Mathematical Tools used in DIP

- Array vs Matrix Operations
- Linear Vs Nonlinear Operations

H is said to be a *linear operator* if

$$H\left[a_i f_i(x, y) + a_j f_j(x, y)\right] = a_i H\left[f_i(x, y)\right] + a_j H\left[f_j(x, y)\right]$$
$$= a_i g_i(x, y) + a_j g_j(x, y) \qquad (2.6\text{-}2)$$

where a_i, a_j, $f_i(x, y)$, and $f_j(x, y)$ are arbitrary constants and images (of the same size), respectively.

  - Sum operator is linear. Max operator is non-linear

- Arithmetic operations:

$$s(x, y) = f(x, y) + g(x, y)$$
$$d(x, y) = f(x, y) - g(x, y)$$
$$p(x, y) = f(x, y) \times g(x, y)$$
$$v(x, y) = f(x, y) \div g(x, y)$$

Mathematical Tools used in DIP

Spatial Operations

- Single Pixel Operations

  s = *T* (z)

  where T is a transformation function that maps a original image pixel value z into a pixel value s (in the output image)

- Neighborhood Operations
- Geometrical Spatial Transformations

Vector and Matrix Operations

Image Transforms

Probabilistic Methods

Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 2
Digital Image Fundamentals

Single-Pixel Operations

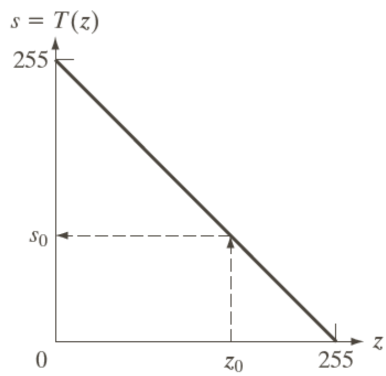


FIGURE 2.34 Intensity transformation function used to obtain the negative of an 8-bit image. The dashed arrows show transformation of an arbitrary input intensity value z_0 into its corresponding output value s_0.

Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 2
Digital Image Fundamentals

Neighborhood Operations

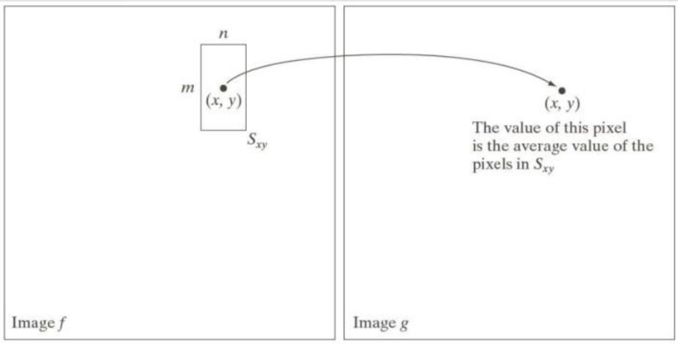
Neighborhood operations

Let S_{xy} denote the set of coordinates of a neighborhood centered on an arbitrary point (x, y) in an image, f. Neighborhood processing generates a corresponding pixel at the same coordinates in an output (processed) image, g, such that the value of that pixel is determined by a specified operation involving the pixels in the input image with coordinates in S_{xy}. For example, suppose that the specified operation is to compute the average value of the pixels in a rectangular neighborhood of size $m \times n$ centered on (x, y). The locations of pixels in this region constitute the set S_{xy}. Figures 2.35(a) and (b) illustrate the process. We can express this operation in equation form as

$$g(x, y) = \frac{1}{mn} \sum_{(r,c) \in S_{xy}} f(r, c) \qquad (2.6\text{-}21)$$

where r and c are the row and column coordinates of the pixels whose coordinates are members of the set S_{xy}.

Chapter 2
Digital Image Fundamentals

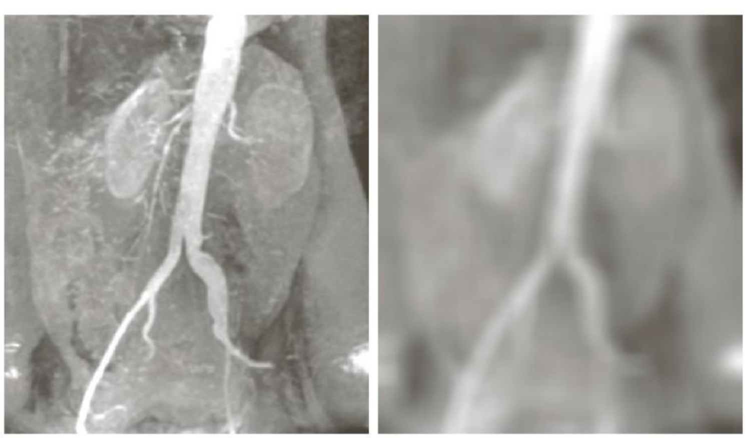


(a) (b)

FIGURE 2.35 Local averaging using neighborhood processing. The procedure is illustrated in (a) and (b) for a rectangular neighborhood.

Digital Image Processing, 3rd ed.
Gonzalez & Woods
www.ImageProcessingPlace.com

Chapter 2
Digital Image Fundamentals



(c) (d)

FIGURE 2.35 (c) The aortic angiogram discussed in Section 1.3.2. (d) The result of using Eq. (2.6-21) with $m = n = 41$. The images are of size 790×686 pixels.

Geometric Transformations

There are two basic steps in geometric transformations:

1. A spatial transformation of the physical rearrangement of pixels in the image, and
2. a grey level interpolation, which assigns grey levels to the transformed image

Spatial transformation

Pixel coordinates (x,y) undergo geometric distortion to produce an image with coordinates (x',y'):

$$x' = r(x,y)$$
$$y' = s(x,y),$$

where r and s are functions depending on x and y.

Examples:

1. Suppose $r(x,y) = \frac{x}{2}$, $s(x,y) = \frac{y}{2}$. This halves the size of the image.

   This transformation can be represented using a matrix equation $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \frac{1}{2} & 0 \\ 0 & \frac{1}{2} \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$

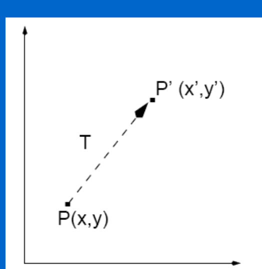
2. Rotation about the origin by an angle θ is given by

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$

Credits: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT5/node5.html

2D Translation

- Moves a point to a new location by adding translation amounts to the coordinates of the point.



$$x' = x + dx, \quad y' = y + dy$$

or

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x \\ y \end{bmatrix} + \begin{bmatrix} dx \\ dy \end{bmatrix}$$

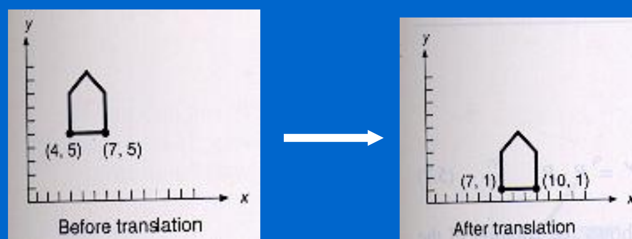
or

$$P' = P + T$$

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

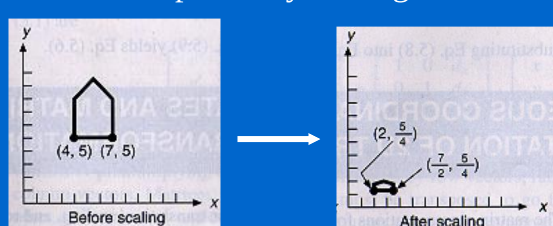
2D Translation (cont'd)

- To translate an object, translate every point of the object by the same amount.



2D Scaling

- Changes the size of the object by multiplying the coordinates of the points by scaling factors.



$x' = x\, s_x,\ y' = y\, s_y$ or $\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} s_x & 0 \\ 0 & s_y \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix},\ s_x, s_y > 0$ or $P' = S\, P$

2D Scaling (cont'd)

- Uniform vs non-uniform scaling

If $s_x = s_y$ uniform scaling

If $s_x \neq s_y$ nonuniform scaling

- Effect of scale factors:

If $s_x, s_y < 1$, size is reduced, object moves closer to origin

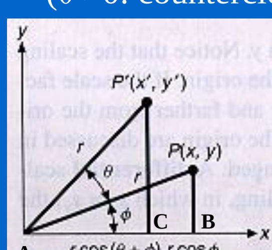
If $s_x, s_y > 1$, size is increased, object moves further from origin

If $s_x = s_y = 1$, size does not change

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

2D Rotation

- Rotates points by an angle θ about origin
  (θ >0: counterclockwise rotation)



- From *ABP* triangle:

$$cos(\phi) = x/r \text{ or } x = rcos(\phi)$$
$$sin(\phi) = y/r \text{ or } y = rsin(\phi)$$

- From *ACP'* triangle:

$$cos(\phi+\theta) = x'/r \text{ or } x' = rcos(\phi+\theta) = rcos(\phi)cos(\theta) - rsin(\phi)sin(\theta)$$
$$sin(\phi+\theta) = y'/r \text{ or } y' = rsin(\phi+\theta) = rcos(\phi)sin(\theta) + rsin(\phi)cos(\theta)$$

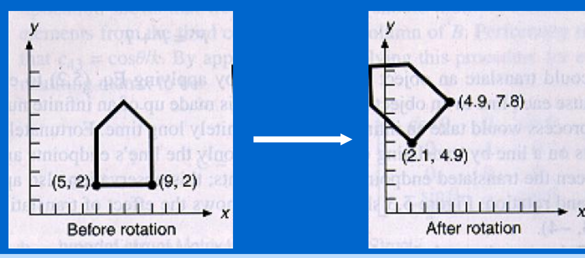
Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

2D Rotation (cont'd)

- From the above equations we have:

$$x' = x\cos(\theta) - y\sin(\theta), \quad y' = x\sin(\theta) + y\cos(\theta)$$ or

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} \cos(\theta) & -\sin(\theta) \\ \sin(\theta) & \cos(\theta) \end{bmatrix} \begin{bmatrix} x \\ y \end{bmatrix}$$ or $$\underline{P' = R\ P}$$



Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

Summary of 2D transformations

$$\underline{\text{Translation:}}\ P' = P + T$$

$$\underline{\text{Scale:}}\ P' = S\ P$$

$$\underline{\text{Rotation:}}\ P' = R\ P$$

- Use *homogeneous* coordinates to express translation as matrix multiplication

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

Homogeneous coordinates

- Add one more coordinate: $(x,y) \rightarrow (x_h, y_h, w)$
- Recover (x,y) by homogenizing (x_h, y_h, w):

$$x = \frac{x_h}{w},\quad y = \frac{y_h}{w},\quad w \neq 0$$

- So, $x_h = xw$, $y_h = yw$, $(w \neq 0)$

$$(x, y) \rightarrow (xw, yw, w)$$

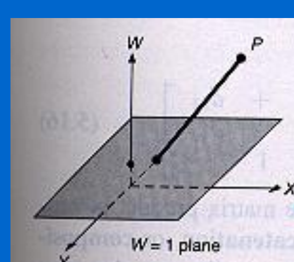
Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

Homogeneous coordinates (cont'd)

- (x, y) has multiple representations in homogeneous coordinates:
  - $w=1$ $(x,y) \rightarrow (x,y,1)$
  - $w=2$ $(x,y) \rightarrow (2x,2y,2)$

$(w \neq 0)$

- All these points lie on a line in the space of homogeneous coordinates !!



projective space

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

2D Translation using homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx \\ 0 & 1 & dy \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

w=1

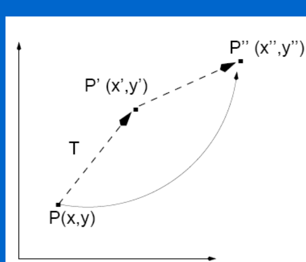
$$x' = x + dx, \quad y' = y + dy$$

$$\underline{P' = T(dx, dy)\ P}$$

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

2D Translation using homogeneous coordinates (cont'd)

- Successive translations:



$$P' = T(dx_1, dy_1)\ P, \quad P'' = T(dx_2, dy_2)\ P'$$

$$P'' = T(dx_2, dy_2) T(dx_1, dy_1)\ P = T(dx_1 + dx_2, dy_1 + dy_2)\ P$$

$$\begin{bmatrix} 1 & 0 & dx_2 \\ 0 & 1 & dy_2 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & dx_1 \\ 0 & 1 & dy_1 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & dx_1 + dx_2 \\ 0 & 1 & dy_1 + dy_2 \\ 0 & 0 & 1 \end{bmatrix}$$

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

2D Scaling using homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

w=1

$$x' = x\, s_x, \quad y' = y\, s_y$$

$$\underline{P' = S(s_x, s_y)\; P}$$

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

2D Scaling using homogeneous coordinates (cont'd)

- Successive scalings:

$$P' = S(s_{x_1}, s_{y_1})\; P, \quad P'' = S(s_{x_2}, s_{y_2})\; P'$$

$$P'' = S(s_{x_2}, s_{y_2}) S(s_{x_1}, s_{y_1})\; P = S(s_{x_1} s_{x_2}, s_{y_1} s_{y_2})\; P$$

$$\begin{bmatrix} s_{x_2} & 0 & 0 \\ 0 & s_{y_2} & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} s_{x_1} & 0 & 0 \\ 0 & s_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix} = \begin{bmatrix} s_{x_2} s_{x_1} & 0 & 0 \\ 0 & s_{y_2} s_{y_1} & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

2D Rotation using homogeneous coordinates

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

w=1

$$x' = xcos(\theta) - ysin(\theta), \quad y' = xsin(\theta) + ycos(\theta)$$

$$\underline{P' = R(\theta)\ P}$$

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

2D Rotation using homogeneous coordinates (cont'd)

- Successive rotations:

$$P' = R(\theta_1)\ P, \quad P'' = R(\theta_2)\ P'$$

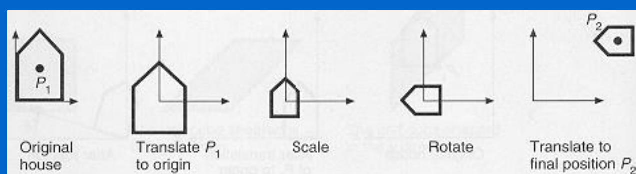
or $$P'' = R(\theta_1)R(\theta_2)\ P = R(\theta_1 + \theta_2)\ P$$

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

Composition of transformations

- The transformation matrices of a series of transformations can be concatenated into a single transformation matrix.

Example:
* Translate P_1 to origin
* Perform scaling and rotation
* Translate to P_2



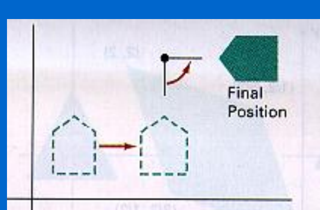
$$M = T(x_2, y_2)R(\theta)S(s_x, s_y)T(-x_1, -y_1)$$

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

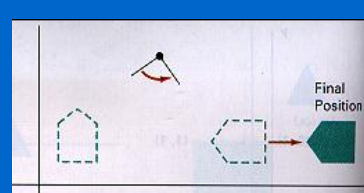
Composition of transformations (cont'd)

- Important: preserve the order of transformations!

translation + rotation



rotation + translation



Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

General form of transformation matrix

rotation, scale translation

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

• Representing a sequence of transformations as a single transformation matrix is more efficient!

$$x' = a_{11}x + a_{12}y + a_{13}$$

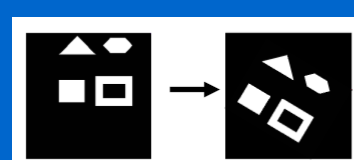
$$y' = a_{21}x + a_{22}y + a_{23}$$

(only 4 multiplications and 4 additions)

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

Special cases of transformations

- Rigid transformations
  - Involves only translation and rotation (3 parameters)
  - Preserve angles and lengths



$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & t_x \\ r_{21} & r_{22} & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

upper 2x2 submatrix is orthonormal

$$u_1 = (r_{11}, r_{12}),\ u_2 = (r_{21}, r_{22})$$

$$u_1 . u_1 = \|u_1\|^2 = r_{11}^2 + r_{12}^2 = 1$$

$$u_2 . u_2 = \|u_2\|^2 = r_{21}^2 + r_{22}^2 = 1$$

$$u_1 . u_2 = r_{11}r_{21} + r_{12}r_{22} = 0$$

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

Example: rotation matrix

$$\begin{bmatrix} cos(\theta) & -sin(\theta) & 0 \\ sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

$$u_1 . u_1 = cos(\theta)^2 + sin(-\theta)^2 = 1$$

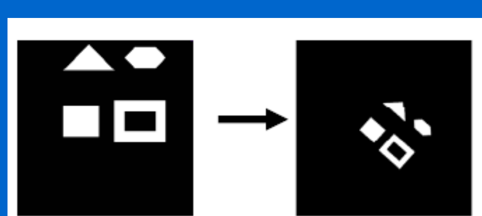
$$u_2 . u_2 = cos(\theta)^2 + sin(\theta)^2 = 1$$

$$u_1 . u_2 = cos(\theta)sin(\theta) - sin(\theta)cos(\theta) = 0$$

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

Special cases of transformations

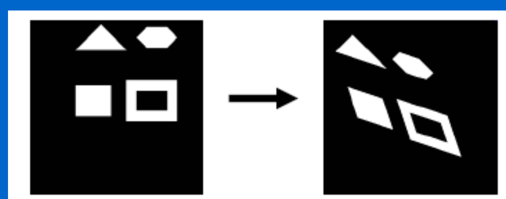
- Similarity transformations
  - Involve rotation, translation, scaling (4 parameters)
  - Preserve angles but not lengths
  - Angles may not be preserved in case of nonuniform scaling transformations



Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

Affine transformations

- Involve translation, rotation, scale, and shear (6 parameters)
- Preserve parallelism of lines but not lengths and angles.



Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

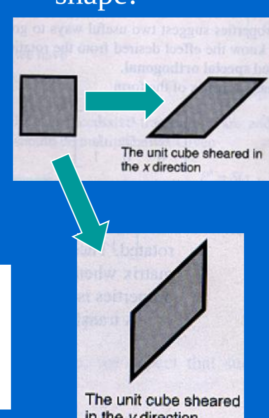
2D shear transformation

- Shearing along x-axis:

$$x' = x + ay,\ y' = y$$

$$SH_x = \begin{bmatrix} 1 & a & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

changes object shape!



- Shearing along y-axis

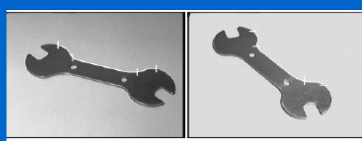
$$x' = x,\ y' = bx + y$$

$$SH_y = \begin{bmatrix} 1 & 0 & 0 \\ b & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

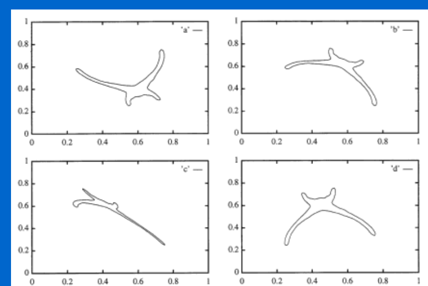
Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)

Affine Transformations

- Under certain assumptions, affine transformations can be used to approximate the effects of perspective projection!

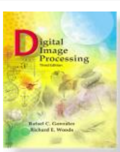


affine transformed object



G. Bebis, M. Georgiopoulos, N. da Vitoria Lobo, and M. Shah, " Recognition by learning affine transformations", **Pattern Recognition**, Vol. 32, No. 10, pp. 1783-1799, 1999.

Credits: Dr. George Bebis, CSE, University of Nevada (UNR), Reno (CS485 notes)



Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 2

Digital Image Fundamentals

Geometric Spatial Transformations (or say, Geometric Operations)

The transformation of coordinates may be expressed as

$$(x, y) = T\{(v, w)\} \qquad (2.6\text{-}22)$$

where (v, w) are pixel coordinates in the original image and (x, y) are the corresponding pixel coordinates in the transformed image. For example, the transformation $(x, y) = T\{(v, w)\} = (v/2, w/2)$ shrinks the original image to half its size in both spatial directions. One of the most commonly used spatial coordinate transformations is the *affine transform* (Wolberg [1990]), which has the general form

$$[x \;\; y \;\; 1] = [v \;\; w \;\; 1]\,\mathbf{T} = [v \;\; w \;\; 1]\begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix} \qquad (2.6\text{-}23)$$

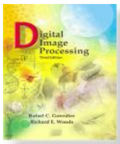
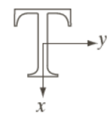
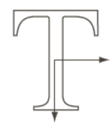
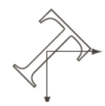
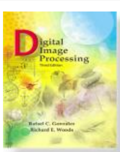


TABLE 2.2
Affine transformations based on Eq. (2.6.–23).

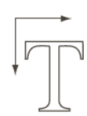
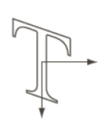
Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Identity	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = w$	
Scaling	$\begin{bmatrix} c_x & 0 & 0 \\ 0 & c_y & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = c_x v$ $y = c_y w$	
Rotation	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	

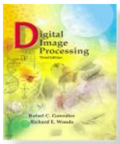


Digital Image Processing, 3rd ed.
Gonzalez & Woods
www.ImageProcessingPlace.com

Chapter 2
Digital Image Fundamentals

TABLE 2.2
Affine transformations based on Eq. (2.6.–23).

Transformation Name	Affine Matrix, T	Coordinate Equations	Example
Translation	$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ t_x & t_y & 1 \end{bmatrix}$	$x = v + t_x$ $y = w + t_y$	
Shear (vertical)	$\begin{bmatrix} 1 & 0 & 0 \\ s_v & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v + s_v w$ $y = w$	
Shear (horizontal)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

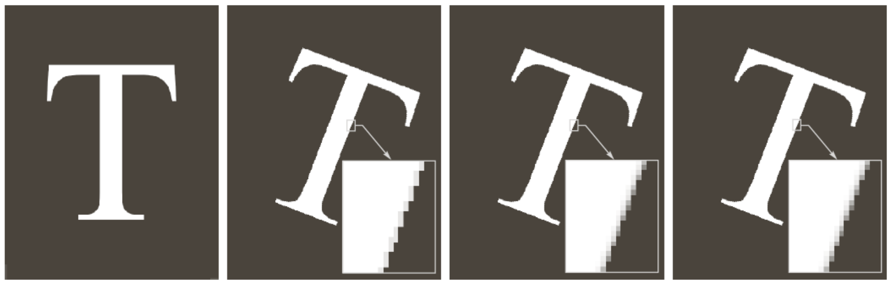


Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 2
Digital Image Fundamentals



a b c d

FIGURE 2.36 (a) A 300 dpi image of the letter T. (b) Image rotated 21° clockwise using nearest neighbor interpolation to assign intensity values to the spatially transformed pixels. (c) Image rotated 21° using bilinear interpolation. (d) Image rotated 21° using bicubic interpolation. The enlarged sections show edge detail for the three interpolation approaches.

Geometric Transformations (Tie Points)

These are points in the distorted image for which we know their corrected positions in the final image. Such tie points are often known for satellite images and aerial photos.

We model such a distortion using a pair of bilinear equations:

$$x' = c_1 x + c_2 y + c_3 xy + c_4$$

$$y' = c_5 x + c_6 y + c_7 xy + c_8.$$

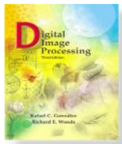
We have 4 pairs of tie point coordinates. This enables us to solve for the 8 coefficients $c_1 \ldots c_8$.

We can set up the matrix equation using the coordinates of the 4 tie points:

Note: This is for mapping the spatial coordinates, and not for assigning the intensity values to output image pixels (For that intensity interpolation to be performed)

$$\begin{bmatrix} x'_1 \\ y'_1 \\ x'_2 \\ y'_2 \\ x'_3 \\ y'_3 \\ x'_4 \\ y'_4 \end{bmatrix} = \begin{bmatrix} x_1 & y_1 & x_1y_1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_1 & y_1 & x_1y_1 & 1 \\ x_2 & y_2 & x_2y_2 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_2 & y_2 & x_2y_2 & 1 \\ x_3 & y_3 & x_3y_3 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_3 & y_3 & x_3y_3 & 1 \\ x_4 & y_4 & x_4y_4 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & x_4 & y_4 & x_4y_4 & 1 \end{bmatrix} \begin{bmatrix} c_1 \\ c_2 \\ c_3 \\ c_4 \\ c_5 \\ c_6 \\ c_7 \\ c_8 \end{bmatrix}$$

Credits: http://homepages.inf.ed.ac.uk/rbf/CVonline/LOCAL_COPIES/OWENS/LECT5/node5.html



Digital Image Processing, 3rd ed.

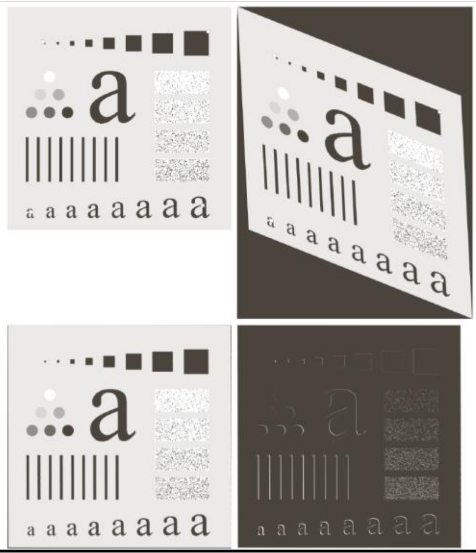
Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 2
Digital Image Fundamentals

Note:

1. **Only 4 tie points needed because distortion is linear shear in both directions.**
2. **Discrepancies - because of error in the manual selection of the tie points. It is generally difficult to achieve oerfect matches when distortion is so severe**



a b
c d

FIGURE 2.37 Image registration. (a) Reference image. (b) Input (geometrically distorted image). Corresponding tie points are shown as small white squares near the corners. (c) Registered image (note the errors in the borders). (d) Difference between (a) and (c), showing more registration errors.

Mathematical Tools used in DIP

Spatial Operations

- Single Pixel Operations

  $s = T(z)$

  where T is a transformation function that maps a original image pixel value z into a pixel value s (in the output image)
- Neighborhood Operations
- Geometrical Spatial Transformations

Vector and Matrix Operations

Image Transforms

Probabilistic Methods

Vector and Matrix Operations

Multispectral image processing is a typical area in which vector and matrix operations are used routinely. Here we see that *each* pixel of an RGB image has three components, which can be organized in the form of a *column vector*

$$\mathbf{z} = \begin{bmatrix} z_1 \\ z_2 \\ z_3 \end{bmatrix} \tag{2.6-26}$$

where z_1 is the intensity of the pixel in the red image, and the other two elements are the corresponding pixel intensities in the green and blue images,

Vector and Matrix Operations

Once pixels have been represented as vectors we have at our disposal the tools of vector-matrix theory. For example, the *Euclidean distance*, *D*, between a pixel vector $\mathbf{z}$ and an arbitrary point $\mathbf{a}$ in n-dimensional space is defined as the vector product

$$\begin{aligned} D(\mathbf{z}, \mathbf{a}) &= \left[(\mathbf{z} - \mathbf{a})^T(\mathbf{z} - \mathbf{a})\right]^{\frac{1}{2}} \\ &= \left[(z_1 - a_1)^2 + (z_2 - a_2)^2 + \cdots + (z_n - a_n)^2\right]^{\frac{1}{2}} \end{aligned} \tag{2.6-27}$$

vector norm, denoted by $\|\mathbf{z} - \mathbf{a}\|$.

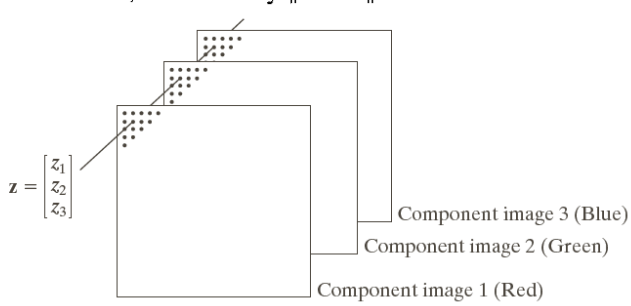
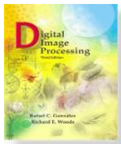


FIGURE 2.38 Formation of a vector from corresponding pixel values in three RGB component images.



Vector and Matrix Operations

Another important advantage of pixel vectors is in linear transformations, represented as $\mathbf{w} = \mathbf{A}(\mathbf{z} - \mathbf{a})$ (2.6-28)

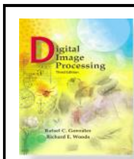
where $\mathbf{A}$ is a matrix of size $m \times n$ and $\mathbf{z}$ and $\mathbf{a}$ are column vectors of size $n \times 1$. As you will learn later, transformations of this type have a number of useful applications in image processing.

Also, we can express an image of size $M \times N$ as a vector of dimension $MN \times 1$

With images formed in this manner, we can express a broad range of linear processes applied to an image by using the notation

$$\mathbf{g} = \mathbf{Hf} + \mathbf{n} \qquad (2.6\text{-}29)$$

where $\mathbf{f}$ is an $MN \times 1$ vector representing an input image, $\mathbf{n}$ is an $MN \times 1$ vector representing an $M \times N$ noise pattern, $\mathbf{g}$ is an $MN \times 1$ vector representing a processed image, and $\mathbf{H}$ is an $MN \times MN$ matrix representing a linear process



Digital Image Processing, 3rd ed.

Gonzalez & Woods

www.ImageProcessingPlace.com

Chapter 2

Digital Image Fundamentals

Probabilistic Methods

Probability finds its way into image processing work in a number of ways. The simplest is when we treat intensity values as random quantities. For example, let z_i, $i = 0, 1, 2, \ldots, L-1$, denote the values of all possible intensities in an $M \times N$ digital image. The probability, $p(z_k)$, of intensity level z_k occurring in a given image is estimated as

$$p(z_k) = \frac{n_k}{MN} \qquad (2.6\text{-}42)$$

where n_k is the number of times that intensity z_k occurs in the image and MN is the total number of pixels. Clearly,

$$\sum_{k=0}^{L-1} p(z_k) = 1 \qquad (2.6\text{-}43)$$

Once we have $p(z_k)$, we can determine a number of important image characteristics. For example, the mean (average) intensity is given by

$$m = \sum_{k=0}^{L-1} z_k\, p(z_k) \qquad (2.6\text{-}44)$$

Probabilistic Methods

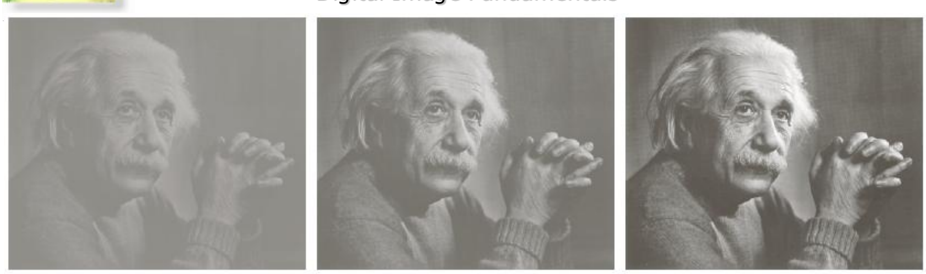
Similarly, the variance of the intensities is

$$\sigma^2 = \sum_{k=0}^{L-1} (z_k - m)^2 p(z_k) \qquad (2.6\text{-}45)$$

The variance is a measure of the spread of the values of z about the mean, so it is a useful measure of image contrast. In general, the nth moment of random variable z about the mean is defined as

$$\mu_n(z) = \sum_{k=0}^{L-1} (z_k - m)^n p(z_k) \qquad (2.6\text{-}46)$$

We see that $\mu_0(z) = 1$, $\mu_1(z) = 0$, and $\mu_2(z) = \sigma^2$. Whereas the mean and variance have an immediately obvious relationship to visual properties of an image, higher-order moments are more subtle.



a b c

FIGURE 2.41 Images exhibiting (a) low contrast, (b) medium contrast, and (c) high contrast.

Chapter 2
Digital Image Fundamentals

Appendix: Images Formats (supported by Matlab Image Processing Toolbox)

Format Name	Full Name	Description	Recognized Extensions
TIFF	*Tagged Image File Format*	A flexible file format supporting a variety of image compression standards, including JPEG. (*container*)	.tif, .tiff
JPEG	*Joint Photographic Experts Group*	A standard for compression of images of photographic quality	.jpg, .jpeg
GIF	*Graphics Interchange Format*	For 1- through 8-bit images. Frequently used to make small animations on the Internet	.gif
BMP	*Windows Bitmap*	Format used mainly for simple uncompressed images	.bmp
PNG	*Portable Network Graphics*	Compresses full colour images with transparency (up to 48 bits/pixel)	.png
XWD	*X Window Dump*		.xwd

Chapter 3

Intensity Transformations and Spatial Filtering

Two principles categories of Spatial Processing

– Intensity Transformations
  - Operate on single pixels of an image, principally for the purpose of contrast manipulation & image thresholding

– Spatial Filtering
  - Deals with performing operation (like image sharpening), by working in a neighborhood (*spatial filter*, *kernel*, *template* , *window or spatial mask*) of every pixel in an image

Chapter 3

Intensity Transformations and Spatial Filtering

About the Examples in This Chapter

Although intensity transformations and spatial filtering span a broad range of applications, most of the examples in this chapter are applications to image enhancement. *Enhancement* is the process of manipulating an image so that the result is more suitable than the original for a specific application. The word *specific* is important here because it establishes at the outset that enhancement techniques are problem oriented. Thus, for example, a method that is quite useful for enhancing X-ray images may not be the best approach for enhancing satellite images taken in the infrared band of the electromagnetic spectrum. There is no general "theory" of image enhancement. When an image is processed for visual interpretation, the viewer is the ultimate judge of how well a particular method works. When dealing with machine perception, a given technique is easier to quantify. For example, in an automated character-recognition system, the most appropriate enhancement method is the one that results in the best recognition rate, leaving aside other considerations such as computational requirements of one method over another.

Background

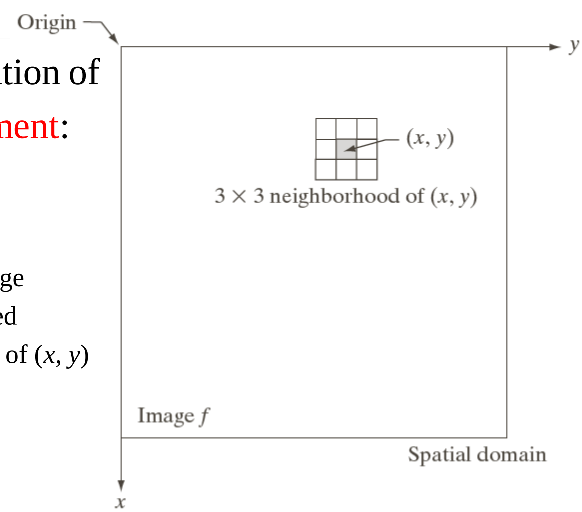
Mathematical representation of spatial domain enhancement:

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

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

$g(x, y)$: the processed image

T: an operator on f, defined over some neighborhood of (x, y)

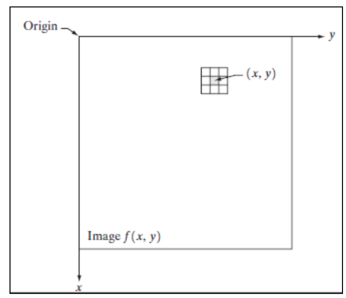


Examples of Enhancement Techniques

Larger neighborhoods allow considerable more flexibility. The general approach is to use a function of the values of *f* in a predefined neighborhood of (x,y) to determine the value of *g* at (x,y).

One of the principal approaches in this formulation is based on the use of so-called *masks* (also referred to as *filters)*

So, a mask/filter: is a small (say 3x3) 2-D array, such as the one shown in the figure, in which the values of the mask coefficients determine the nature of the process, such as *image sharpening*. Enhancement techniques based on this type of approach often are referred to as *mask processing* or *filtering*.



- The simplest form of ***T***, is when the neighborhood of size 1X1 (that is a single pixel). In this case, g depends only on the value of f at (x,y), and ***T*** becomes a *grey-level* (also called *intensity* or *mapping*) *transformation function* of the form:

$$s = T\,(r)$$

Where, for simplicity in notation, r and s are variables denoting, respectively, the grey level of f(x,y) and g(x,y) at any point (x,y)

Gray-Level Mapping

$$g(x, y) = f(x, y) + b \qquad g(x, y) = a \cdot f(x, y) + b$$

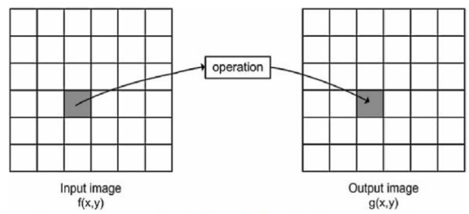


Fig. The principle of point processing.

Decreased brightness | Input image | Increased brightness

Credits: S. Murala, EEL484 Lecture notes/slides, IIT Ropar

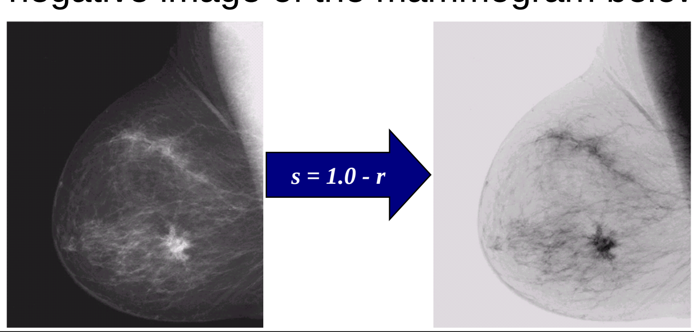
52

Point Processing Example: Negative Images

Negative images are useful for enhancing white or grey detail embedded in dark regions of an image

– Note how much clearer the tissue is in the negative image of the mammogram below

Original Image

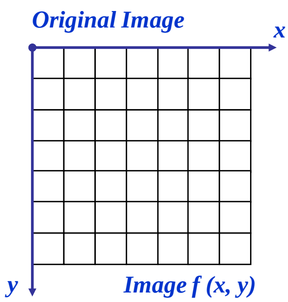


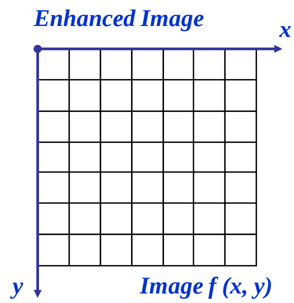
Negative Image

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

53

Point Processing Example: Negative Images (cont…)





$$s = intensity_{max} - r$$

Image Negative: $s = L - 1 - r$

where L is # of different intensity levels

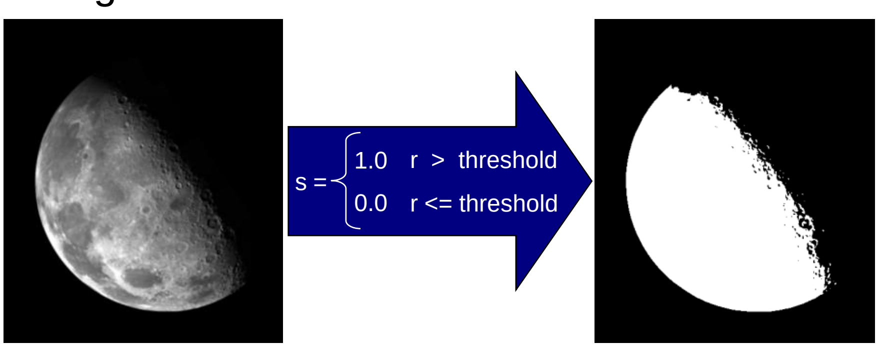
Credits: Brian Mac, Dublin Institute of Technology

54

Point Processing Example: Thresholding

Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Thresholding transformations are particularly useful for segmentation in which we want to isolate an object of interest from a background

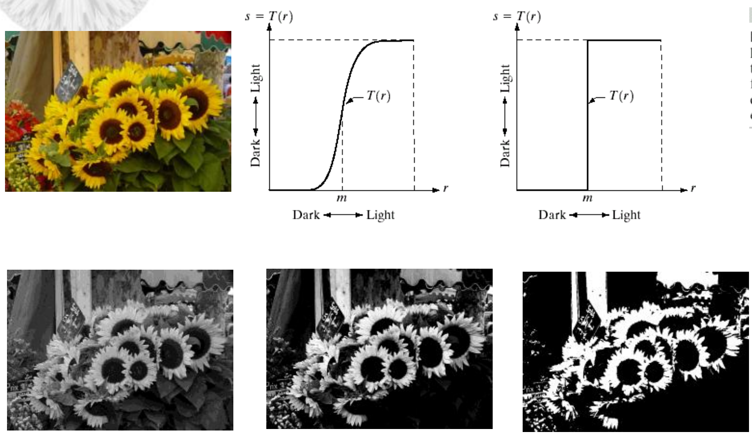


Credits: Brian Mac, Dublin Institute of Technology

Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

Gray-level Transformation (Intensity Transformations)



a b

FIGURE 3.2 Gray-level transformation functions for contrast enhancement.

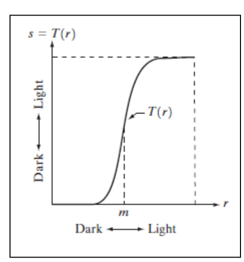
Examples of Enhancement Techniques

- **Contrast Stretching:**

  If T(r) has the form as shown in the figure below, the effect of applying the transformation to every pixel of f to generate the corresponding pixels in g would:

  Produce higher contrast than the original image, by:

  - Darkening the levels below m in the original image
  - Brightening the levels above m in the original image

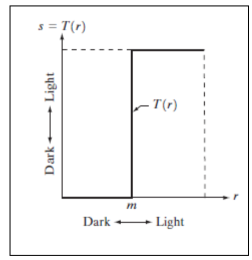


So, Contrast Stretching: is a simple image enhancement technique that improves the contrast in an image by ‘stretching’ the range of intensity values it contains to span a desired range of values. Typically, it uses a linear function

Examples of Enhancement Techniques

- **Thresholding**

  Is a limited case of contrast stretching, it produces a two-level (binary) image.



Some fairly simple, yet powerful, processing approaches can be formulated with grey-level transformations. Because enhancement at any point in an image depends only on the gray level at that point, techniques in this category often are referred to as *point processing*.

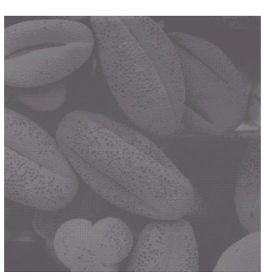
58

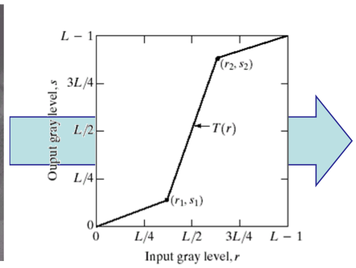
Piecewise Linear Transformation Functions

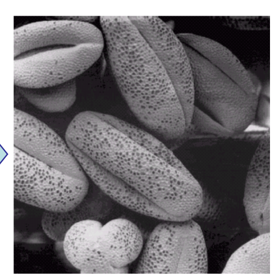
Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Rather than using a well defined mathematical function we can use arbitrary user-defined transforms

The images below show a contrast stretching linear transform to add contrast to a poor quality image



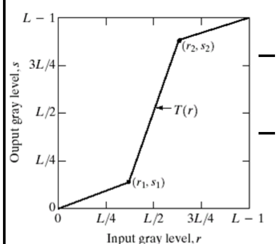




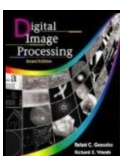
59

Contrast-Stretching

The locations of (r_1,s_1) and (r_2,s_2) control the shape of the transformation function.



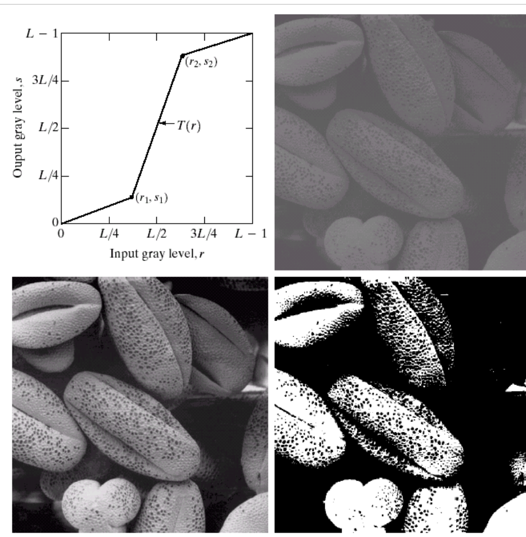
- If $r_1= s_1$ and $r_2= s_2$ the transformation is a linear function and produces no changes.
- If $r_1=r_2$, $s_1=0$ and $s_2=L-1$, the transformation becomes a thresholding function that creates a binary image.
- Intermediate values of (r_1,s_1) and (r_2,s_2) produce various degrees of spread in the gray levels of the output image, thus affecting its contrast.
- Generally, $r_1 \le r_2$ and $s_1 \le s_2$ is assumed.



Digital Image Processing, 2nd ed.

www.imageprocessingbook.com

Piecewise-Linear Transformation Functions
Contrast Stretching



a b
c d

FIGURE 3.10 Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)