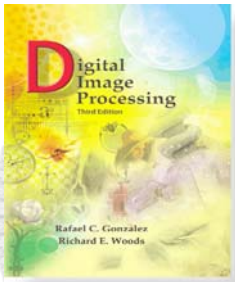
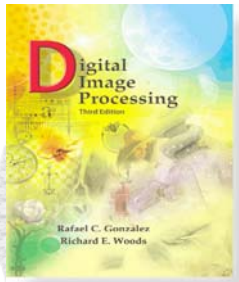


Lecture #26

- Moments; invariant moments
- Eigenvector, principal component analysis
- Boundary coding
- Image primitives
- Image representation: trees, graphs
- Object recognition and classes
- Minimum distance classifiers
- Correlation
- Statistical pattern recognition; Bayes decision rule
- Neural network classifiers



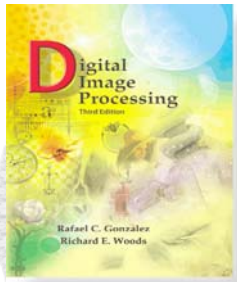
Moments

The calculation of moments of for images and objects in images is very similar to that used to calculate statistical moments

$$m_{pq} = \int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} x^p y^q f(x,y)\,dxdy$$

For an LxL discrete image we can write

$$m_{pq} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} x_i^p y_j^q f(x_i, y_j)$$



Moments

We can also define central moments in a similar manner

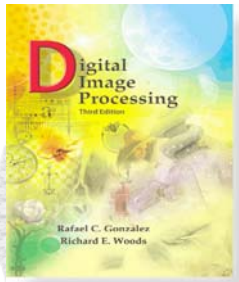
$$\mu_{pq} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\left(x_i - \bar{x}\right)^p \left(y_j - \bar{y}\right)^q f\left(x_i, y_j\right)$$

Some simple moments and means

$$\mu_{00} = m_{00} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} f\left(x_i, y_j\right)$$

$$m_{10} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} x_i f\left(x_i, y_j\right) \qquad m_{01} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1} y_i f\left(x_i, y_j\right)$$

$$\bar{x} = \frac{m_{10}}{m_{00}} \qquad \bar{y} = \frac{m_{01}}{m_{00}}$$



Moments

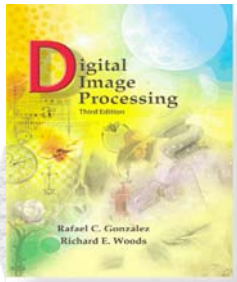
Computing higher moments and means

$$\mu_{20} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\left(x_i - \bar{x}_i\right)^2 f\left(x_i, y_j\right) = m_{20} - \bar{x}m_{10} \qquad \mu_{02} = \sum_{i=0}^{L-1}\sum_{j=0}^{L-1}\left(y_j - \bar{y}_j\right)^2 f\left(x_i, y_j\right) = m_{02} - \bar{y}m_{01}$$

$$\eta_{20} = \frac{\mu_{20}}{\mu_{00}^2} \qquad \eta_{02} = \frac{\mu_{02}}{\mu_{00}^2}$$

We can use these second moments to define the first invariant moment

$$\phi_1 = \eta_{20} + \eta_{02}$$



Invariant Moments

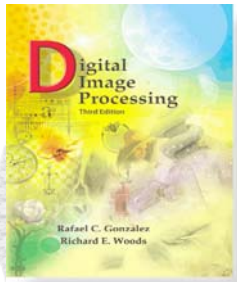
$$\phi_1 = \eta_{20} + \eta_{02}$$

$$\phi_2 = (\eta_{20} - \eta_{02})^2 + 4\eta_{11}^2$$

$$\phi_3 = (\eta_{30} - 3\eta_{12})^2 + (\eta_{21} - 3\eta_{03})^2$$

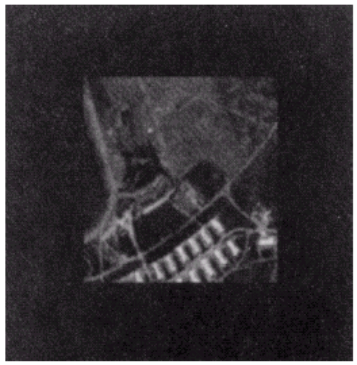
$$\phi_7 = (3\eta_{21} - \eta_{03})(\eta_{30} + \eta_{12})\left[(\eta_{30} + \eta_{12})^2 - 3(\eta_{21} + \eta_{03})^2\right]$$

$$+(3\eta_{12} - \eta_{30})(\eta_{21} + \eta_{03})\left[3(\eta_{30} + \eta_{12})^2 - (\eta_{21} + \eta_{03})^2\right]$$



Moments

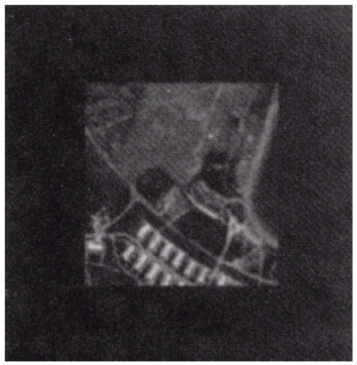
Original



a
b c
d e

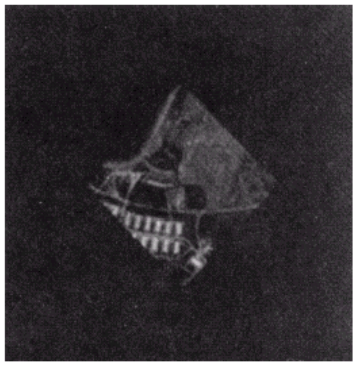
FIGURE 11.25 Images used to demonstrate properties of moment invariants (see Table 11.3).

Reduced 50% in size

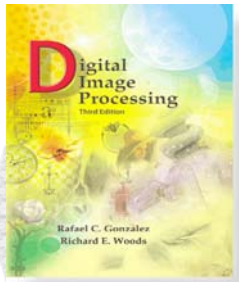


Mirrored

Rotated 2°



Rotated 45°



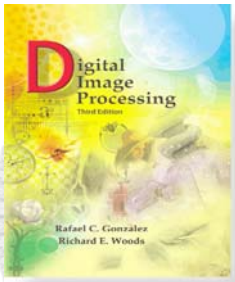
Moments

There are seven moments which are invariant to translation, rotation, and scale.

Each row in this table should remain constant. The constancy shows the utility of the moment

Invariant (Log)	Original	Half Size	Mirrored	Rotated 2°	Rotated 45°
ϕ_1	6.249	6.226	6.919	6.253	6.318
ϕ_2	17.180	16.954	19.955	17.270	16.803
ϕ_3	22.655	23.531	26.689	22.836	19.724
ϕ_4	22.919	24.236	26.901	23.130	20.437
ϕ_5	45.749	48.349	53.724	46.136	40.525
ϕ_6	31.830	32.916	37.134	32.068	29.315
ϕ_7	45.589	48.343	53.590	46.017	40.470

TABLE 11.3 Moment invariants for the images in Figs. 11.25(a)-(e).



Moments

Translated

Original

Reduced 50% in size

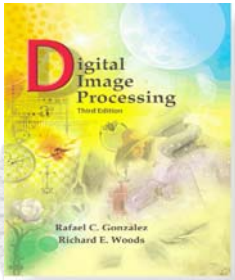
Mirrored

Rotated 90°

Rotated 45°

a b c
d e f

FIGURE 11.37 (a) Original image. (b)–(f) Images translated, scaled by one-half, mirrored, rotated by 45° and rotated by 90°, respectively.



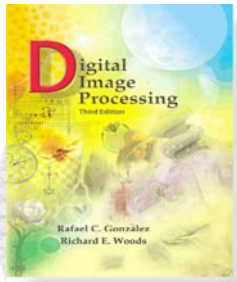
Representation and Description

Each row in this table should also remain constant. The constancy shows the utility of the moment

Moment Invariant	Original Image	Translated	Half Size	Mirrored	Rotated 45°	Rotated 90°
ϕ_1	2.8662	2.8662	2.8664	2.8662	2.8661	2.8662
ϕ_2	7.1265	7.1265	7.1257	7.1265	7.1266	7.1265
ϕ_3	10.4109	10.4109	10.4047	10.4109	10.4115	10.4109
ϕ_4	10.3742	10.3742	10.3719	10.3742	10.3742	10.3742
ϕ_5	21.3674	21.3674	21.3924	21.3674	21.3663	21.3674
ϕ_6	13.9417	13.9417	13.9383	13.9417	13.9417	13.9417
ϕ_7	−20.7809	−20.7809	−20.7724	20.7809	−20.7813	−20.7809

TABLE 11.5 Moment invariants for the images in Fig. 11.37.

This example is much better than the example from the 2nd edition



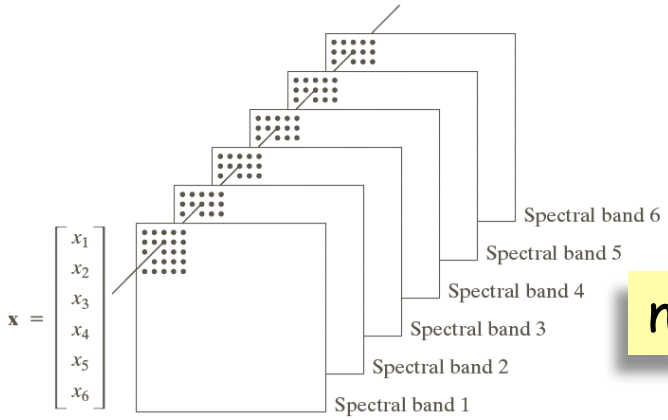
Eigenvectors and Eigenvalues

For RGB images (with 3 components) we can write each pixel as $\underline{x}$

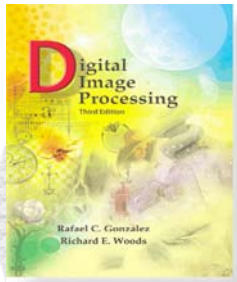
$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ x_3 \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

For registered multi-spectral images (with n components) we can write $\underline{x}$ as

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix}$$



n=6 components

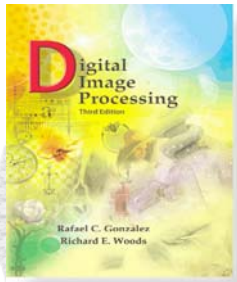


Eigenvectors and Eigenvalues

For these n registered images we can compute the mean vector (nx1) and covariance matrix (nxn) for the set of all $\underline{x}$

$$\underline{x} = \begin{bmatrix} x_1 \\ x_2 \\ \vdots \\ x_n \end{bmatrix} \qquad \underline{m}_x = E\{\underline{x}\} = \begin{bmatrix} E\{x_1\} \\ E\{x_2\} \\ \vdots \\ E\{x_n\} \end{bmatrix}$$

$$C_x = E\left\{(\underline{x} - \underline{m}_x)(\underline{x} - \underline{m}_x)^T\right\} = \begin{bmatrix} E\{(x_1 - m_1)(x_1 - m_1)\} & E\{(x_1 - m_1)(x_2 - m_2)\} & \cdots & E\{(x_1 - m_1)(x_n - m_n)\} \\ E\{(x_2 - m_2)(x_1 - m_1)\} & E\{(x_2 - m_2)(x_2 - m_2)\} & & \vdots \\ \vdots & & \ddots & \vdots \\ E\{(x_n - m_n)(x_1 - m_1)\} & \cdots & \cdots & E\{(x_n - m_n)(x_n - m_n)\} \end{bmatrix}$$



Eigenvectors and Eigenvalues

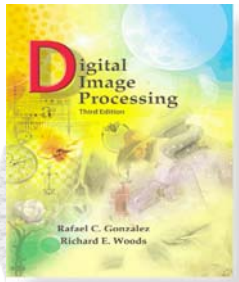
Transform the data by a Hotelling* transformation

$$\underline{y} = \underline{A}\left(\underline{x} - \underline{m}_x\right)$$

where the rows of $\underline{A}$ are the eigenvectors of the covariance matrix $\underline{C}_x$

$$\underline{A} = \begin{bmatrix} \underline{e}_1^T \\ \underline{e}_2^T \\ \vdots \\ \underline{e}_n^T \end{bmatrix}$$

* Also known as the Karhunen-Loeve transformation



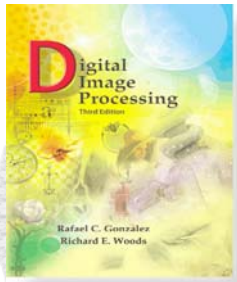
Eigenvectors and Eigenvalues

The transformed variable $\underline{y}$ has the properties that

$$\underline{m}_y = E\{\underline{y}\} = 0 \quad \text{and} \quad \underline{C}_y = \underline{A}\underline{C}_x\underline{A}^T = \begin{bmatrix} \lambda_1 & 0 & \cdots & 0 \\ 0 & \lambda_2 & & \vdots \\ \vdots & & \ddots & \vdots \\ 0 & \cdots & \cdots & \lambda_n \end{bmatrix}$$

where $\lambda_1 > \lambda_2 > \cdots > \lambda_n$

C_y is diagonal which indicates that the transformed $\underline{y}$ vectors are uncorrelated.



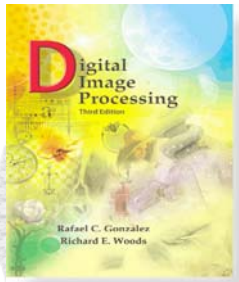
Eigenvectors and Eigenvalues

This transformation can be inverted to give back the original image vectors $\underline{x}$

$$\underline{x} = \underline{A}^T \underline{y} + \underline{m}_x$$

where

$$\underline{A} = \begin{bmatrix} \underline{e}_1^T \\ \underline{e}_2^T \\ \vdots \\ \underline{e}_k^T \end{bmatrix}$$



Eigenvectors and Eigenvalues

What if I replace A by B which has only the k eigenvectors corresponding to the k largest eigenvalues?

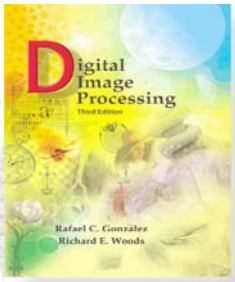
$$\underline{B} = \begin{bmatrix} \underline{e}_1^T \\ \underline{e}_2^T \\ \vdots \\ \underline{e}_k^T \end{bmatrix}$$

We now have an approximation to x given by

$$\underline{\hat{x}} = \underline{B}^T \underline{y} + \underline{m}_x$$

The difference (rms error) between this approximation and the original x is given by the sum of the eigenvalues corresponding to the removed eigenvectors.

$$\underline{e}_{rms} = \sum_{j=k+1}^{n} \lambda_j$$

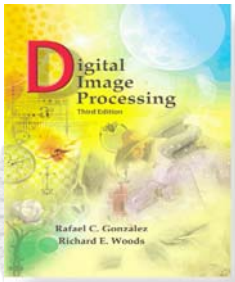


Eigenvectors and Eigenvalues

The transformed vectors y are orthogonal and form a orthoginal basis set for describing the original set of images.

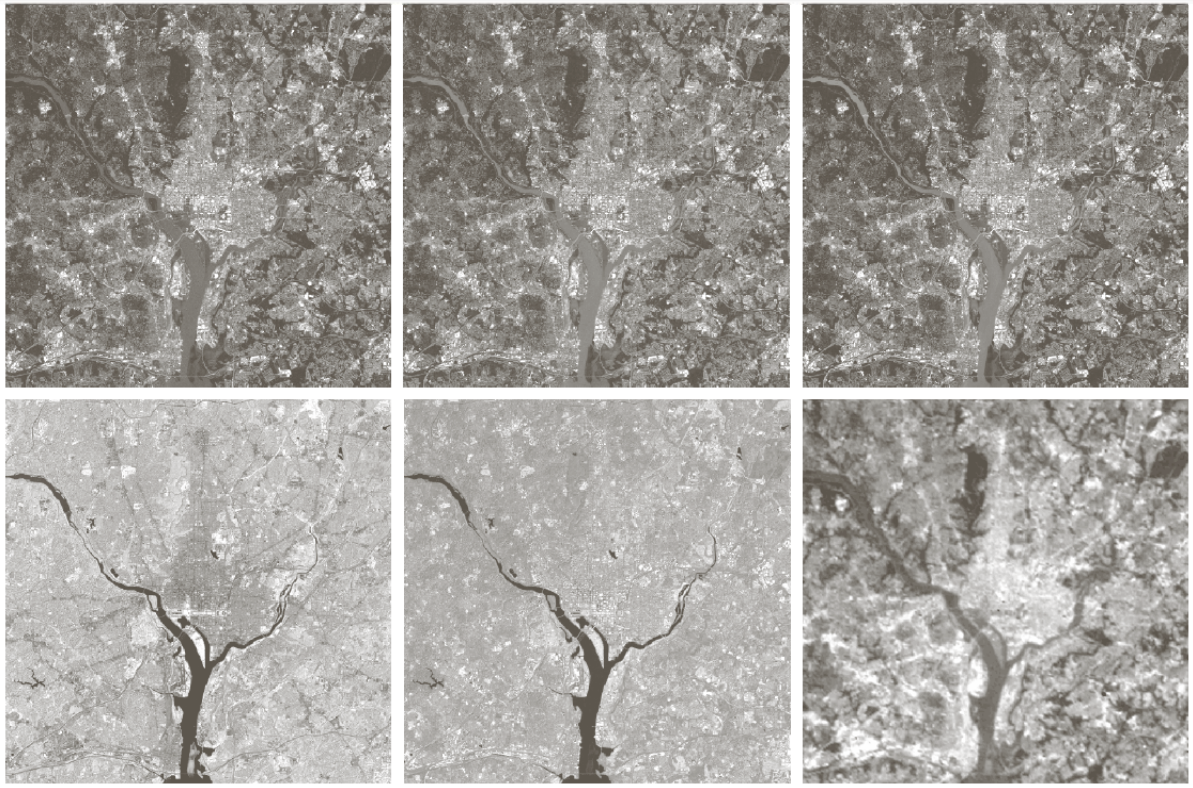
$$\hat{\underline{x}} = \sum_{i=1}^{k} a_i \underline{y}_i$$

Each $\underline{y}_i$ is an eigenvector of C_x and represents a "feature" of the original set of images. $\underline{y}_1$ corresponds to the most significant feature since $\underline{y}_1$ corresponds to the largest eigenvalue λ_1. $\underline{y}_2$ corresponds to the next most significant feature since $\underline{y}_2$ corresponds to the next largest eigenvalue λ_2.



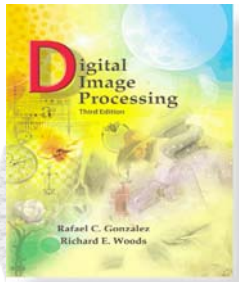
Eigenvector Example

Six registered multi-spectral images: blue, green, red, near infrared, middle infrared, and far (thermal) infrared bands.



a b c
d e f

FIGURE 11.38 Multispectral images in the (a) visible blue, (b) visible green, (c) visible red, (d) near infrared, (e) middle infrared, and (f) thermal infrared bands. (Images courtesy of NASA.)



Eigenvector Example

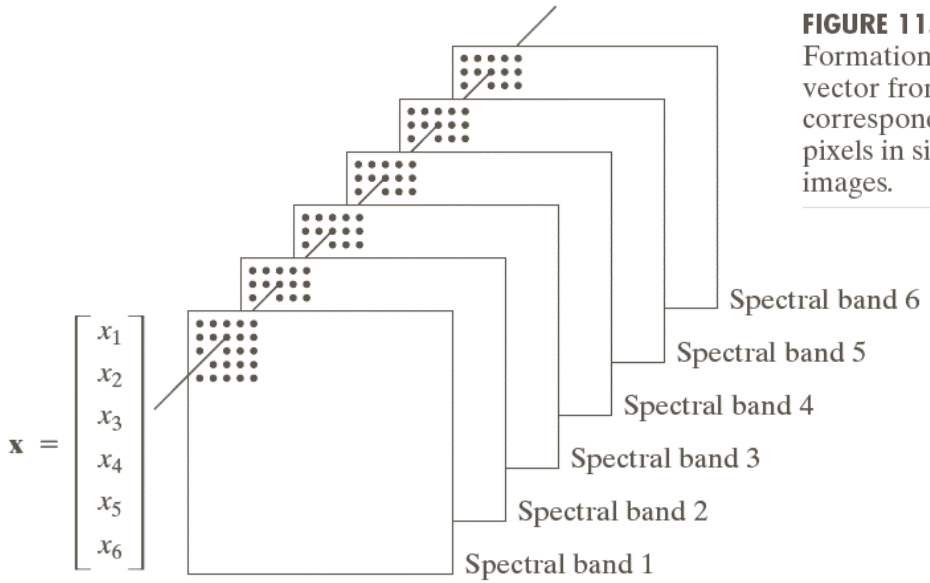
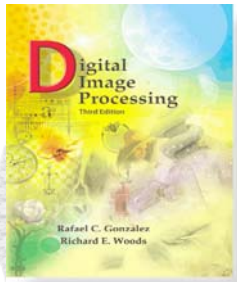


FIGURE 11.39 Formation of a vector from corresponding pixels in six images.

Each pixel in this multi-spectal image can be described by a six-component pixel vector $\underline{x}$

These are 384x239 pixels so there are 91776 pixel vectors describing the complete image

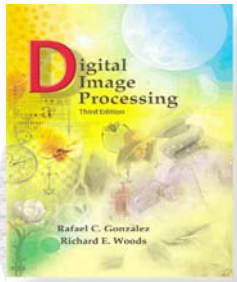


Eigenvector Example

We can use a computer program to compute the mean and 6x6 covariance matrix C_x for these 91776 pixels. We can then calculate the eigenvectors e_1 to e_6 for C_x. For this example the eigenvalues are

λ_1	λ_2	λ_3	λ_4	λ_5	λ_6
10344	2966	1401	203	94	31

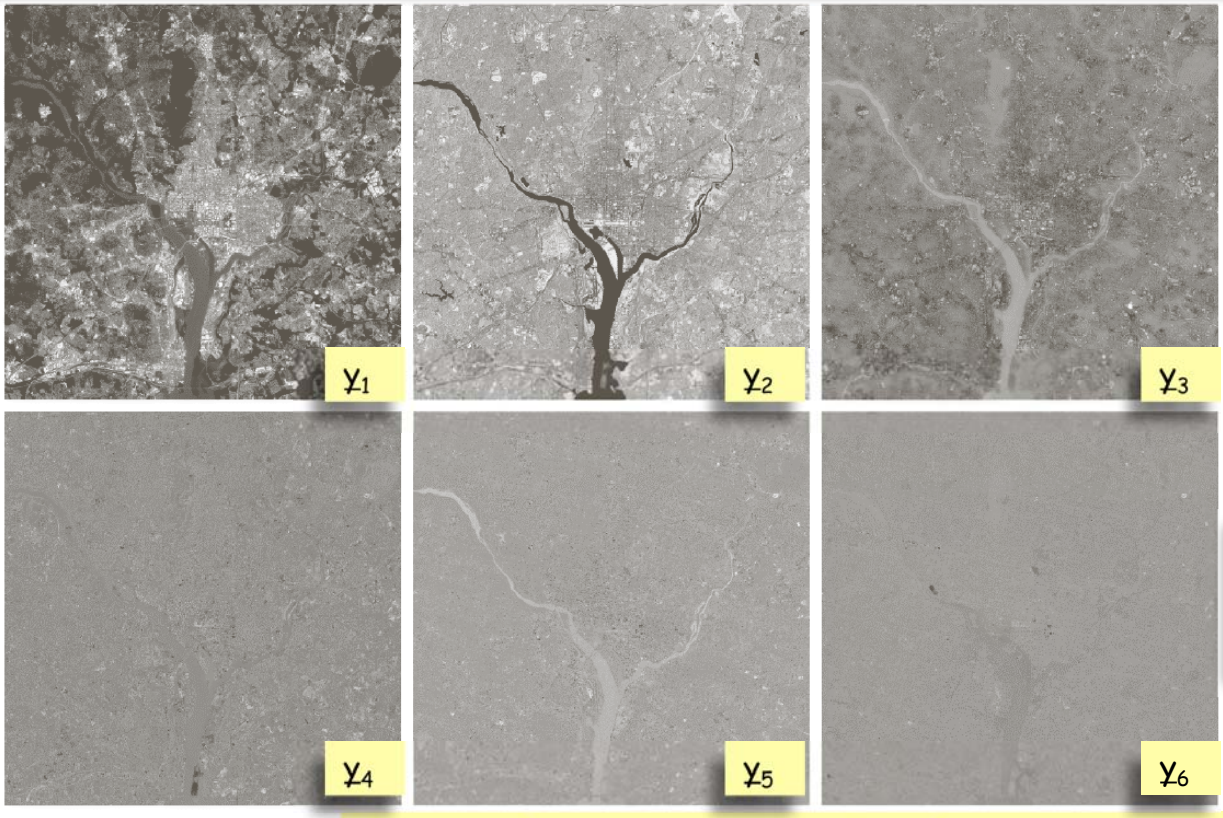
TABLE 11.6
Eigenvalues of the covariance matrices obtained from the images in Fig. 11.38.



Eigenvector Example

These are the six principal component images given by $\underline{y}_i = e_i^T(\underline{x} - \underline{m}_x)$

This is the most significant image with the most information (λ_1)

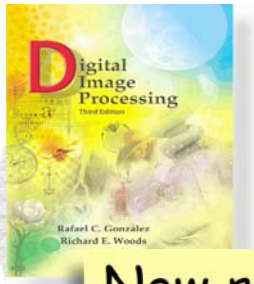


This is the least significant image with the least information (λ_1)

This transformation rearranges the information presented in the multi-spectral image.

a b c
d e f

FIGURE 11.40 The six principal component images obtained from vectors computed using Eq. (11.4-6). Vectors are converted to images by applying Fig. 11.39 in reverse.



Eigenvector Example

Now reconstruct the six original images using only the two most significant eigenvectors.

Compute

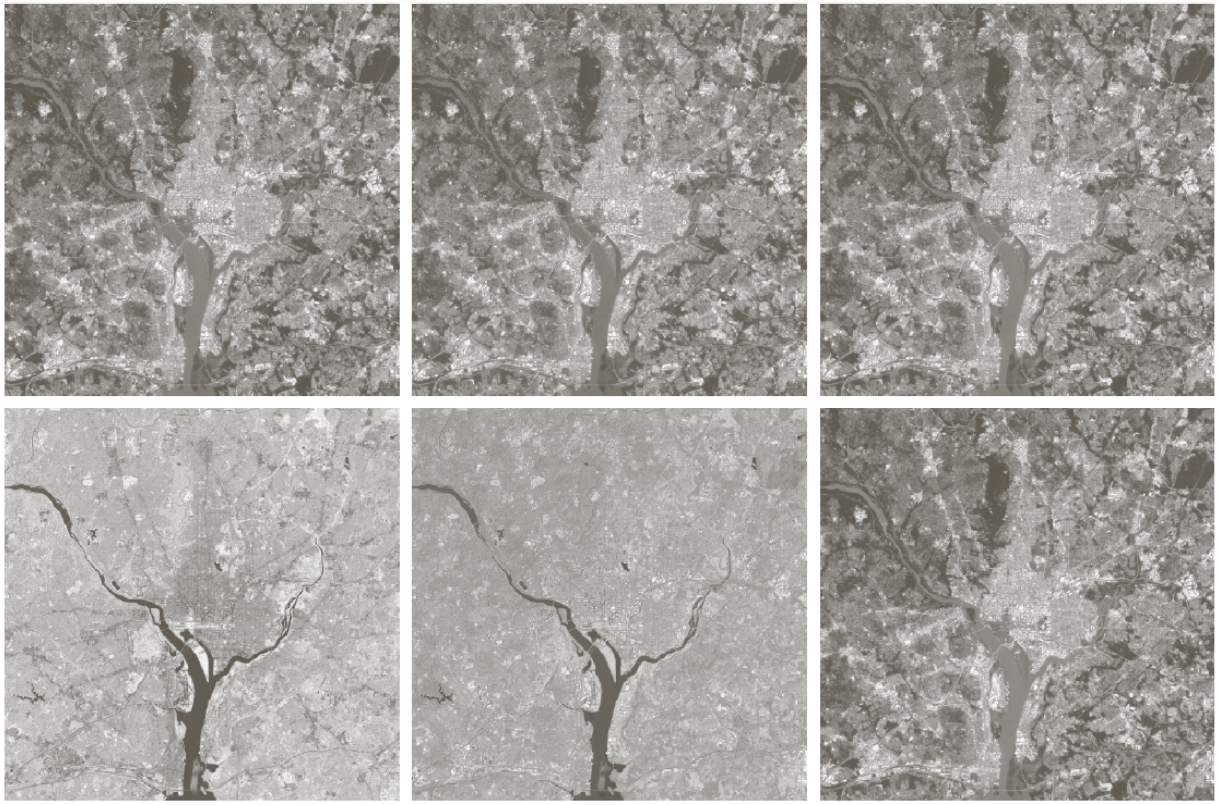
$$\underline{y} = \underline{A}\left(\underline{x} - \underline{m}_x\right)$$

Use only

$$\underline{B} = \begin{bmatrix} \underline{e}_1^T \\ \underline{e}_2^T \end{bmatrix}$$

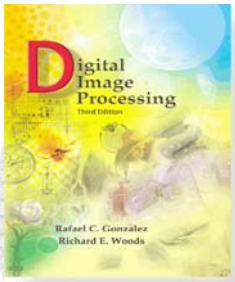
Invert

$$\hat{\underline{x}} = \underline{B}^T \begin{bmatrix} \underline{y}_1 \\ \underline{y}_2 \end{bmatrix} + \underline{m}_x$$



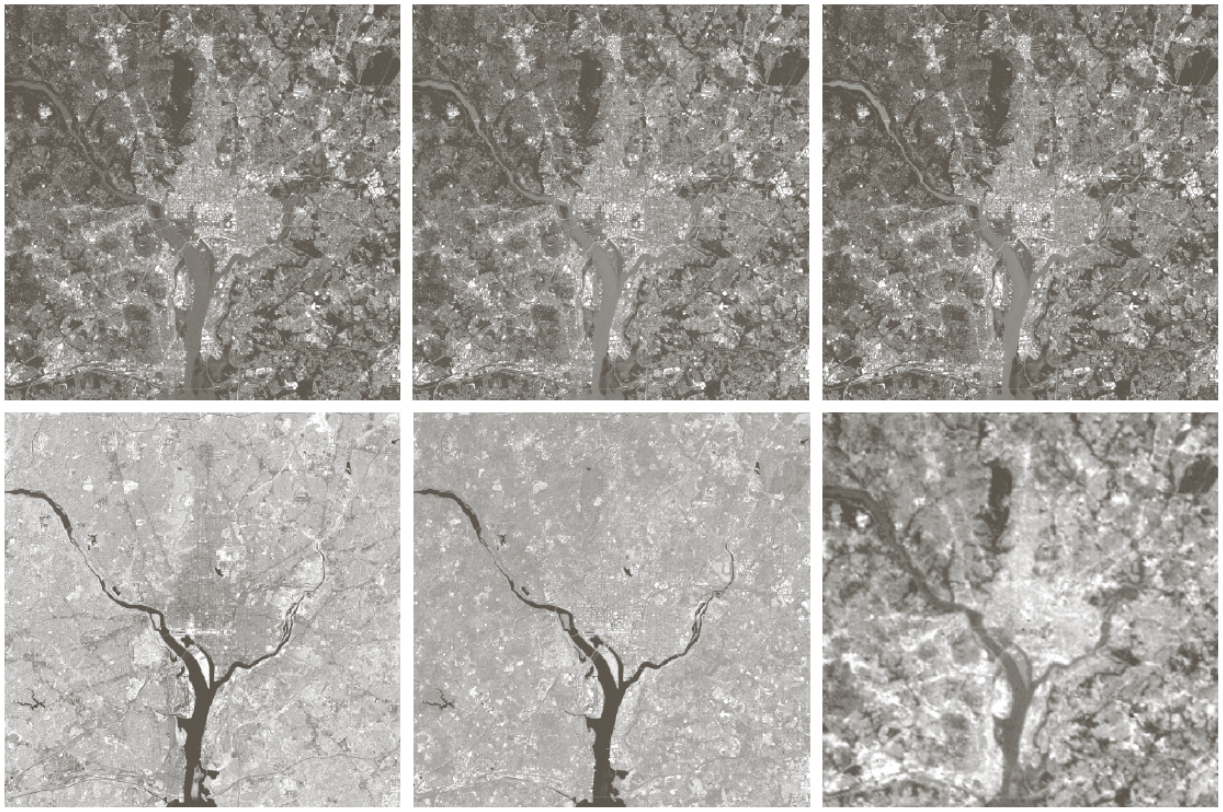
a b c
d e f

FIGURE 11.41 Multispectral images reconstructed using only the two principal component images corresponding to the two principal component images with the largest eigenvalues (variance). Compare these images with the originals in Fig. 11.38.



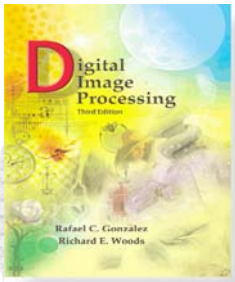
Eigenvector Example

Original images



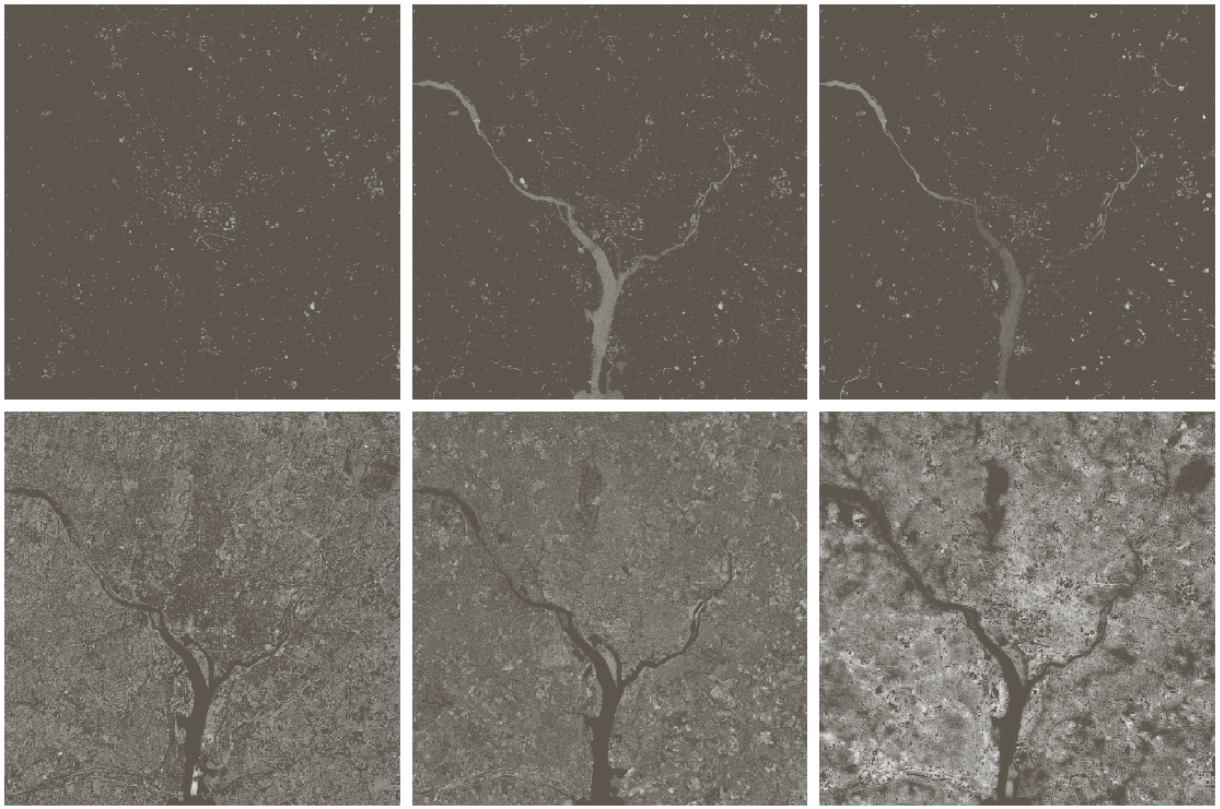
a b c
d e f

FIGURE 11.38 Multispectral images in the (a) visible blue, (b) visible green, (c) visible red, (d) near infrared, (e) middle infrared, and (f) thermal infrared bands. (Images courtesy of NASA.)



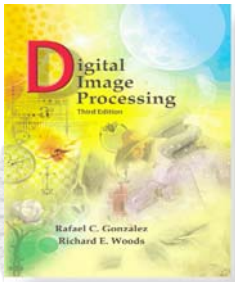
Eigenvector Example

Differences between the original images and the images reconstructed using only two eigenvectors.



a b c
d e f

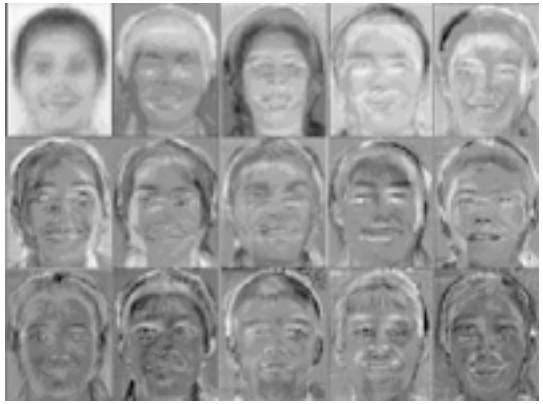
FIGURE 11.42 Differences between the original and reconstructed images. All difference images were enhanced by scaling them to the full [0, 255] range to facilitate visual analysis.



Eigenface Analysis

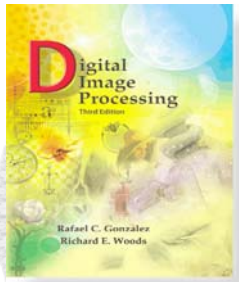
Original dataset of registered faces

Corresponding 15 eigenfaces

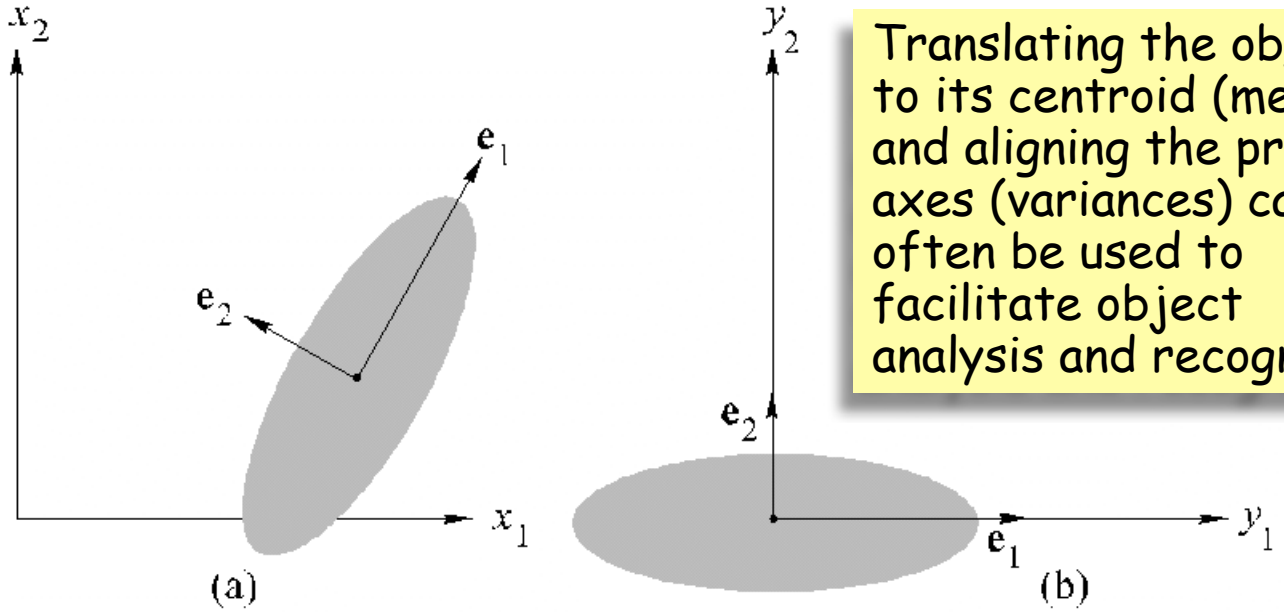


Eigenvector vector expansions have been finding increasing application in such areas as face recognition.

http://www.geop.ubc.ca/CDSST/eigenfaces.html.

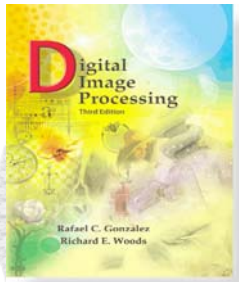


Eigenvalues & Eigenvectors



Translating the object to its centroid (mean) and aligning the principal axes (variances) can often be used to facilitate object analysis and recognition.

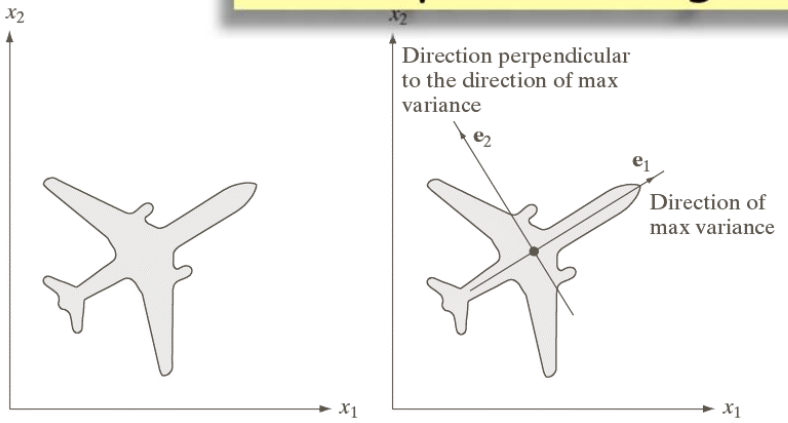
The first eigenvector $\underline{e}_1$ corresponds to the axis of largest variance, i.e., the principal axis.



Eigenvector Example #2

1. Represent the boundary points of an object as a set of 2-D vectors $\underline{x}$ and compute $\underline{C}_x$

2. Compute the eigenvectors of $\underline{C}_x$

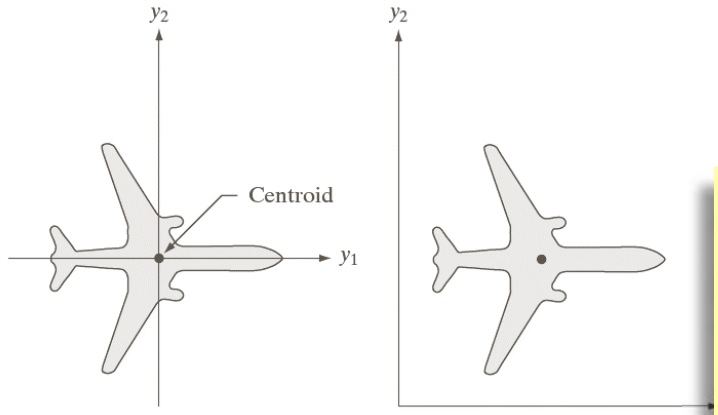


$$\mathbf{m}_x = \begin{bmatrix} 3 \\ 3 \end{bmatrix}$$

$$\mathbf{C}_x = \begin{bmatrix} 3.333 & 2.00 \\ 2.00 & 3.333 \end{bmatrix}$$

$$\mathbf{e}_1 = \begin{bmatrix} 0.707 \\ 0.707 \end{bmatrix}$$

$$\mathbf{e}_2 = \begin{bmatrix} -0.707 \\ 0.707 \end{bmatrix}$$

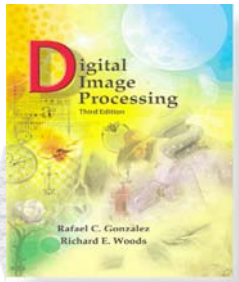


a b
c d

FIGURE 11.43
(a) An object. (b) Object showing eigenvectors of its covariance matrix. (c) Transformed object, obtained using Eq. (11.4-6). (d) Object translated so that all its coordinate values are greater than 0.

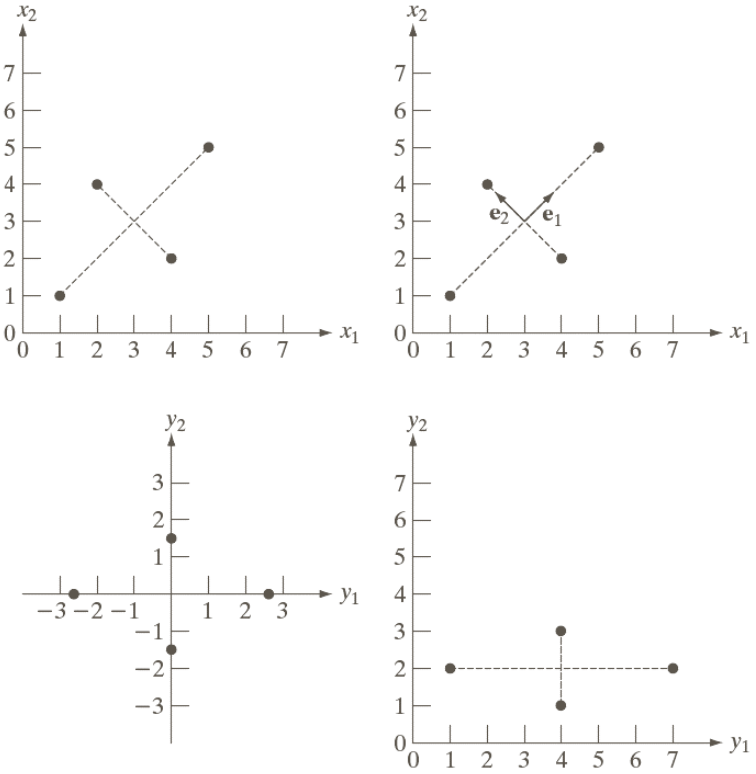
3. Compute the mean $\underline{m}_x$ and translate the mean to the origin

4. Rotate the object so that $\underline{e}_1$ corresponds to the horizontal axis and $\underline{e}_2$ to the vertical axis.



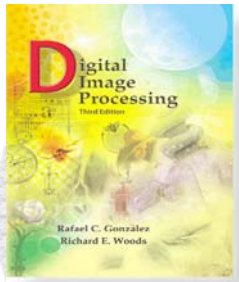
Manual Eigenvector Example

This is a simple boundary analysis which you can replicate by hand.



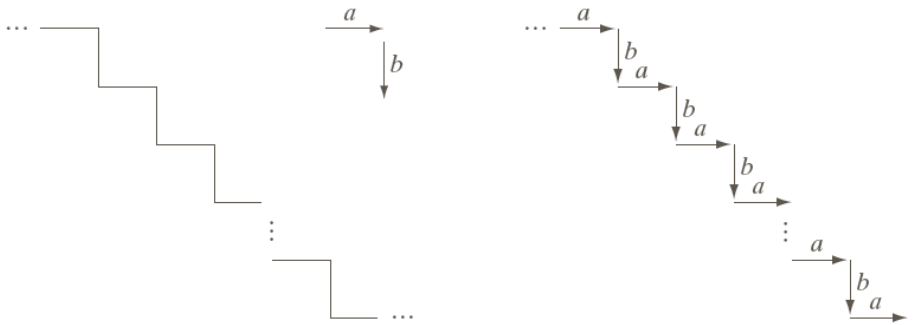
a b
c d

FIGURE 11.44 A manual example. (a) Original points. (b) Eigenvectors of the covariance matrix of the points in (a). (c) Transformed points obtained using Eq. (11.4-6). (d) Points from (c), rounded and translated so that all coordinate values are integers greater than 0. The dashed lines are included to facilitate viewing. They are not part of the data.



Relational Descriptors

This is similar to the texture grammars which we discussed earlier.



a b

FIGURE 11.45 (a) A simple staircase structure. (b) Coded structure.

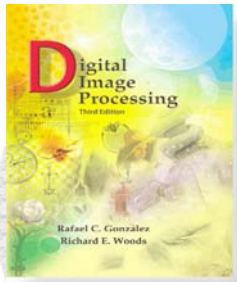
a,b are the elements shown above; S is the starting symbol; A is a variable

(1) $S \rightarrow aA$,
(2) $A \rightarrow bS$, and
(3) $A \rightarrow b$,

These are the re-writing rules which can be used to generate abab… from S

S
aA
abS
abaA
abab

etc.



Representation and Description

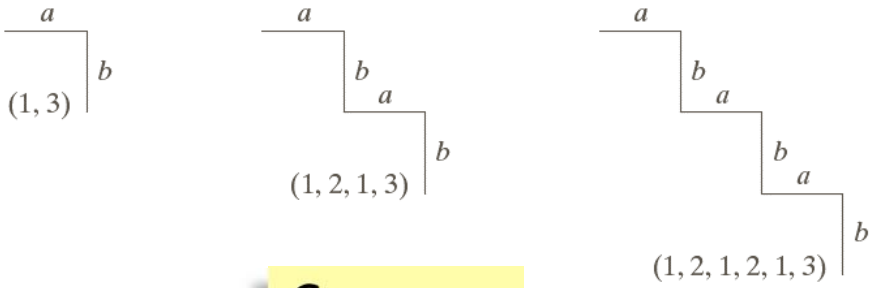
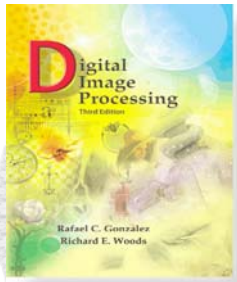


FIGURE 11.46 Sample derivations for the rules $S \rightarrow aA$, $A \rightarrow bS$, and $A \rightarrow b$.

S
aA
abS
abaA
abab

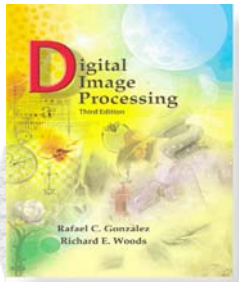
The rules can be repeated to describe larger objects. The numbers correspond to the rules which were applied.



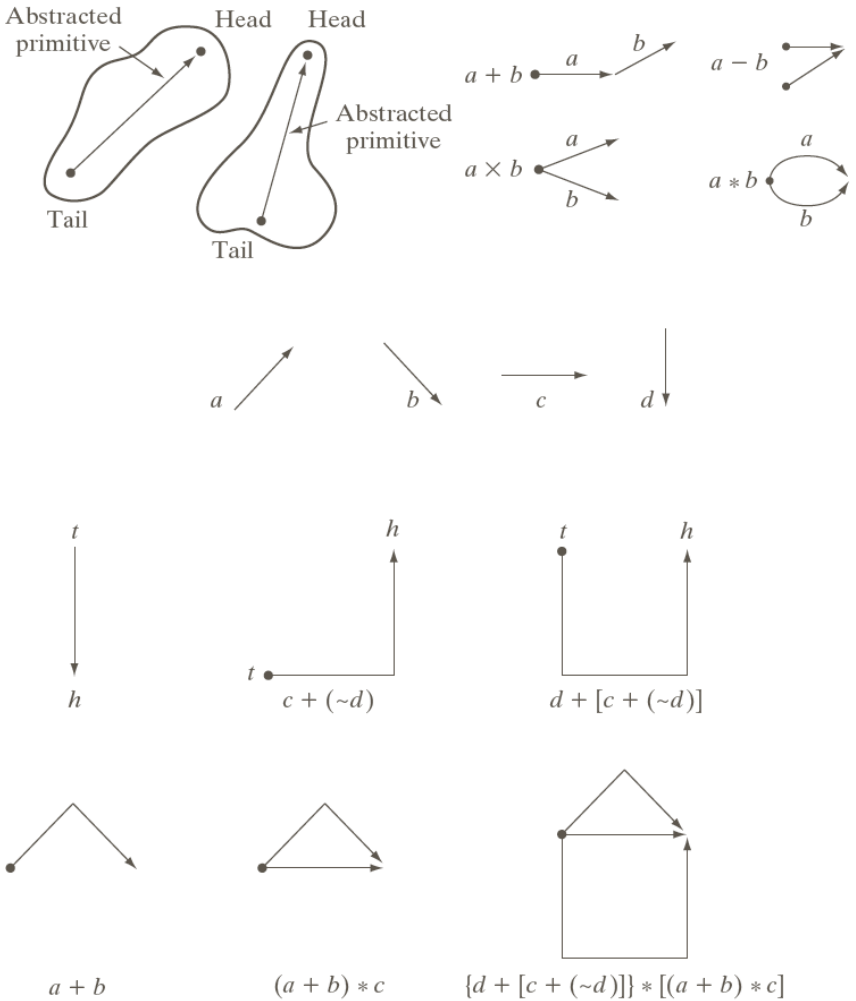
Representation and Description

FIGURE 11.47 Coding a region boundary with directed line segments.

Object boundaries can be coded using head-to-tail connected directed line segments.



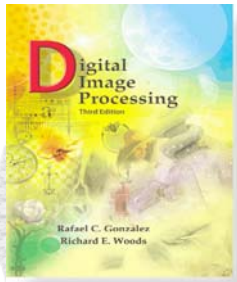
Representation and Description



a b
c
d

FIGURE 11.48
(a) Abstracted primitives. (b) Operations among primitives. (c) A set of specific primitives. (d) Steps in building a structure.

More complex relationships can be defined for directed line segments.



Representation and Description

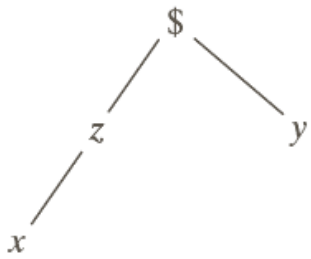
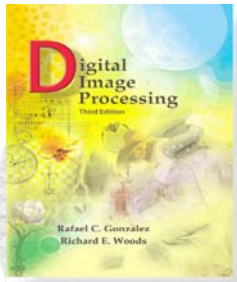
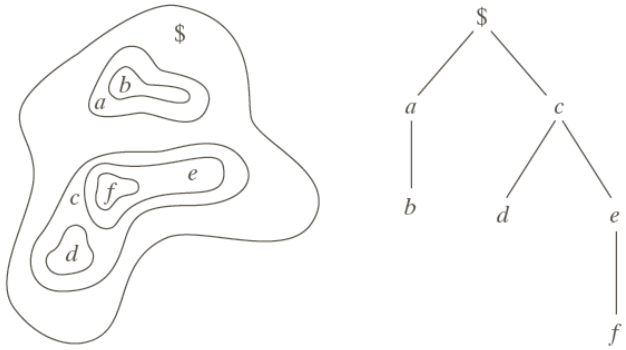


FIGURE 11.49 A simple tree with root $ and frontier xy.

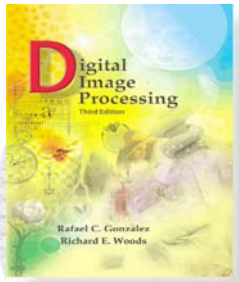


Representation and Description



a b

FIGURE 11.50
(a) A simple composite region. (b) Tree representation obtained by using the relationship "inside of."



Object Recognition

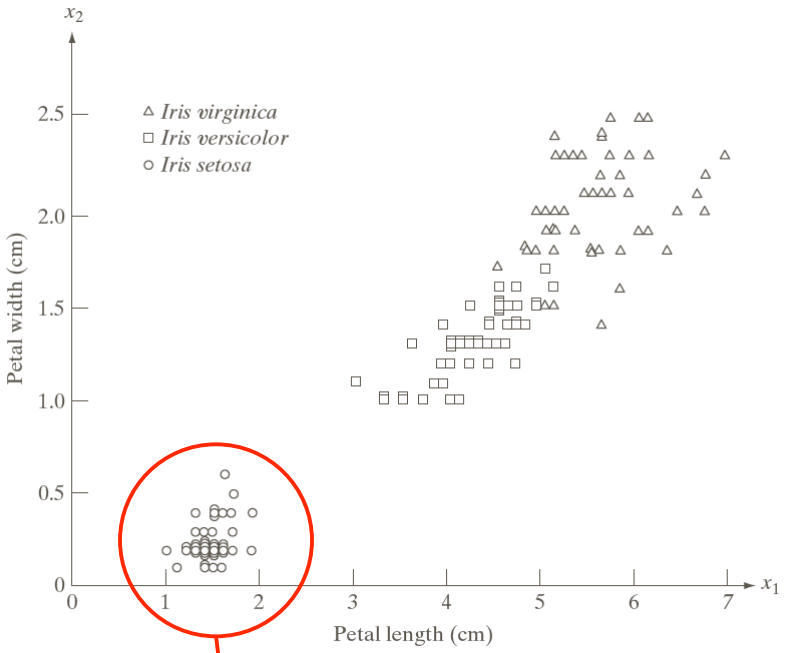
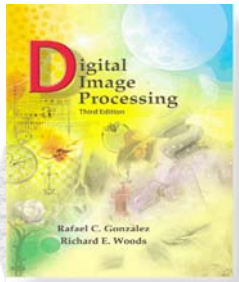
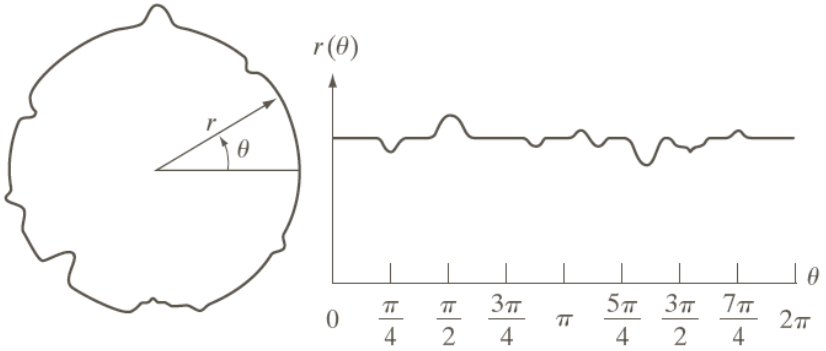


FIGURE 12.1
Three types of iris flowers described by two measurements.

This shows a clearly defined region for *Iris setosa* based upon petal length and width.



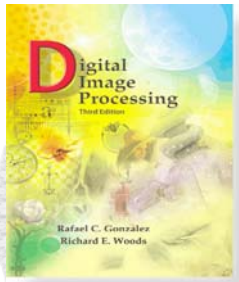
Object Recognition



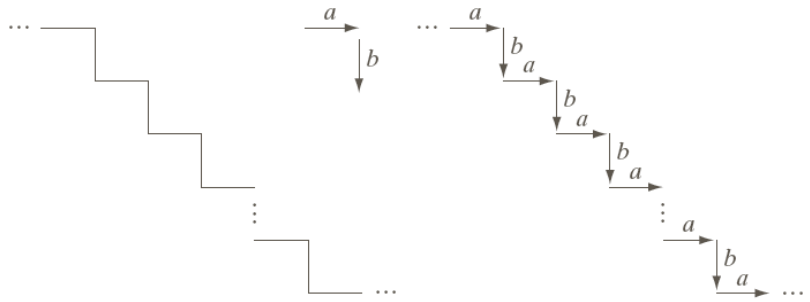
a b

FIGURE 12.2 A noisy object and its corresponding signature.

Signatures can be used to code objects to be recognized.



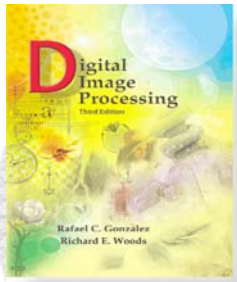
Object Recognition



a b

FIGURE 12.3
(a) Staircase structure.
(b) Structure coded in terms of the primitives *a* and *b* to yield the string description . . . *ababab* . . . .

String descriptions can also be used to code objects.



Object Recognition

More complex structures such as trees are needed to describe the objects in a typical image

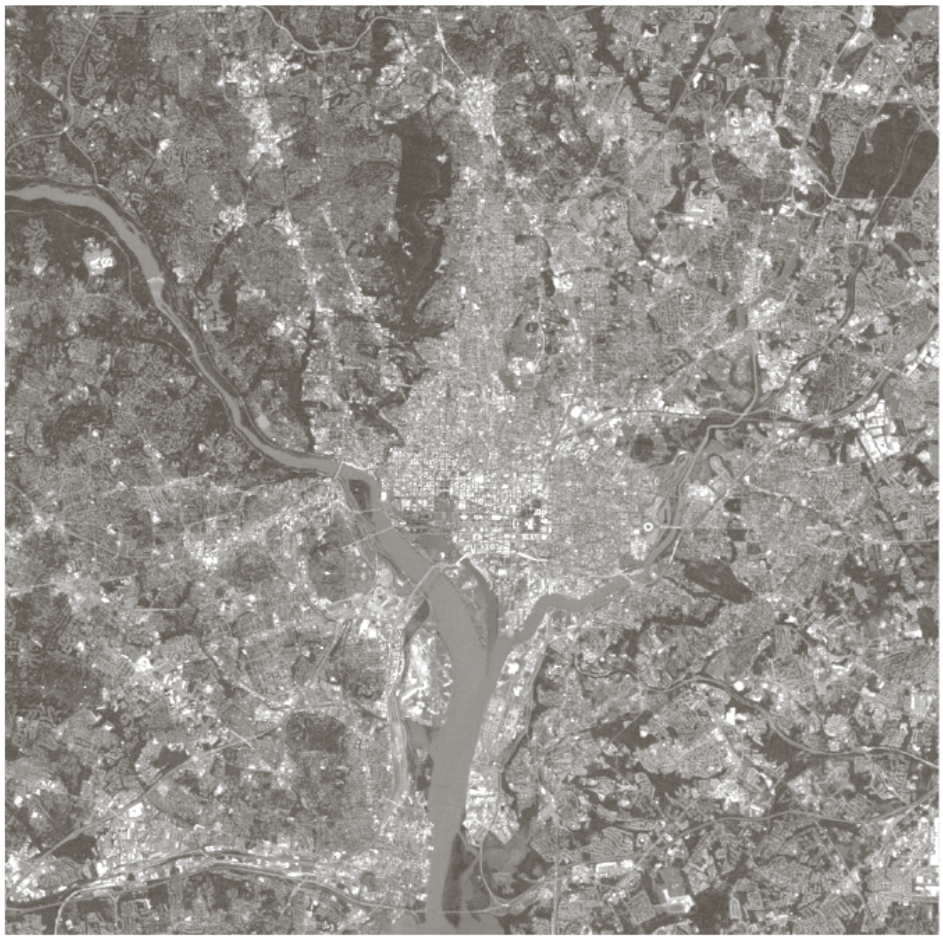
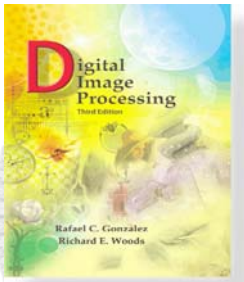


FIGURE 12.4 Satellite image of a heavily built downtown area (Washington, D.C.) and surrounding residential areas. (Courtesy of NASA.)



Object Recognition

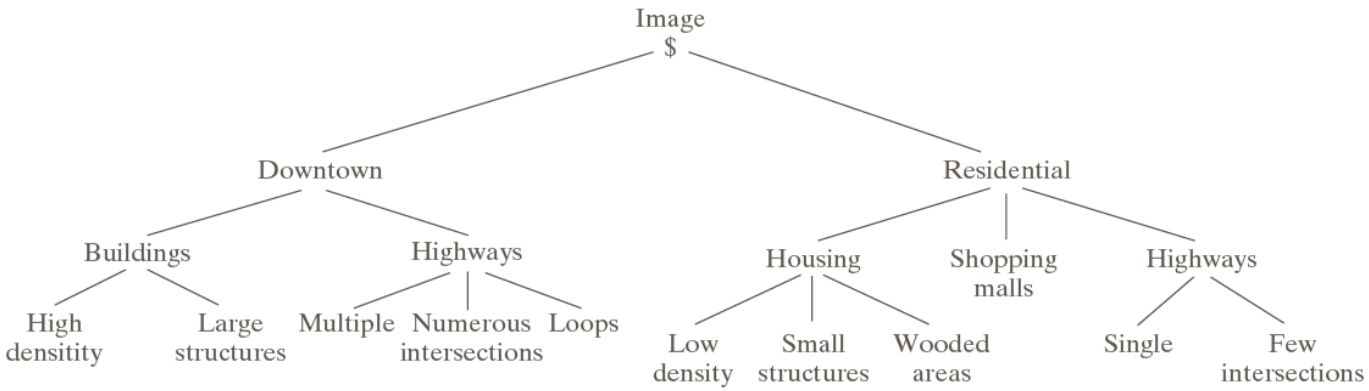
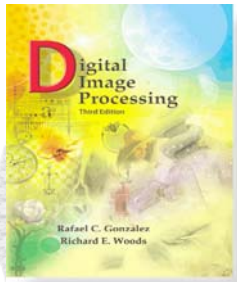


FIGURE 12.5 A tree description of the image in Fig. 12.4.



Minimum Distance Classifiers

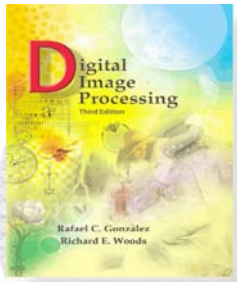
The prototype for each class (out of W classes) is the mean vector of that class

$$\underline{m}_j = \frac{1}{N_j} \sum_{x \in w_j} x_j \qquad j = 1, 2, \ldots, W$$

Compute "closeness" using Euclidian distance

$$D_j(\underline{x}) = \left\| \underline{x} - \underline{m}_j \right\| \qquad j = 1, 2, \ldots, W$$

Assign $\underline{x}$ to class j if $D_j(\underline{x})$ is the smallest distance



Minimum Distance Classifiers

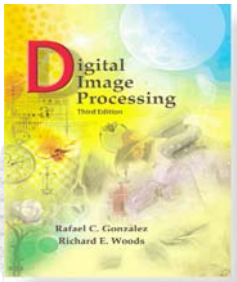
More mathematically the distance from a candidate pattern $\underline{x}$ to the pairs of mean vectors $\underline{m}_i$ and $\underline{m}_j$ can be written as

$$d_{ij}(\underline{x}) = d_i(\underline{x}) - d_j(\underline{x}) = \underline{x}^T(\underline{m}_i - \underline{m}_j) - \frac{1}{2}(\underline{m}_i - \underline{m}_j)^T(\underline{m}_i + \underline{m}_j)$$

This describes a plane marking the classification boundary between classes i and j, i.e.,

$$d_{ij}(\underline{x}) = 0$$

If $d_{ij}(\underline{x}) > 0$ then assign $\underline{x}$ to class j otherwise assign to class i



Object Recognition

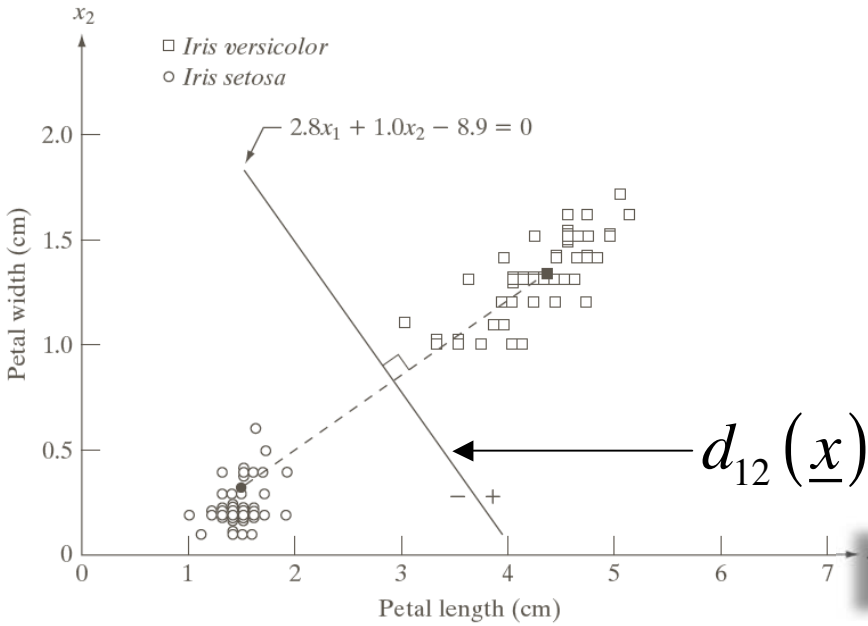
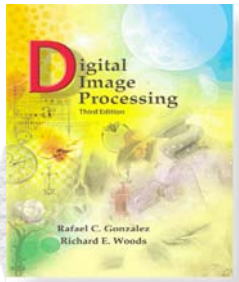


FIGURE 12.6 Decision boundary of minimum distance classifier for the classes of *Iris versicolor* and *Iris setosa*. The dark dot and square are the means.

$$d_{12}(\underline{x}) = 0$$

Decision classifier



Object Recognition

Check characters are read horizontally by a single vertical sensor which travels from left to right to generate a distinct waveform for each character

The characters and sensors are optimized to give distinct responses in a simple x-y grid.

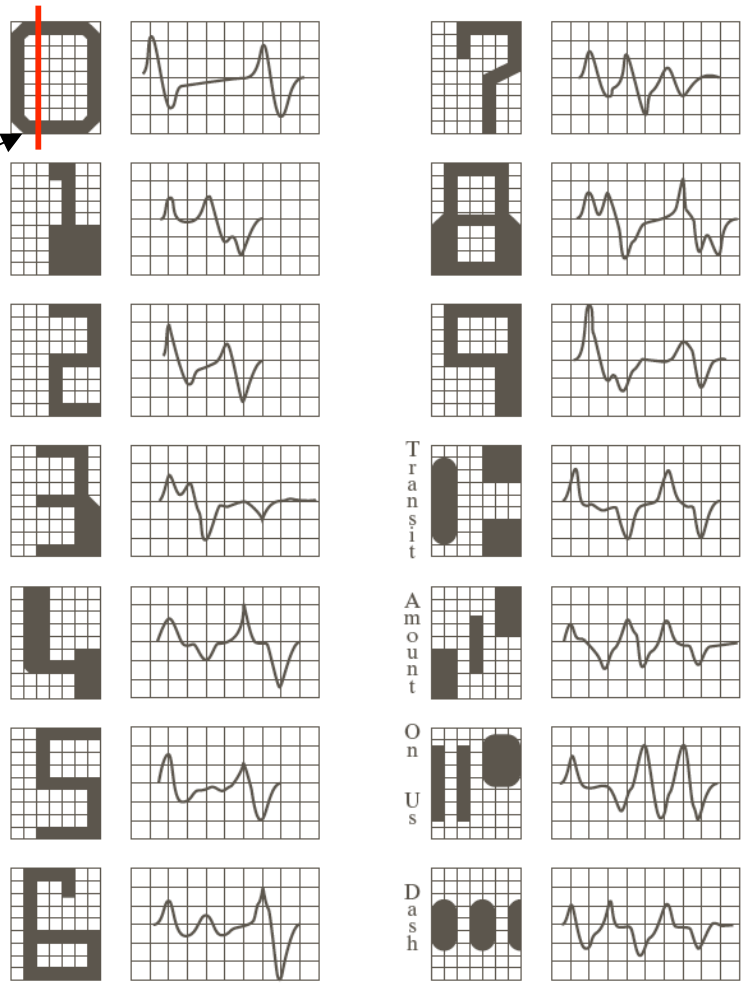
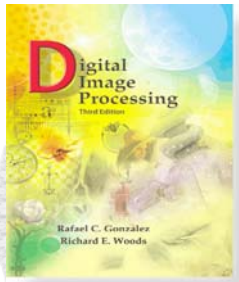


FIGURE 12.7 American Bankers Association E-13B font character set and corresponding waveforms.



Object Recognition

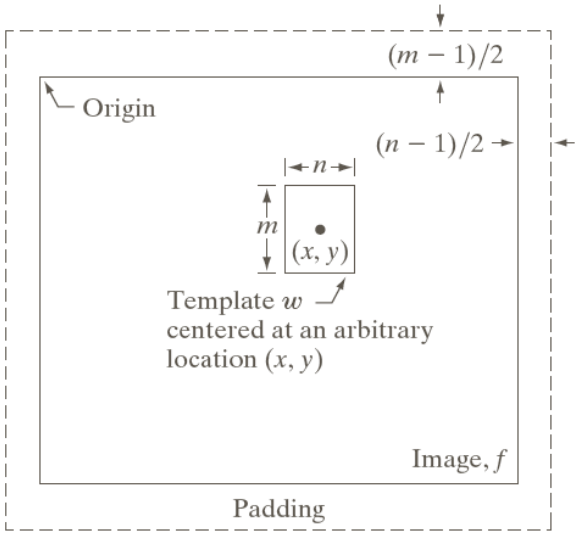
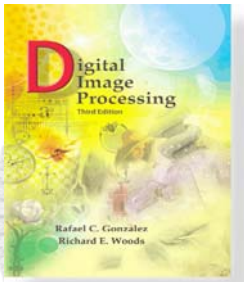
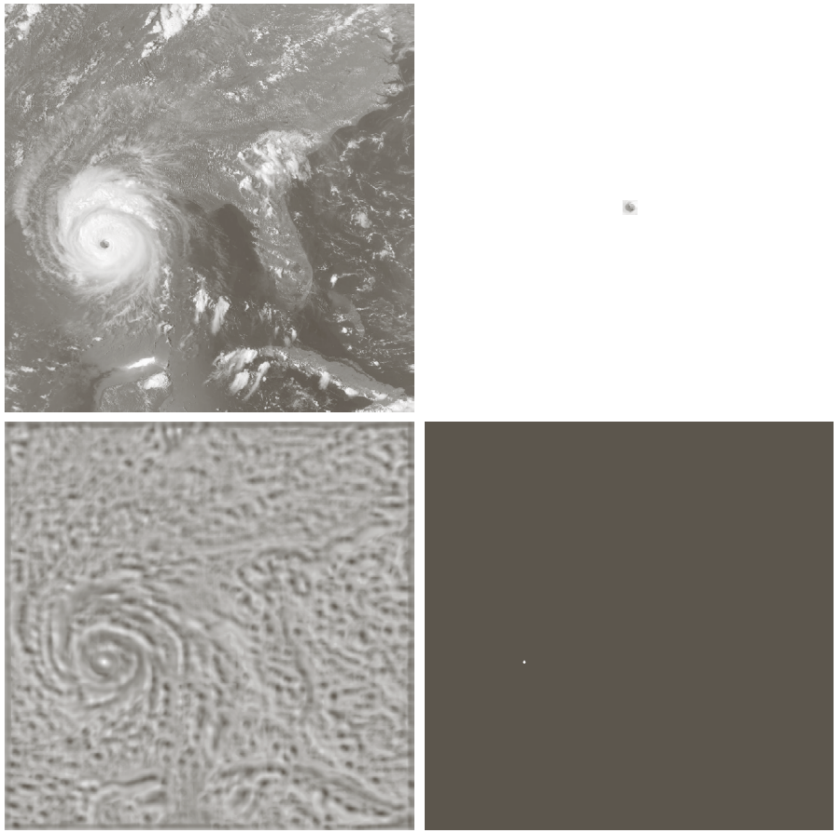


FIGURE 12.8
The mechanics of template matching.

Correlation can be used to find similar patterns

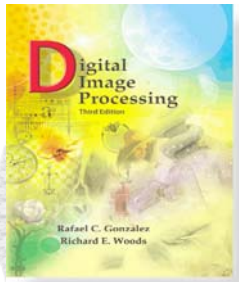


Object Recognition



a b
c d

FIGURE 12.9
(a) Satellite image of Hurricane Andrew, taken on August 24, 1992. (b) Template of the eye of the storm. (c) Correlation coefficient shown as an image (note the brightest point). (d) Location of the best match. This point is a single pixel, but its size was enlarged to make it easier to see. (Original image courtesy of NOAA.)



Statistical Pattern Recognition

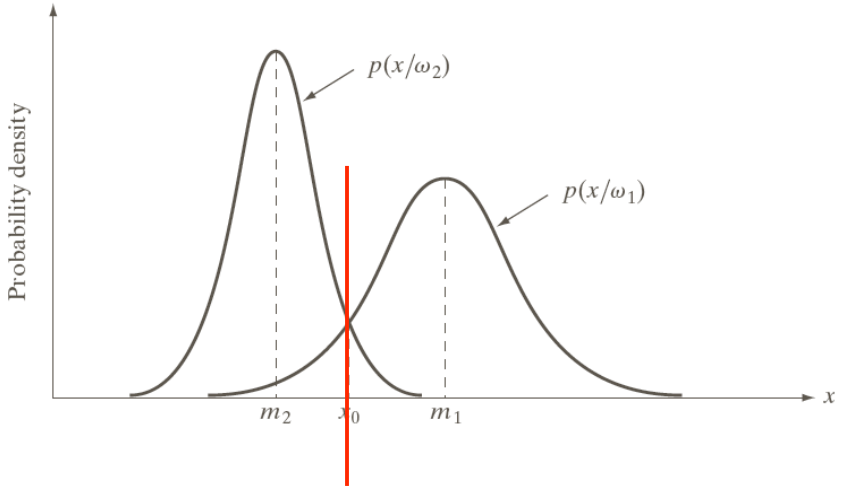
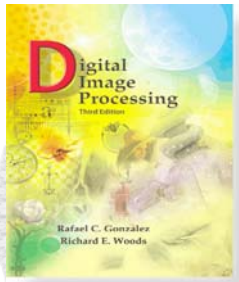


FIGURE 12.10 Probability density functions for two 1-D pattern classes. The point x_0 shown is the decision boundary if the two classes are equally likely to occur.

The threshold for classifying patterns is where the pdf's overlap

If the class probabilities are not equal we must use a Bayesian classifier and use the maximum to classify unknown patterns

$$d_j(\underline{x}) = p(\underline{x} \mid \omega_j) P(\omega_j) \qquad j = 1, 2, \ldots, W$$



Object Recognition

In more dimensions the classifier forms a plane (or hyperplane) for Gaussian pattern classes

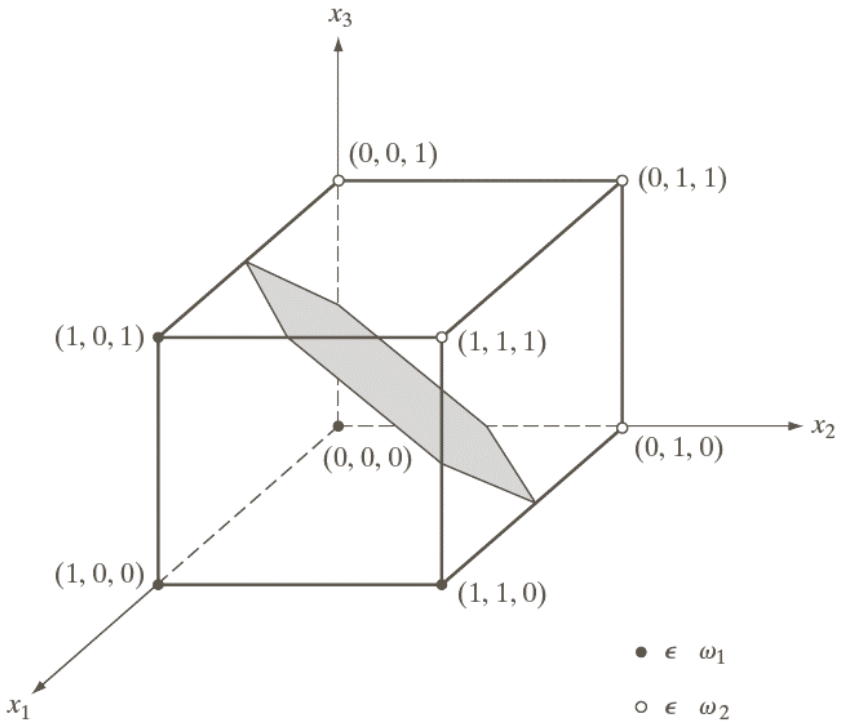
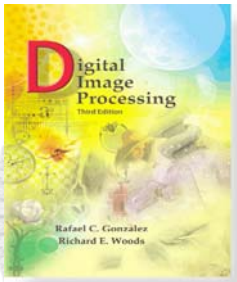


FIGURE 12.11 Two simple pattern classes and their Bayes decision boundary (shown shaded).

$$\underline{d}_j(x) = \ln P(\omega_j) - \frac{1}{2}\ln|\underline{C}_j| - \frac{1}{2}\left[(\underline{x} - \underline{m}_j)^T \underline{C}_j^{-1}(\underline{x} - \underline{m}_j)\right] \qquad j = 1,\ldots,W$$



Object Recognition

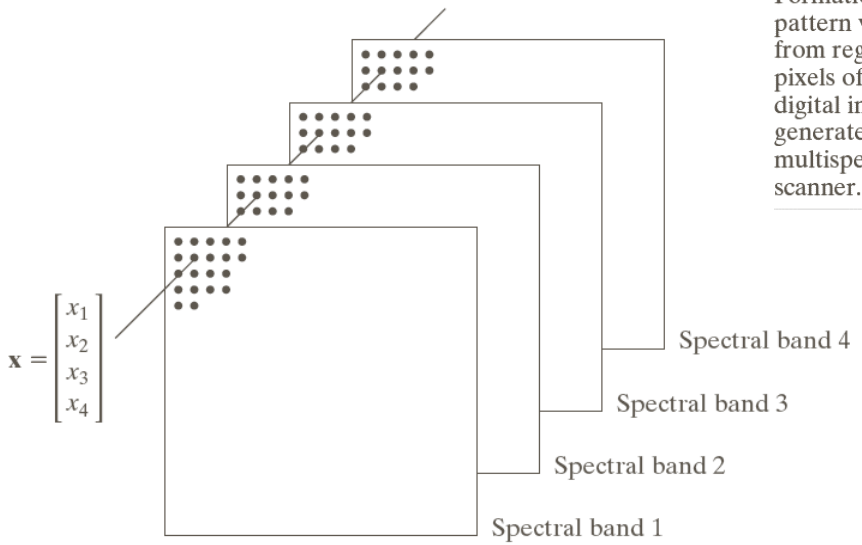
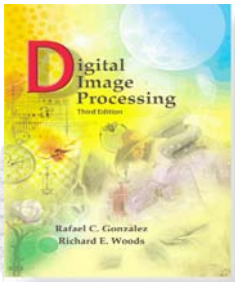
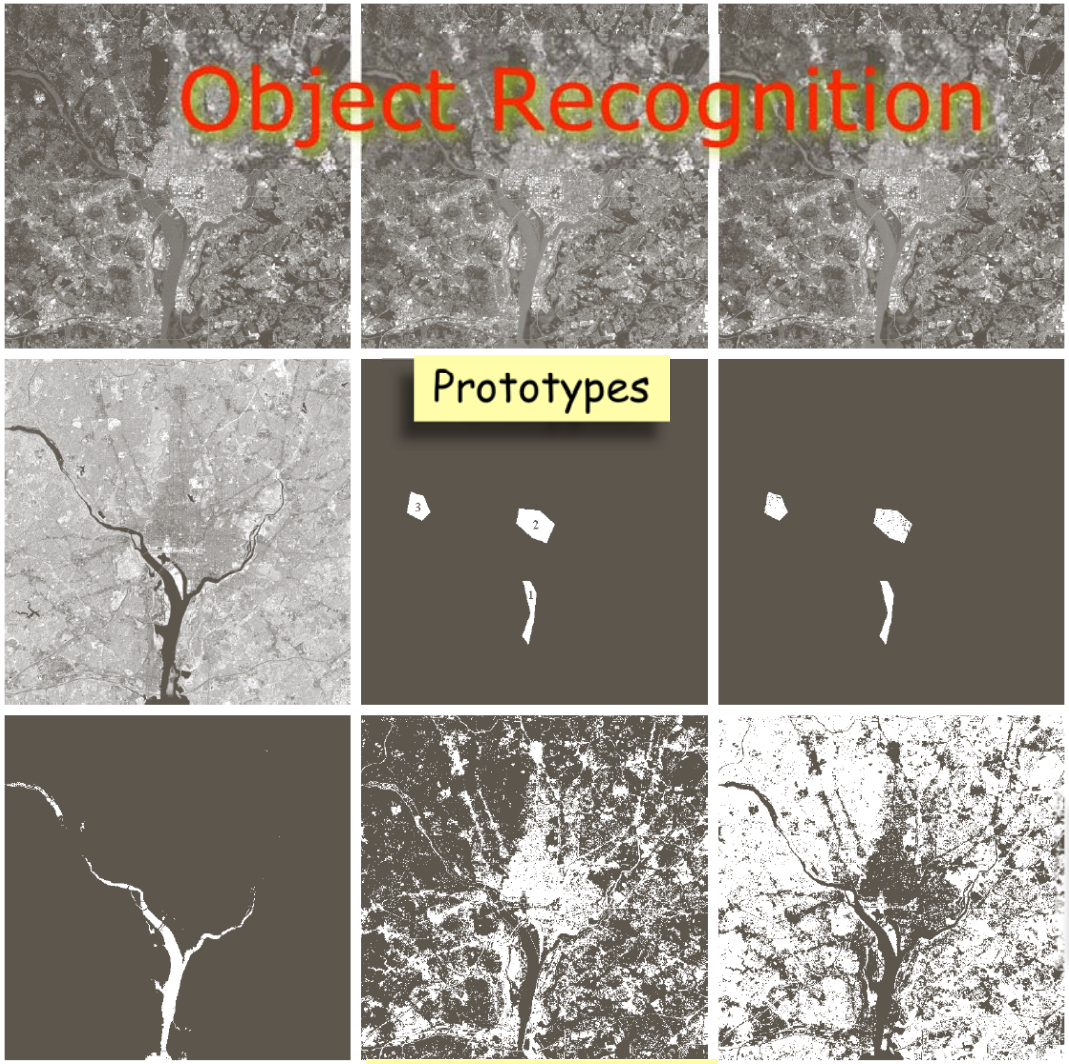


FIGURE 12.12 Formation of a pattern vector from registered pixels of four digital images generated by a multispectral scanner.

We can use the same registered multi-spectral image data as input to a pattern classifier



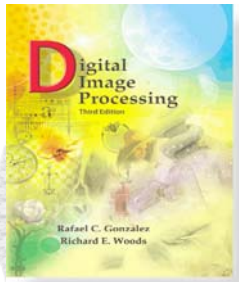
Object Recognition



Errors using half of the prototypes to train, the other to classify

a b c
d e f
g h i

FIGURE 12.13 Bayes classification of mu ages in the visible blue, visible green, visible red, and near infrared wavele sample regions of water (1), urban development (2), and vegetation (3). (f) Results of classification, the black dots denote points classified incorrectly. The other (white) points were classified correctly. (g) All image pixels classified as water (in white). (h) All image pixels classified as urban development (in white). (i) All image pixels classified as vegetations (in white).

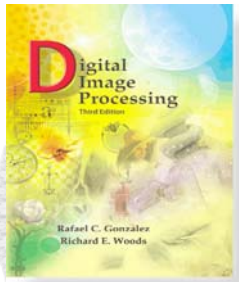


Object Recognition

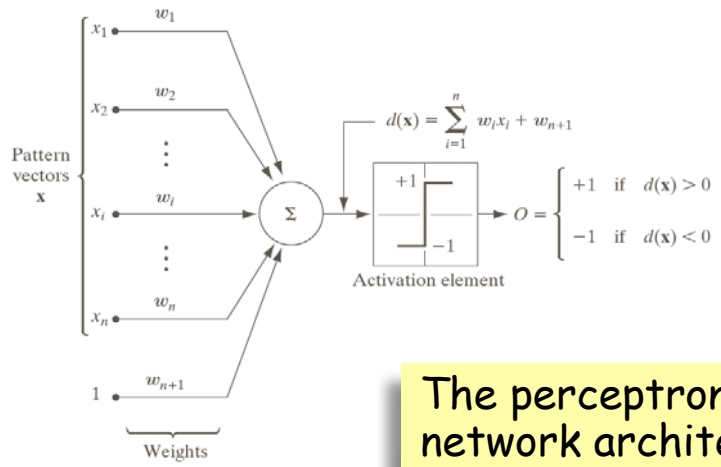
TABLE 12.1
Bayes classification of multispectral image data.

Training Patterns						Independent Patterns					
		Classified into Class						Classified into Class			
Class	No. of Samples	1	2	3	% Correct	Class	No. of Samples	1	2	3	% Correct
1	484	482	2	0	99.6	1	483	478	3	2	98.9
2	933	0	885	48	94.9	2	932	0	880	52	94.4
3	483	0	19	464	96.1	3	482	0	16	466	96.7

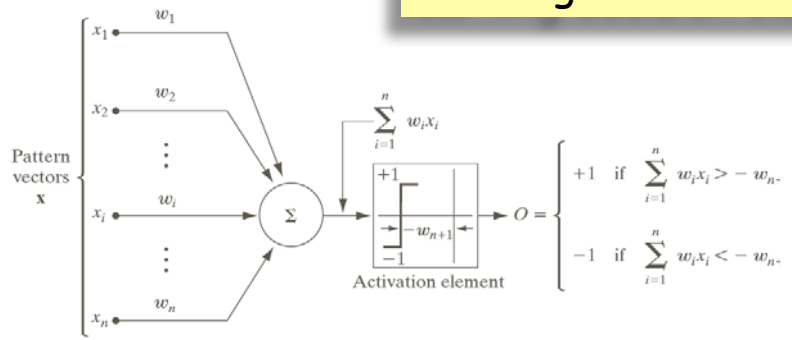
Bayes classification of pixels in the training regions 1,2,3



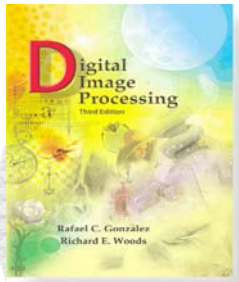
Object Recognition



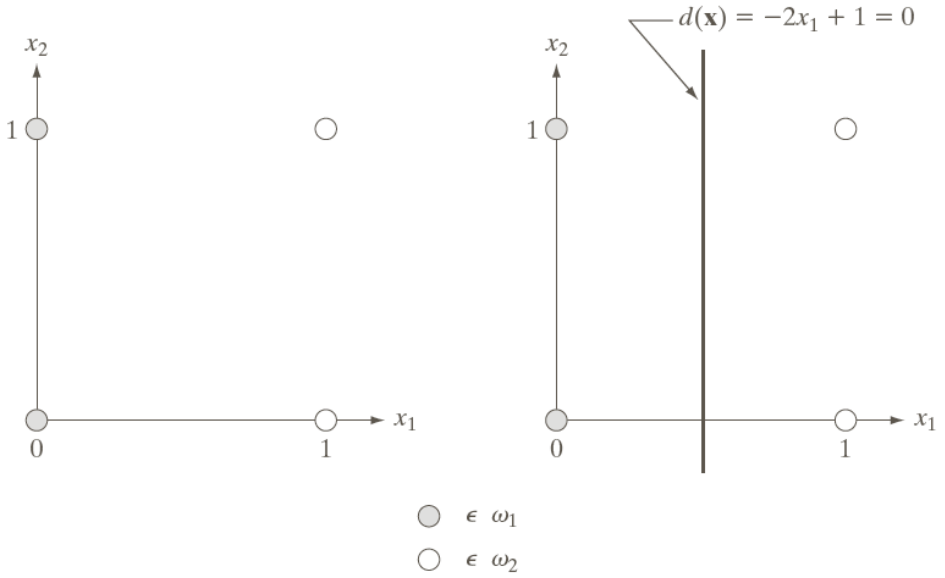
The perceptron is an early neural network architecture for learning decision classifiers



a
b **FIGURE 12.14** Two equivalent representations of the perceptron model for two pattern classes.

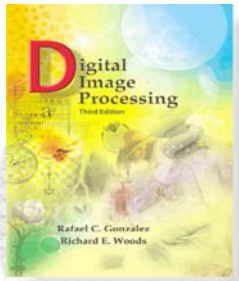


Object Recognition



a b

FIGURE 12.15 (a) Patterns belonging to two classes. (b) Decision boundary determined by training.



Object Recognition

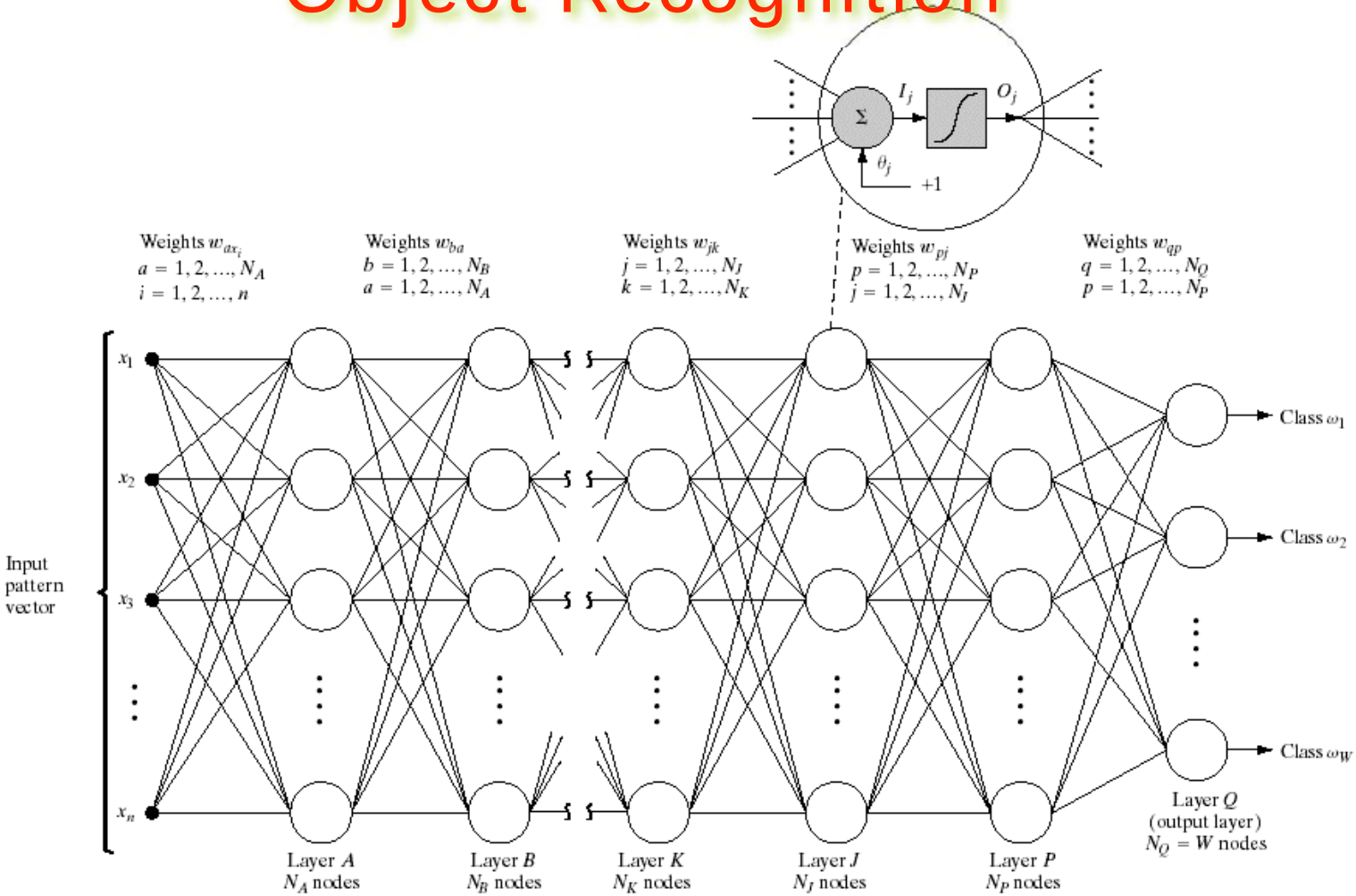
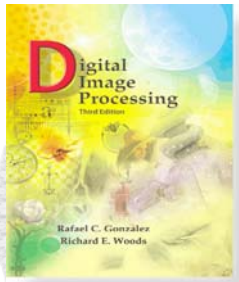


FIGURE 12.16 Multilayer feedforward neural network model. The blowup shows the basic structure of each neuron element throughout the network. The offset, θ_j, is treated as just another weight.



Object Recognition

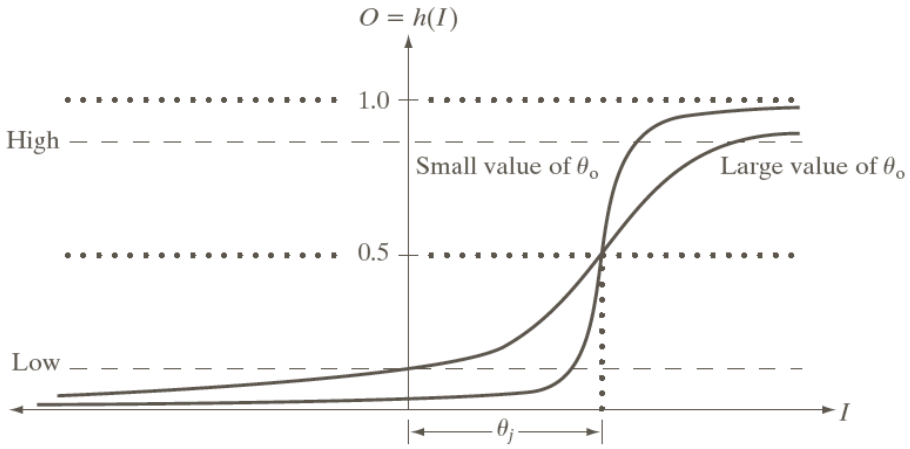
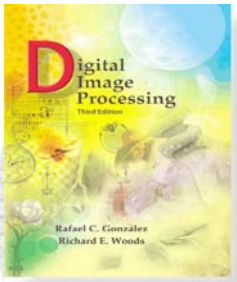
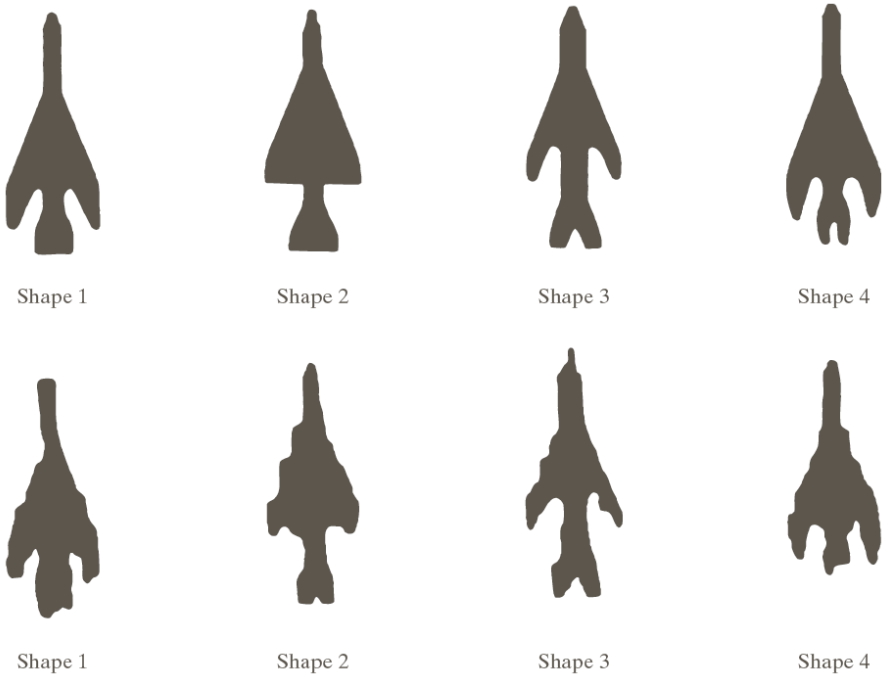


FIGURE 12.17 The sigmoidal activation function of Eq. (12.2-47).

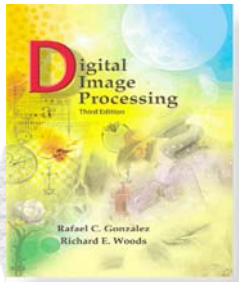


Object Recognition



a
b

FIGURE 12.18 (a) Reference shapes and (b) typical noisy shapes used in training the neural network of Fig. 12.19. (Courtesy of Dr. Lalit Gupta, ECE Department, Southern Illinois University.)



Object Recognition

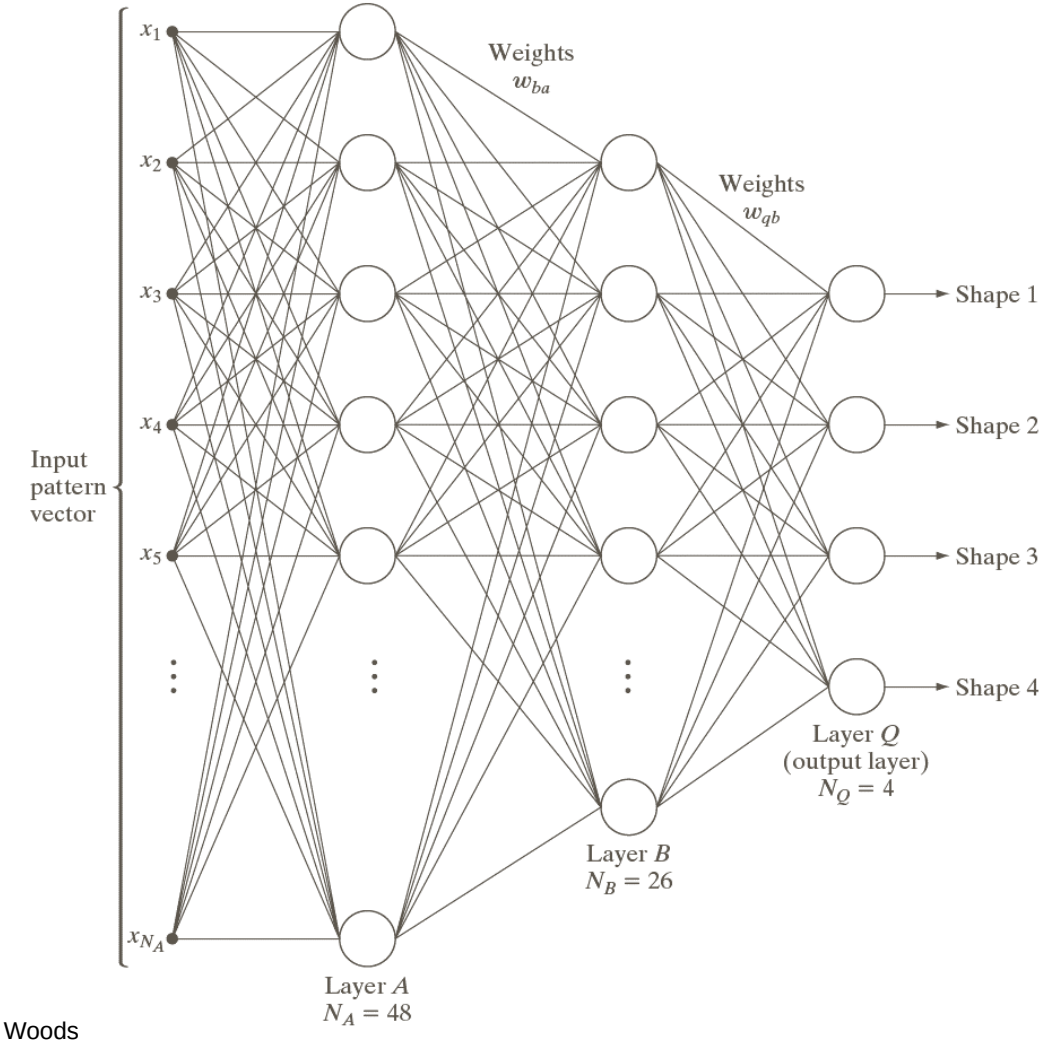
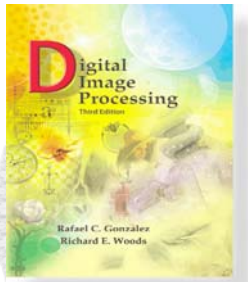


FIGURE 12.19 Three-layer neural network used to recognize the shapes in Fig. 12.18. (Courtesy of Dr. Lalit Gupta, ECE Department, Southern Illinois University.)



Object Recognition

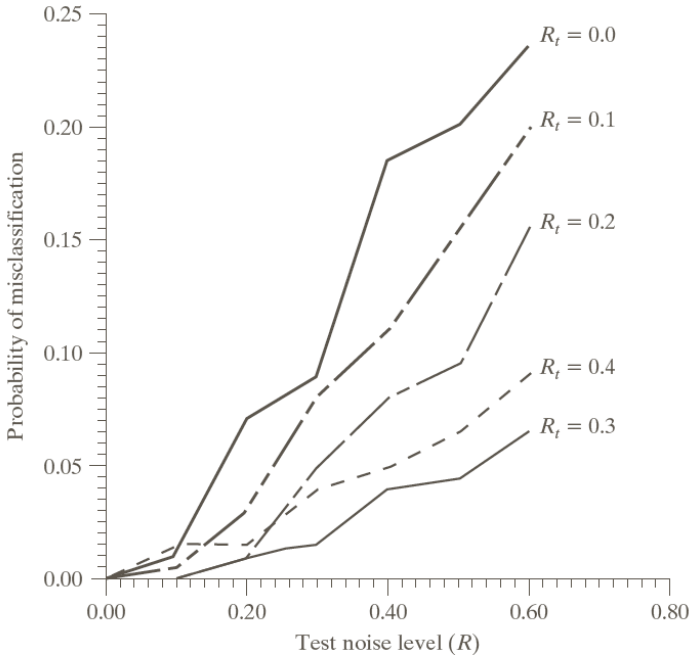
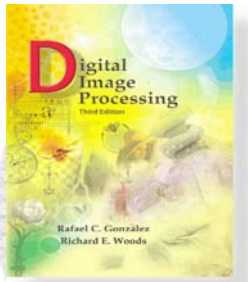


FIGURE 12.20 Performance of the neural network as a function of noise level. (Courtesy of Dr. Lalit Gupta, ECE Department, Southern Illinois University.)



Object Recognition

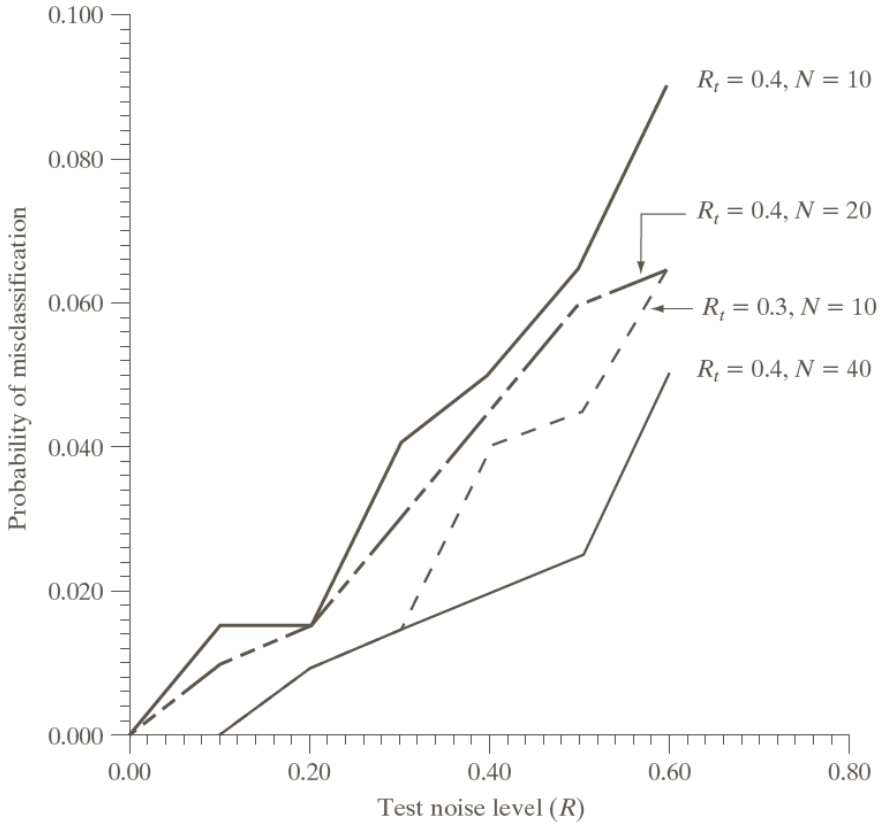
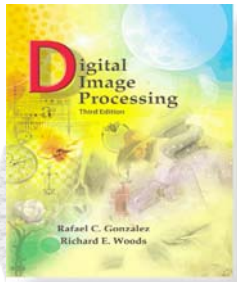
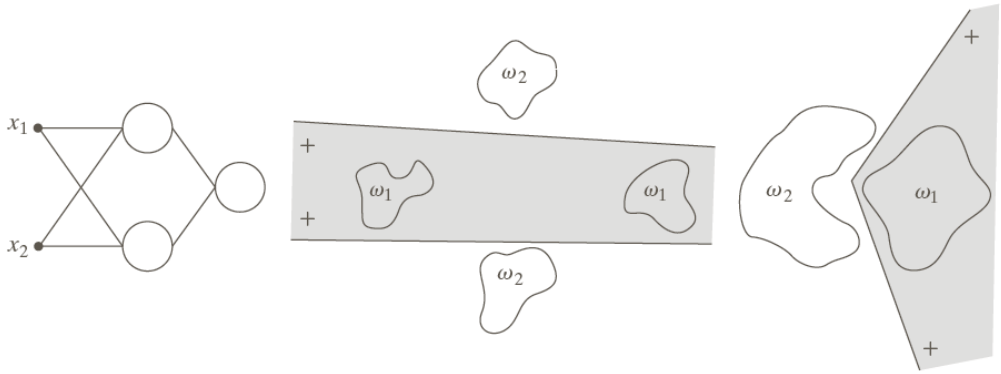


FIGURE 12.21 Improvement in performance for $R_t = 0.4$ by increasing the number of training patterns (the curve for $R_t = 0.3$ is shown for reference). (Courtesy of Dr. Lalit Gupta, ECE Department, Southern Illinois University.)



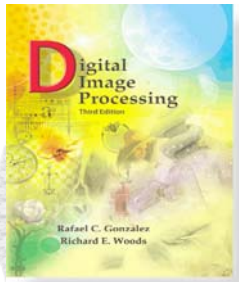
Object Recognition



a b c

FIGURE 12.22 (a) A two-input, two-layer, feedforward neural network. (b) and (c) Examples of decision boundaries that can be implemented with this network.

Neural networks can be used to implement (and learn) geometrically complex decision functions.



Object Recognition

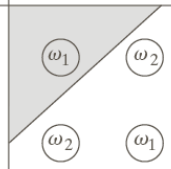
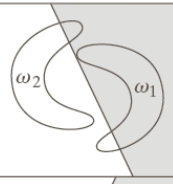
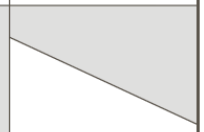
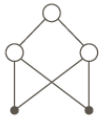
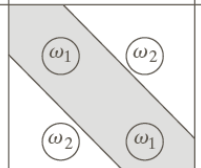
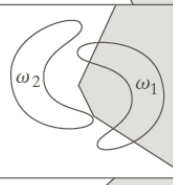
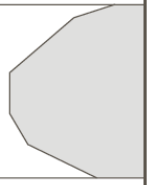
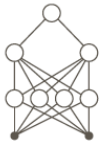
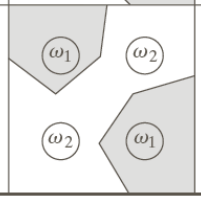
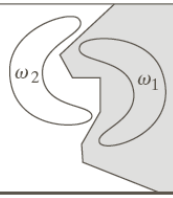
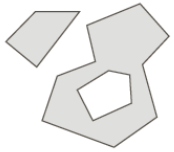
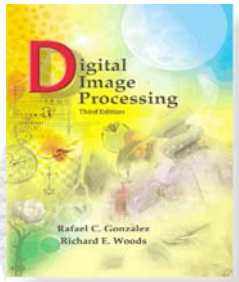
Network structure	Type of decision region	Solution to exclusive-OR problem	Classes with meshed regions	Most general decision surface shapes
Single layer 	Single hyperplane			
Two layers 	Open or closed convex regions			
Three layers 	Arbitrary (complexity limited by the number of nodes)			

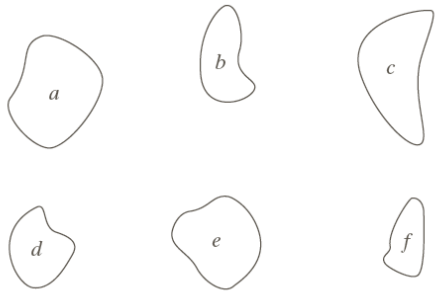
FIGURE 12.23 Types of decision regions that can be formed by single- and multilayer feed-forward networks with one and two layers of hidden units and two inputs. (Lippman.)

Multi-layer networks can be used to approximate (learn) more complex decision functions.



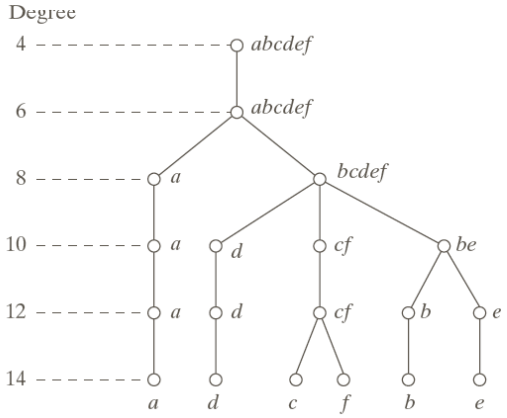
Shape Number Classification

Objects can be similar up to different orders of shape number. This can be used to classify objects.

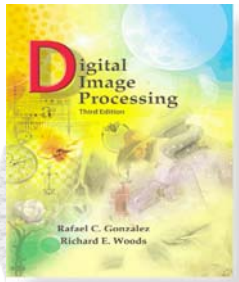


a
b c

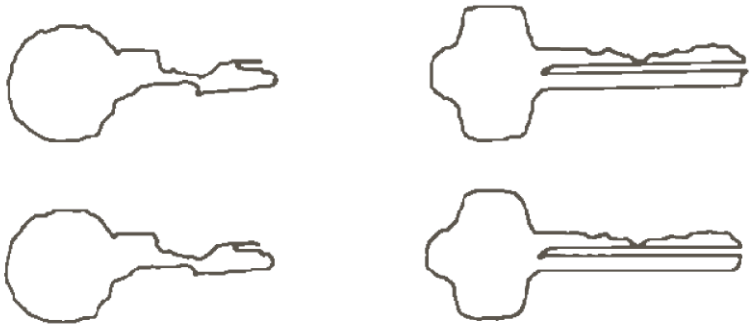
FIGURE 12.24
(a) Shapes.
(b) Hypothetical similarity tree.
(c) Similarity matrix.
(Bribiesca and Guzman.)



	a	b	c	d	e	f
a	∞	6	6	6	6	6
b		∞	8	8	10	8
c			∞	8	8	12
d				∞	8	8
e					∞	8
f						∞



Shape Number Classification



a b
c d
e f
g

FIGURE 12.25 (a) and (b) Sample boundaries of two different object classes; (c) and (d) their corresponding polygonal approximations; (e)–(g) tabulations of R. (Sze and Yang.)

R	1.a	1.b	1.c	1.d	1.e	1.f
1.a	∞					
1.b	16.0	∞				
1.c	9.6	26.3	∞			
1.d	5.1	8.1	10.3	∞		
1.e	4.7	7.2	10.3	14.2	∞	
1.f	4.7	7.2	10.3	8.4	23.7	∞

R	2.a	2.b	2.c	2.d	2.e	2.f
2.a	∞					
2.b	33.5	∞				
2.c	4.8	5.8	∞			
2.d	3.6	4.2	19.3	∞		
2.e	2.8	3.3	9.2	18.3	∞	
2.f	2.6	3.0	7.7	13.5	27.0	∞

R	1.a	1.b	1.c	1.d	1.e	1.f
2.a	1.24	1.50	1.32	1.47	1.55	1.48
2.b	1.18	1.43	1.32	1.47	1.55	1.48
2.c	1.02	1.18	1.19	1.32	1.39	1.48
2.d	1.02	1.18	1.19	1.32	1.29	1.40
2.e	0.93	1.07	1.08	1.19	1.24	1.25
2.f	0.89	1.02	1.02	1.24	1.22	1.18