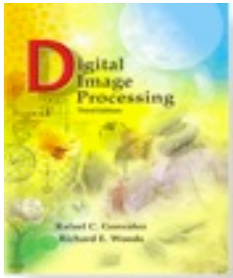


Last time:

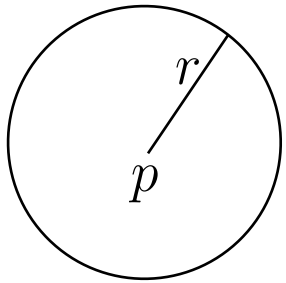
Distance measures

A distance function (or metric) is a function $D(p,q)$ where:

- $D(p,q) \geq 0$ and $D(p,q) = 0$ if and only if $p = q$ (positivity)
- $D(p,q) = D(q,p)$ (symmetry)
- $D(p,r) \leq D(p,q) + D(q,r)$ (triangle inequality).

Assume p has coordinates (x, y) and q has coordinates (s, t).

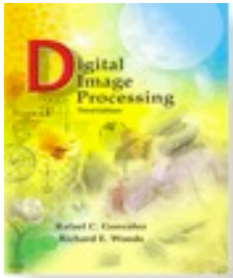
Euclidean distance: $D_e(p,q) = \sqrt{(x-s)^2 + (y-t)^2}$



City-block distance or D_4 distance: $D_4(p,q) = |x-s| + |y-t|$.

(1-norm in vector spaces)

		2		
	2	1	2	
2	1	0	1	2
	2	1	2	
		2		



Chessboard distance or D_8 distance: $D_8(p,q) = \max\{|x-s|, |y-t|\}$
(infinity distance)

$$\begin{matrix} 2 & 2 & 2 & 2 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 1 & 0 & 1 & 2 \\ 2 & 1 & 1 & 1 & 2 \\ 2 & 2 & 2 & 2 & 2 \end{matrix}$$

The definition of distance involves only the coordinates of the points hence is *independent* of adjacency.

There is however another distance D_m, defined as the length of the shortest path between the points.

Example:

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

$D_m = 6$

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

$D_m = 5$

What are the Euclidean, D_4 and D_8 distances?

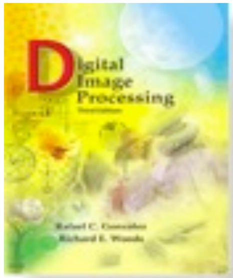
Arithmetic operations on images (must be of same size):

a_{11}	a_{12}	a_{13}
a_{21}	a_{22}	a_{23}
a_{31}	a_{32}	a_{33}

b_{11}	b_{12}	b_{13}
b_{21}	b_{22}	b_{23}
b_{31}	b_{32}	b_{33}

Operations (and matrix mult/division) are done **componentwise,** and not as classical row times column matrix multiplication.

In Matlab: use `A.*B` instead of `A*B`



Arithmetic operations on images (must be of same size):

$$\begin{array}{|c|c|c|}\hline a_{11} & a_{12} & a_{13} \\ \hline a_{21} & a_{22} & a_{23} \\ \hline a_{31} & a_{32} & a_{33} \\ \hline \end{array} + \begin{array}{|c|c|c|}\hline b_{11} & b_{12} & b_{13} \\ \hline b_{21} & b_{22} & b_{23} \\ \hline b_{31} & b_{32} & b_{33} \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline a_{11}+b_{11} & a_{12}+b_{12} & a_{13}+b_{13} \\ \hline a_{21}+b_{21} & a_{22}+b_{22} & a_{23}+b_{23} \\ \hline a_{31}+b_{31} & a_{32}+b_{32} & a_{33}+b_{33} \\ \hline \end{array}$$

Operations (and matrix mult/division) are done **componentwise,** and not as classical row times column matrix multiplication.

In Matlab: use `A.*B` instead of `A*B`

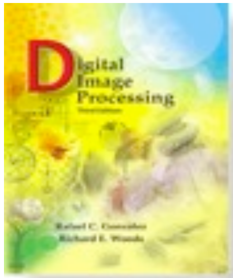
Arithmetic operations on images (must be of same size):

a_{11}	a_{12}	a_{13}
a_{21}	a_{22}	a_{23}
a_{31}	a_{32}	a_{33}

b_{11}	b_{12}	b_{13}
b_{21}	b_{22}	b_{23}
b_{31}	b_{32}	b_{33}

Operations (and matrix mult/division) are done **componentwise,** and not as classical row times column matrix multiplication.

In Matlab: use `A.*B` instead of `A*B`

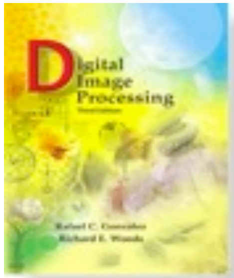


Arithmetic operations on images (must be of same size):

$$\begin{array}{|c|c|c|}\hline a_{11} & a_{12} & a_{13} \\ \hline a_{21} & a_{22} & a_{23} \\ \hline a_{31} & a_{32} & a_{33} \\ \hline \end{array} \times \begin{array}{|c|c|c|}\hline b_{11} & b_{12} & b_{13} \\ \hline b_{21} & b_{22} & b_{23} \\ \hline b_{31} & b_{32} & b_{33} \\ \hline \end{array} = \begin{array}{|c|c|c|}\hline a_{11}\,b_{11} & a_{12}\,b_{12} & a_{13}\,b_{13} \\ \hline a_{21}\,b_{21} & a_{22}\,b_{22} & a_{23}\,b_{23} \\ \hline a_{31}\,b_{31} & a_{32}\,b_{32} & a_{33}\,b_{33} \\ \hline \end{array}$$

Operations (and matrix mult/division) are done **componentwise,** and not as classical row times column matrix multiplication.

In Matlab: use `A.*B` instead of `A*B`



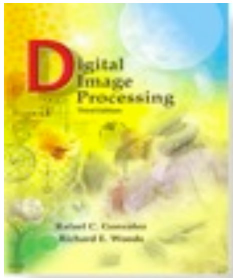
Linear/nonlinear operations:

Assume f and g are arbitrary $M \times N$ images, and α, β arbitrary real numbers.
We say that a function H is *linear* if

$$H(\alpha f + \beta g) = \alpha H(f) + \beta H(g),$$

that is, H obeys *additivity* and *homogeneity*.

Sum (+) is an example of linear operator. exp, max, min, ... are examples of nonlinear operations.



Example of useful imaging tasks using elementary arithmetic:

- Removal of gaussian noise by averaging
- Removal of shading (if shading patter is known)
- Regions Of Interest (ROI) (mask)

Removal of gaussian noise

Consider an image $g(x, y)$ that we assume to be a noiseless image $f(x, y)$ corrupted by some noise $\eta(x, y)$. We further assume that the noise in each pixel is *uncorrelated* and that is has zero average.

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

$=$

$+$

Review of Probability

Consider a random variable x with probability density function $p(x)$ or distribution $P(x)$.

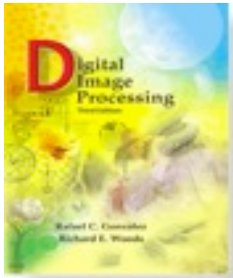
The expected value is defined as:

$$E[x] = \overline{x} = m = \int_{-\infty}^{\infty} xp(x)dx$$

when x is continuos and

$$E[x] = \overline{x} = m = \sum_{i=1}^{N} x_i P(x_i)$$

when x is discrete. The expected value of x is equal to its ***average*** (or ***mean***) ***value***, hence the use of the equivalent notation $\overline{x}$ and m.



Expected Value & Moments (Con't)

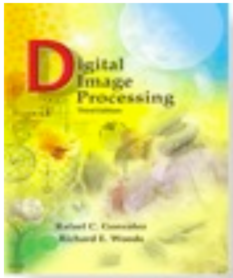
The ***variance*** of a random variable x, denoted by σ^2

$$\sigma^2 = E[(x-m)^2] = \int_{-\infty}^{\infty} (x-m)^2 p(x)dx$$

for continuous random variables and

$$\sigma^2 = E[(x-m)^2] = \sum_{i=1}^{N} (x_i - m)^2 P(x_i)$$

for discrete variables.



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

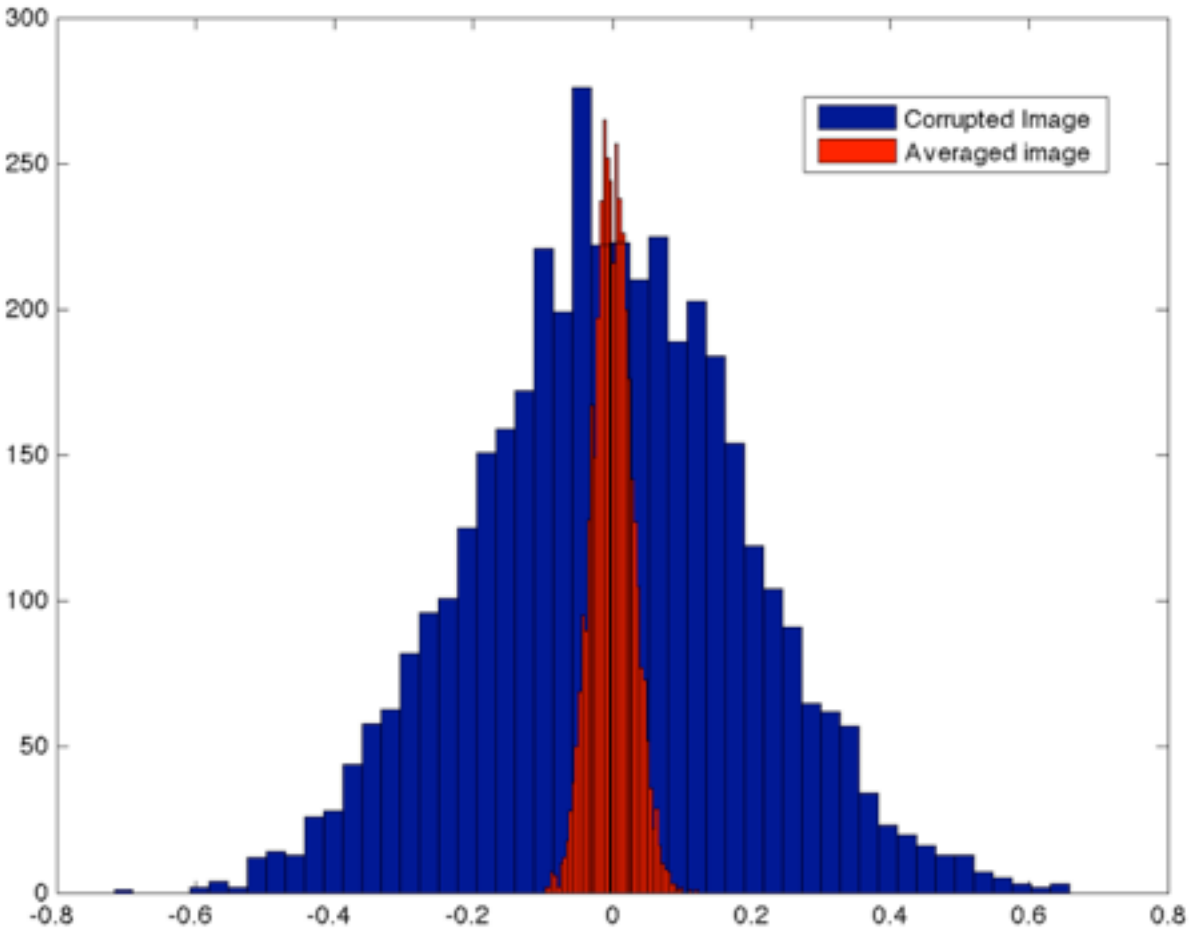
$$\sigma^2_{\bar{g}(x,y)} = \frac{1}{K}\sigma^2_{\eta(x,y)} \quad \Rightarrow \quad \sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{K}}\sigma_{\eta(x,y)}$$

That is, the standard deviation is reduced by a factor of $\sqrt{K}$.

This means that the variance from *f(x,y)* of the averaged image $\bar{g}(x,y)$ is decreased and the averaged image has less error than the original corrupted image.

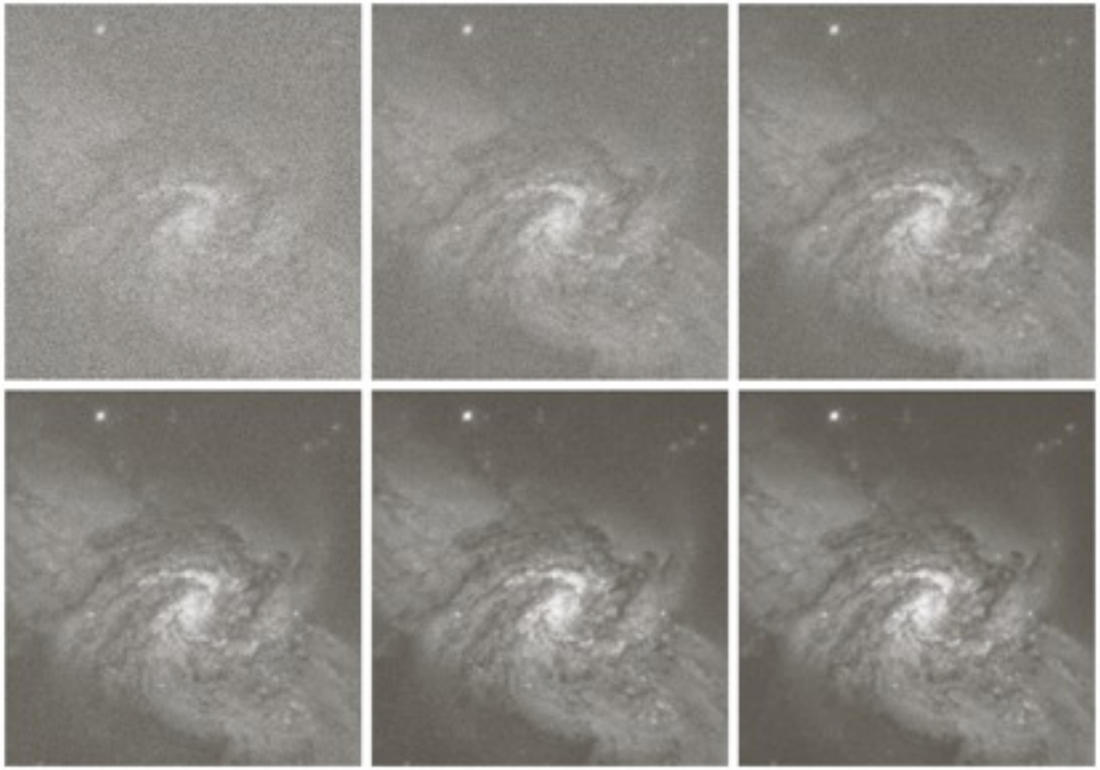
This means that the variance from *f(x,y)* of the averaged image $\bar{g}(x, y)$ is decreased and the averaged image has less error than the original corrupted image.

uncorrupted	corrupted by noise	filtered by 50 averages
		



eraged image $\bar{g}(x, y)$
r than the original

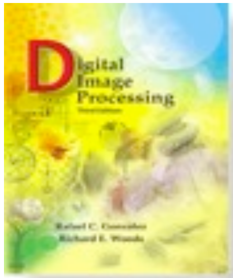
ltered by 50 averages



a b c
d e f

FIGURE 2.26 (a) Image of Galaxy Pair NGC 3314 corrupted by additive Gaussian noise. (b)–(f) Results of averaging 5, 10, 20, 50, and 100 noisy images, respectively. (Original image courtesy of NASA.)

The averaging technique is used also in other imaging modalites
(ex. MRI)



Chapter 2

Digital Image Fundamentals

Example: 69 images of the moon taken by prof Hans ZMK with a telescope (24.01.2010).

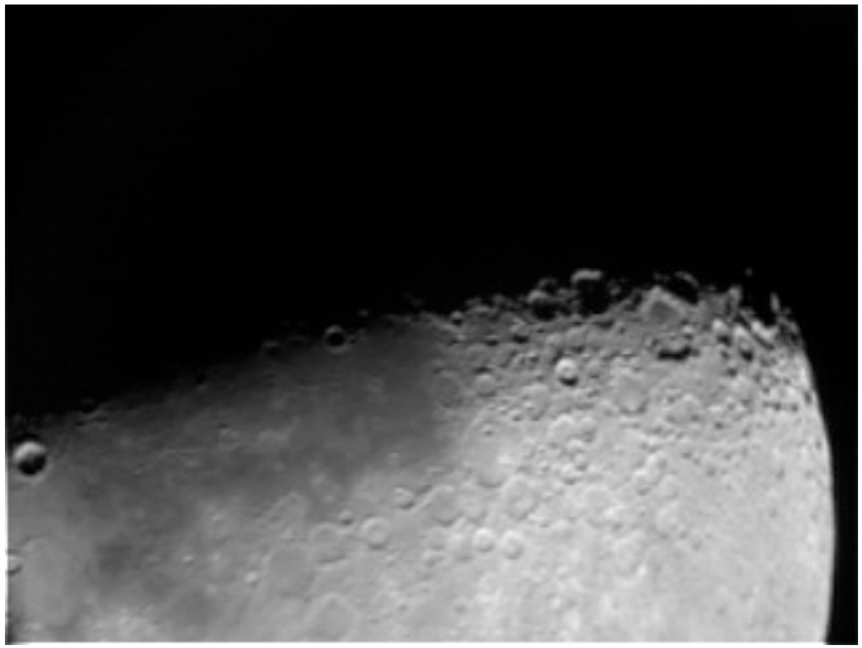
Example: 69 images of the moon taken by prof Hans ZMK with a telescope (24.01.2010).

Image obtained by the averaging method

Example: 69 images of the moon taken by prof Hans ZMK with a telescope (24.01.2010).

Image obtained by the averaging method

Image enhanced using unsharp masking

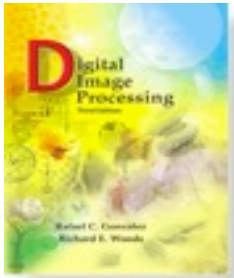


Removal of shading (knowing the shading function)

$$h(x, y)$$

$$g(x, y) = h(x, y) f(x, y)$$

$$f(x, y) = g(x, y) / h(x, y)$$



Removal of shading (knowing the shading function)

We assume the shading to be a known function $h(x, y)$ and that the observed image

$$g(x, y) = h(x, y)f(x, y)$$

Then,

$$f(x, y) = g(x, y)/h(x, y)$$

(NB. Pointwise division!).

Removal of shading (knowing the shading function)

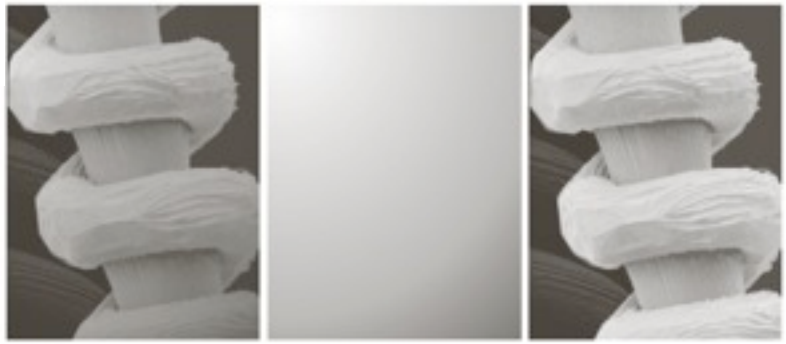
We assume the shading to be a known function $h(x, y)$ and that the observed image

$$g(x, y) = h(x, y) f(x, y)$$

Then,

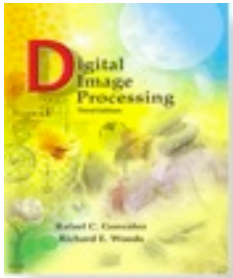
$$f(x, y) = g(x, y) / h(x, y)$$

(NB. Pointwise division!).



a b c

FIGURE 2.29 Shading correction. (a) Shaded SEM image of a tungsten filament and support, magnified approximately 130 times. (b) The shading pattern. (c) Product of (a) by the reciprocal of (b). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)



Removal of shading (knowing the shading function)

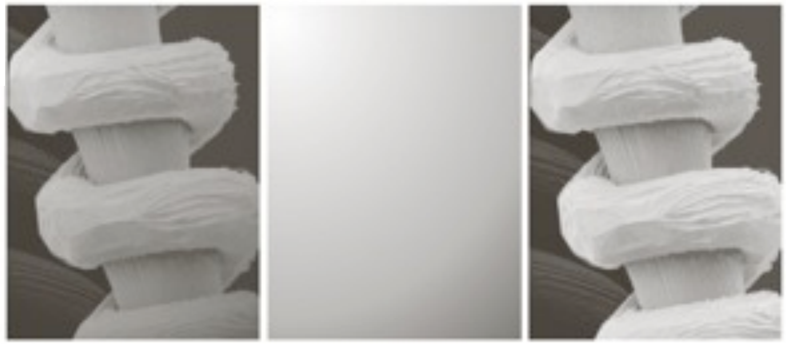
We assume the shading to be a known function $h(x, y)$ and that the observed image

$$g(x, y) = h(x, y)f(x, y)$$

Then,

$$f(x, y) = g(x, y)/h(x, y)$$

(NB. Pointwise division!).



a b c

FIGURE 2.29 Shading correction. (a) Shaded SEM image of a tungsten filament and support, magnified approximately 130 times. (b) The shading pattern. (c) Product of (a) by the reciprocal of (b). (Original image courtesy of Mr. Michael Shaffer, Department of Geological Sciences, University of Oregon, Eugene.)

Care should be used if $h(x, y) = 0$. In that case, $h(x, y)$ is replaced by a small nonzero number.

ROIs and masks

Regions of interests can be extracted by multiplying (pointwise) with an appropriate binary image (mask)

a b c

FIGURE 2.30 (a) Digital dental X-ray image. (b) ROI mask for isolating teeth with fillings (white corresponds to 1 and black corresponds to 0). (c) Product of (a) and (b).

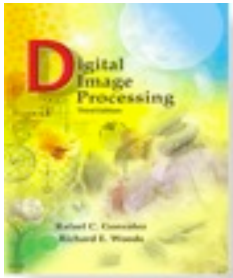
Sets and logical operations

The way sets are used in image processing is to let the elements of the set to be ordered couples $a=(a_1,a_2)$ of the spatial coordinates of the image.

Sets are relevant especially when manipulation ROIs

Let A be a subset of the image. If a pixel a is an element of A, we write

$$a \in A$$



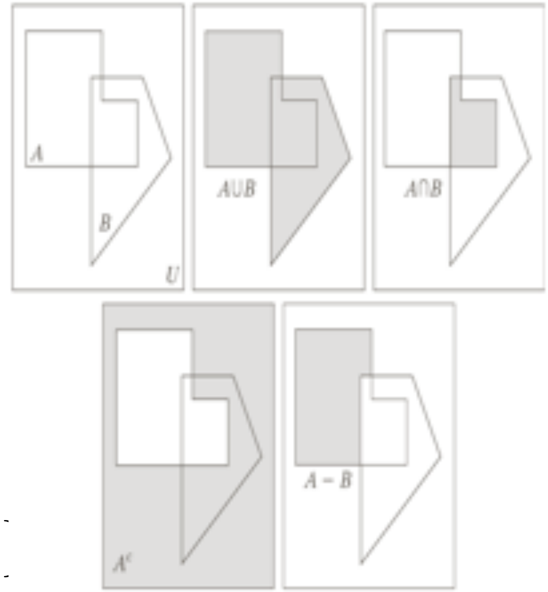
Usual set manipulations: given two sets, A, B,

Union: $A \cup B = \{w|w \in A \text{ or } w \in B\}$

Intersection: $A \cap B = \{w|w \in A \text{ and } w \in B\}$

Complement: $A^C = \{w|w \notin A\}$

Set difference: $A - B = \{w|w \in A \text{ and } w \notin B$



Subsets of pixels of an image can be handled using a binary mask:

$A \equiv$

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

If this pixel is in the set A

If this pixel is not in the set A

Operations involving sets

Set operation	Logical operation	Matlab
Union	OR	max(A,B) , or(A,B)
Intersection	AND	and(A,B)
Complement	NOT	not(A)
Subtraction	AND NOT	and(A,not(B))
Exclusive union	XOR	xor(A,B)

A =

$$\begin{matrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{matrix}$$

B =

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

A =

$$\begin{matrix} 1 & 1 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 1 & 0 \\ 0 & 0 & 0 & 0 & 1 & 0 \end{matrix}$$

B =

$$\begin{matrix} 0 & 0 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 1 \\ 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 \end{matrix}$$

A =

```
1  1  0  0  0  0
1  1  0  0  0  0
0  0  1  1  1  0
0  0  0  0  1  0
```

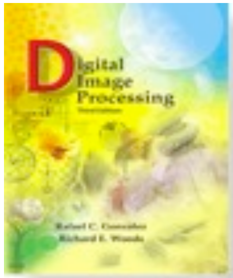
B =

```
0  0  0  0  0  1
0  0  0  0  1  1
0  0  0  0  0  0
0  0  0  0  0  0
```

```
>> or(A,B)

ans =

     1     1     0     0     0     1
     1     1     0     0     1     1
     0     0     1     1     1     0
     0     0     0     0     1     0
```




```
A =

     1     1     0     0     0     0
     1     1     0     0     0     0
     0     0     1     1     1     0
     0     0     0     0     1     0
```

```
B =

     0     0     0     0     0     1
     0     0     0     0     1     1
     0     0     0     0     0     0
     0     0     0     0     0     0
```

```
>> or(A,B)

ans =

     1     1     0     0     0     1
     1     1     0     0     1     1
     0     0     1     1     1     0
     0     0     0     0     1     0
```

```
>> not(B)

ans =

     1     1     1     1     1     0
     1     1     1     1     0     0
     1     1     1     1     1     1
     1     1     1     1     1     1
```

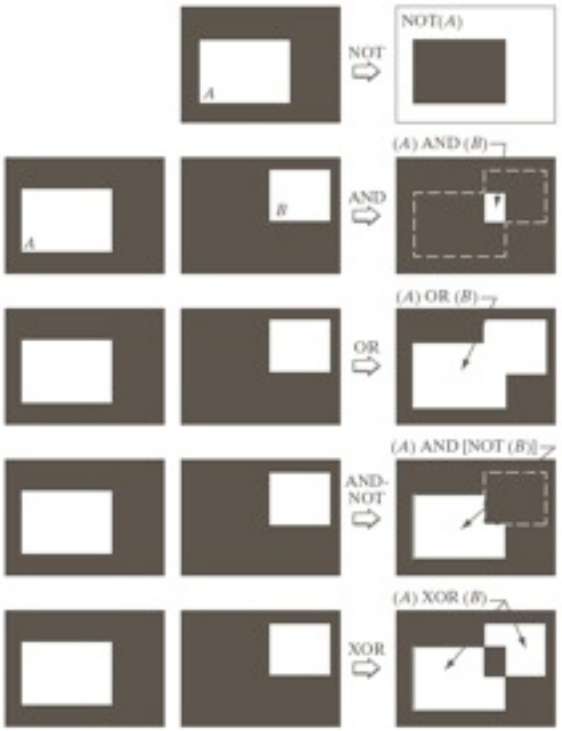
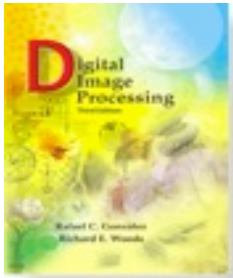


FIGURE 2.33 Illustration of logical operations involving foreground (white) pixels. Black represents binary 0s and white binary 1s. The dashed lines are shown for reference only. They are not part of the result.



For gray-scale images, these concept might not apply (because we need to deal with intensities as well).

Union: pixel-wise maximum
Intersection: pixel-wise minimum
Complement: difference between a constant and the pixel-wise value

For gray-scale images, these concept might not apply (because we need to deal with intensities as well).

Union: pixel-wise maximum
Intersection: pixel-wise minimum
Complement: difference between a constant and the pixel-wise value

Example: Complement (negative) of a grayscale image.

For gray-scale images, these concept might not apply (because we need to deal with intensities as well).

Union: pixel-wise maximum
Intersection: pixel-wise minimum
Complement: difference between a constant and the pixel-wise value

Example: Complement (negative) of a grayscale image.

Let k be the number of bits used to represent the image, so that the intensity range is between $[0, K]$, where $K = 2^k - 1$. Then, the complement is computed as

$$K - f(x, y).$$

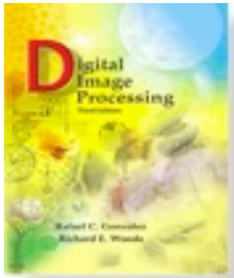
If we think of $A = \{(x, y, z) | (x, y) \text{ spatial coordinates}, z \text{ intensity, of all pixels in } f\}$, then,

$$A^C = \{(x, y, K - z) | (x, y, z) \in A\}$$

jpeg (8 bits, K=255)

jpeg (8 bits, K=255)

$$g(x,y) = 255 - f(x,y)$$



Spatial operations.

Spatial operations are performed directly on the pixels of the image in 3 ways

1) Single pixel operations

2) Operations involving neighbors

3) Geometric spatial transformations

1) Single pixel operations:

Assume given a function T of a single variable, we transform the intensities z to $s=T(z)$ pixel per pixel.

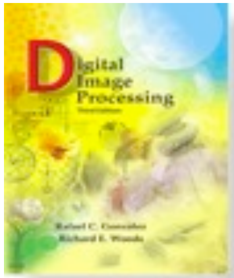
(example: the negative, the power)

1) Single pixel operations:

Assume given a function T of a single variable, we transform the intensities z to $s=T(z)$ pixel per pixel.

(example: the negative, the power)

$$T(z) = z^3 \longrightarrow$$

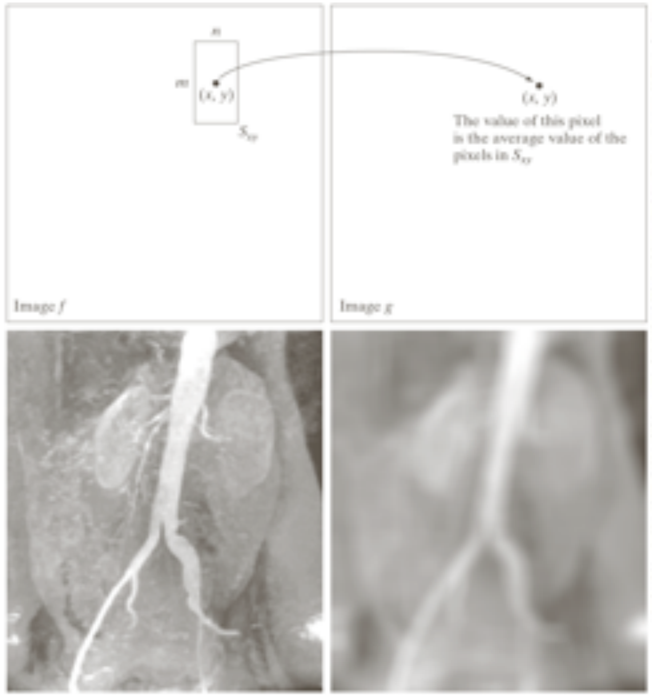


2) Operations involving neighbors

Define S_{xy} a neighborhood of the pixel (x, y). The transformed intensity will be dependent on the intensities of all the pixels in S_{xy}.

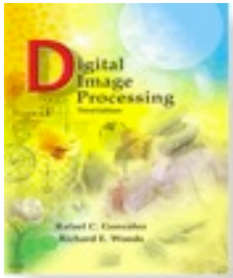
Example: If S_{xy} consists of a subset of m rows and n columns, considering the *blurring* transformation

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



a b
c d

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



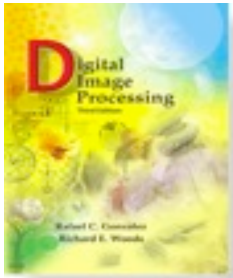
3) Geometric spatial transformations and image registration

These are transformations that modify the spatial relationship between the pixels in an image (for example by shearing and tearing)

$$(x,y) = T(v,w)$$

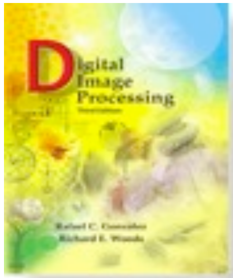
Pixel coordinates of the new image

Pixel coordinates of the original image



An important class of transformation is that of *affine transformations*, with the general form:

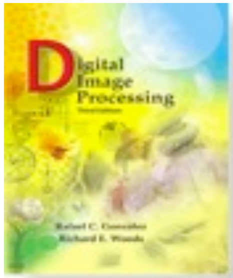
$$[\,x \quad y \quad 1\,] = [\,v \quad w \quad 1\,] \underbrace{\begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}}_{\mathrm{T}}$$



An important class of transformation is that of *affine transformations*, with the general form:

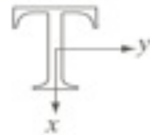
$$[\, x \quad y \quad 1 \,] = [\, v \quad w \quad 1 \,] \underset{\mathrm{T}}{\begin{bmatrix} t_{11} & t_{12} & 0 \\ t_{21} & t_{22} & 0 \\ t_{31} & t_{32} & 1 \end{bmatrix}}$$

NB. These are geometric transformations and the product is the classical matrix-vector product (and not entry-wise).



Chapter 2
Digital Image Fundamentals

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	$\begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v\cos\theta - w\sin\theta$ $y = v\cos\theta + w\sin\theta$	
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)	$\begin{bmatrix} 1 & s_h & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}$	$x = v$ $y = s_h v + w$	

The transformation relocates pixels to a new location. To obtain the transformed image, we need to assign intensity values to the relocated pixels

Interpolation

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.

Image registration is an important application of digital image processing, used to align two or more images of the same object/ scene.

Image registration is an important application of digital image processing, used to align two or more images of the same object/scene.