

## Lectures 3 , 4, and 5:

### **L3:Representing Digital Images;**

**Zooming. Bilinear Bi-cubic interpolations;  
Relationships, Connectivity, Regions, Boundaries;**

### **L4:Arithmetic and Logical Operations with Images.**

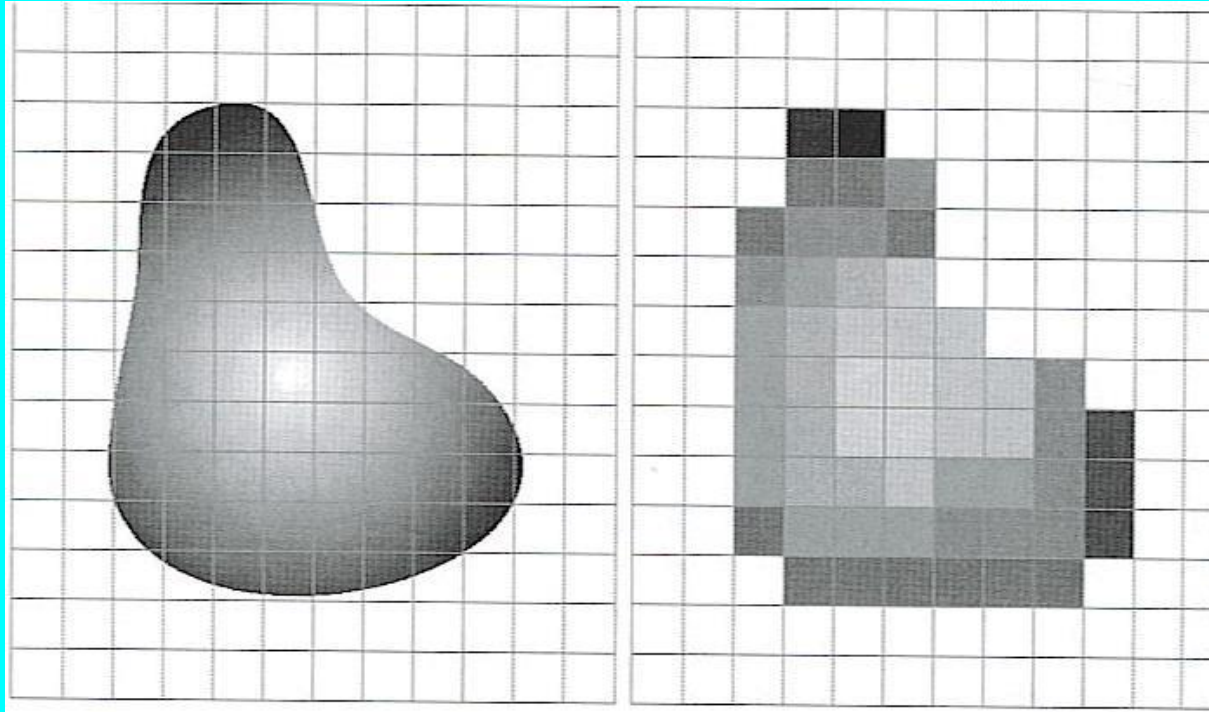
### **L5:Transformations**

**Gray Level, Log, Power-Law, Piecewise-Linear.**

**Experiments with software performing: arithmetic;  
Logic; Log and Power function operations.**

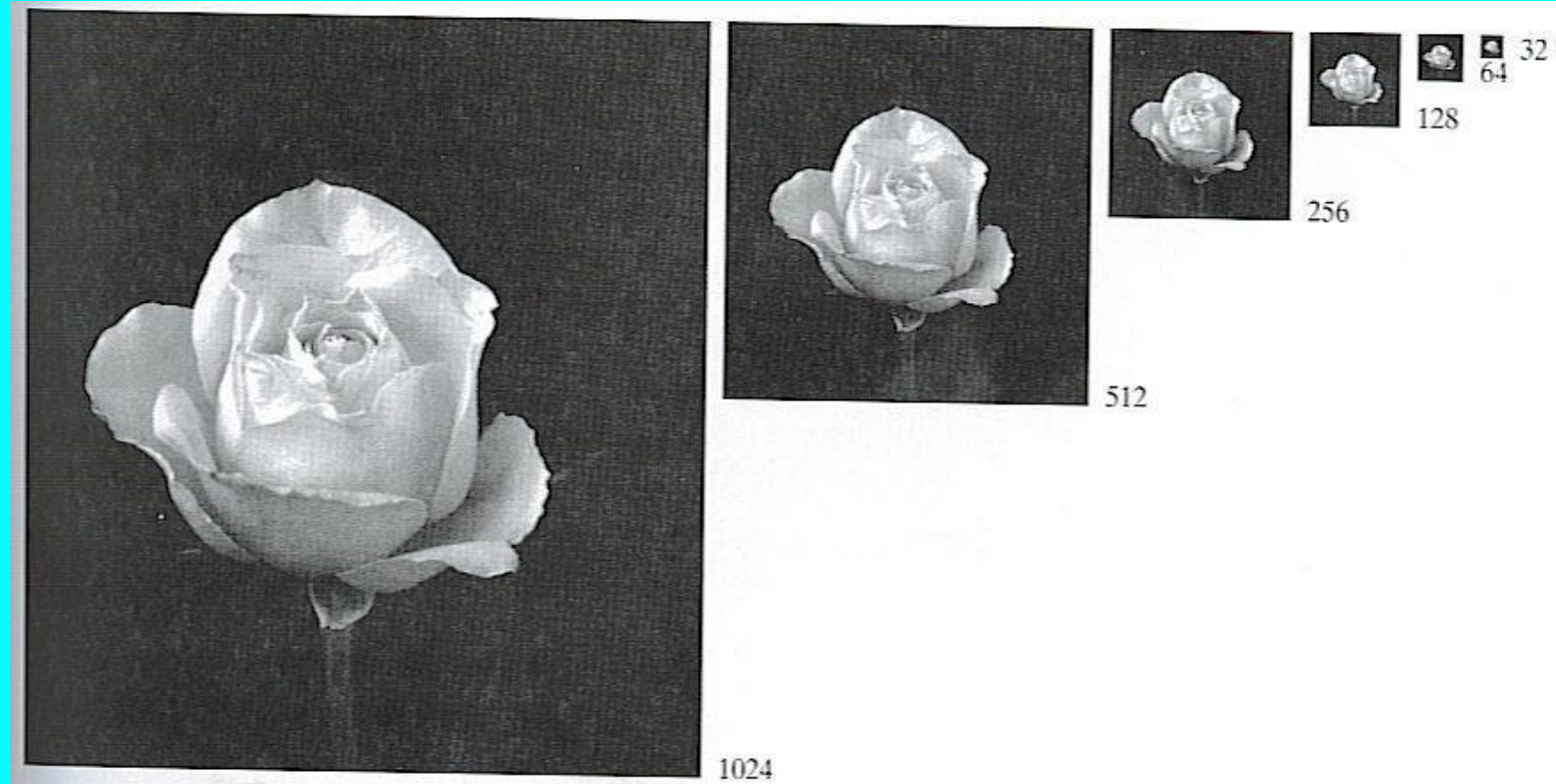
**The software is coded by students of this class- 2005,2008.**

## Image Processing with Applications-CSCI597/MATH597/MATH489



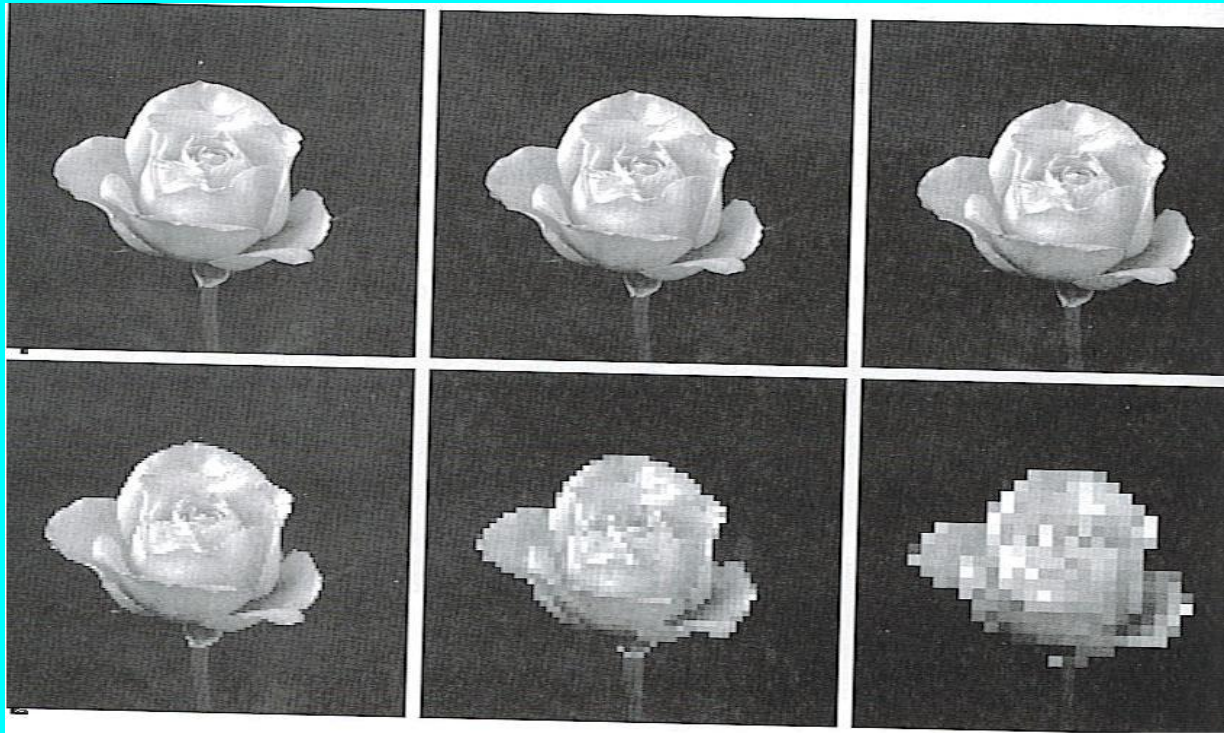
**Figure 1.** a) Continuous image projection onto a sensor array.  
b) Result of sampling and quantization. (**Digital Image Processing, 2<sup>nd</sup> E, by Gonzalez, Richard**).

## Math Definition of an Image



**Figure 2.** Spatial sub-sampling of  $1024 \times 1024$ , 8 bit image.

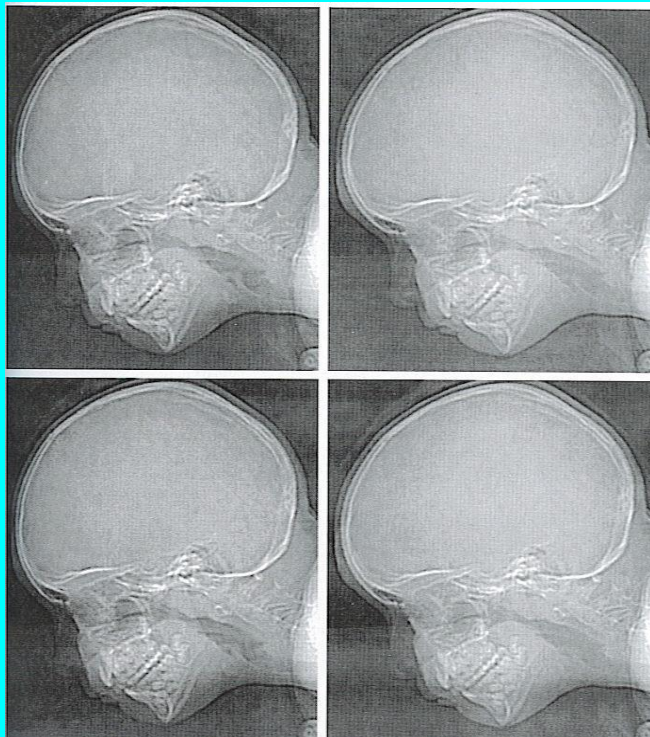
The gray level quantization is the same for all images. Every image is produced from the previous by deleting every other column and row. (Digital Image Processing, 2<sup>nd</sup> E, by Gonzalez, Richard).



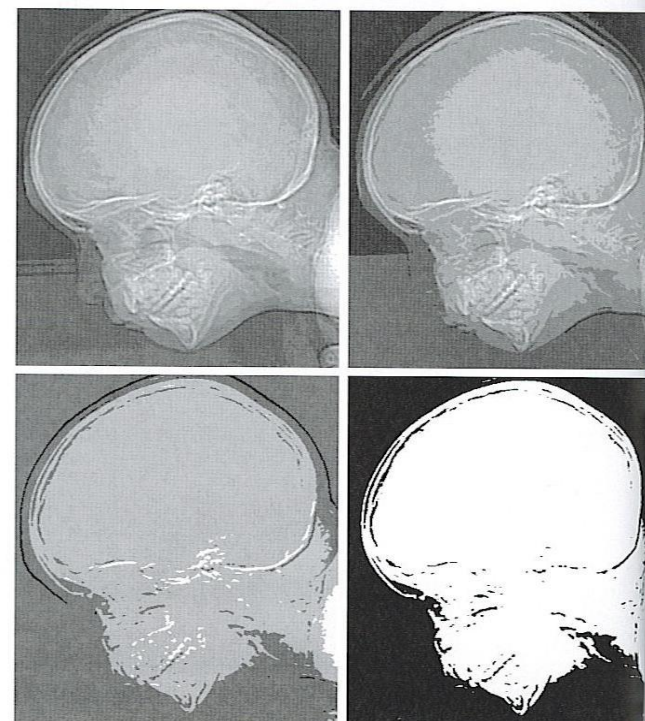
**Figure 3.** Spatial re-sampling of the images from Fig.2, 8 bit image. The size of each image is enlarged to the size of the 1024x1024 image. The gray level quantization is the same for all images. (Digital Image Processing, 2<sup>nd</sup> E, by Gonzalez, Richard).



## Gray Level Sampling



a) b)  
c) d)



e) f)  
g) h)

**Figure 4. We keep the spatial sampling but decrease  $k=8$  to  $k=1$  (gray levels from 256 to 2). (Digital Image Processing, 2<sup>nd</sup> E, by Gonzalez, Richard).**

## **Image Processing with Applications-CSCI597/MATH597/MATH489**

Experimental prove made by Huang (1965)-

**Image with high level of detail could be presented with only a few gray levels.**

**Image with low level of detail need more gray levels.**

## **ZOOMING AND SHRINKING**

**It is the same procedure as re-sampling and sub-sampling.**

**The difference is that in this case we work with digital images.**

## Gray level Interpolation for Scaling

- Nearest Neighbor Interpolation

Nearest neighbor interpolation is the simplest method and basically makes the pixels bigger. The color of a pixel in the new image is the color of the nearest pixel of the original image. If you enlarge 200%, one pixel will be enlarged to a 2 x 2 area of 4 pixels with the same color as the original pixel. **Most image viewing and editing software use this type of interpolation to enlarge a digital image for the purpose of closer examination because it does not change the color information of the image and does not introduce any anti-aliasing.** For the same reason, it is not suitable to enlarge photographic images because it increases the visibility of jaggies.

## Gray level Interpolation for Scaling



**Figure 5.** Gray level interpolation by using the nearest neighbor in case of zooming.



## Gray level Interpolation for Scaling

- **Bilinear Interpolation**

Bilinear Interpolation determines the value of a new pixel based on a weighted average of the 4 pixels in the nearest  $2 \times 2$  neighborhood of the pixel in the original image. The averaging has an anti-aliasing effect and therefore produces relatively smooth edges with hardly any jaggies.

## Gray level Interpolation for Scaling



**Figure 6.** Gray level interpolation by using the bilinear method in case of zooming.

## Gray level Interpolation for Scaling

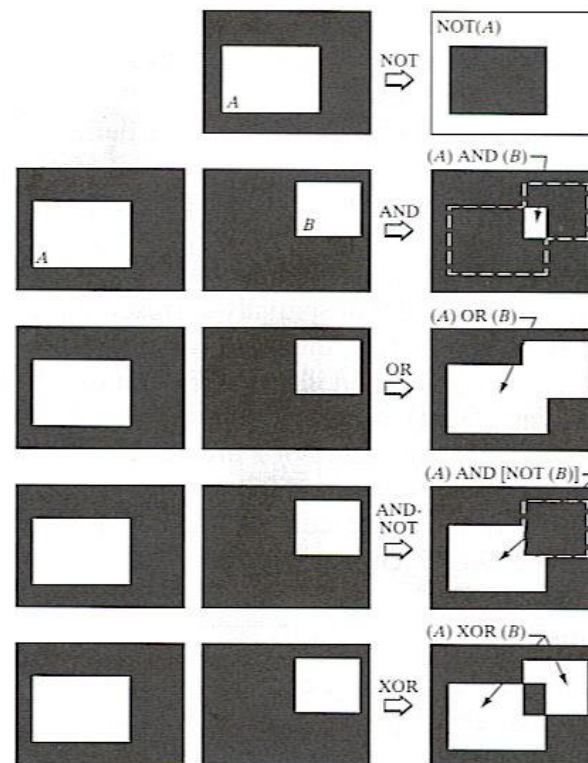
- **Bicubic interpolation**
- Bicubic interpolation is more sophisticated and produces smoother edges than bilinear interpolation. Here, a new pixel is a bicubic function using 16 pixels in the nearest 4 x 4 neighborhood of the pixel in the original image. This is the method most commonly used by image editing software, printer drivers and many digital cameras for re-sampling images.

# Operations with Images

84 Chapter 2 ■ Digital Image Fundamentals

**FIGURE 2.33**

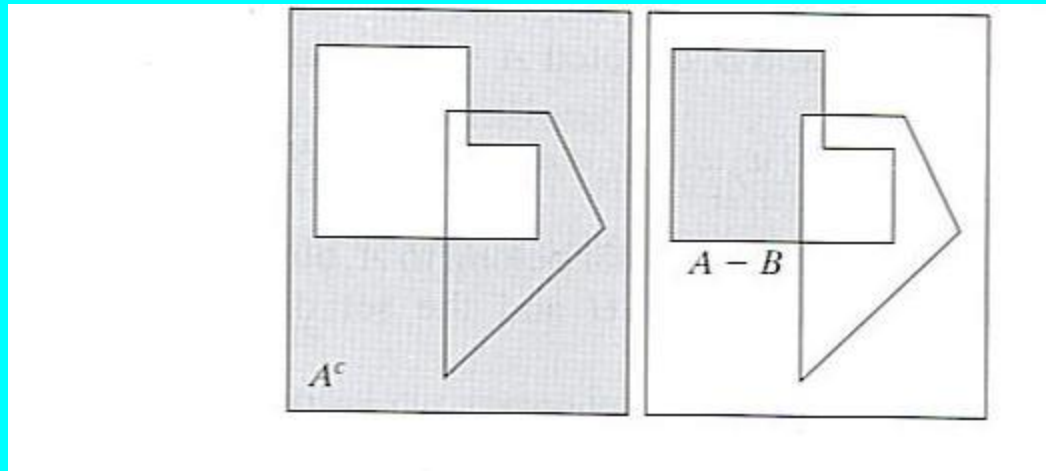
Illustration of logical operations involving foreground (white) pixels. Black represents binary 0s and white binary 1s. The dashed lines are shown for reference only. They are not part of the result.



**Figure 7. Logical Operations**

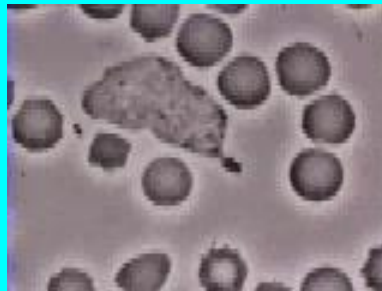
Digital Image Processing, 3rd E, by Gonzalez, Richard

## Operations with Images



**Figure 8.** left) Complement; right) Subtraction.

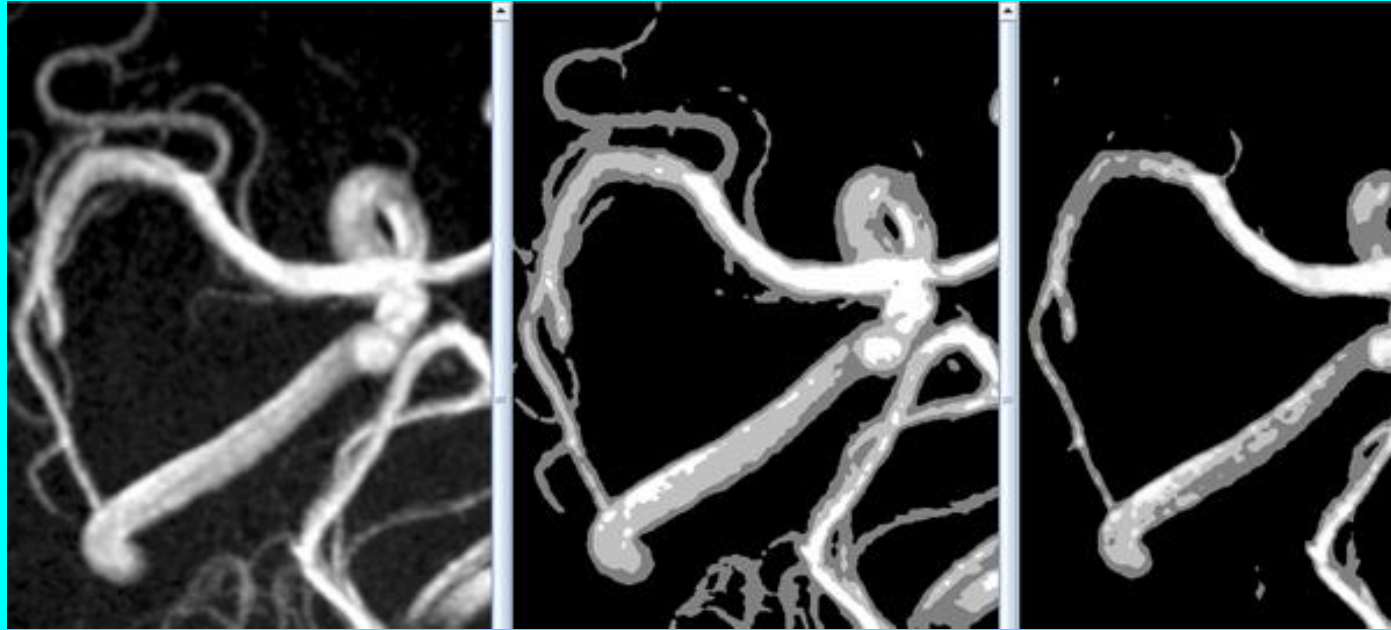
Digital Image Processing, 3rd E, by Gonzalez, Richard



**Figure 9.** A noisy image of neutrophil.

The image is a frame from a movie from: [http://www.youtube.com/watch?v=I\\_xh-bkiv\\_c](http://www.youtube.com/watch?v=I_xh-bkiv_c)

# Logical and Arithmetic Operations



a)

b)

c)

**Figure 10:** The image in c) is received as “and” of a) and b).

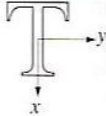

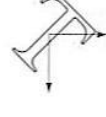

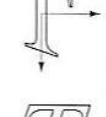
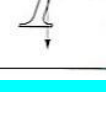
The original image is a courtesy of Dr. Val Runge, Medical Center, Temple Texas

- Software coded Spring 2008 in Java by:
- SAYED HAFIZUR RAHMAN in a team with
- PRADEEP REDDY DAMEGUNTA , SURESH BANDARU , LAKSHMI PYDIKONDALA



# Affine Transformations

Affine transformations based on Eq. (2.6-23).

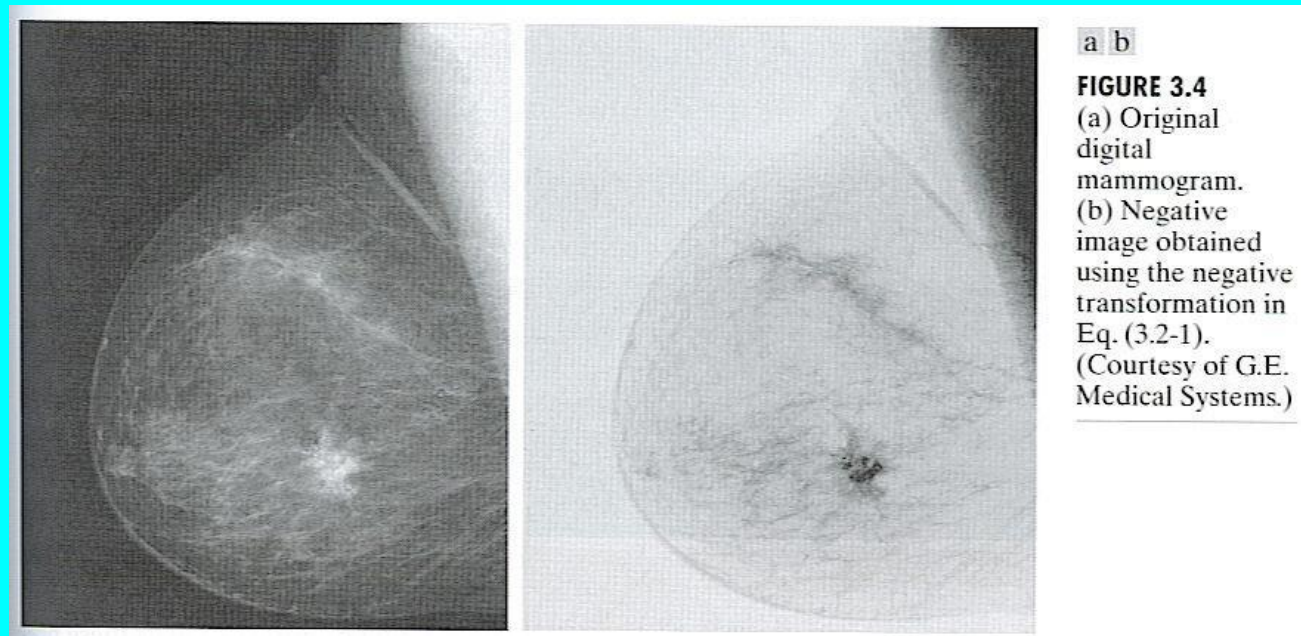
| Transformation Name | Affine Matrix, T   | Coordinate Equations   | Example  |
|---------------------|--|--|--|
| Identity            | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  | $x = v$<br>$y = w$   |   |
| Scaling             | $\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$                                      | $x = c_x v$<br>$y = c_y w$   |   |
| Rotation            | $\begin{bmatrix} \cos \theta & \sin \theta & 0 \\ -\sin \theta & \cos \theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$ | $x = v \cos \theta - w \sin \theta$<br>$y = v \sin \theta + w \cos \theta$ |   |
| Translation         | $\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$                                      | $x = v + t_x$<br>$y = w + t_y$   |   |
| Shear (vertical)    | $\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  | $x = v + s_v w$<br>$y = w$   |   |
| Shear (horizontal)  | $\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$  | $x = v$<br>$y = s_h v + w$   |  |

The affine transforms are obtained by the equations:  
where (x,y) are the new coordinates of a pixel.  
while (u,w) are the old

$$\begin{bmatrix} x & y & 1 \end{bmatrix} = \begin{bmatrix} v & w & 1 \end{bmatrix} \mathbf{T} = \begin{bmatrix} v & w & 1 \end{bmatrix} \begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}$$

Digital Image Processing, 3rd E, by Gonzalez, Richard

## Image Transformations



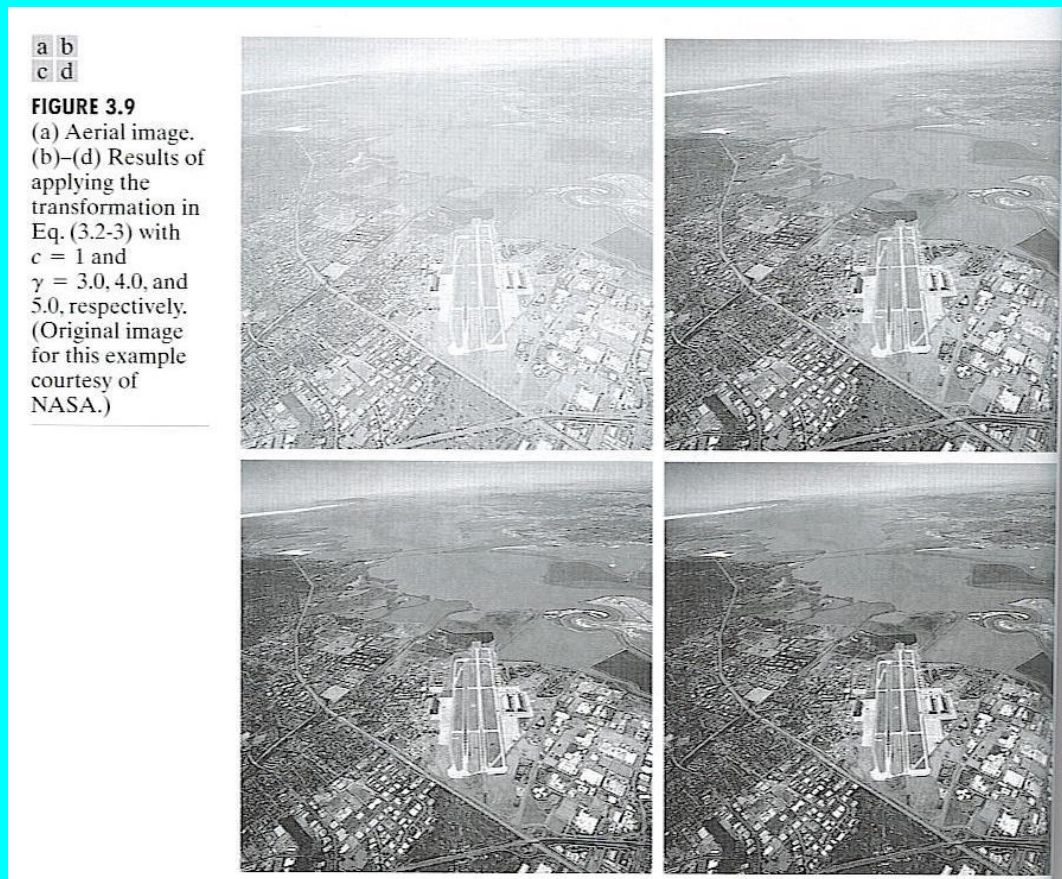
a)

b)

**Figure 11.a) An image; b) The image after negative transformation.**

(Digital Image Processing, 2<sup>nd</sup> E, by Gonzalez, Richard).

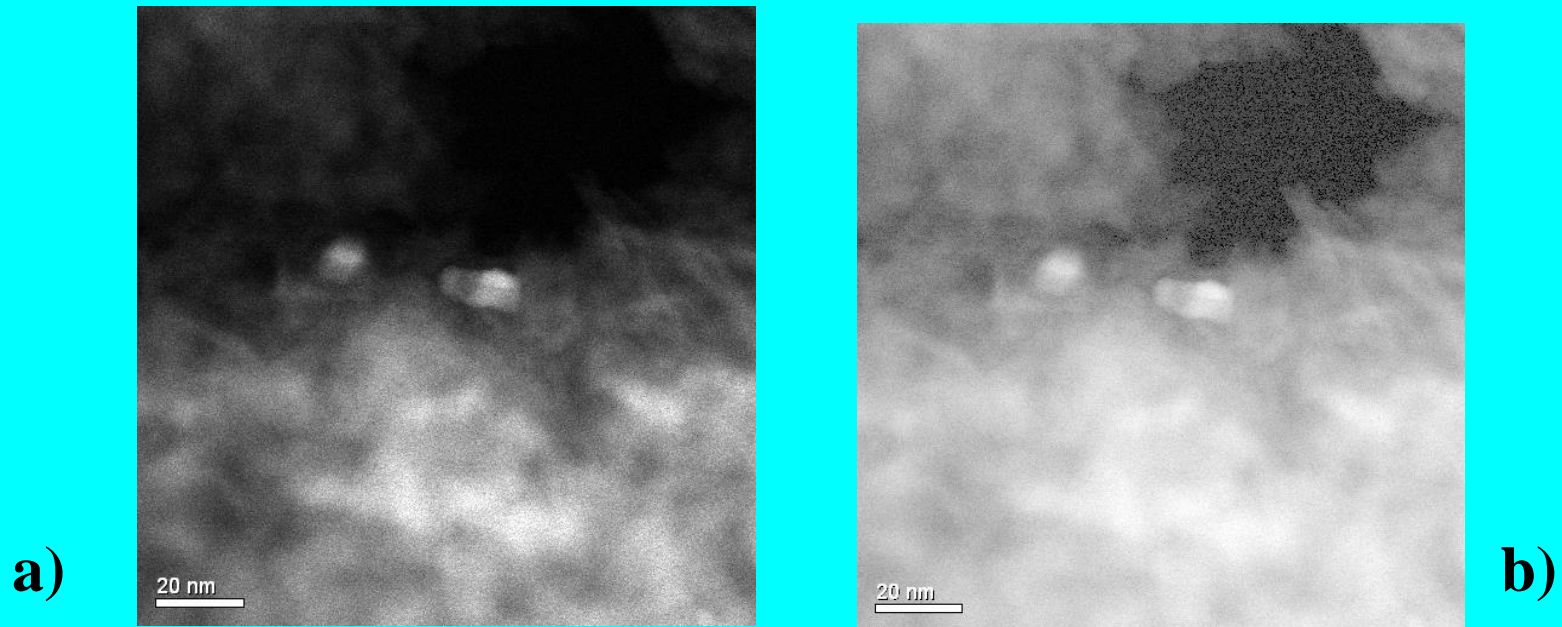
## Image Transformations



**Figure 12. An urban image and the results after applying power transformation with different power.**

(Digital Image Processing, 2<sup>nd</sup> E, by Gonzalez, Richard).

## Power and Log Operators

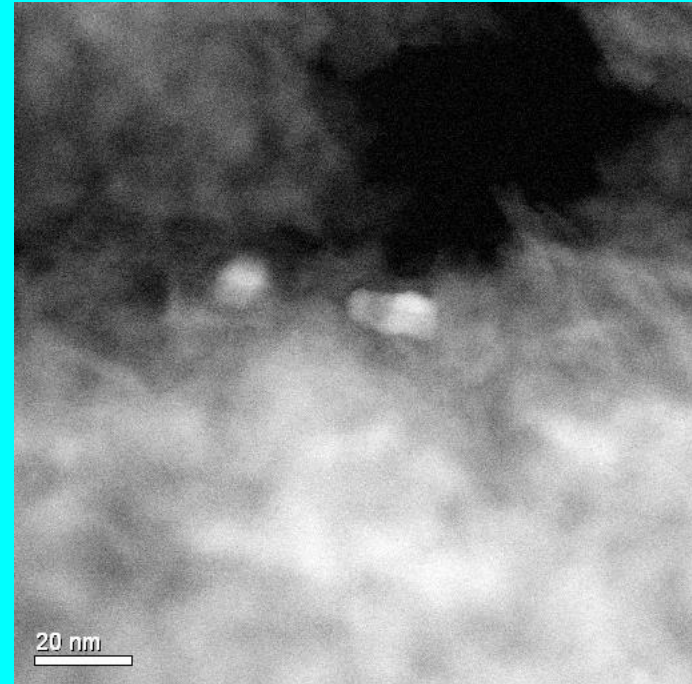
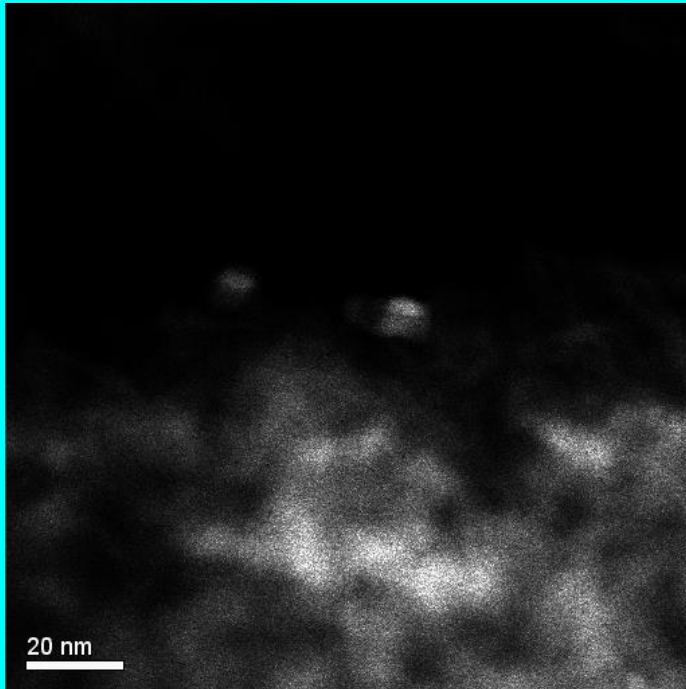


**Figure 13. a)** Original image courtesy of Dr. B. Jang Dept. of Chemistry TAMUC; **b)** the result after applying power operator with 0.3.

The tool was coded in C++ under a project assignment in the IP Class Spring 2005, by Nathaniel Rowland, in a team with Jarrod Robinson

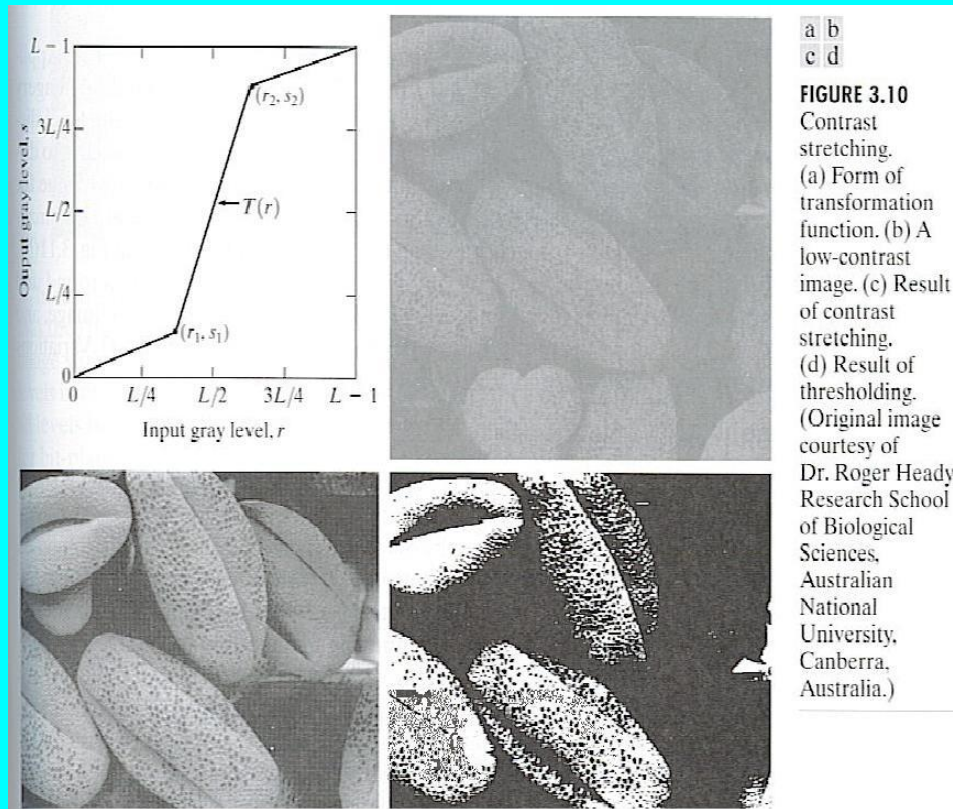


## Power and Log Operators



**Figure 14.** a) The image from Fig. 13a) after applying power operator with power 3; b) the result after applying logarithmic operator with base 3.

## Image Transformations



**Figure 8.** Image enhancement by contrast stretching and thresholding.  
 (Digital Image Processing, 2<sup>nd</sup> E, by Gonzalez, Richard).