

# Lecture-2

## ECE4810/ECE5810

### Spring 2019

#### **Lecture Summary:**

1. Digital Images
2. Spatial Resolution
3. Intensity Resolution
4. Levels of Computations
  - Point
  - Local
  - Global
  - Object
5. Background
6. Look-up tables
7. Contrast Stretching

# A SIMPLE IMAGE MODEL

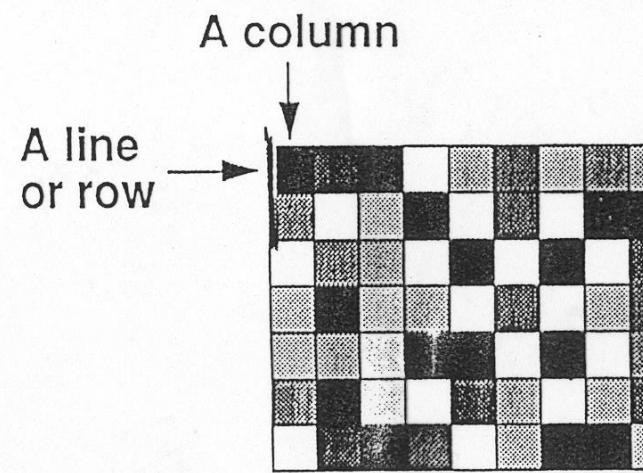
- image - 2D light intensity function  $f(x,y)$ , where the value of  $f$  at spatial coordinates  $(x,y)$  gives the intensity of the image at that point:  
$$0 < f(x,y) < \infty$$
- the intensity of image  $f$  at coordinates  $(x,y)$  is the gray level ( $l$ ) of the image  
$$L_{\min} \leq l \leq L_{\max}$$
- the interval  $[L_{\min}, L_{\max}]$  is called the gray scale;  
typical scale for 8-bit images:  $[0,255]$ , where 0 is considered black, 255 - white, and all intermediate values are shades of gray varying from black to white
- continuous image  $f(x)$  is approximated by equally spaced samples arranged in the form of an  $N \times M$  array:

$$f(x) \approx \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,M-1) \\ f(1,0) & f(1,1) & \dots & f(1,M-1) \\ \vdots & \vdots & \ddots & \vdots \\ \vdots & \vdots & \ddots & \vdots \\ f(N-1,0) & f(N-1,1) & \dots & f(N-1,M-1) \end{bmatrix} \quad \xleftarrow{\text{digital image}}$$

- usually  $N$  and  $M$  are integer powers of two:  $N=2^n$ ,  $M=2^k$ ; also - number of gray  $G=2^m$ ; therefore number of bits required to store an image:  $b=N \times M \times m$ ; for example: 128 by 128 image with 256 gray levels requires 131,072 bits = 16,0284 bytes = 16KB of storage

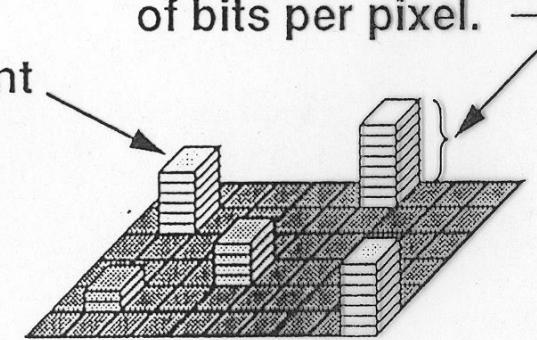
# The Digital Image

Spatial resolution is number of columns by number of lines.



A "pixel" or picture element

Intensity resolution or "depth" is number of bits per pixel.



# Examples

## Spatial resolution

4 x 4  
16 x 16  
128 x 64  
256 x 256  
512 x 512  
1024 x 1024  
etc.

## Intensity resolution

8 - bit unsigned  
8 - bit signed  
16 - bit unsigned  
16 - bit signed  
32 - bit unsigned  
32 - bit signed  
etc.

# Signed and Unsigned Bytes

- unsigned bytes

Value	Binary
0	0000 0000
1	0000 0001
127	0111 1111
128	1000 0000
254	1111 1110
255	1111 1111

- signed bytes

Value	Binary
-128	0000 0000
-127	0000 0001
-1	0111 1111
0	1000 0000
1	1111 1110
127	1111 1111

- to change the sign of a stored number you 'flip' all of the bits (0s to 1s and 1s to 0s), and then add 1 - this convention is called twos-complement
- you must know whether your image is signed or unsigned byte - there is no way to know by looking at the data!
- image transfer - always set the transfer mode to binary

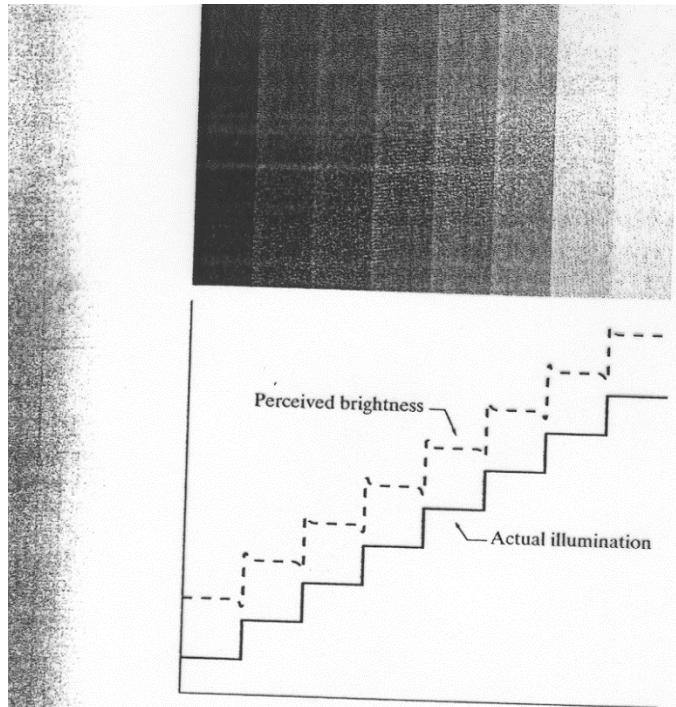
*For Real Techies...*

## Twos-Complement vs. Ones-Complement

Most computers currently use twos-complement arithmetic as described above, but back in the computational stone ages, machines such as the Control Data Cybers used ones-complement arithmetic. In ones-complement, to change the sign of a stored number you just flip all of the bits. No 'add by 1.'

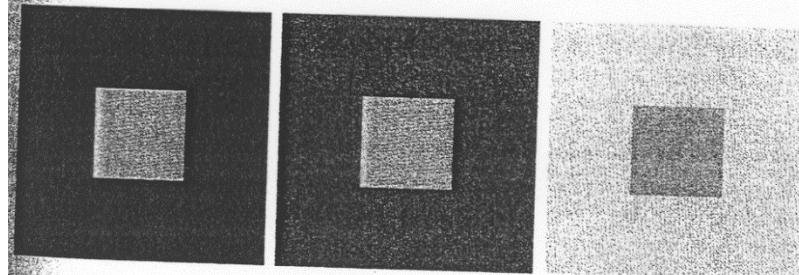
This was thought to be more efficient, but it created a problem— now there were two representations for zero: 00000000 and 11111111b. Programs had to check for 'positive' zero and 'negative' zero (!).

*Ask an old Cyber programmer  
about negative zero.  
Carefully.*



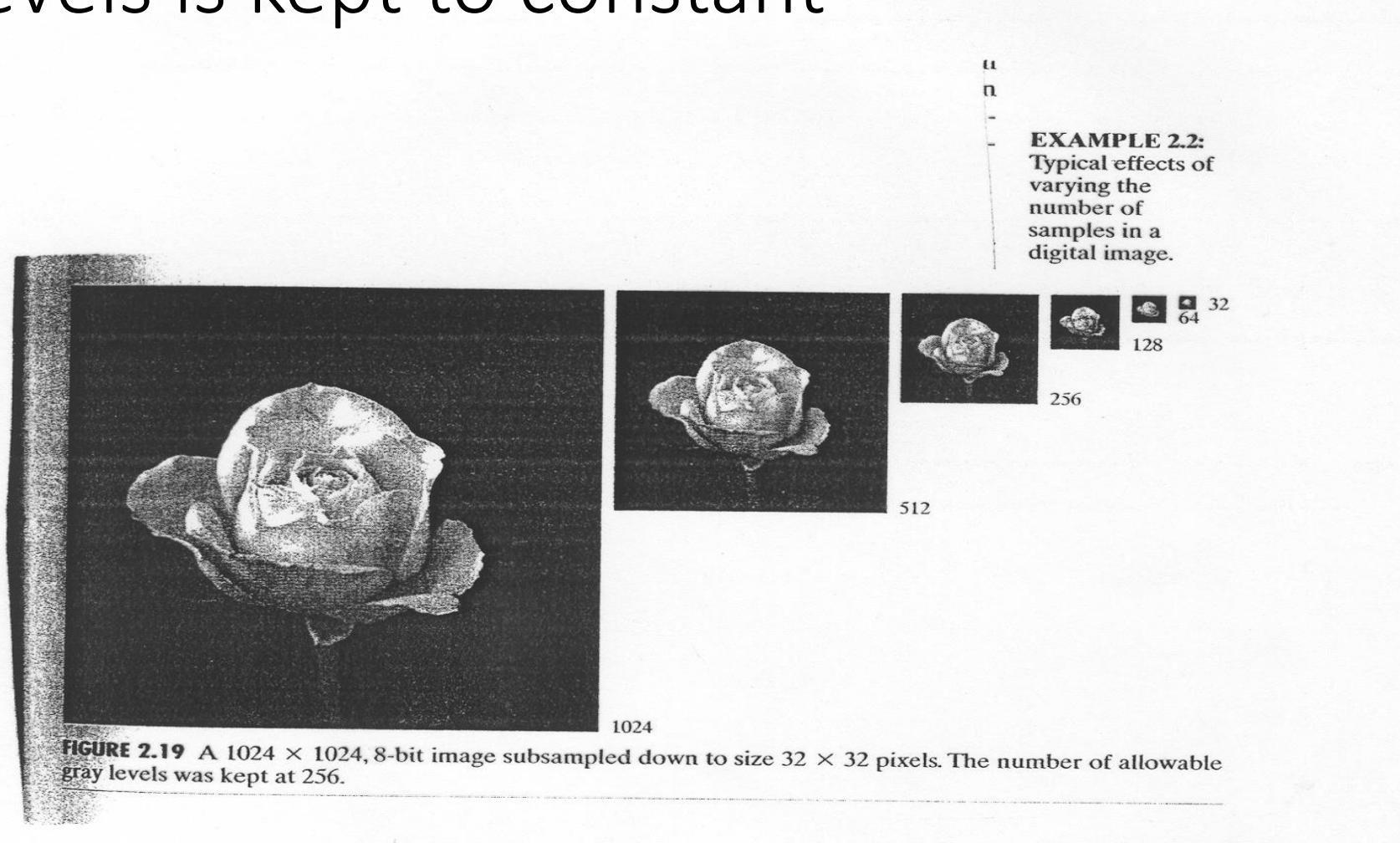
**FIGURE 2.7**  
 (a) An example showing that perceived brightness is not a simple function of intensity. The relative vertical positions between the two profiles in (b) have no special significance; they were chosen for clarity.

However, they appear to the eye to become darker as the background gets lighter. A more familiar example is a piece of paper that seems white when lying on a desk, but can appear totally black when used to shield the eyes while looking directly at a bright sky.

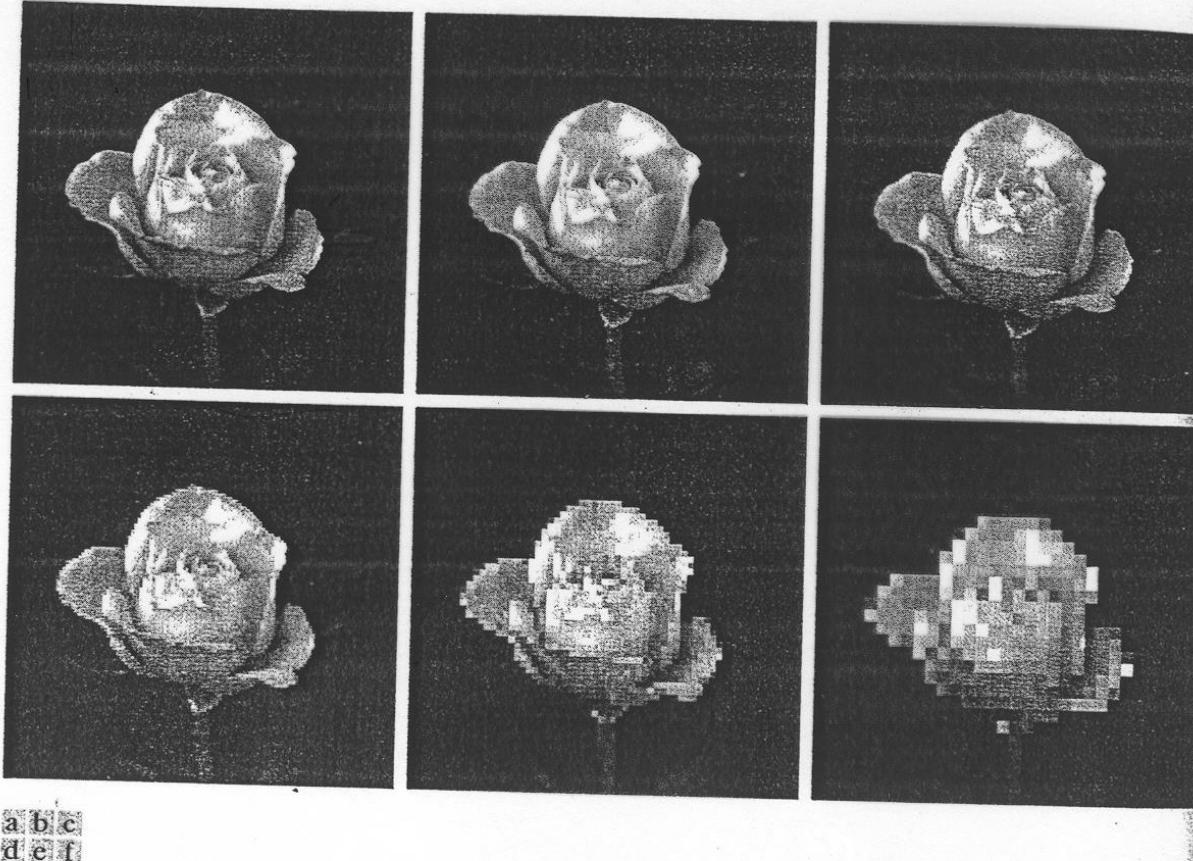


**FIGURE 2.8** Examples of simultaneous contrast. All the inner squares have the same intensity, but they appear progressively darker as the background becomes lighter.

# Effect of Image size when the number of allowable gray levels is kept to constant



# Effect of image sampling



**FIGURE 2.20** (a)  $1024 \times 1024$ , 8-bit image. (b)  $512 \times 512$  image resampled into  $1024 \times 1024$  pixels by row and column duplication. (c) through (f)  $256 \times 256$ ,  $128 \times 128$ ,  $64 \times 64$ , and  $32 \times 32$  images resampled into  $1024 \times 1024$  pixels.

# Effect of variation of the number of pixels used

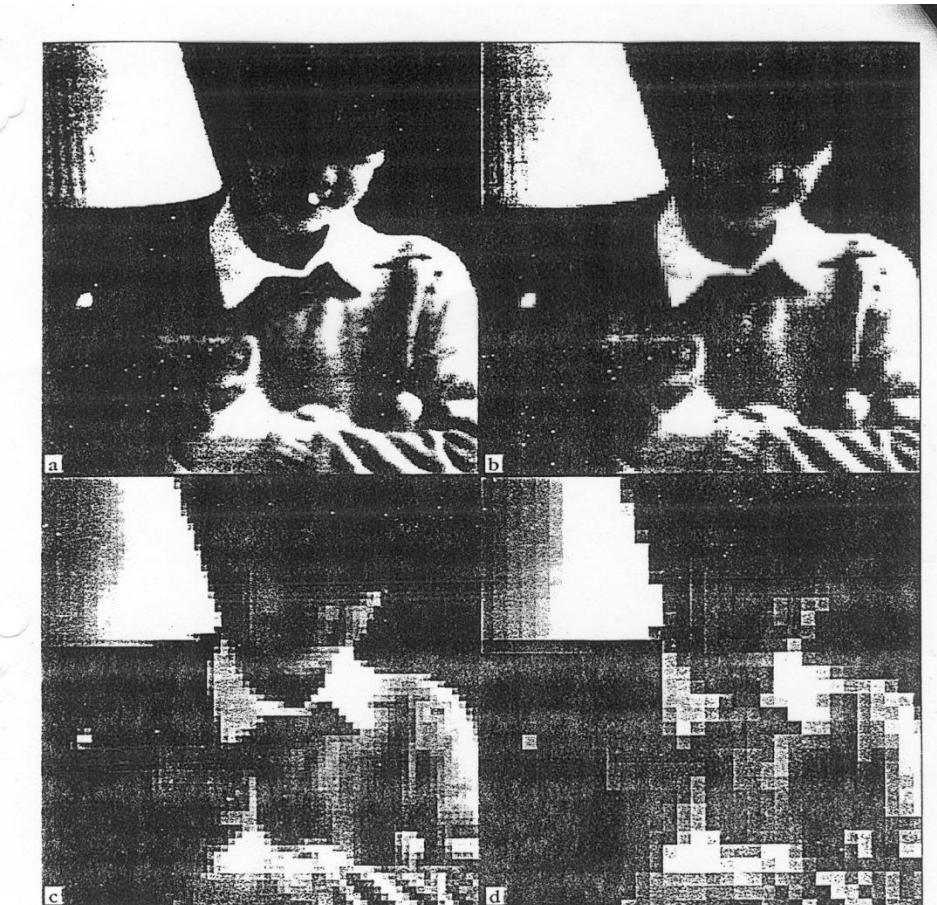


Figure 25. Four representations of the same image, with variation in the number of pixels used:

a)  $256 \times 256$ ; b)  $128 \times 128$ ; c)  $64 \times 64$ ; d)  $32 \times 32$ .

In all cases, a full 256 grey values are retained. Each step in coarsening of the image is accomplished by averaging the brightness of the region covered by the larger pixels.

# Effect on the image of the number of gray levels used

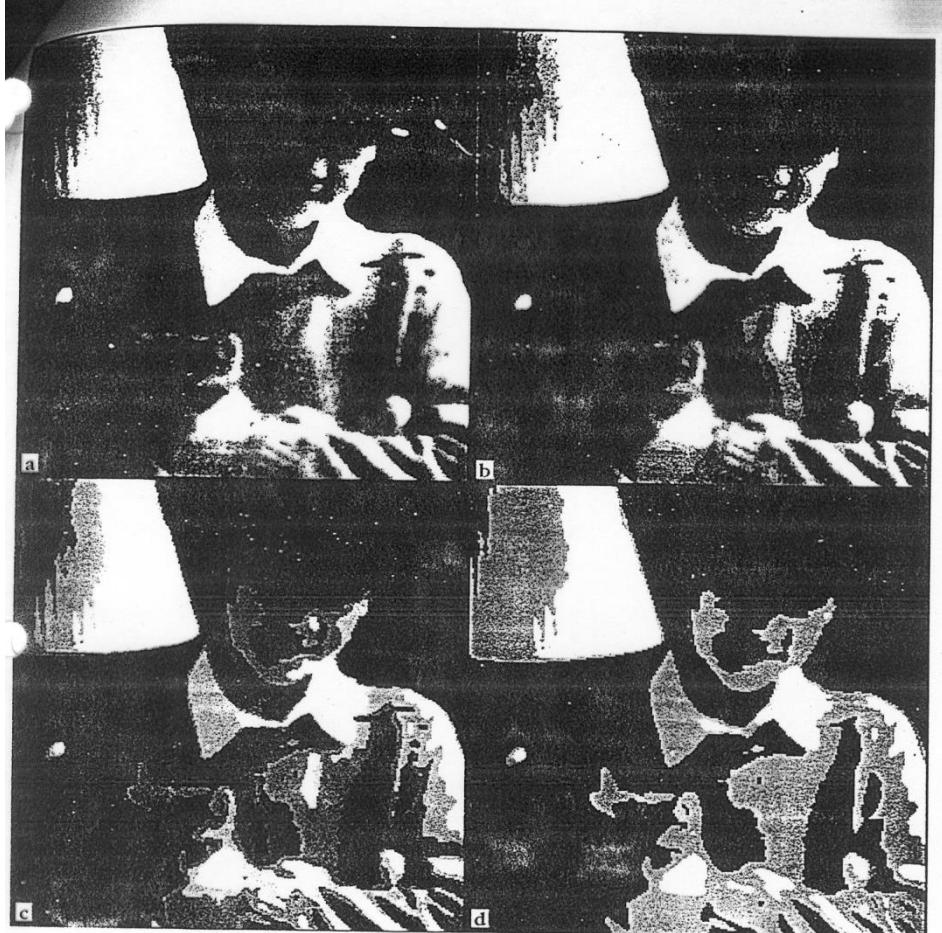


Figure 26. Four representations of the same image, with variation in the number of grey levels used: a) 32; b) 16; c) 8; d) 4.  
In all cases, a full  $256 \times 256$  array of pixels are retained. Each step in the coarsening of the image is accomplished by rounding the brightness of the original pixel value.

# Levels of Computation

an image usually consists of several objects; a vision application usually involves computing certain properties of an object, not the image as a whole

first, the individual objects must be identified - using algorithms for connectivity and segmentation

**point level processing** - operations that produce an output based on only a point in an image

**local level processing** - operations that produce an output image in which the intensity at a point depends on the **neighborhood** of the corresponding point in the input image

**global level processing** - operations that produce an output (not necessary an output image) which depends on the whole input image

**object level processing** - operations that produce object characteristics, such as size, average intensity, shape. etc.

# Point Level Processing

Point processes are the simplest of basic image processing operations. A point operation takes a single input image into a single output image in such a way that each output pixel's gray level depends only upon the gray level of the corresponding input pixel. Thus, a point operation cannot modify the spatial relationships within an image.

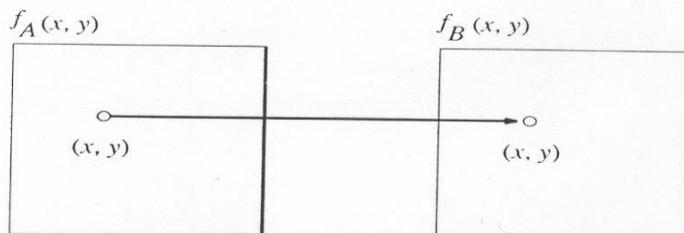
$$f_B[i, j] = O_{\text{point}}\{f_A[i, j]\}$$

where  $f_A$  - input image and  $f_B$  - output image

- point operations transform the gray scale of an image
- linear and nonlinear point operations
- they are useful in photometric calibration, display calibration, enhancement and histogram modification

examples:

Contrast Stretching  
Image Negatives  
Intensity-level Slicing  
Bit-plane Slicing  
Histogram Equalization



salted binary image

← Thresholding  
is done using  
Point operator.

There is a difference between image1 and image3:



image1



image3

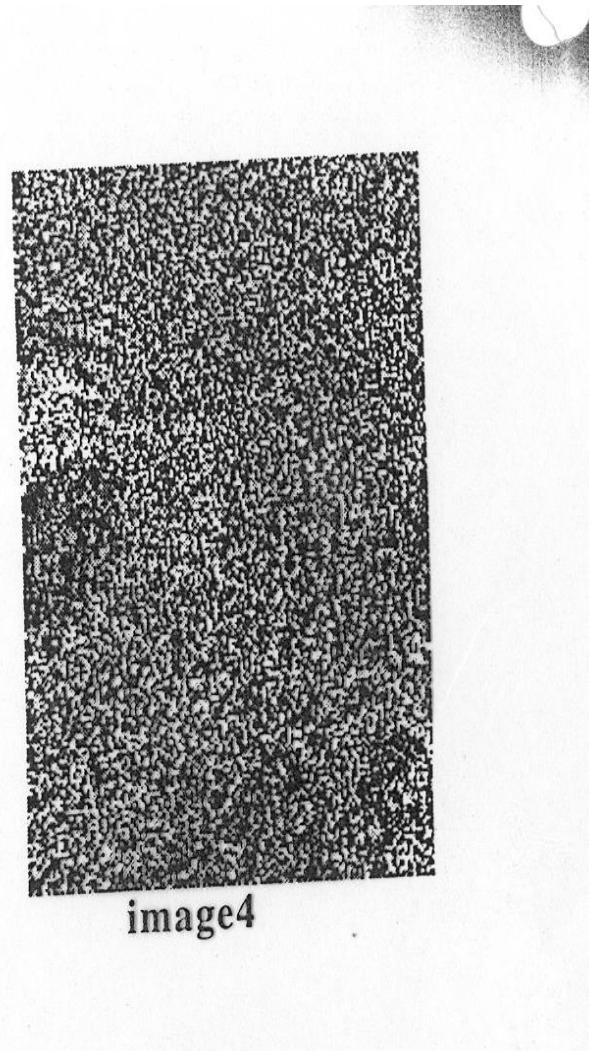


image4

# Local Level Processing

A local operation takes a single input image into a single output image in such a way that each output pixel's gray level depends upon the gray levels of the pixels in the neighborhood of the corresponding input pixel.

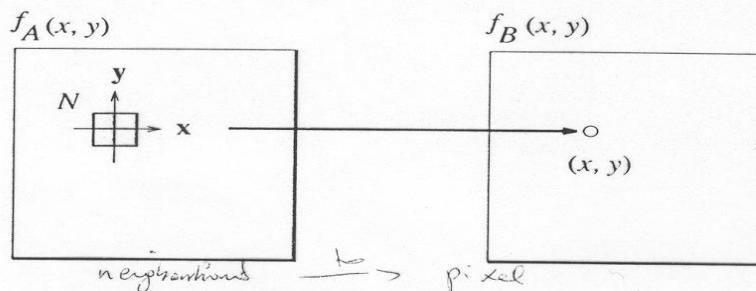
$$f_B[i, j] = O_{\text{local}} \{ f_A[i_k, j_l]; [i_k, j_l] \in N[i, j] \}$$

where  $f_A$  - input image and  $f_B$  - output image  $N[i, j]$  - neighborhood of  $[i, j]$

area processing typically is formulated in the context of so-called mask operations (the terms **template**, **kernel**, **window**, **filter**, and **convolution mask** are also used to denote a mask); mask is an array of numbers (typically 3x3 or 5x5, etc.); mask is usually very small with respect to the processed image; mask values are called coefficients; values of coefficients determine the nature of the process

examples:

smoothing  
sharpening  
median  
max and min



← smoothing

# GLOBAL LEVEL PROCESSING

A global operation takes a single input image into a single output image (or a single symbolic output) in such a way that the output depends on the whole input image

*symbolic output*

$$P = O_{\text{global}}\{f_A[i, j]\}$$

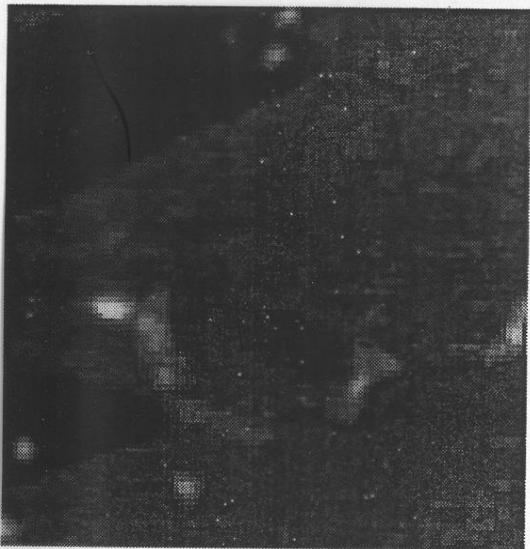
where  $f_A$  - input image and  $P$  - output image or symbolic output

- examples:

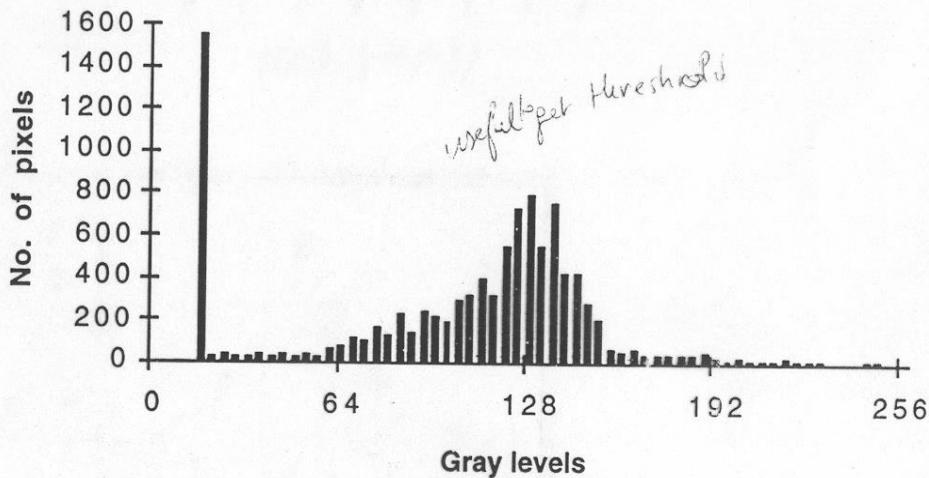
**histogram**  
**Fourier transform**  
**image subtraction**  
**averaging**  
**logical operations**

# Histogram

- gray level histogram of an image - a function showing for each gray level the number of pixels in the image that have that gray level; it is simply a bar graph of the pixel intensities



CT image



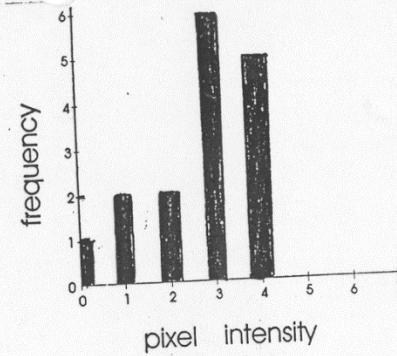
Histogram of the CT image

- histogram gives us a convenient, easy-to-read representation of concentration of pixels versus intensity in an image
- dynamic range - a range of intensity values that occur in an image
- contrast stretching - if image has low-dynamic range; low-dynamic range can result from poor illumination, lack of dynamic range in imaging sensor, wrong setting of the sensor parameters, etc.
- compression of dynamic range - if the dynamic range of the image far exceeds the capability of the display device

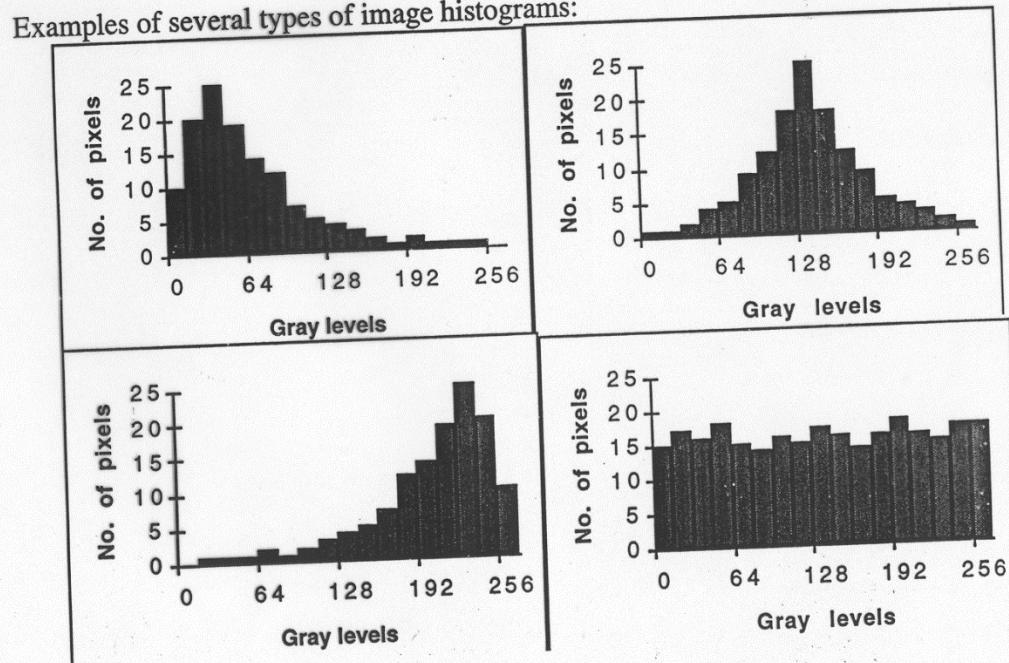
Histogram calculation:

4	4	3	3
4	4	3	3
4	1	2	3
0	1	2	3

Image

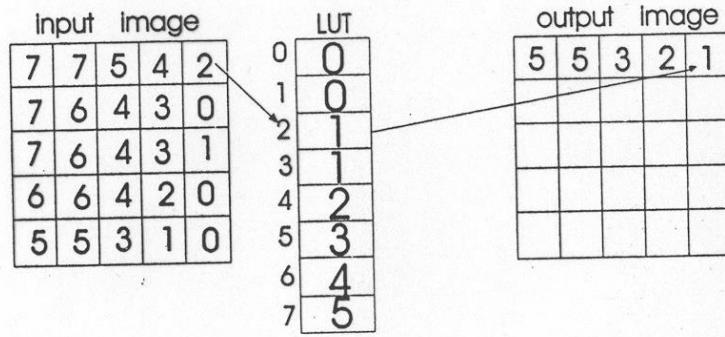


Examples of several types of image histograms:



# Lookup Tables (LUTs)

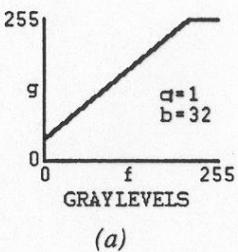
- LUTs are arrays that use the current pixel value as the array index; the new value is the array element pointed by this index:



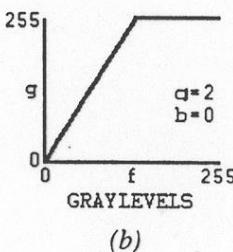
- computational savings - using LUTs avoids needless repeated computations for each pixel; for example, if you were to add some value to every pixel in a 512 x 512 image, it would require 262,144 operations without LUT, and 256 operations with LUT

# Examples

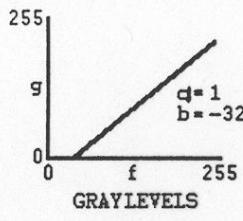
linear LUTs



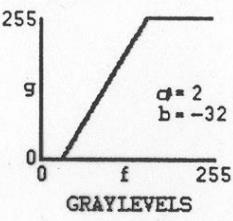
(a)



(b)

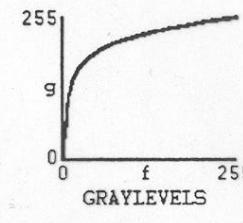


(c)

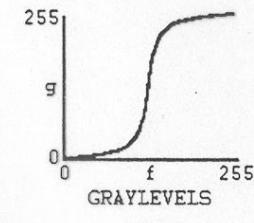


(d)

nonlinear LUTs



(a)

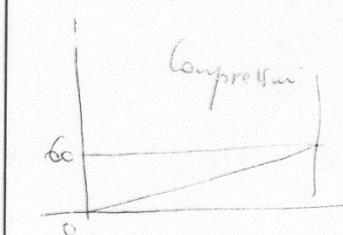
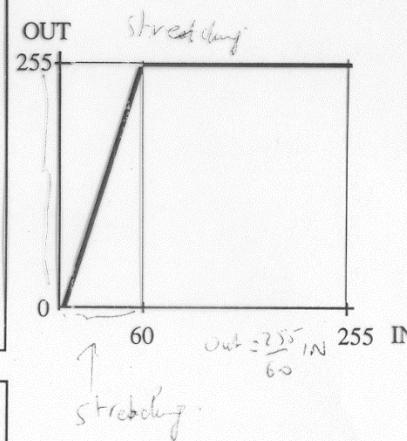
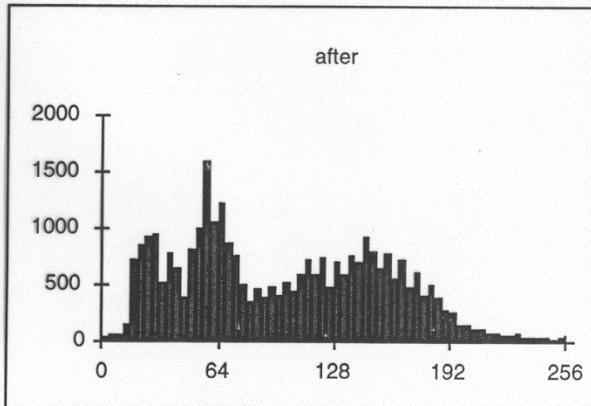
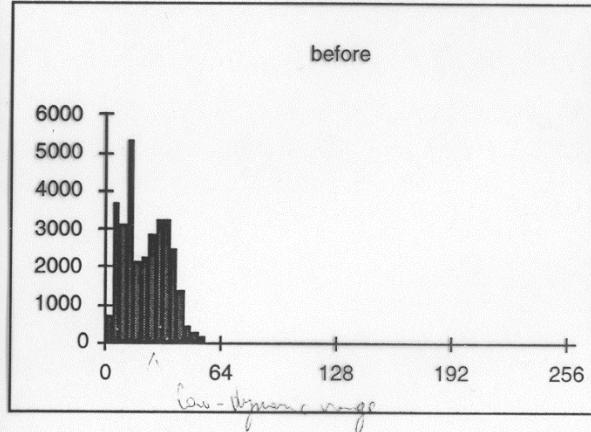


(b)

$$g = 31.9 * \log_2(f+1)$$

$$g = a * f + b$$

# Contrast Stretching



$$Out = \frac{60}{255} IN$$