs. These pairs form a sinusoidal in the Hough space. Take note of the 2 red lines.



## Hough Space Quantization



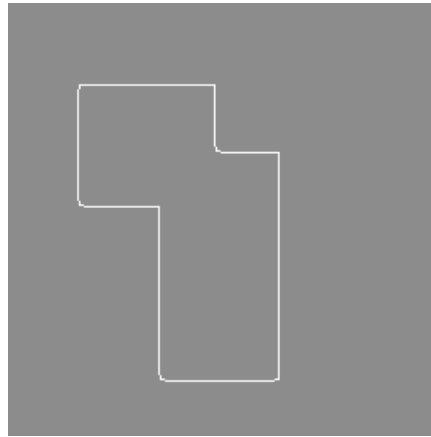
x-y space



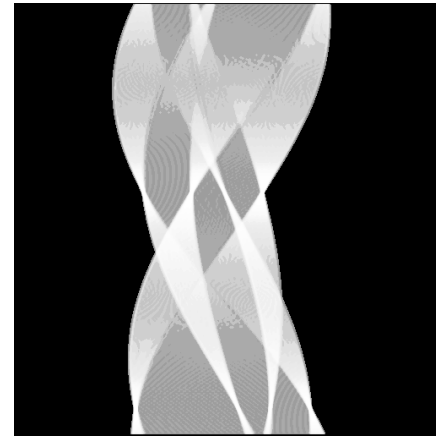
## – Example of HT for straight line



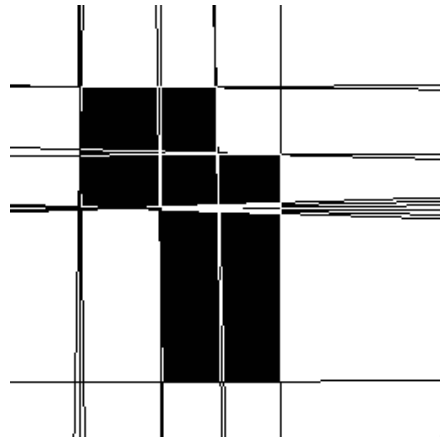
original



Canny edge



Hough space



Back projection  
from Hough space

- the accuracy of alignment of detected and original image lines not perfect due to Hough space quantization.
- many edges have several detected lines due to several nearby Hough-space peaks.

## Hough Transform for arc/circle detection

- The same procedure can be used to detect other features with analytical descriptions, such as arcs and circles.
- The parametric equation is now

$$(x - a)^2 + (y - b)^2 = r^2$$

where  $a$  and  $b$  are the coordinates of the center of the circle and  $r$  is the radius.

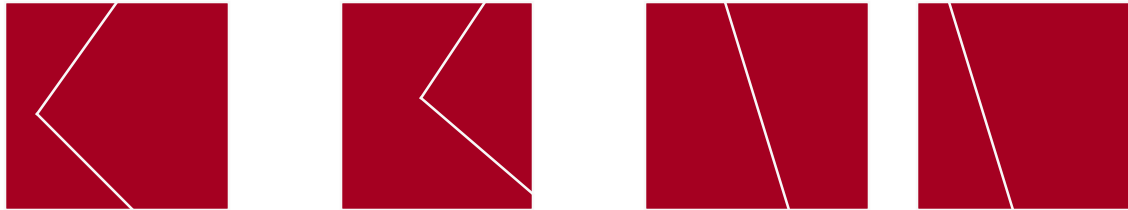
- The computational complexity of the algorithm increases as the Hough space becomes a 3-D accumulator.

# **Corner detection**

- Corner detection is an important step in many computer vision tasks.
- Corners are considered as second order derivatives whereas edges are considered as first order derivatives.
- Corners are as curvature extrema whereas edges are gradient extrema.
- Corners are less ambiguous as edges exhibit the aperture problem (we know how the object has moved from corner but not from the edge)



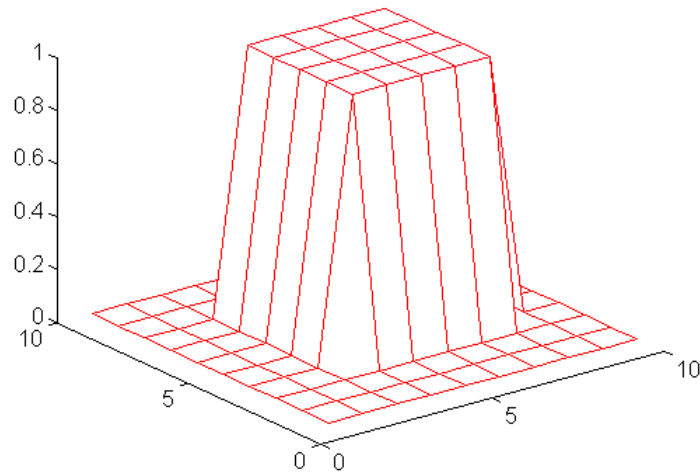
## The aperture problem



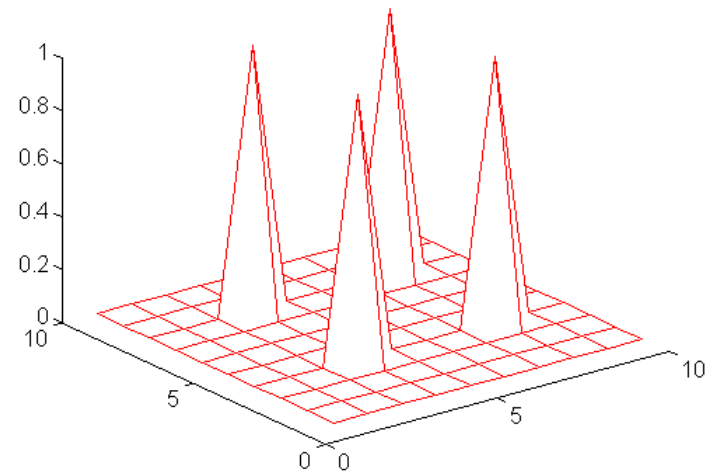
- object can be seen moved horizontally by observing the difference in position of the corner in the images.
- The change in position of the edge line cannot reveal the object movement.
- Corner is said to occur at edge points where there is a significant change in gradient directions.

## Gray level corner versus Geometric corner

- Gray level corner:
  - corner detection is performed on the gray level image.
  - Corner is detected through enhancing the second order gray level variation, i.e. the gray level curvature.
- Geometric corner (covered in slides 185-187):
  - corner detection is performed on the boundary of object.
  - corner is detected by following the boundary points and by seeking for locally extreme curvature points.



Gray level profile



Desired corner  
detection output

- Consider the image gray levels forming a intensity surface over the image plane.
- The objective of corner detection is to find a gray level region that exhibit large surface curvature.

## Beaudet DET corner detector

- A rotationally invariant operator called DET is defined.

$$DET(x, y) = I_{xx}(x, y)I_{yy}(x, y) - I_{xy}(x, y)^2$$

where


$I_{xx}$  and  $I_{yy}$  are second directional derivatives in  $x$  and  $y$  respectively, and

$I_{xy}$  is the derivative taken first in  $x$  and then in  $y$ .

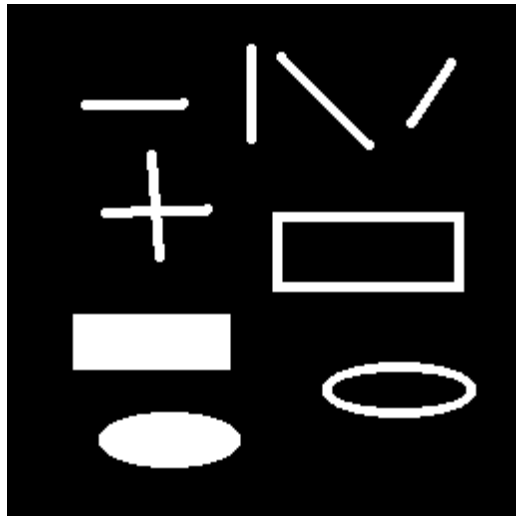
- The corner detection is based on the thresholding of the absolute value of the extrema of this operator.

- DET can be calculated by convolution of the image with the corresponding directional derivative kernels.
- 3x3 and convolution kernels for corner detection are given as follows.

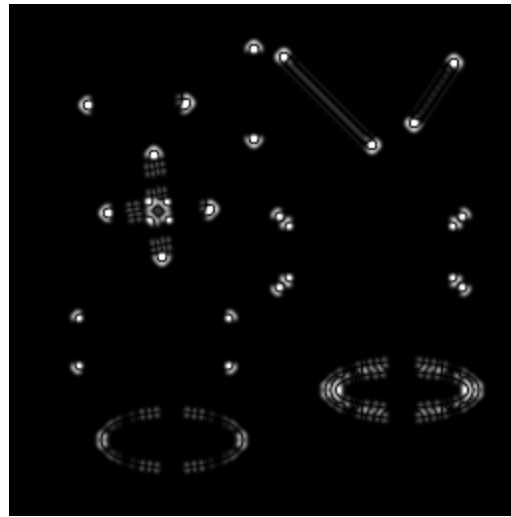
$$I_x = \frac{1}{6} \begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix} \quad I_y = \frac{1}{6} \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix} \quad I_{xx} = \frac{1}{3} \begin{bmatrix} 1 & -2 & 1 \\ 1 & -2 & 1 \\ 1 & -2 & 1 \end{bmatrix}$$

$$I_{yy} = \frac{1}{3} \begin{bmatrix} 1 & 1 & 1 \\ -2 & -2 & -2 \\ 1 & 1 & 1 \end{bmatrix} \quad I_{xy} = \frac{1}{4} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$


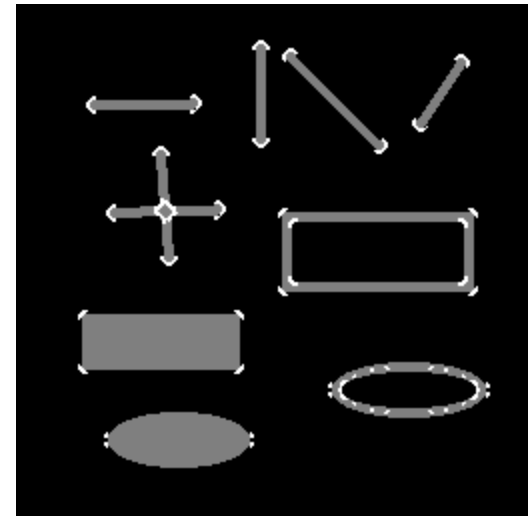
Consider 3x3 region of grey values  
and apply differencing in x, y directions. <sup>126</sup>



Original



DET output



Threshold with 100

## DET in C

```
convolve(tmp, tmp1, xsize, ysize, gaussian, 7, 7, 115.0);
convolve(tmp1, tmp2, xsize, ysize, lxx, 5, 5, 35.0);
convolve(tmp1, tmp3, xsize, ysize, lyy, 5, 5, 35.0);
convolve(tmp1, tmp4, xsize, ysize, lxy, 5, 5, 100.0);
for (i = 0; i < xsize*ysize; i++)
    out[i] = 0.0;
for (i = 7; i < ysize - 7; i++)
    for (j = 7; j < xsize - 7; j++) {
        pt_loc = i*xsize+j;
        out[pt_loc] = (tmp2[pt_loc]*tmp3[pt_loc]
            - tmp4[pt_loc]*tmp4[pt_loc]);
    }
```

```

void convolve (float *in_image, float *out_image, int xsize, int
ysize, int *mask, int mxs, int mys, double normalizer)
{
    int i,j,k,l,halfwx,halfwy,mwxy;
    float sum;

    halfwx = mxs/2;  halfwy = mys/2;  mwxy = mxs*mys - 1;
    for (i = halfwy; i < ysize - halfwy; i++)
        for (j = halfwx; j < xsize - halfwx; j++) {
            sum = 0;      for (k = 0; k < mys; k++)
                for (l = 0; l < mxs; l++)
                    sum += in_image[(i+k-halfwy)*xsize+j+l-halfwx]
                        *(float)mask[mwxy-k*mxs - l];
            out_image[i*xsize+j] = sum/normalizer;      }
}

```



## Plessey corner detector

- A good corner detector that can locate corner position more accurately.
- Corner detection involves first derivatives
- Less sensitive to image noise than DET.
- For each pixel, a matrix is constructed:

$$A = \begin{bmatrix} \langle I_x^2 \rangle & \langle I_x I_y \rangle \\ \langle I_x I_y \rangle & \langle I_y^2 \rangle \end{bmatrix}$$

where  $\langle . \rangle$  means Gaussian weighted averages

- In theory, the cornerness value is defined as

$$C_p = \frac{Tr(A)}{Det(A)}$$

- $C_p$  are all negative and smallest  $C_p$  corresponds to corner.
- In another approach, the eigenvalues of A is calculated and the smallest of the two eigenvalues is taken as the output. A large +ve value indicates existence of a corner.

## Plessy corner detection in C

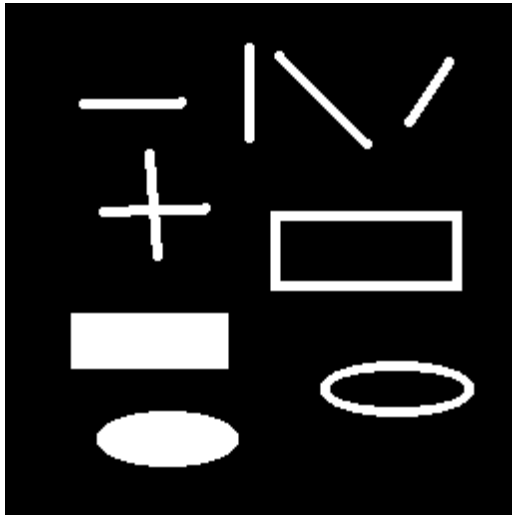
```
convolve(tmp, tmp2,xsize, ysize,Ix,5,5,35.0);  
convolve(tmp, tmp3,xsize, ysize,Iy,5,5,35.0);
```

```
for (i = 0; i < xsize*ysize; i++) {  
    tmp1[i] = tmp2[i]*tmp3[i];/* IxIy */  
    tmp2[i] = tmp2[i]*tmp2[i]; /* Ix2 */  
    tmp3[i] = tmp3[i]*tmp3[i];/* Iy2 */  
}  
convolve(tmp3, tmp4, xsize, ysize, low_pass,3,3,16.0);/* <Iy2> */  
convolve(tmp2, tmp3, xsize, ysize, low_pass,3,3,16.0);/* <Ix2> */  
convolve(tmp1, tmp2, xsize, ysize, low_pass,3,3,16.0); /* <IxIy> */
```

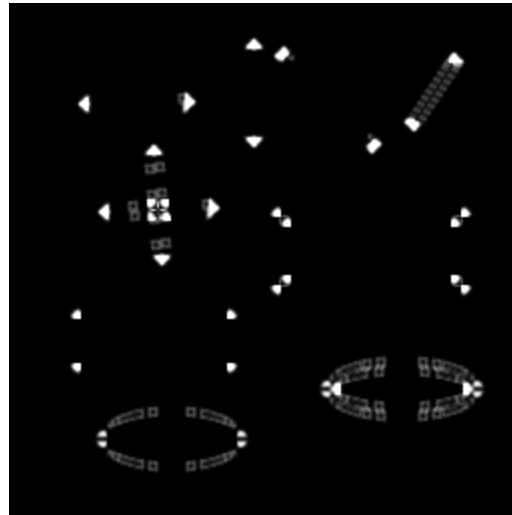
```

for (i = 0; i < xsize*ysize; i++)
    out[i]=0.0;
for (i = 7; i < ysize - 7; i++)
    for (j = 7; j < xsize - 7; j++) {
        pt_loc = i*xsize+j;
        tracea = tmp3[pt_loc]+ tmp4[pt_loc];
        deta = tmp3[pt_loc]*tmp4[pt_loc]-tmp2[pt_loc]*tmp2[pt_loc];
        if ((pow(tracea,2)-4*deta) > 0.0) {
            if (tracea < 0){
                u1 = -0.5*(tracea - sqrt(pow(tracea,2)-4*deta));
                u2 = deta/(0.001+u1);    }
            else {
                u1 = -0.5*(tracea + sqrt(pow(tracea,2)-4*deta));
                u2 = deta/(0.001+u1);    }
            out[pt_loc] = (u1 < u2) ? u2 : u1;
        }
    }
}

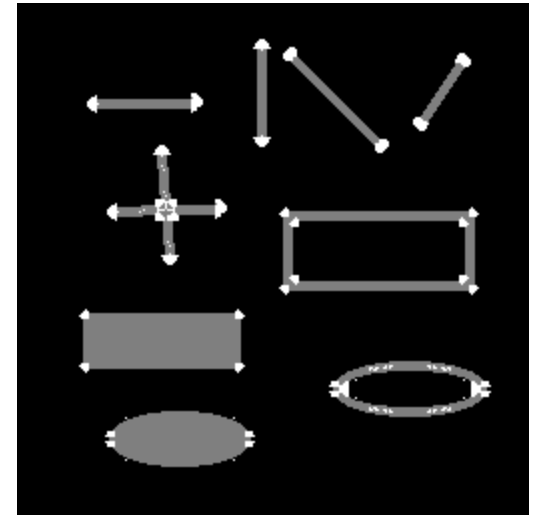
```



Original



Plessy output



Threshold with -100

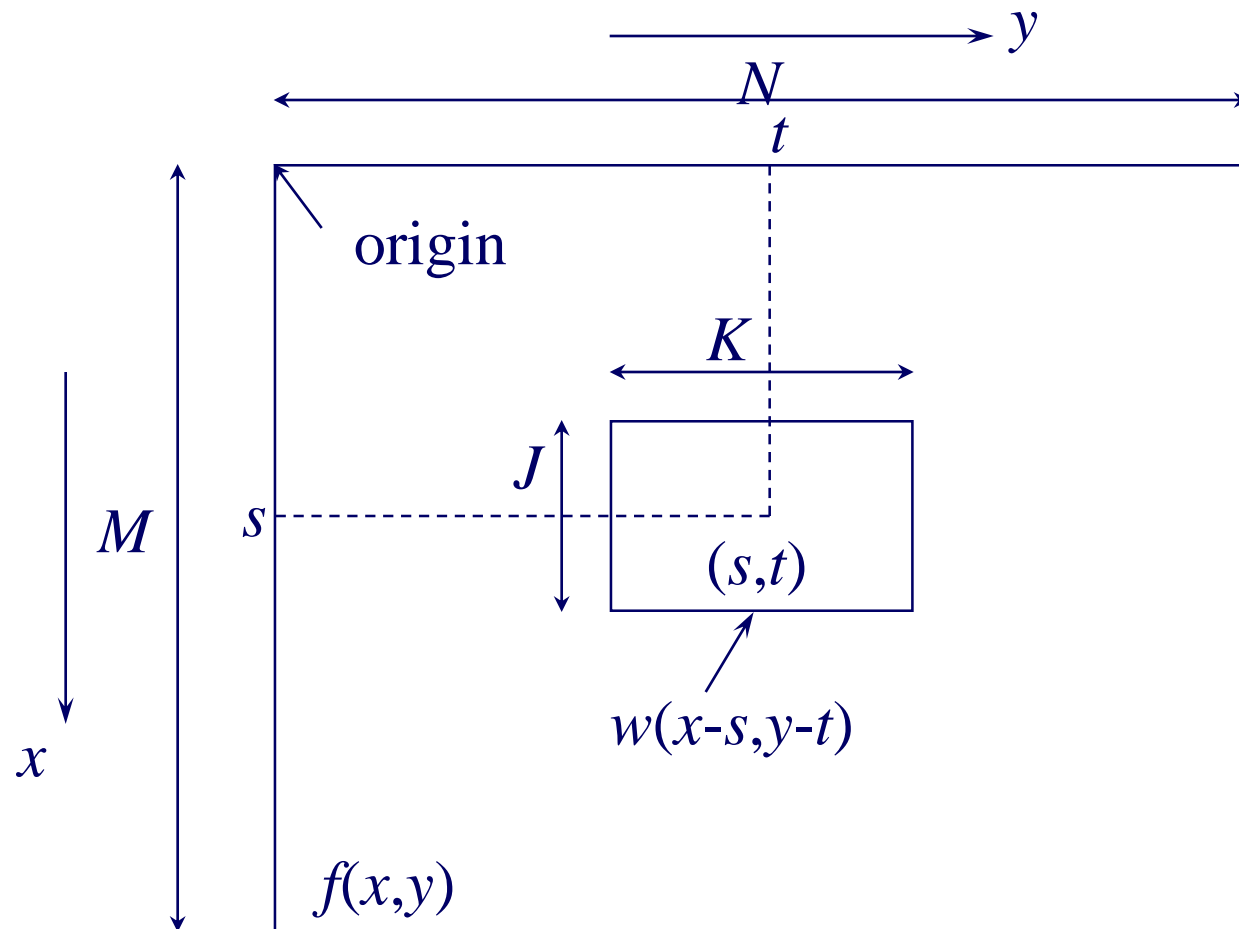
# Pattern (Model) extraction by Template Matching (G & W)

## 1. Matching by normalized cross correlation

- find matches of a sub-image  $w(x,y)$  of size  $J \times K$  within an image  $f(x,y)$  of size  $M \times N$ , by

$$c(s,t) = \frac{\sum_x \sum_y f(x,y)w(x-s,y-t)}{\left\{ \sum_x \sum_y [f(x,y)]^2 \sum_x \sum_y [w(x-s,y-t)]^2 \right\}^{1/2}}$$

where  $s = 0, 1, 2, \dots, M-1$ ,  $t = 0, 1, 2, \dots, N-1$ .



## Properties:

- the numerator is the cross correlation function.
- the denominator is the normalizer.
- $0 \leq c(s,t) \leq 1$
- summation is taken over the image region where  $w$  and  $f$  overlap.
- The *maximum* value of  $c(s,t)$  appears at the position where  $w(x,y)$  best matches  $f(x,y)$ .
- Advantage : easy to implement.
- Disadvantages : sensitive to image rotation and image scale changes.



*Numerical example illustrating normalized cross-correlation:*

$f =$

2	4	66	7	8	9
34	5	6	7	2	4
4	44	5	6	9	10
3	6	7	8	9	7
5	7	3	5	7	5

$x$

$y$

$w =$

5	6	7
44	5	6
6	7	8

$x$

$y$

$c =$

	0.5868	0.2877	0.2712	0.6410	
	0.2879	1.0000	0.5719	0.5872	
	0.2840	0.3309	0.6531	0.6638	

$x$

$y$

e.g. calculate  $c(2,3)$  : - i.e.  $x = s = 2$ ,  $y = t = 3$ ;

coordinates range of  $3 \times 3$  template :  $-1 \leq x \leq 1$ ,  $-1 \leq y \leq 1$ ;

centre of mask at  $f(2,3)$  because  $w(0,0) = w(2-2, 3-3)$ ;

coordinates range of image and mask overlap region :

$$1 \leq x \leq 3, 2 \leq y \leq 4 ;$$

$$c(2,3) = (6*5+7*6+2*7+5*44+6*5+9*6+7*6+8*7+9*8)$$

$$/[(6^2+7^2+2^2+5^2+6^2+9^2+7^2+8^2+9^2)$$

$$*(5^2+6^2+7^2+44^2+5^2+6^2+6^2+7^2+8^2)]^{0.5}$$

$$c(2,3) = 560/(425*2256)^{0.5} = 0.5719$$

Image

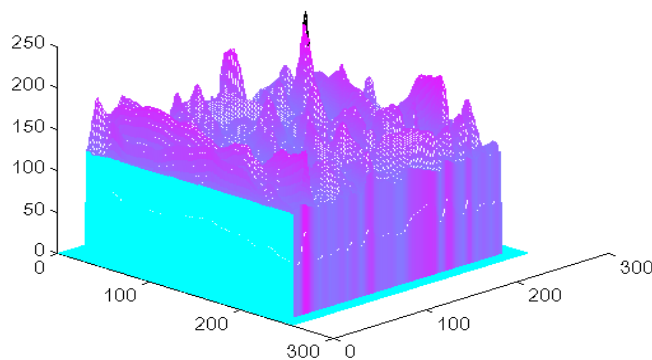
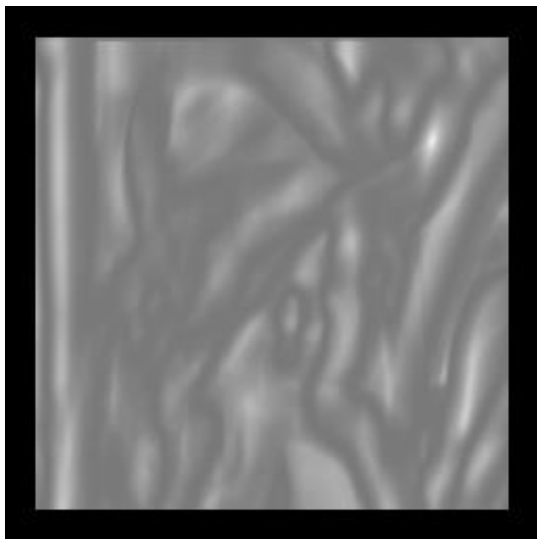


An example of  
template matching

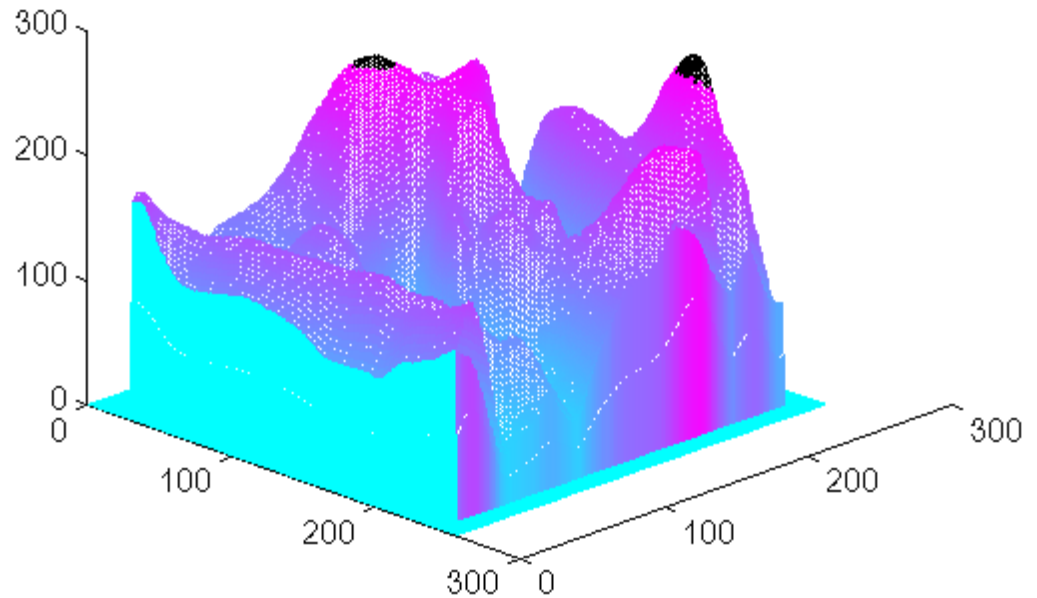


template

Result of  
template matching



What happens if performing cross correlation without normalization?



# 9. Morphology Operations

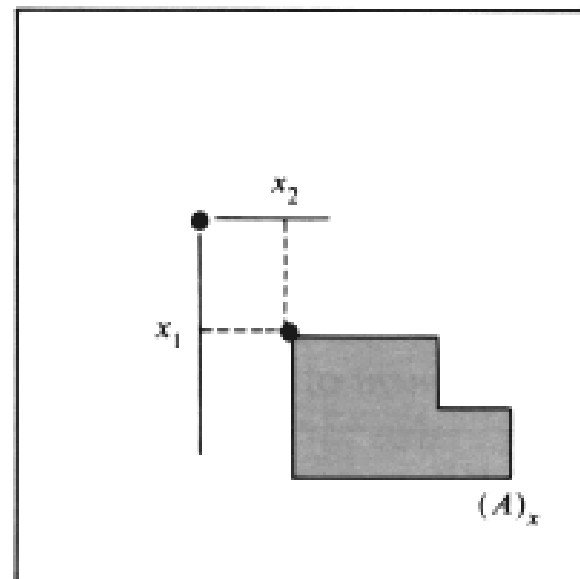
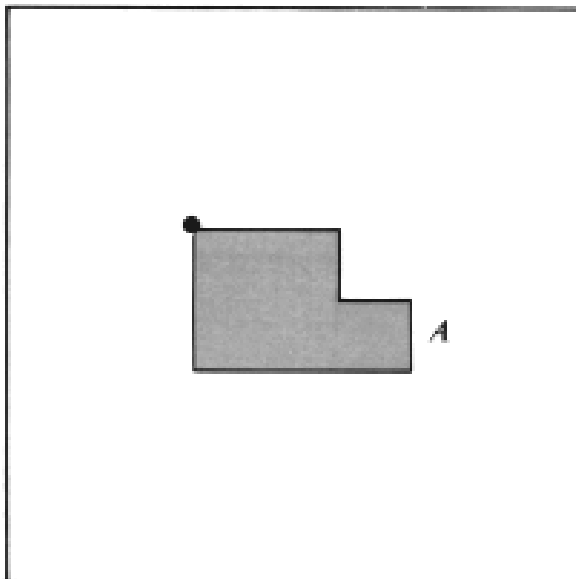
- Morphology (G & W)
  - a branch of biology that deals with the form and structure of animals and plants.
- Mathematical morphology (MM)
  - a tool for extracting image components that are useful in the representation and description of region shape.
- The language of MM is set theory.
- Sets in MM represent the shapes of objects in an image.
- First binary images.

## Basic Definitions :

- Let  $A$  and  $B$  be sets in the spatial plane with components  $a=(a_1, a_2)$  and  $b=(b_1, b_2)$

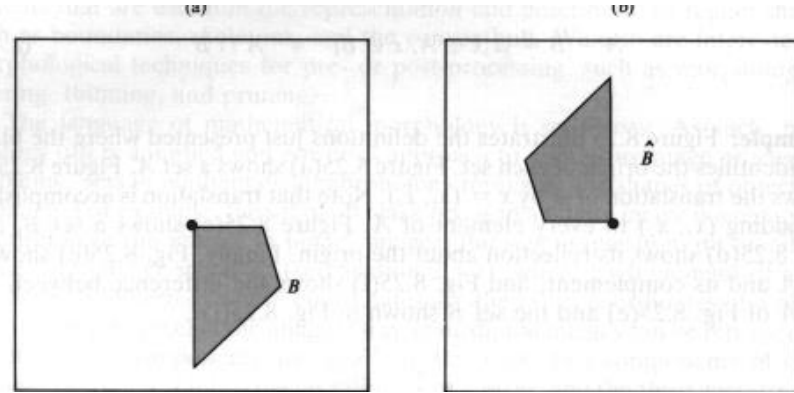
The translation of  $A$  by  $x=(x_1, x_2)$ , denoted by  $(A)_x$  is defined as

$$(A)_x = \{c \mid c = a + x, \text{ for } a \in A\}$$



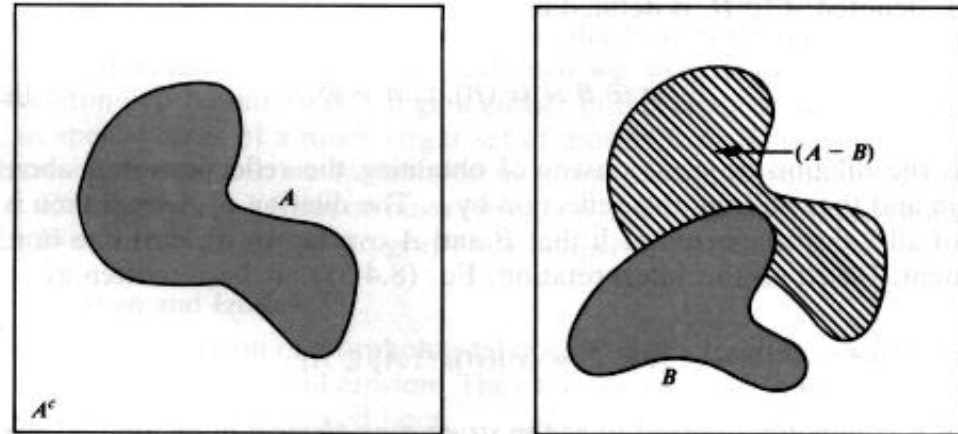
The reflection of  $B$  is defined as

$$\hat{B} = \{x \mid x = -b, \text{ for } b \in B\}$$



The complement of  $A$  is defined as

$$A^c = \{x \mid x \notin A\}$$



The difference of  $A$  and  $B$  is defined as

$$A - B = \{x \mid x \in A, x \notin B\} = A \cap B^c$$

## Dilation

Let  $A$  and  $B$  be sets in the spatial plane and  $\phi$  be the empty set, dilation of  $A$  by  $B$  is

$$\begin{aligned} A \oplus B &= \{x \mid (\hat{B})_x \cap A \neq \phi\} \\ &= \{x \mid [(\hat{B})_x \cap A] \subseteq A\} \end{aligned}$$

Hence  $A \oplus B$  is the set of all  $x$  displacements such that  $\hat{B}$  and  $A$  overlap by at least one non-zero element.

$B$  is called the structuring element.

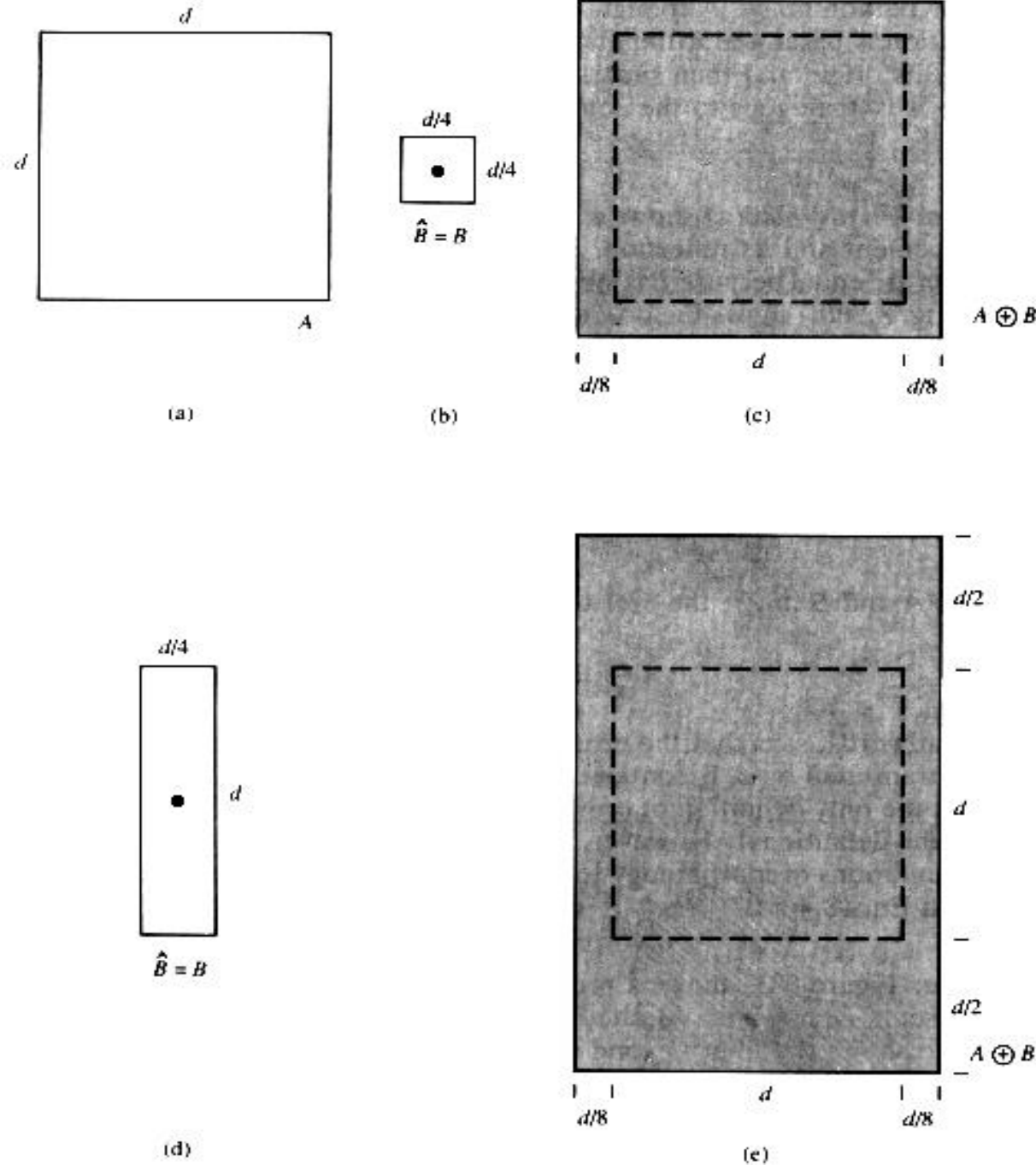


## Illustration of dilation

Note the expansion of the original shape by  $d/8$  on four sides when dilated by a square structuring element of size  $d/4$  by  $d/4$ .

&

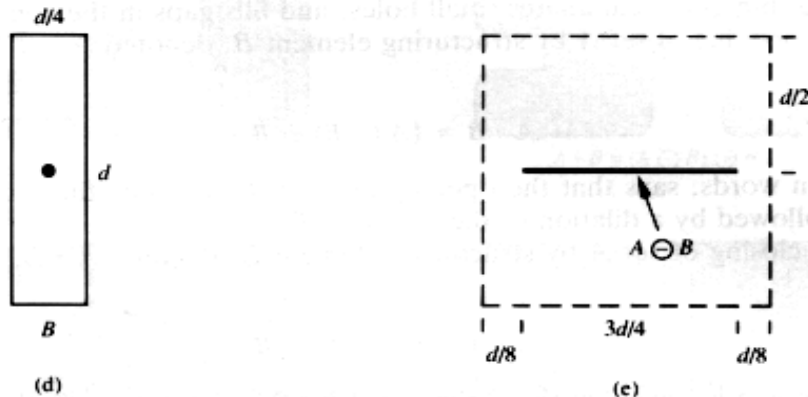
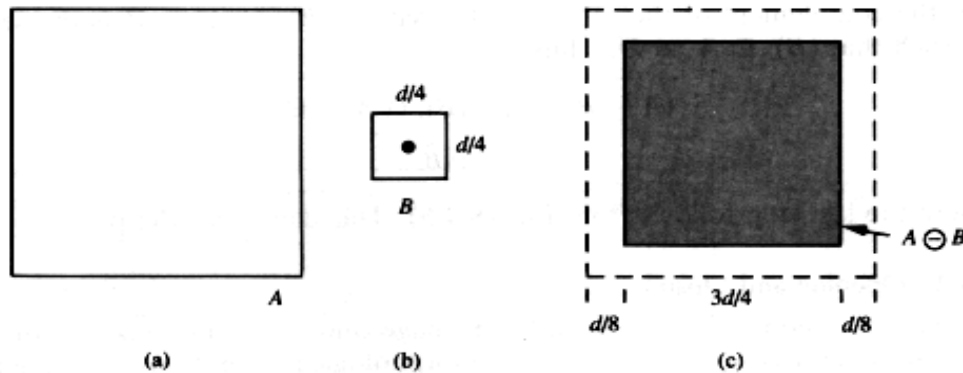
$d/2$  vertically for the 2<sup>nd</sup> case.



## Erosion

$$A \ominus B = \{x \mid (B)_x \subseteq A\}$$

Hence  $A \ominus B$  is the set of all points  $x$  such that  $B$ , translated by  $x$ , is **fully** contained in  $A$ .



## Illustration of erosion

Note the shrinking of the original shape by  $d/8$  on four sides when eroded by a square structuring element of size  $d/4$  by  $d/4$  in the 1<sup>st</sup> case.

## Opening for noise reduction

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

## Closing for noise reduction

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

## Duals

$$(A \bullet B)^c = (A^c \circ \hat{B})$$

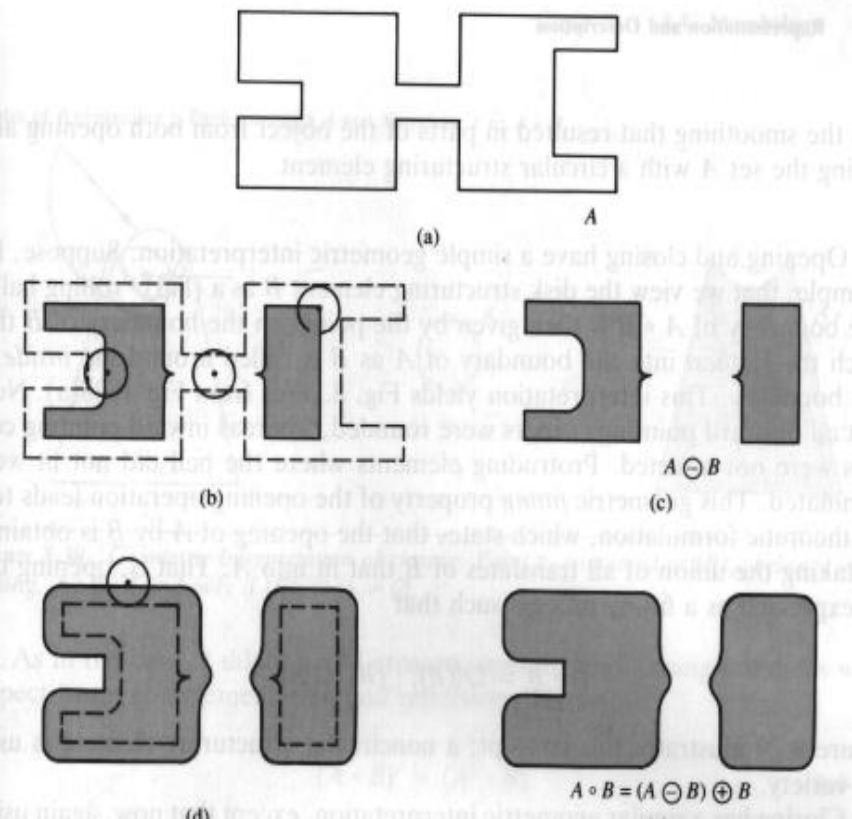


Illustration of opening

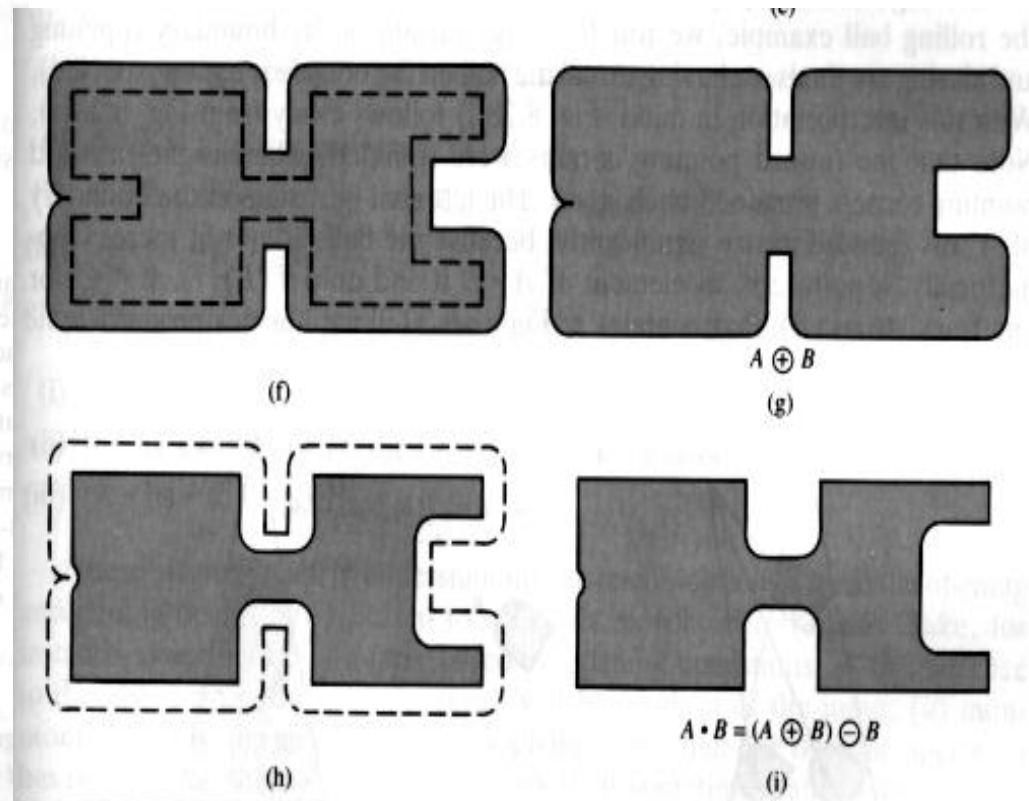


Illustration of closing

## Properties of Dilation

$$(A \oplus B) = (B \oplus A) \text{ (commutative)}$$

## Properties of Erosion

$$(A \ominus B) \neq (B \ominus A) \text{ (not commutative)}$$

## Properties of Opening

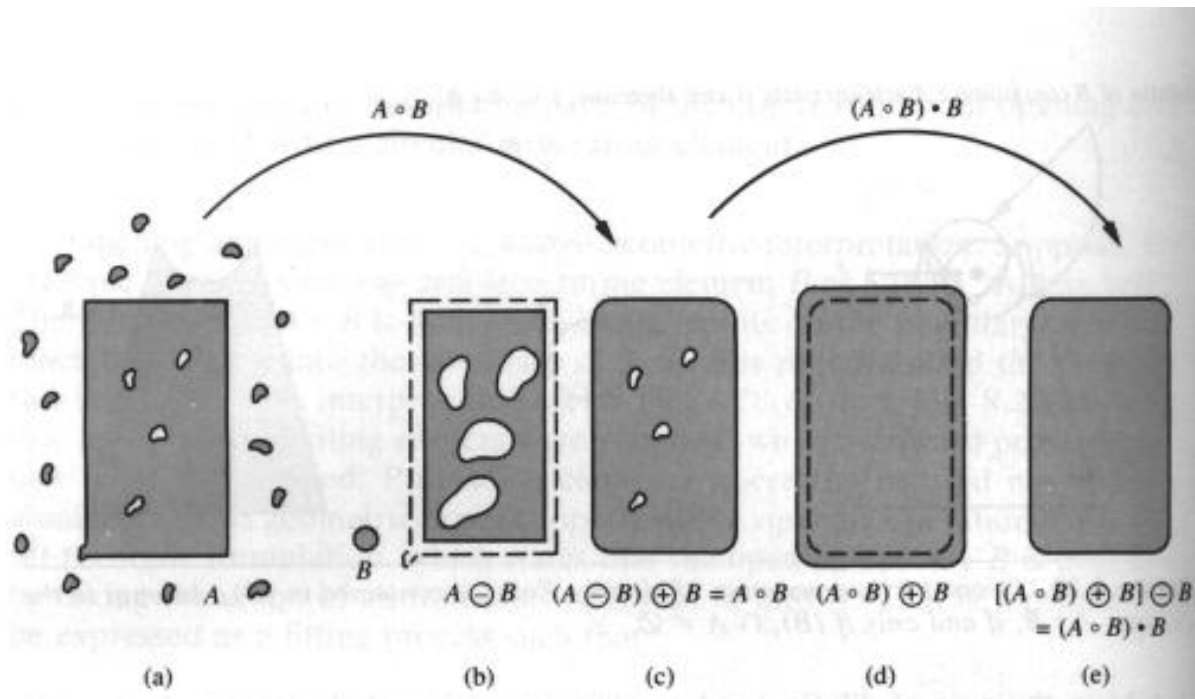
- (i)  $(A \circ B) \subseteq A$  (result will be a subset of the input  $A$ )
- (ii) If  $C \subseteq D$ , then  $(C \circ B) \subseteq (D \circ B)$  (monotonicity)
- (iii)  $(A \circ B) \circ B = A \circ B$  (idempotence)

## Properties of Closing

- (i)  $A \subseteq (A \bullet B)$  (input  $A$  will be a subset of the result)
- (ii) If  $C \subseteq D$ , then  $(C \bullet B) \subseteq (D \bullet B)$  (monotonicity)
- (iii)  $(A \bullet B) \bullet B = A \bullet B$  (idempotence)

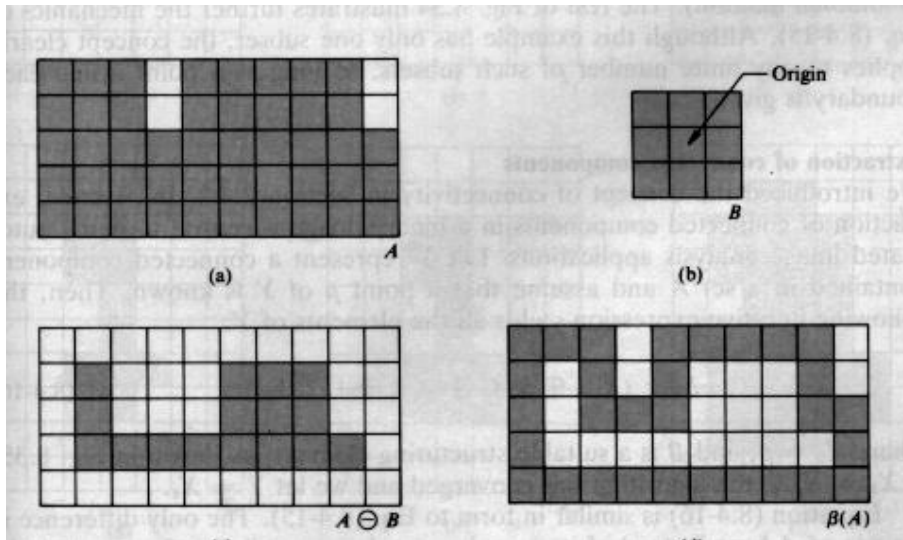
## Morphological filtering

- $(A \ominus B) \oplus B$  can be used to eliminate the noise and its effect on the object.
- Noisy pixels outside object area are removed by opening with  $B$  and noise pixels inside object area are removed by closing with  $B$



# Boundary extraction using binary morphology

$$\beta(A) = A - (A \ominus B)$$



```
void morph_edge1(BYTE *img,BYTE *oimg,int xsize,int ysize,int
*mask,int mxs,int mys)
{
    erosion(img,oimg,xsize,ysize,mask,mxs,mys);
    subtract_images(img,oimg,xsize*ysize);
}
```

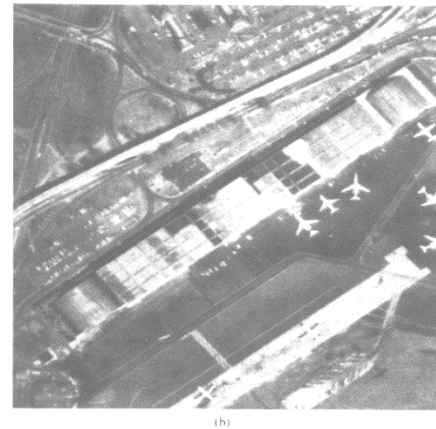
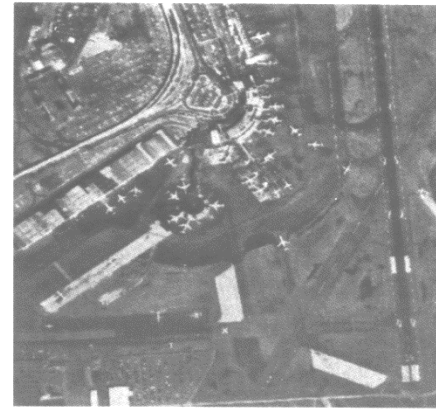
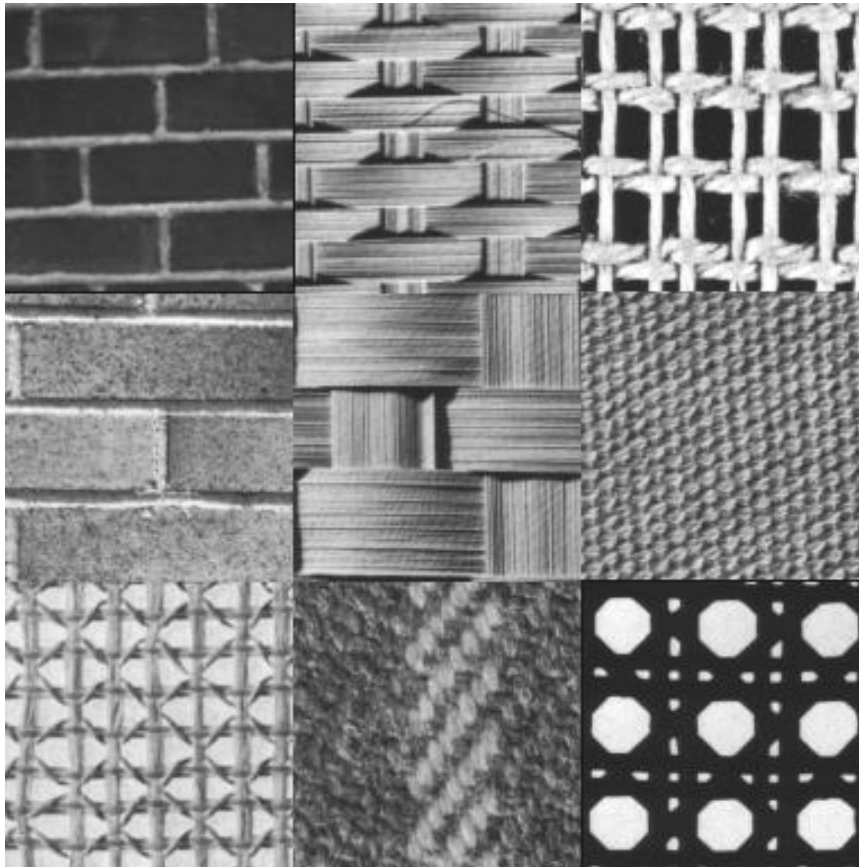
# Gray level morphology

- In gray level morphology, we deal with the functions  $f(x,y)$  and  $b(x,y)$ , where  $f(x,y)$  is the input image and  $b(x,y)$  is the structuring element.
- Min and Max functions are used instead of set operations.
- Gray level morphological operations are similar to **convolution / correlation**, but multiplications become additions or subtractions and summation is replaced by min or max operations.



# 10. Texture Analysis

- What is texture ? (Pratt)





- Texture is an important characteristic in image analysis.
- Image types in which texture plays an important Role:
  - multi-spectral images obtained from aircraft or satellite
  - microscopic images of cell cultures or tissue samples
  - wood, fabrics, food, etc.
- Despite its importance, a formal approach or precise definition of texture does not exist.
- Various models have been used by different people based on their conceptualization of texture.

- Issues in texture analysis:
  - Given a textured region, determine to which of a finite number of classes the region belongs  
(a pattern recognition task)
  - Given a textured region, determine a description or model for it.  
(a texture modeling task)
  - Given an image having many textured areas, determine the boundaries between the differently textured regions.  
(a texture segmentation task)
- Many preprocessing methods often produce confusing results when applied to texture.

- An image texture is characterized by the gray level primitive properties as well as the spatial relationships between them.
- The gray level primitives are regions with gray level properties:
  - e.g. average level or the maximum and minimum levels of a gray level region, area and shape of a gray level region.
- The spatial relationships between primitives may be random, pair-wise dependent or  $n$ -primitive dependent.
- The dependence may be structural, probabilistic, or functional.

## Gray Level Co-occurrence (Method 1)

- The gray level co-occurrence characterize the gray level spatial dependence.
- The gray level means and variances obtained from gray level histograms are first order statistics.
- The gray level co-occurrence gives second or higher order statistics.
- The gray level co-occurrence is specified in a matrix, known as the gray level co-occurrence matrix (GLCM).

- The elements in GLCM are the relative frequencies  $P_{ij}$  with which two neighboring pixels separated by distance  $d$  occur on the image, one with gray level  $i$  and the other with gray level  $j$ .
- Example, for angle quantization of  $45^\circ$  interval:
  - $P(i,j,d,0^\circ) = \#\{[(k,l),(m,n)] \mid k-m=0, |l-n|=d, I(k,l)=i, I(m,n)=j\}$
  - $P(i,j,d,45^\circ) = \#\{[(k,l),(m,n)] \mid (k-m=d, l-n=-d), (k-m=-d, l-n=d), I(k,l)=i, I(m,n)=j\}$
  - $P(i,j,d,90^\circ) = \#\{[(k,l),(m,n)] \mid |k-m|=d, l-n=0, I(k,l)=i, I(m,n)=j\}$
  - $P(i,j,d,135^\circ) = \#\{[(k,l),(m,n)] \mid (k-m=d, l-n=d), (k-m=-d, l-n=-d), I(k,l)=i, I(m,n)=j\}$

## An example of calculation of GLCM

0	0	1	1
0	0	1	1
0	2	2	2
2	2	3	3

Gray  
Level

Gray Level

	0	1	2	3
0	#(0,0)	#(0,1)	#(0,2)	#(0,3)
1	#(1,0)	#(1,1)	#(1,2)	#(1,3)
2	#(2,0)	#(2,1)	#(2,2)	#(2,3)
3	#(3,0)	#(3,1)	#(3,2)	#(3,3)

$$P(i,j,d=1,0^\circ) = \begin{pmatrix} 4 & 2 & 1 & 0 \\ 2 & 4 & 0 & 0 \\ 1 & 0 & 6 & 1 \\ 0 & 0 & 1 & 2 \end{pmatrix}$$

$$P(i,j,d=1,90^\circ) = \begin{pmatrix} 6 & 0 & 2 & 0 \\ 0 & 4 & 2 & 0 \\ 2 & 2 & 2 & 2 \\ 0 & 0 & 2 & 0 \end{pmatrix}$$

*Can you find the GLCM for 45° and 135 ° ?*

- By dividing all the elements in  $P$  with the total no. of point pairs, the matrix element in  $P$  becomes an estimate of the joint probability of a point pair with gray levels  $(i, j)$ .
- Second order statistical features can be computed from GLCM.

e.g. Uniformity or energy:  $\sum_{ij} P_{ij}^2$

Entropy :  $-\sum_{ij} P_{ij} \log P_{ij}$

## Textural energy as features (Method 2)

- In the textural energy approach, the image is first convolved with a variety of kernels.

– e.g. Laws' Kernels

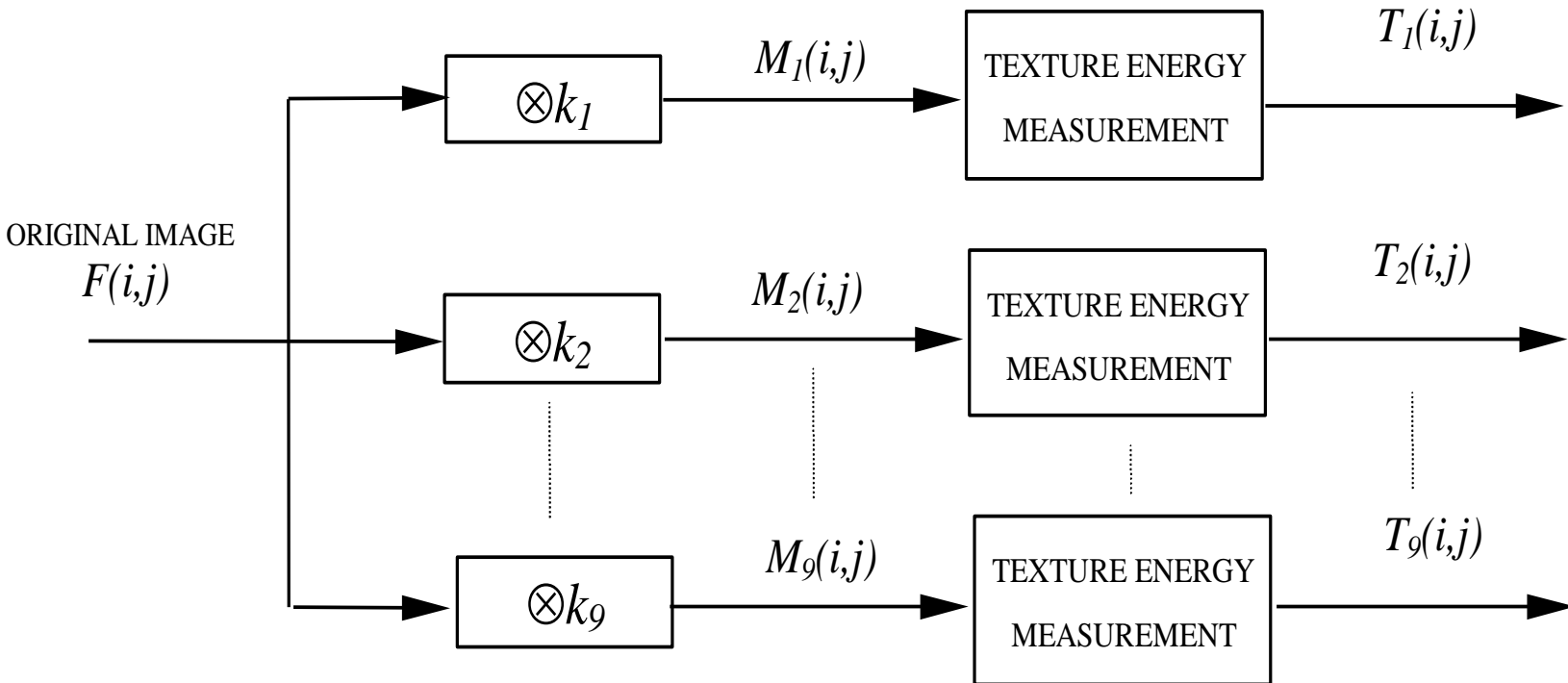
$$k_1 = \frac{1}{36} \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \quad k_2 = \frac{1}{12} \begin{bmatrix} 1 & 0 & -1 \\ 2 & 0 & -2 \\ 1 & 0 & -1 \end{bmatrix} \quad k_3 = \frac{1}{12} \begin{bmatrix} -1 & 2 & -1 \\ -2 & 4 & -2 \\ -1 & 2 & -1 \end{bmatrix}$$

$$k_4 = \frac{1}{12} \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \quad k_5 = \frac{1}{4} \begin{bmatrix} 1 & 0 & -1 \\ 0 & 0 & 0 \\ -1 & 0 & 1 \end{bmatrix} \quad k_6 = \frac{1}{4} \begin{bmatrix} -1 & 2 & -1 \\ 0 & 0 & 0 \\ 1 & -2 & 1 \end{bmatrix}$$

$$k_7 = \frac{1}{12} \begin{bmatrix} -1 & -2 & -1 \\ 2 & 4 & 2 \\ -1 & -2 & -1 \end{bmatrix} \quad k_8 = \frac{1}{4} \begin{bmatrix} -1 & 0 & 1 \\ 2 & 0 & -2 \\ -1 & 0 & 1 \end{bmatrix} \quad k_9 = \frac{1}{4} \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix}$$



The original image is convolved with Laws masks, to obtain  $M_1(i,j)$ ,  $M_2(i,j)$ ...  $M_9(i,j)$ .



$\otimes$  denotes convolution;  $k_1, k_2 \dots k_9$  are Laws' filter masks.

- Then, texture energy  $T_1(i,j)$ ,  $T_2(i,j)$ ...  $T_9(i,j)$  are calculated over a windowed neighborhood of  $W_x$  by  $W_y$  at  $(i,j)$ .
- Texture energy is defined as:

$$T_n(i, j) = \frac{1}{W_x W_y} \sum_{x=-\frac{W_x}{2}}^{\frac{W_x}{2}} \sum_{y=-\frac{W_y}{2}}^{\frac{W_y}{2}} |M_n(i + x, j + y)|$$

- A feature vector can be defined at  $(i,j)$ :

$$\mathbf{T}(i, j) = [T_1(i, j) \quad T_2(i, j) \quad \dots \quad T_9(i, j)]^T$$

- Statistical features can then be derived from the textural energy measures.

## Multi-channel features (Method 3)

- Motivated by studies on biological visual system which suggest that human visual system is performing some form of local spatial-frequency analysis on the retinal image.
- The analysis is done by a bank of tuned band-pass filters.
- Bank of band-pass filters are therefore used to extract spatial frequency components that match with the tuned filters.
- Texture features derived from the band-passed images are known as multi-channel features.

## Multi-channel features

- Multi-channel filtering can use filters designed from Gabor functions or wavelet functions.
- Filtering can be performed in spatial domain or spatial frequency domain.
- An “energy” function is usually defined over a neighborhood.
- Feature vector is constructed using each filter output as a feature dimension.
- To reduce the dimensionality of the feature space, a filter selection algorithm is used to select a subset of filters that produces more useful features.

# 11. Feature Extraction II

- Why feature selection/ extraction?
  - dimensionality reduction. The raw image is too big for decision making, recognition or interpretation tasks
  - Useful information is hidden in the volume of pixels. Hence, need to extract the useful information and represent appropriately.
  - Feature selection methods optimize the feature set against recognition rate or/ & computation cost.
  - Beneficial to reduce/eliminate redundant features.

# Pattern / Object Recognition

- Major component of machine vision.
- Problem description:
  - a computer is provided with information about some shapes, objects, patterns and their labels - called model database or prototypes
  - When the computer is presented with an unknown (test) entity, determine if it is similar to one of the objects, shapes, patterns in the model database.
  - to efficiently solve this problem feature extraction / selection can be useful.
  - Restricted to 2-D intensity images in this module.
  - In the next module you'll learn about 3-D recognition problems.

# What features to use

- Highly problem dependent.
- Selection of sensing modality (camera, scanners), preprocessing methods, features, feature representation methods, recognition or interpretation methods all depend on the specific problem that we're solving.
- Necessary to consider the constraints imposed by the problem, *eg.* accuracy of the system, cost, computation speed, etc.

# Some pattern recognition scenarios

- Patterns isolated or cluttered (isolated hand written characters vs cursive writing)? If cluttered, is segmentation possible prior to recognition?
- Patterns undergone some transformations (eg. rotation, translation, scale, etc.)? Is it necessary to recognize invariant of these transformations?
- Do patterns/objects have shape &/or color &/or texture? (eg. machined parts, textile, printed characters, color magazines, natural scenes, etc.).
- Do these features have discriminatory (between different classes or categories) information.



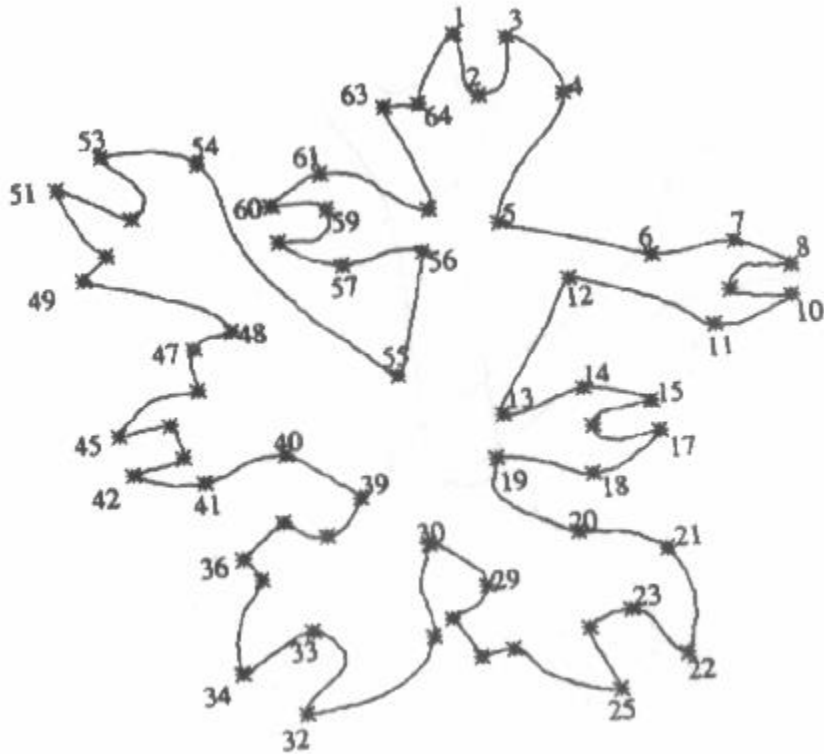
# Global & local features

- If the objects are always isolated or can be isolated prior to interpretation then global &/or local features can be used.
- If objects cannot be isolated prior to recognition, then local features should be used.

# Example of cluttered & isolated objects

**An isolated hand written character – global or local features**

**Cluttered multiple hand tools – local features**



Overlapped Tools

Isolated single Chinese character

# Example. 1: Template matching

- This method search for specified pattern(s) (templates) in a given image.
- This method expects the template to appear in the same size and orientation.
- Not suitable if there is rotation or scaling.
- Commonly used in manufacturing environments.
- Fast & cost effective hardware available
- The details have been given earlier (slides ~135-141).

# Example 2: Natural scene classification

- This may be treated as classification of individual pixels or segmented uniform regions.
- Color and texture of individual pixels or uniform regions are useful (feature vectors).
- Shape or geometry may not be suitable.
- Potential region labels may be sky, grass, trees, water, building, road, etc.

## Example 3: Character recognition

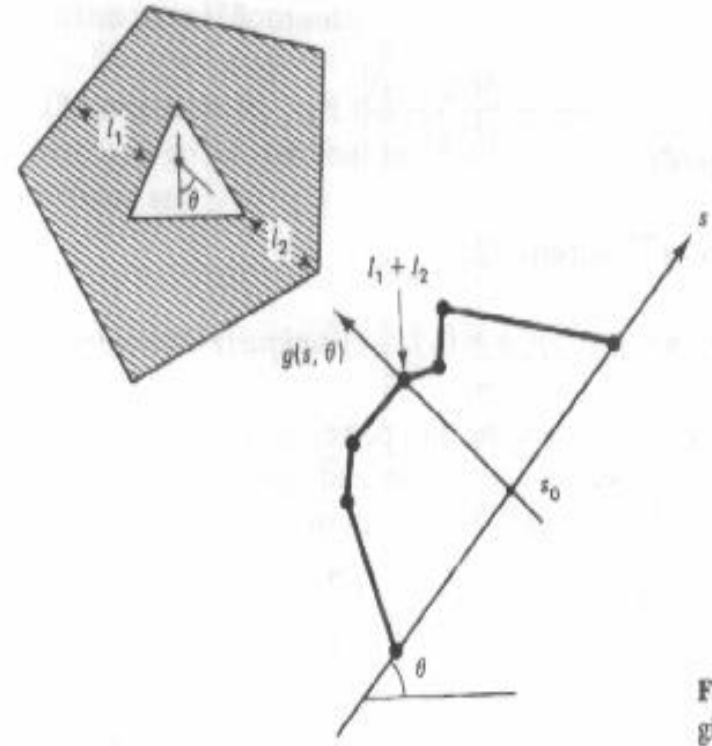
- Commonly treated as isolated pattern recognition problem. Hence, global features can be used.
- bi-level information is commonly used.
- Generally possible to normalize the isolated character to a suitable size.
- Some times invariance to rotation may be necessary in the extracted features.
- Binary character matrices can be represented as feature vectors.

# Ex. 4: Overlapped objects recognition

- Local features should be used
- Generally invariance to scale, rotation, translation required in the chosen features.
- Shape information as well as texture and color may be used depending on the objects under consideration.
- Objects can be represented as relational structures.
- Next we look at some feature extraction methods.

# Projection profile at angle “theta”

- Shows projection of binary object at an angle “theta”.
- Common angles: 0,90,45,135.
- Can obtain from 8-bit images
- projection profile is the summation of pixel intensity values along a given direction.
- Useful for segmentation too. (eg columns/lines separation in printed documents.)
- How to segment document images??



# Boundary representation

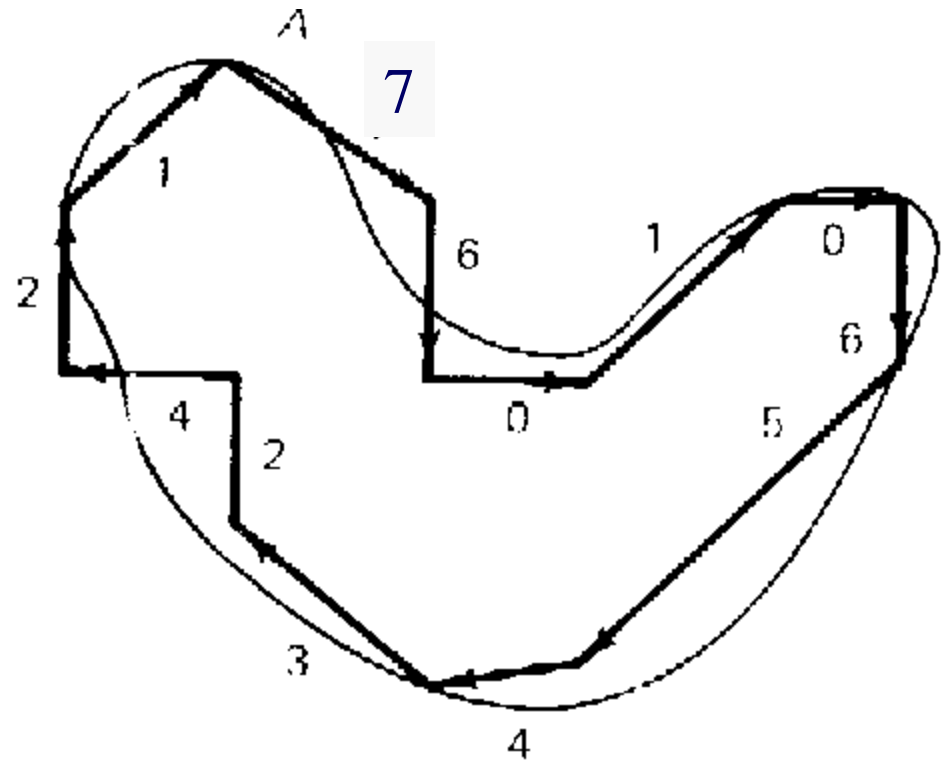
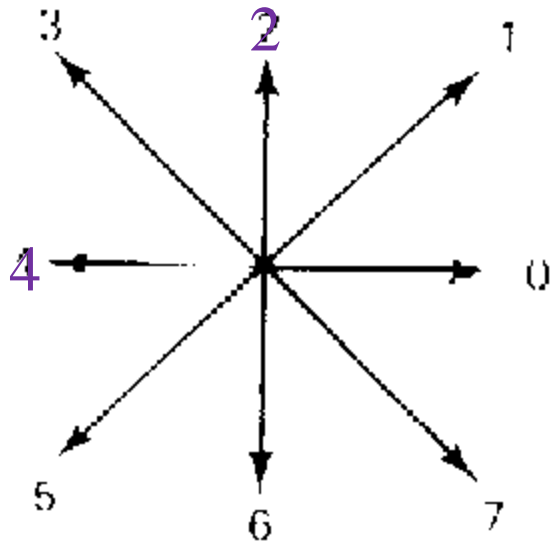
- Chain codes, difference chain codes, shape numbers
- Polygonal approximation
- Break points/Extreme curvature points
- Fourier descriptors
- Moments
- Curve fitting *eg.* B - spline, polynomials, etc.
- Boundary curve may be extracted by edge linking, boundary following or morphological operations.



# Chain codes

- The chain code may use 4 or 8 directions.
- Compute the direction by considering 4-5 pixels, if we use consecutive pixels, codes will be noisy and long.
- Chain code will be a sequence of directions from a starting point.
- Code depends on the starting point. And also variant to rotation and scaling.
- In the next slide, an example of 8 direction chain code. In the following slide, 4 direction with difference code and shape number.

# Example: Chain codes



(b) Contour

Boundary pixel orientations: (A), 76010655432421

Chain code: A 111 110 000 001 000 110 101 101 110 011 010 100 010 001

# Difference of chain codes & Shape number

- Take difference between consecutive codes.  
Difference is approximately invariant to rotation.
- Align 1 direction of the code with the major axis of the object. (see the next slide)
- Shape number starts with the smallest in the difference code sequence so that overall number is the smallest.
- Order of the chain codes can compensate for scaling variations.

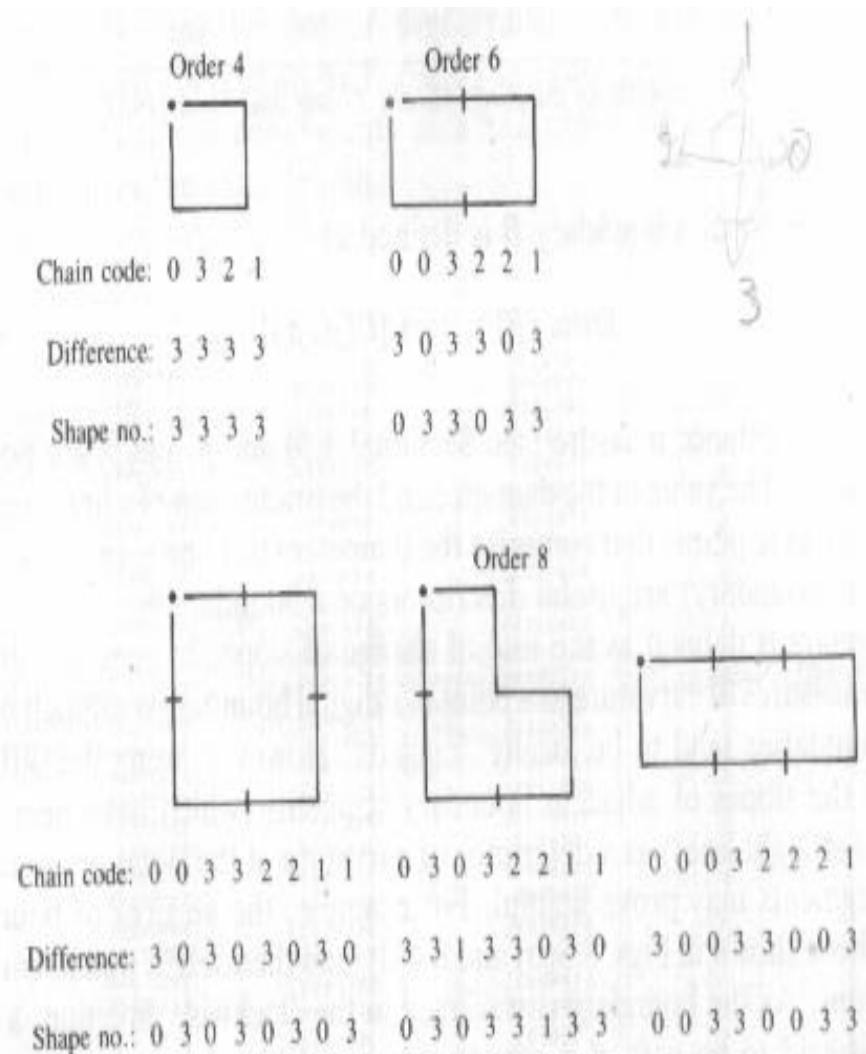
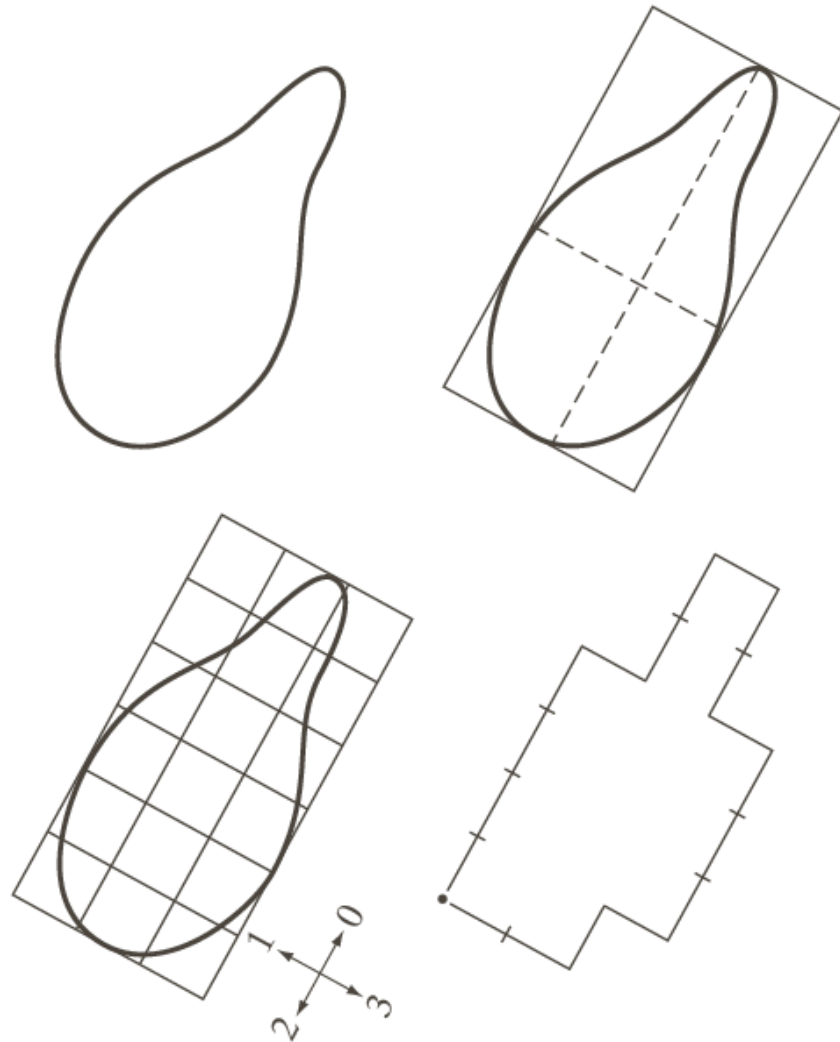


Figure 8.12 All shapes of order 4, 6, and 8. The directions are from Fig. 8.1(a), and dot indicates the starting point.

# Aligning Axes for chain codes & Shape number



a	b
c	d

**FIGURE 11.18**  
Steps in the  
generation of a  
shape number.

---

Chain code: 0 0 0 0 3 0 0 3 2 2 3 2 2 2 1 2 1 1

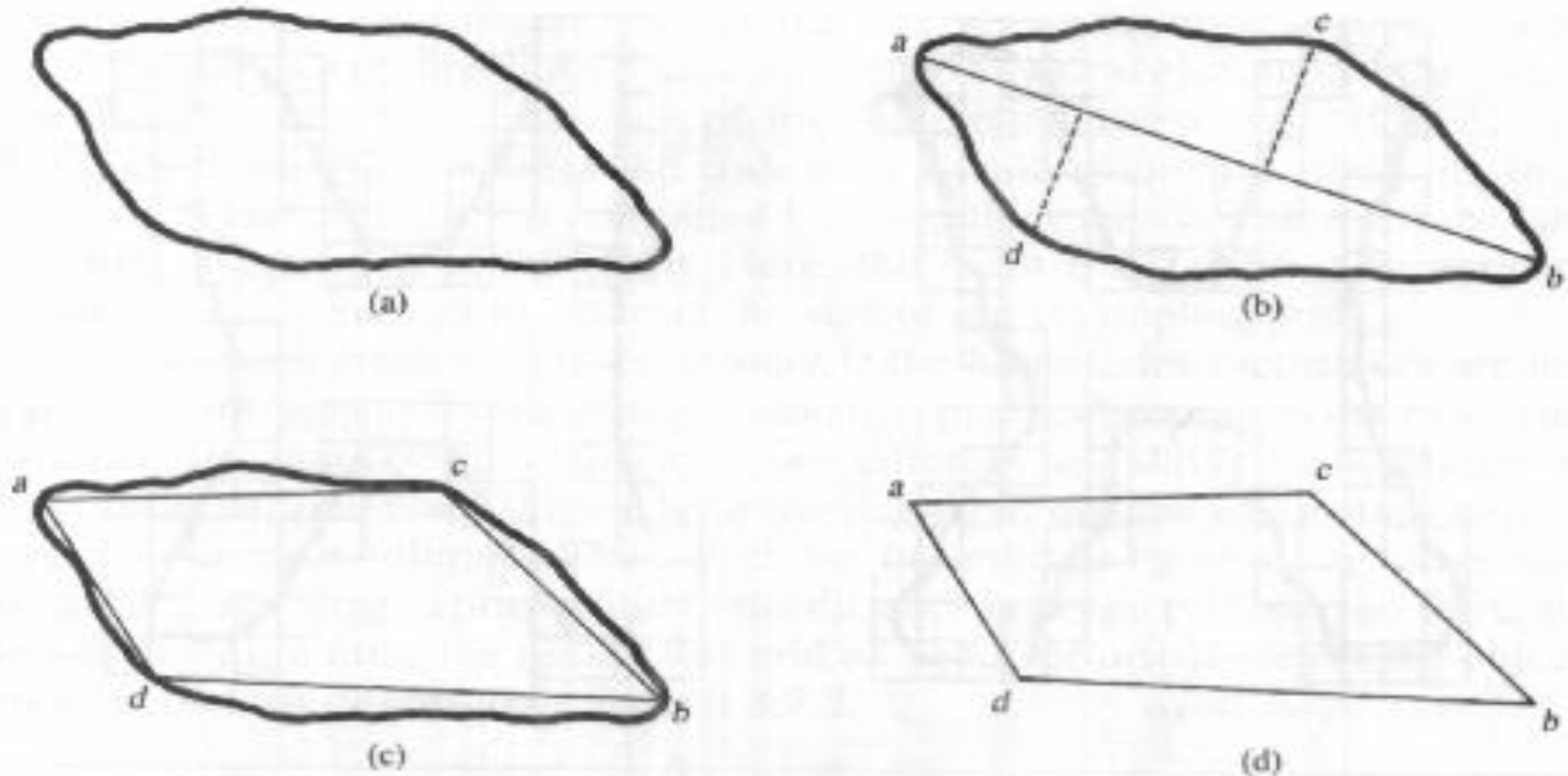
Difference: 3 0 0 0 3 1 0 3 3 0 1 3 0 0 3 1 3 0

Shape no.: 0 0 0 3 1 0 3 3 0 1 3 0 0 3 1 3 0 3

# Polygonal approximation

- A method to approximate curves by line segments of arbitrary orientation.
- One approach is to subdivide a curve segment into 2 parts until a criterion is satisfied.
- A suitable criterion may be the largest distance between any point on the curve to the chord joining the two end points should be less than a specified value.
- An example is shown on the next slide.

# Example: Polygonal approximation



**Figure 8.4** (a) Original boundary; (b) boundary divided into segments based on distance computations; (c) joining of vertices; (d) resulting polygon.

# Curvature-based approximation

- If the boundary contour is represented by  $s(k)=[x(k),y(k)]$ ,  $k=0,\dots,N-1$ , then we can compute the curvature.
- Before computing the curvature, the boundary curve should be smoothed to reduce the effect of noise.
- 1-D Gaussian smoother is applied separately to  $x$  and  $y$  coordinates.
- **Extreme curvature points** are the break points or **geometric corner points** of the curve.

# Gaussian smoothing

- Gaussian mask:

$$h(k, \omega) = \frac{1}{\sqrt{2\pi\omega}} \exp[-0.5(k / \omega)^2]$$

- Curvature is defined as follows:

$$\kappa(k, \omega) = \frac{\dot{X}\ddot{Y} - \dot{Y}\ddot{X}}{(\dot{X}^2 + \dot{Y}^2)^{3/2}}$$

- 1st and 2nd order differences:

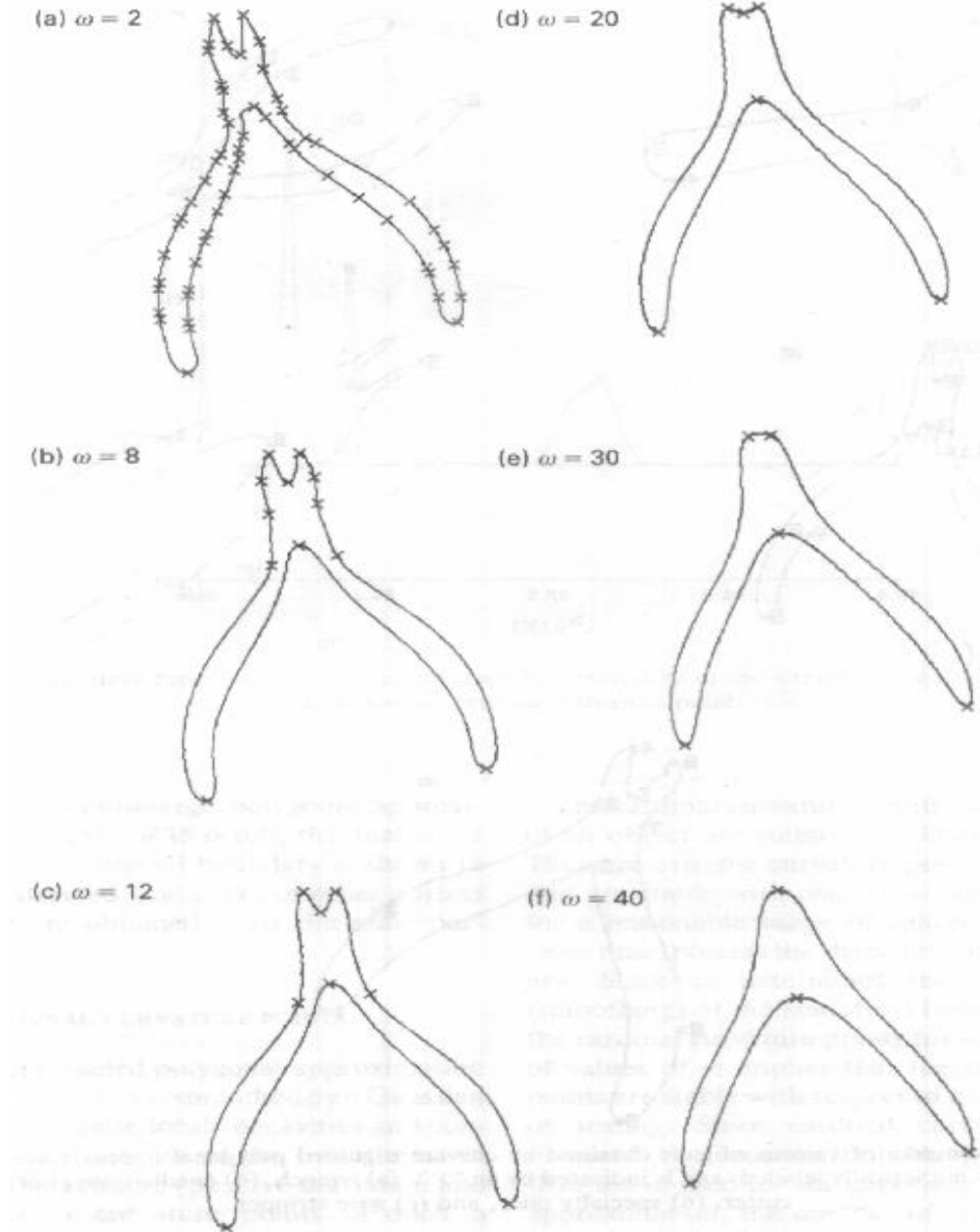
$$\dot{X} = X(k+1) - X(k-1); \quad \ddot{X} = X(k+1) + X(k-1) - 2X(k)$$

- find a suitable width  $\omega$  for the smoother to obtain consistent corner points.



# Example: Corner vs Gaussian width

- The figure shows the effect of smoothing using various width Gaussian masks.
- We can see that omega value of 12 gives fairly stable corner points.



# Fourier descriptors (FDs)

- The FD is the coefficient of the DFT of the boundary coordinates  $s(k)=[x(k),y(k)]$ .
- $S(k)=x(k)+jy(k)$ . Now obtain the DFT
- the DFT equations:

$$a(u) = \frac{1}{N} \sum_{k=0}^{k=N-1} s(k) \exp[-2j\pi uk / N]$$

where  $a(u)$  are the FDs.

- The curve can be reconstructed by inverse DFT.
- These DFTs can be used to represent the curve in a recognition application.

# FDs of transformed object

- The table shows FD after transformation such as Rotation, Translation, Scale, Shift in starting point of the curve.
- Can you show that these 4 relations given in the table below hold, given the “identity”? (homework)
- How can we make the FD invariant to R-T-S-S? (i.e. how to make the FD comparable to “identity” even after R-T-S-S?) (homework & past exam question)

Transformation	Boundary	FD
identity	$s(k)$	$a(u)$
Rotation	$s_r(k) = s(k)e^{j\theta}$	$a(u)e^{j\theta}$
Translation	$s_t(k) = s(k) + \Delta_{xy}$	$a(u) + \Delta_{xy}\delta(u)$
Scaling	$s_s(k) = \alpha s(k)$	$\alpha a(u)$
Starting point	$s_p(k) = s(k - k_o)$	$a(u)e^{-j2\pi k_o u / N}$

# Region-based features

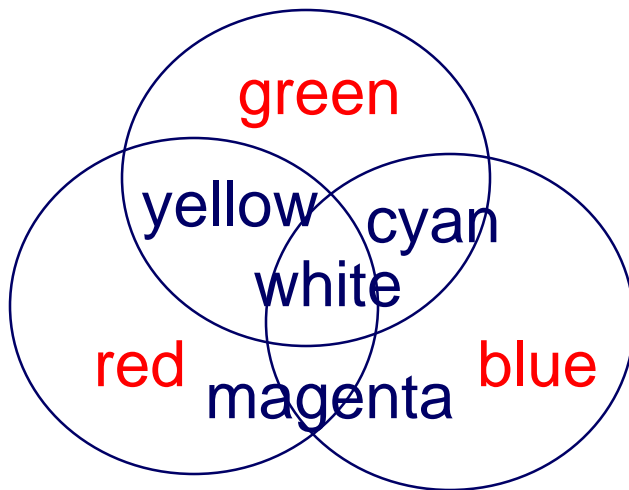
- Area, perimeter -- invariant to rotation, translation
- projections, histograms of regions.
- compactness= $\text{perimeter}^2 / \text{area}$ .
- Euler number= number of connected components - total number of holes (suitable for binary images).
- If it is possible to segment the image into uniform regions, then the above features as well as color/texture (if applicable), moments can also be used to describe interior regions.

# Color attributes (refer to Gonzalez & Woods)

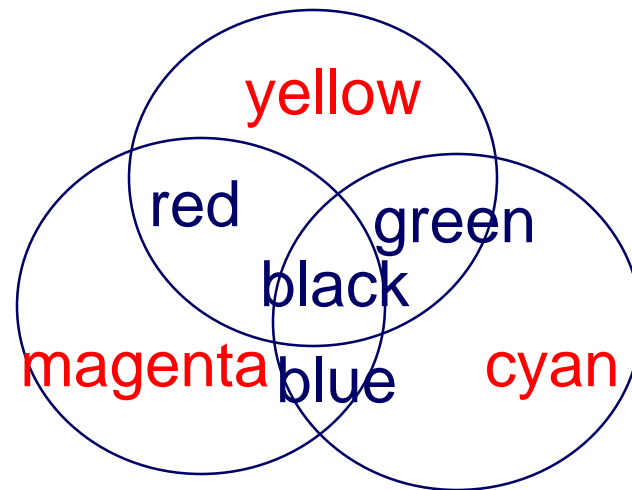
Why color images ?

- Color is a powerful description that often simplifies object identification and extraction from a scene.
- human eye can discern thousands of color shades and intensities compared to only 24 shades of grays
- Types of color images:
  - full color (24-bits), pseudo-color (8-bits indexed)
- Color is often distinguished by 3 attributes:
  - brightness : the chromatic notion of intensity
  - hue : represents the color as perceived by an observer
  - saturation : the relative purity or the amount of white light mixed with a hue.

- Hue and Saturation are chromatic attributes.
- Color can be characterized by
  - brightness (luminance) or intensity
  - Chromaticity (Hue & Saturation)
- Color Primaries: Red(R), Green(G), Blue(B)



Mixture of light  
(additive primaries)



Mixture of pigments used in  
printers (subtractive primaries)  
(i.e. yellow surface subtracts  
blue from white light)

- The amount of RGB needed to form a color is called *Tristimulus values*, denoted as  $X$ ,  $Y$ ,  $Z$ .
- A color is defined as

$$x = \frac{X}{X + Y + Z} \quad y = \frac{Y}{X + Y + Z} \quad z = \frac{Z}{X + Y + Z}$$

$$x + y + z = 1$$

- Color models are used to specify colors in some standard generally accepted ways.
- A color model is a specification of a coordinate system (usually 3D).

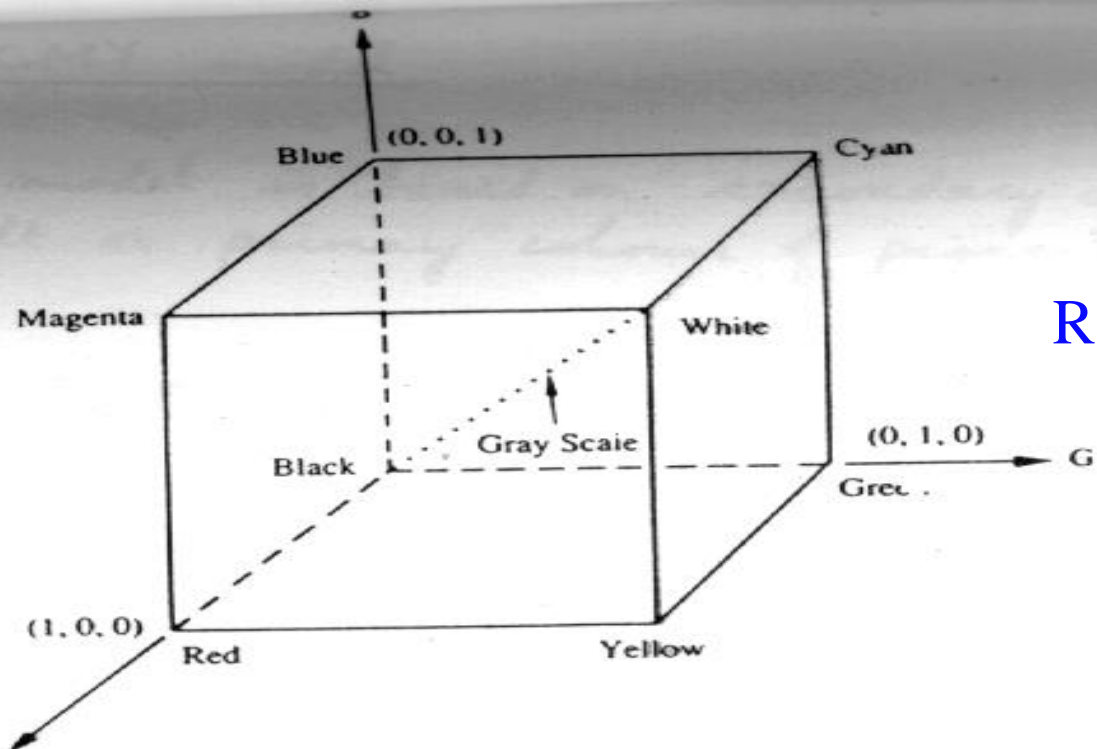
## Common Color Spaces:

- RGB - hardware oriented, used for monitors, video cameras
  - rgb - Normalized RGB
  - CMY (Cyan-Magenta-Yellow) - used for color printer
  - YIQ (luminance, in-phase, quadrature. ) - color TV broadcast (Y has complete information for monochrome TV).
  - HSI (HSV) (Hue, saturation, intensity/value) - used for color manipulation
  - CIE-Lab (lightness, red-green, yellow-blue) - used for color differentiation.
- RGB, YIQ and HSI are often used in image processing

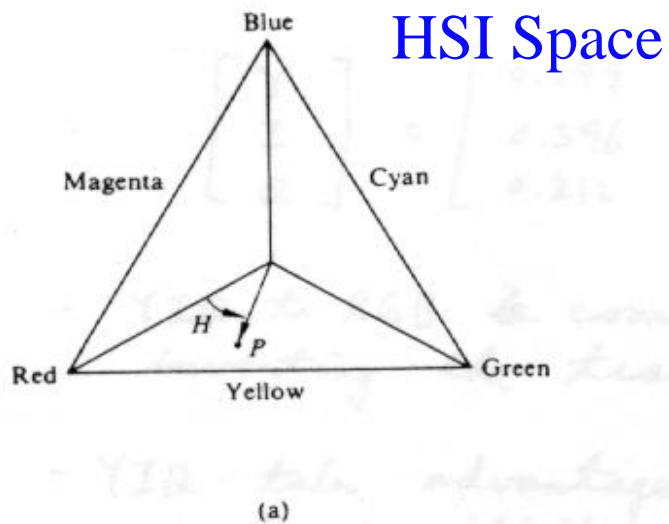


# The HSI model:

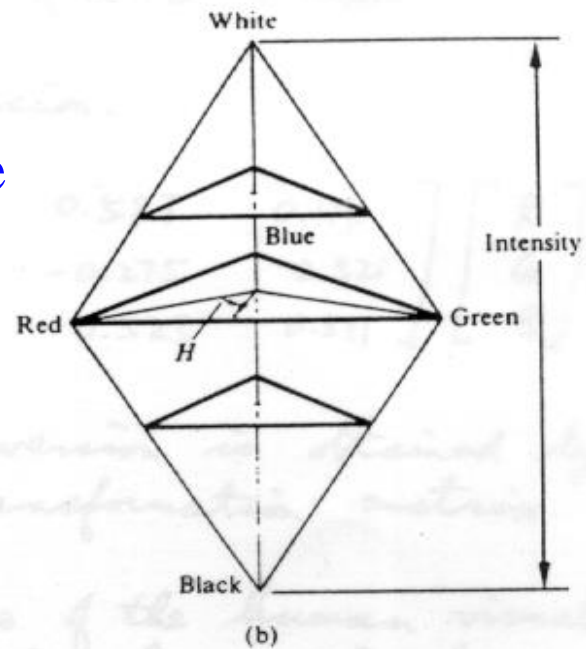
- useful due to 2 main reasons:
  - Intensity component (I) is decoupled from the color information in the image
  - Hue (H) and saturation (S) components are related to the way in which human perceive color
- Ideal tool for developing image processing algorithms based on some color sensing properties of the human visual system.
  - E.g. determining the ripeness of fruits and vegetable, inspecting quality of finished color surfaces.
- Not perceptually uniform.



RGB Space



HSI Space



## From RGB to HSI Conversion:

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases}$$

where

$$\theta = \cos^{-1} \left\{ \frac{0.5[(R - G) + (R - B)]}{[(R - G)^2 + (R - B)(G - B)]^{0.5}} \right\}$$

and

$$S = 1 - \frac{3}{R + G + B} [\min(R, G, B)]$$

$$I = \frac{R + G + B}{3}$$

## From HSI to RGB:

- Sector  $0^\circ \leq H < 120^\circ$

$$R = I \left[ 1 + \frac{S \cos H}{\cos(60^\circ - H)} \right]$$

$$B = I(1 - S) \quad G = 3I - (R + B)$$

- Sector  $120^\circ \leq H < 240^\circ$

$$G = I \left[ 1 + \frac{S \cos H}{\cos(60^\circ - H)} \right]$$

$$R = I(1 - S) \quad B = 3I - (R + G)$$

- Sector  $240^\circ \leq H < 360^\circ$

$$B = I \left[ 1 + \frac{S \cos H}{\cos(60^\circ - H)} \right]$$

$$G = I(1 - S) \quad R = 3I - (G + B)$$

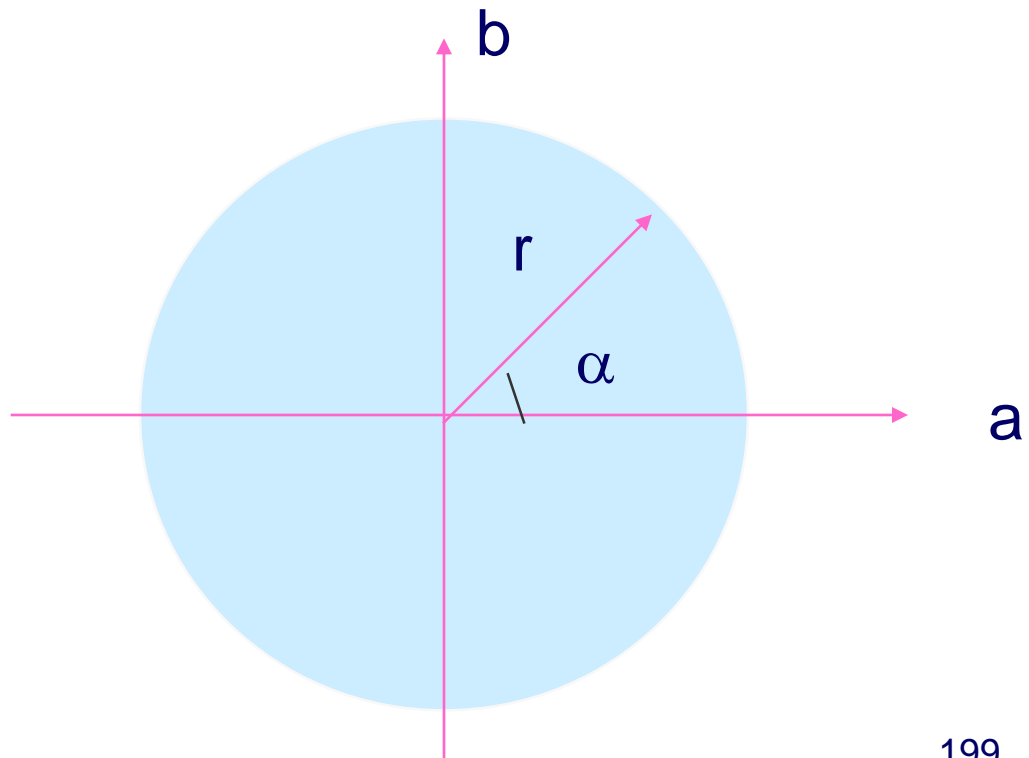
# Perceptually Uniform Color Space:

- A color space designed for measuring color difference in a way that is perceptually agreed, *i.e.* distance in Euclidean distance matches our visual differencing.
- Conversion from RGB involves:
  - RGB to XYZ
  - XYZ to Lab

L - lightness

r - saturation

$\alpha$  - hue



– Reference white :  $R=G=B=1$

$$\begin{bmatrix} X \\ Y \\ Z \end{bmatrix} = \begin{bmatrix} 0.607 & 0.174 & 0.200 \\ 0.299 & 0.587 & 0.114 \\ 0 & 0.066 & 1.116 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

$$L = 25(100Y / Y_0)^{1/3} - 16; \quad 1 \leq 100Y \leq 100$$

$$a = 500[(X / X_0)^{1/3} - (Y / Y_0)^{1/3}]$$

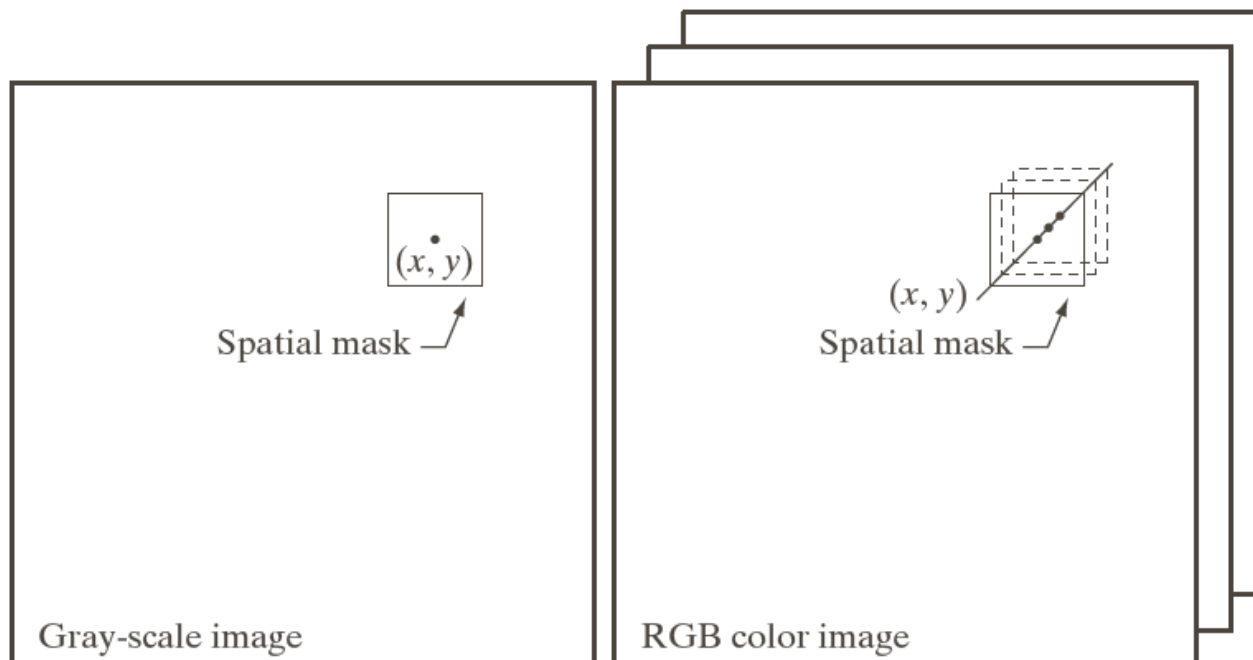
$$b = 200[(X / X_0)^{1/3} - (Z / Z_0)^{1/3}]$$

Color difference,  $\Delta C^2 = (\Delta L)^2 + (\Delta a)^2 + (\Delta b)^2$

–  $(X_0, Y_0, Z_0)$  is the reference white

# Color Image Processing:

- Since color images have 3 image planes, we can perform the processing independently on each plane.
- If we use the HSI color representation, then it is also possible to perform the operation only on the intensity component,  $I$  while the color information in  $H$  &  $S$  are left unchanged.



a b

**FIGURE 6.29**  
Spatial masks for  
gray-scale and  
RGB color  
images.

## Noise in Color Images:

- As color images are usually captured in RGB space, it can be advantageous to perform noise cleaning operations in RGB space.
- In particular, if noise affects only 1 component during image capture, then it is necessary to apply noise cleaning only to that component
- Converting the RGB (with noise only in 1 component) to HSI, will spread the noise to all 3 components of the HSI.



## Smoothing for Color Images:

- We can employ the same methods as we used for grey scale images with only one difference: We perform the smoothing separately for each color plane (*i.e.* RGB, CMY, HSI, etc.).
- It is also possible to do the smoothing of intensity only if we use the HSI color space, *i.e.* do the smoothing of  $I$  component only, if we know the noise is in  $I$  only.

## Sharpening for Color Images:

- It is possible to obtain the Laplacian required for sharpening in all 3 color planes separately. If we use the HSI color space, we can sharpen only the intensity component,  $I$ .

# Potential features/attributes

- Grey intensity and its mean variance, histograms, projections in horizontal/vertical/diagonal directions
- Area, perimeter, shape numbers, chain codes, number of holes, and so on
- Colour, texture and their properties (HSI, co-occurrence, texture energy), gray level corner points.
- Geometric corner points, curvature, line or curve segments
- Moment invariants, Fourier descriptors
- 3-D surface types/area, principal curvatures (you'll study some of these in the next 5 lectures.)
- There is a need to consider the invariance of the chosen features and required invariance by the application.