Professor Brendan Morris, SEB 3216, brendan.morris@unlv.edu

ECG782: Multidimensional Digital Signal Processing

Spring 2014
TTh 14:30-15:45 CBC C313

Lecture 06
Image Structures
13/02/06

http://www.ee.unlv.edu/~b1morris/ecg782/

Outline

- Wavelets
- Eigen Decomposition
- Singular Value Decomposition
- Principle Component Analysis
- Stochastic Images

Wavelet Transform

- Decompose signals as linear combination of another set of basis functions (not sinusoid)
  - Can be more complex basis
- Mother wavelets

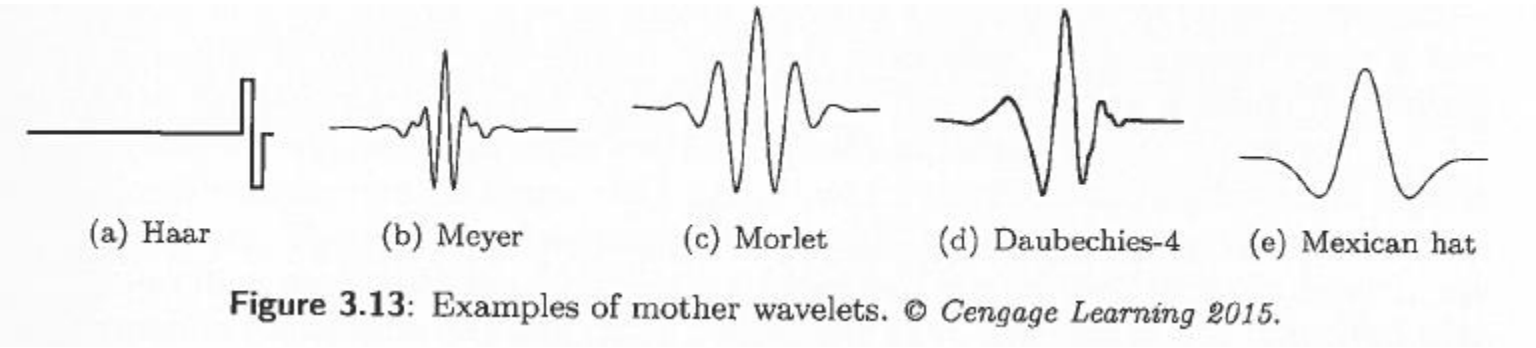


Figure 3.13: Examples of mother wavelets. © *Cengage Learning 2015.*

- Multiscale analysis
  - Provide localization in space
  - Search for particular "pattern" a different scales
- Wavelets are better designed for digital images
  - Less coefficients required than for sinusoidal
    - Think about how many coefficients are required for a single on pixel (delta)

1D Continuous Wavelet Transform

- $c(s,\tau) = \int_R f(t)\Psi^*_{s,\tau}(t)dt$
  - $s \in R^+ - \{0\}$ – indicates scale
  - $\tau \in R$ – indicates a time shift
- Wavelets at scale and shift generated from a "mother" wavelet
  - $\Psi_{s,\tau}(t) = \frac{1}{\sqrt{s}}\Psi\left(\frac{t-\tau}{s}\right)$
- Wavelet functions must have two properties
  - Admissibility – must have bandpass spectrum
    - Use oscillatory functions
  - Regularity – must have smoothness and concentration in time/frequency domains
    - Fast decrease with decreasing scale

Haar Wavelet

- "Mother" function (basis)
  - $\Psi_{ji}(x) = 2^{\frac{j}{2}}\Psi(2^j x - i)$
  - $\Psi(x) = \begin{cases} 1 & 0 \le x < \frac{1}{2} \\ -1 & \frac{1}{2} \le x < 1 \\ 0 & else \end{cases}$
- Scaling ("Father") function (multi-resolution/scale)
  - $\Phi_{ji}(x) = 2^{\frac{j}{2}}\Phi(2^j x - i)$
  - $\Phi(x) = \begin{cases} 1 & 0 \le x < 1 \\ 0 & else \end{cases}$
  - Scaled and translated box functions

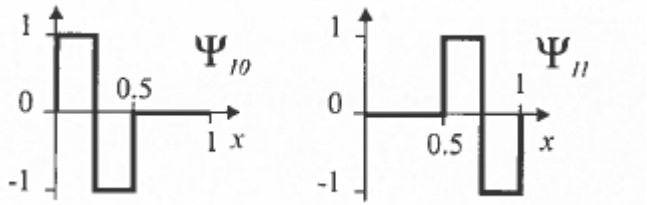


Figure 3.15: Haar wavelets Ψ_{11}, Ψ_{12}.
© *Cengage Learning 2015.*

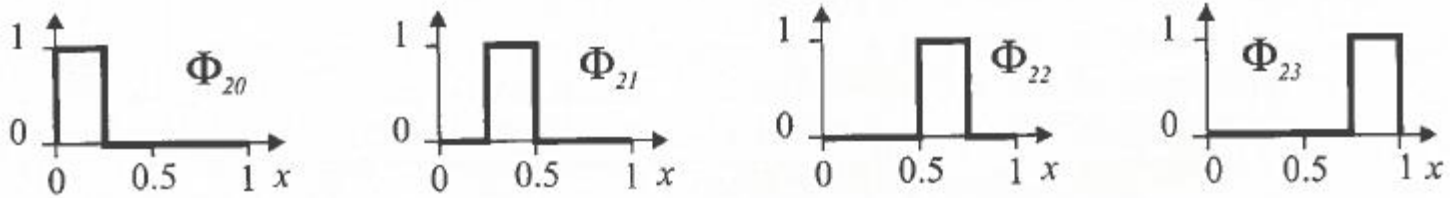


Figure 3.14: 'Box-like' scaling functions Φ. © *Cengage Learning 2015.*

Discrete Wavelet Transform

- Computationally efficient implementation
  - Herringbone algorithm exploits relationship between coefficients at various scales
- 1D case:
- At each level produce approximation coefficients and details
  - Approximation from lowpass
  - Detail from highpass
  - Use downsample to change scale
- Better approximation with more coefficients (more levels/scale)

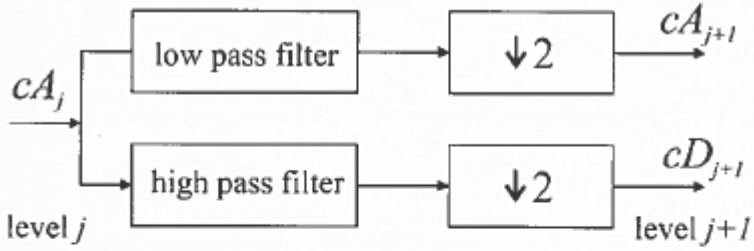


Figure 3.16: A single decomposition step of the 1D discrete wavelet transform consists of the convolution of coefficients from previous level j by a low/high pass filter and down-sampling by dyadic decimation. Approximation and detail coefficients at level $j+1$ are obtained. © *Cengage Learning 2015.*

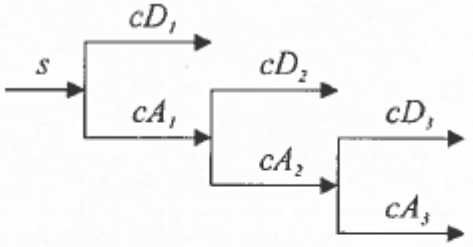


Figure 3.17: Example illustrating the structure of approximation and detail coefficients for levels up to a level $j = 3$. © *Cengage Learning 2015.*

2D Wavelet Transform

- Similar idea and extension from 1D to 2D
- 2D case
  - 4 decomposition types
    - Approximation
    - 3 detail – horizontal, vertical, and diagonal

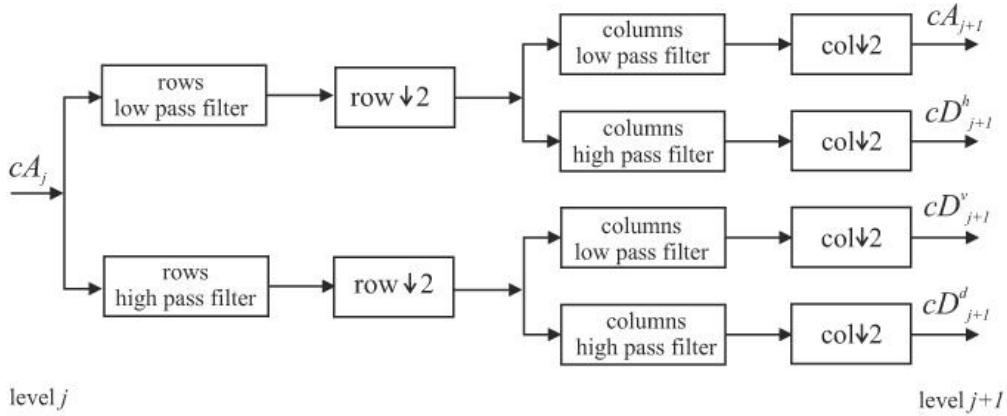


Figure 3.19: 2D discrete wavelet transform. A decomposition step. © *Cengage Learning 2015.*

2D Wavelet Transform Example

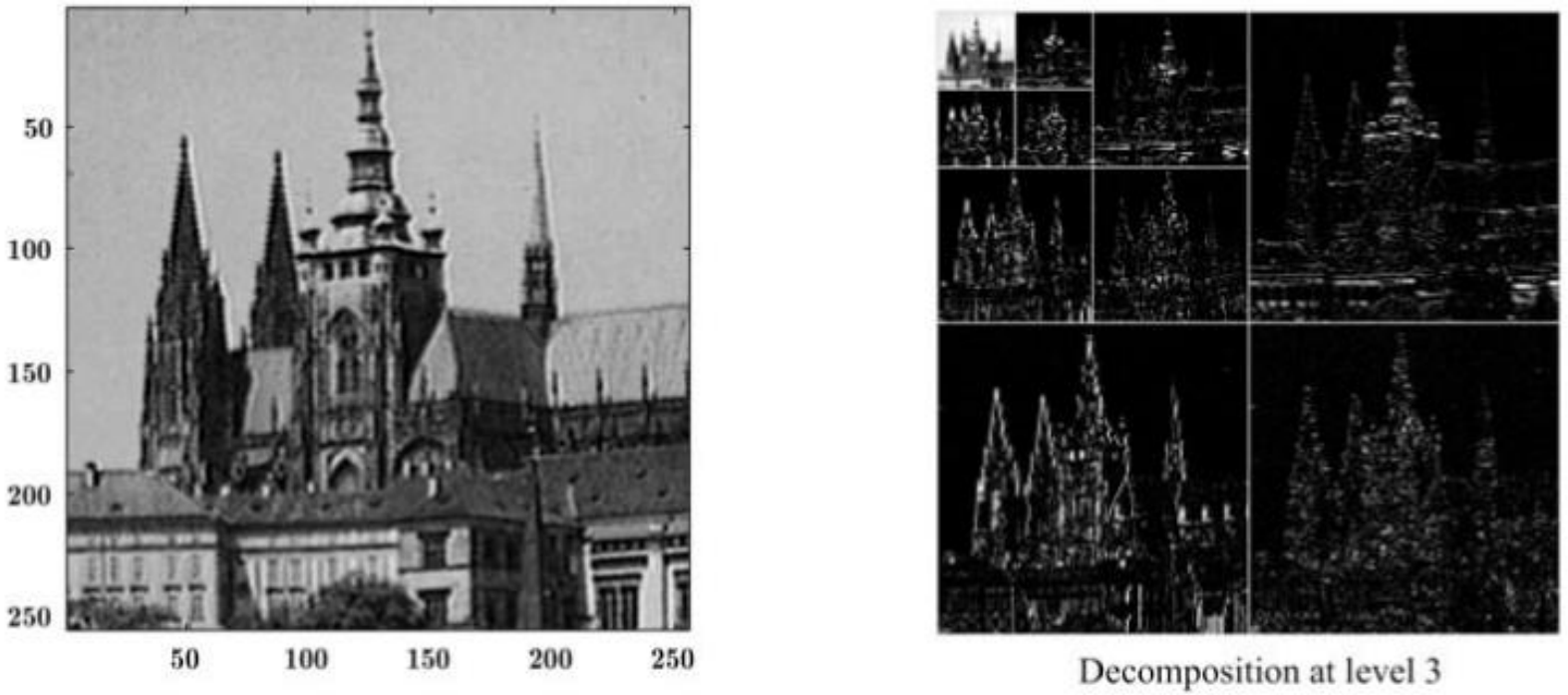


Figure 3.21: Decomposition to three levels by the 2D discrete Haar wavelet transform. Left is the original 256×256 gray-scale image, and right four quadrants. The undivided southwestern, southeastern and northeastern quadrants correspond to detailed coefficients of level 1 at resolution 128×128 in vertical, diagonal and horizontal directions, respectively. The northwestern quadrant displays the same structure for level 2 at resolution 64×64. The northwestern quadrant of level 2 shows the same structure at level 3 at resolution 32×32. The lighter intensity 32×32 image at top left corresponds to approximation coefficients at level 3.

2D Wavelet Transform Example

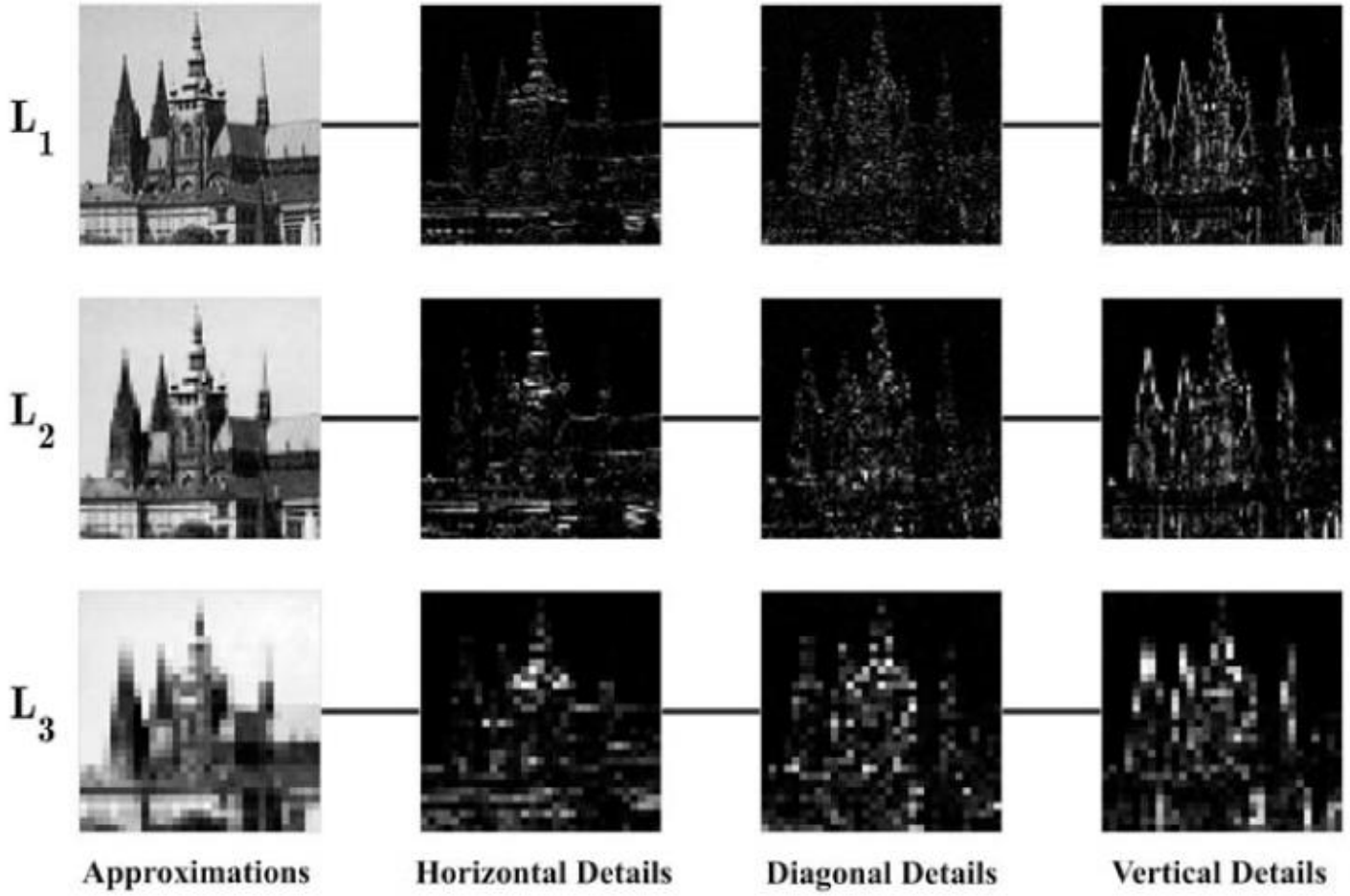
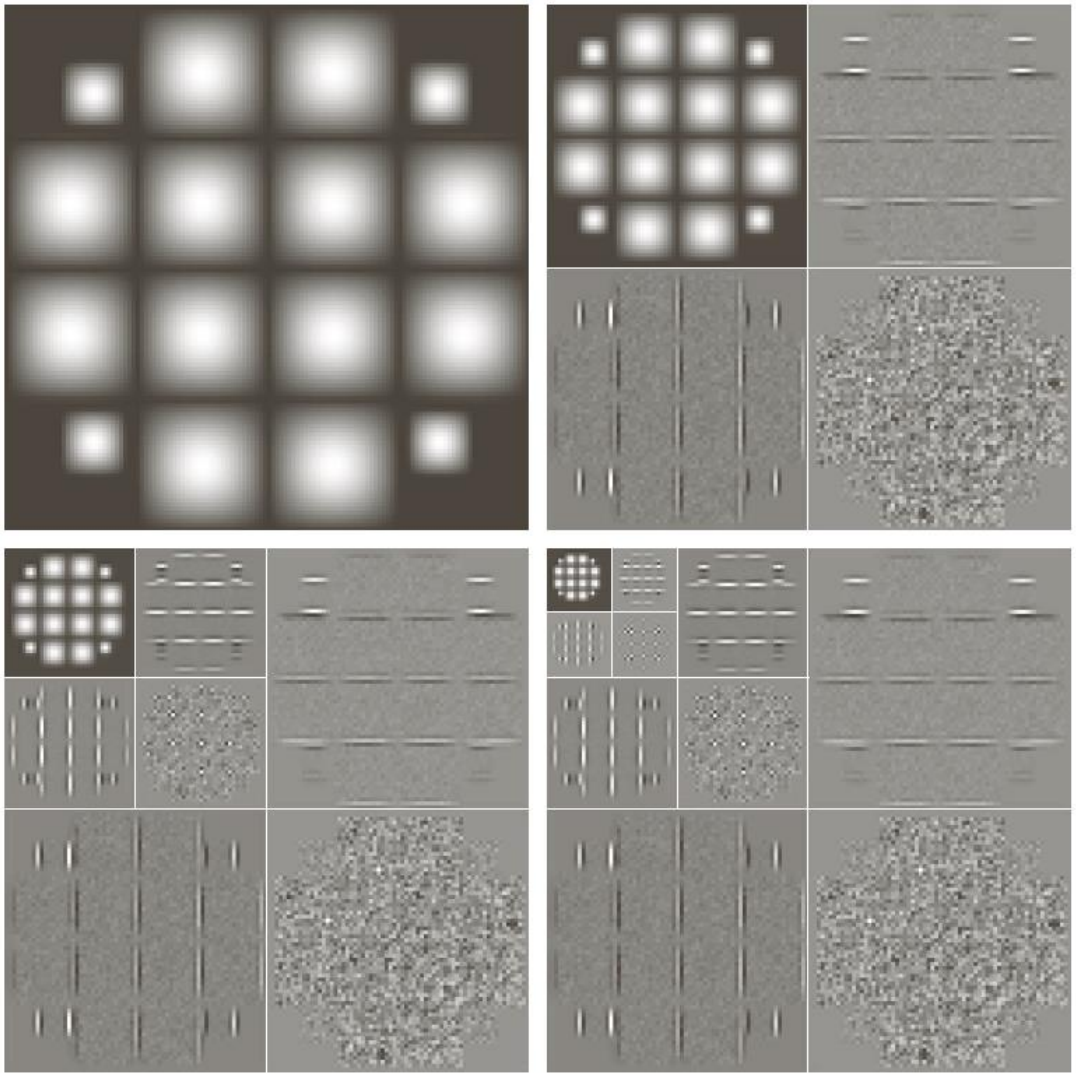


Figure 3.22: 2D wavelet decomposition; another view of the same data as Figure 3.21. *© Cengage Learning 2015.*

2D Wavelet Transform Example



a b
c d

FIGURE 7.25 Computing a 2-D three-scale FWT: (a) the original image; (b) a one-scale FWT; (c) a two-scale FWT; and (d) a three-scale FWT.

Eigen-Analysis

- Represent observations in a form that enhances mutual independence of contributory components
  - Fundamental of linear algebra
- Find a new "natural" basis
  - Orthogonal basis vectors
- Eigen solution
  - $Ax = \lambda x$
    - A – $n \times n$ square matrix
    - λ – eigenvalue (can be complex)
    - x – eigenvector
- There will be n eigenvalues
  - Can find as roots of characteristic polynomial
    - $\det(A - \lambda I)$
  - May have repeated eigenvalues
- The n eigenvectors is a compact representation of the space
  - E.g. (i, j, k) unit vectors in R^3

Eigen-Analysis Idea

- Imagine you have some data x
- You have a transformation matrix A that maps points
  - $Ax = b$

- The eigenvectors are the directions that are preserved under this mapping
- They provide a basis for describing the new transformed space

Singular Value Decomposition

- Generalization of eigen-analysis to operate on rectangular matrices (not just square)
- SVD relationship
  - $Av = \sigma u$ and $A^* u = \sigma v$
    - A – rectangular matrix
    - u – right singular vector
    - v – left singular vector
- This is a powerful matrix factorization tool
  - $A = UDV^*$
    - A - $m \times n$ matrix ($m \geq n$)
    - U- $m \times m$ orthonormal columns (right singular vectors)
    - V - $n \times n$ orthonormal rows (left singular vectors)
    - D – non-negative diagonal matrix of singular values
  - Solves linear equations in the least-squares sense
    - Used in signal processing, statistics, etc.

SVD In Practice

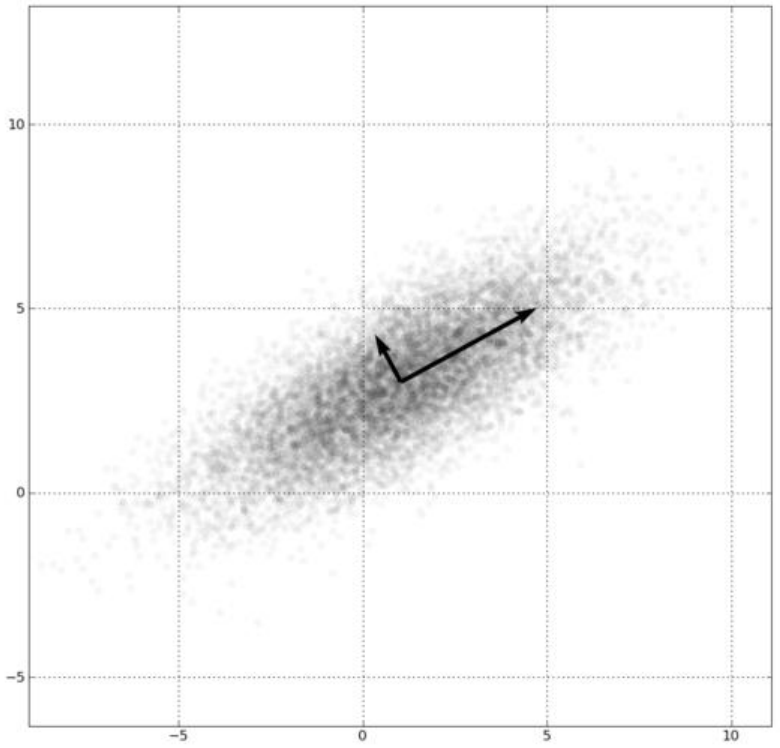
- SVD is an input output relationship
- V are the orthonormal input basis vectors
- U are the orthonormal output basis vectors
- D are the scalar gains to go from input to output
  - This diagonal matrix is usually ordered in non-increasing order

- The SVD is often used even for eigen-analysis because it is numerically stable
  - Given noisy measurements eigen-analysis may not have a solution
  - Least-square solution

Principle Component Analysis

- PCA is quite popular and known by other names
  - Karhunen-Loeve and Hotelling transforms
- This is a dimensionality reduction technique
- Multidimensional data is projected onto a lower dimensional space for analysis or visualization
  - E.g. $R^n \rightarrow R^2$ for print
- Determine new basis for data to best account for variance
  - Basis vectors are in directions of greatest variance

Wikipedia



- Change from Euclidean coordinates to data centric coordinates

PCA Formulation

- Given N observations of M dimensions, want to represent each datapoint by only L
  - $N \gg M$
  - $1 \leq L < M$
- Arrange data into columns of raw observation matrix R
  - R is large $M \times N$ matrix
- Create normalized matrix X
  - $X = R - u1^T$
    - $u(m) = \frac{1}{N}\sum_{n=1}^{N} R(m,n)$
    - 1 – vector of ones
  - Normalization centers at mean of data
- Compute data covariance matrix
  - $C_X = \frac{1}{N} X X^T$
- Diagonalize covariance
  - $C_Y = A^T C_X A$
    - Use `svd.m`
- C_y is a diagonal matrix with same eigen values as C_x
- Rows of A are eigenvectors
  - $A^T A = A A^T = I$
- Map datapoints to new vectors
  - $y = A_L(x - u)$
  - A is like a rotation matrix
  - Choose to use only a certain L top eigen vectors

PCA Approximation Error

- Recover original data from transformed
  - $y = A(x - u)$
  - $A^T y = A^T A(x - u)$
  - $x = A^T y - u$
- Using only top L eigen vectors
  - $\hat{x} = A_L y - u$
  - y are L dimensional
- Reconstruction is no longer exact
  - Similar to Fourier series when you drop higher coefficients
- $x = \hat{x}$ when the eigenvectors that are dropped do not have informational content
  - Redundant directions
  - $\lambda_k = 0$
- Mean square error of approximation
  - $\epsilon^2 = \frac{1}{N}\sum_n (x_n - A^T y_n)^2$
- Without transformation
- $\epsilon^2 = \frac{1}{N}\sum_{n=1}^{N} |x_n|^2 - \sum_{i=1}^{L} b_i^T \left(\frac{1}{N}\sum_{n=1}^{N} x_n x_n^T\right) b_i$
- This is minimized by maximizing
  - $\sum_{i=1}^{L} b_i^T \left(\frac{1}{N}\sum_{n=1}^{N} x_n x_n^T\right) b_i$
  - $\sum_{i=1}^{L} b_i^T \operatorname{cov}(x) b_i$
- Based on eigenvalues
  - $\epsilon^2 = \sum_{j=1}^{M} \lambda_j - \sum_{j=1}^{L} \lambda_j = \sum_{j=L+1}^{M} \lambda_j$

Eigenfaces

- Application of PCA to face recognition
- Given a set of face images

Figure 3.23: 32 original images of a boy's face, each 321 × 261 pixels. © *Cengage Learning 2015.*

- Each face can be treated as a point in a high dimensional image space
  - Each face image is made into a vector by stacking rows (e.g. `col=I(:))` in Matlab)
  - Example $321 \times 261 = 83781$ pixels
    - R^{83781} dimensional image space
    - Fewer images (datapoints) than dimensions

Eigenface PCA

- Collect data matrix
  - $R = [x_1, x_2, \ldots, x_k]_{n \times k}$
- Remove mean to center
  - $X = R - u1^T$
- Compute data covariance
  - $C_x = \frac{1}{k} X X^T$
    - $n \times n$ matrix
- Do eigen decomposition
  - $C_x e = \lambda e \rightarrow X X^T e = \lambda e$
    - Use SVD
- Each eigenvector is known as an eigenface
  - Basis of faces (can compose a face as mixture of basis)

- Notice this procedure requires the decomposition of $n \times n$ matrix
  - This can be computationally expensive for images
- Trick to save memory
  - Decompose a $k \times k$ matrix
  - $X^T X f = \lambda f$
  - $X X^T (Xf) = \lambda (Xf)$
- Let $e = Xf$
  - Be sure to normalize $||e|| = 1$

Figure 3.24: Reconstruction of the image from four basis vectors $\mathbf{b}_i$, $i = 1, \ldots, 4$ which can be displayed as images. The linear combination was computed as $q_1\mathbf{b}_1 + q_2\mathbf{b}_2 + q_3\mathbf{b}_3 + q_4\mathbf{b}_4 = 0.078\,\mathbf{b}_1 + 0.062\,\mathbf{b}_2 - 0.182\,\mathbf{b}_3 + 0.179\,\mathbf{b}_4$. © *Cengage Learning 2015.*

Stochastic Images

- Images can be considered statistical in nature
  - The same image of a scene will be slightly different each time
- Think of an image as a stochastic process ϕ
  - $\phi(x, y)$ – is the random intensity variable for pixel at (x, y)
- An image $f(x, y)$ is a specific realization or the random process
  - It is a real deterministic function

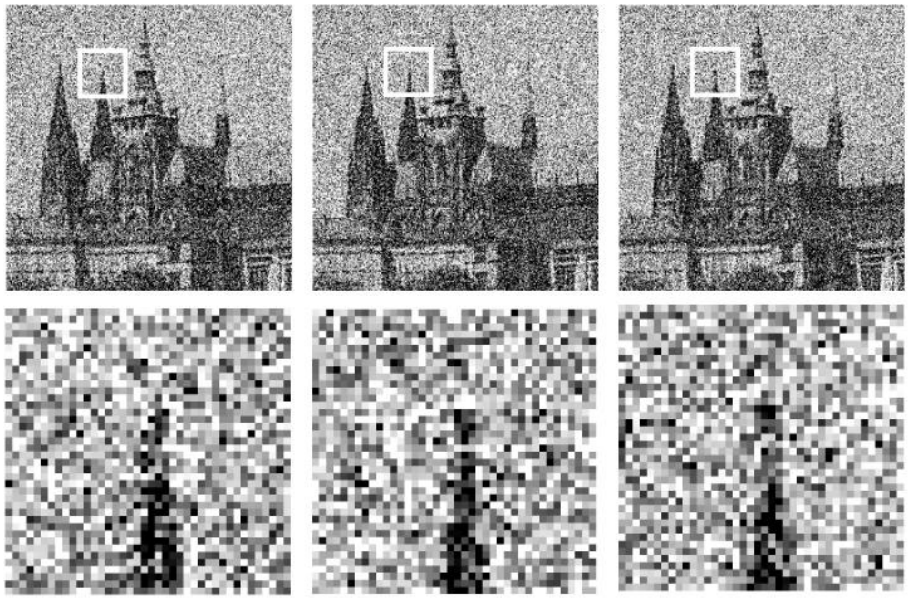


Figure 3.28: Above: three 256 × 256 images are shown as realizations of a stochastic process ϕ. The crop windows are marked by white squares in the images. Below: the content of these windows is enlarged. Notice that pixels really differ in the same locations in three realizations.

Image Processes

- The joint distribution of all pixels is large and complicated
- Often will simplify to distributions on single pixels $p_1(z; x, y)$
- Characterize distribution by simple statistics
  - Mean (first order expectation)
    - $\mu_\phi(x, y) = E[\phi(x, y)] = \int_{-\infty}^{\infty} z p_1(z; x, y) dz$
  - Cross correlation
    - $R_{\phi\gamma}(x_1, y_1, x_2, y_2) = E[\phi(x_1, y_1)\gamma(x_2, y_2)]$
    - Autocorrelation → $R_{\phi\phi}$
    - Uncorrelated processes have zero correlation

Image Data Representation Levels

- Computer vision (machine understanding) aims to make sense from visual image data
- Must transform raw image data into a more semantically meaningful model

- Relational models
  - High level of abstraction
  - Use of a priori knowledge
- Geometric representations
  - Knowledge about 2D and 3D shapes
  - Quantification of shape
- Segmented images
  - Semantically meaningful grouping (e.g. object)
  - Some domain knowledge is usually required
- Iconic images
  - “Raw” data – pixel image brightness

Matrix Data

- Most common data structure
  - Output of capture devices (cameras)
  - Each entry represents brightness
- Explicit spatial relationships through matrix coordinates
  - Can define neighborhood relationships
- Common matrix images
  - Binary image – only two intensity values $\{0, 1\}$
    - Threshold image
  - Multispectral image – several matrices together with each matrix the response to a particular spectral band
    - RGB image - $M \times N \times 3$ matrix
  - Hierarchical image structures – images at different resolutions
    - Image pyramid – useful for analysis at scale

Co-Occurrence Matrix

- Estimate of the probability of two pixels appearing in a spatial relationship
- Define matrix $C_r(z, y)$
  - Examine pixel $f(i_1, j_1) = z$ and pixel $f(i_2, j_2) = y$
  - Count the number of pixels that exhibit relationship r
    - r – can be a neighborhood relation
- This is useful for describing texture

Algorithm 4.1: Co-occurrence matrix $C_r(z, y)$ for the relation r

1. Set $C_r(z, y) = 0$ for all z, $y \in [0, L]$, where L is the maximum brightness.
2. For all pixels (i_1, j_1) in the image, determine all (i_2, j_2) which have the relation r with the pixel (i_1, j_1), and perform

$$C_r\left[f(i_1, j_1), f(i_2, j_2)\right] = C_r\left[f(i_1, j_1), f(i_2, j_2)\right] + 1\,.$$

Integral Image

- Cumulative sum image
  - $ii(i,j) = \sum_{k \le i, l \le j} f(k,l)$
  - Each entry is the sum of pixels to the above left
- Used for rapid calculation of simple rectangle feature at various scales

Algorithm 4.2: Integral image construction

1. Let $s(i,j)$ denote a cumulative row sum, and set $s(i,-1) = 0$.
2. Let $ii(i,j)$ be an integral image, and set $ii(-1,j) = 0$.
3. Make a single row-by-row pass through the image. For each pixel (i,j) calculate the cumulative row sums $s(i,j)$ and the integral image value $ii(i,j)$ using

$$s(i,j) = s(i,j-1) + f(i,j)\,, \tag{4.2}$$
$$ii(i,j) = ii(i-1,j) + s(i,j)\,. \tag{4.3}$$

4. After completing a single pass through the image, the integral image ii is constructed.

Integral Image Utility

- Any rectangular sum can be computed from the integral image in only 4 array references
  - With basic matrix, need to reference each pixel in rectangular region
- Only overhead is the single pass through the image to compute the integral image

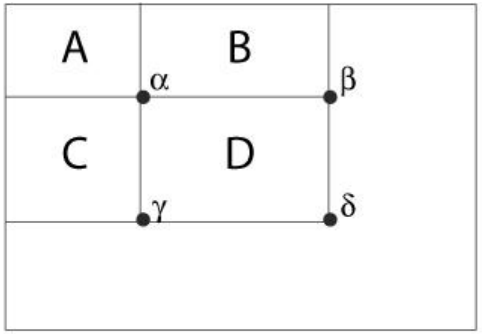


Figure 4.1: Calculation of rectangle features from an integral image. The sum of pixels within rectangle D can be obtained using four array references. $D_{sum} = ii(\delta) + ii(\alpha) - (ii(\beta) + ii(\gamma))$, where $ii(\alpha)$ is the value of the integral image at point α (and similarly for β, γ, δ). © *Cengage Learning 2015.*

Haar-Like Features

- Sum of rectangular regions
  - Add white and subtract black regions

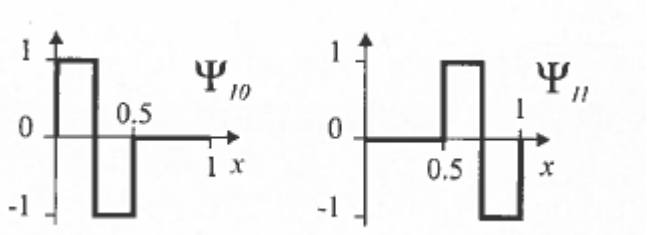


Figure 3.15: Haar wavelets Ψ_{11}, Ψ_{12}. © *Cengage Learning 2015.*

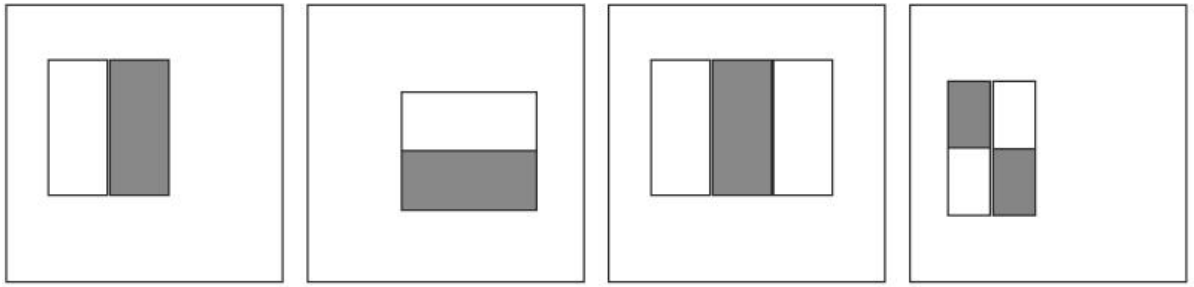


Figure 4.2: Rectangle-based features may be calculated from an integral image by subtraction of the sum of the shaded rectangle(s) from the non-shaded rectangle(s). The figure shows (a,b) two-rectangle, (c) three-rectangle, and (d) four-rectangle features. Sizes of the individual rectangles can be varied to yield different features as well as features at different scales. Contributions from the regions may be normalized to account for possibly unequal region sizes. © *Cengage Learning 2015.*

- These look for intensity patterns
  - Used in face recognition [Viola and Jones]

Chains

- Sequence to describe object borders
- Can use language theory models for pattern recognition
  - E.g. letters and grammar rules

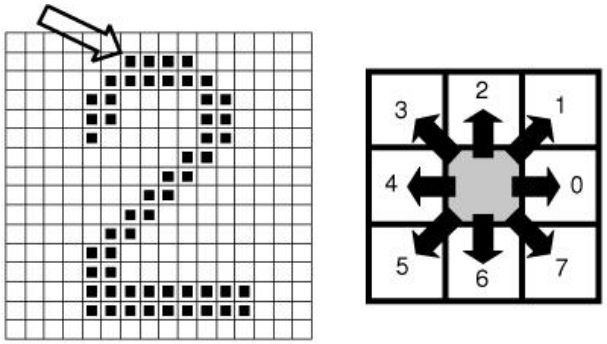


Figure 4.3: An example chain code; the reference pixel starting the chain is marked by an arrow: 0007766555555660000000644444444222111111223444565221 1. © *Cengage Learning 2015.*

Topological Structures

- Image described by elements and relationships
- Represented by graphs
  - $G = (V, E)$
    - $V = \{v_1, v_2, \ldots, v_n\}$ – nodes
    - $E = \{e_1, e_2, \ldots, e_n\}$ – relationship edges (arcs)
    - Degree of node is the number of incident edges
  - Weighted graphs have values (weight, cost) for edges
- Region adjacency graph
  - Nodes are image regions and edges connect neighboring regions
  - Created from region map (labeled image)
  - A graph cut can extract inside regions simply

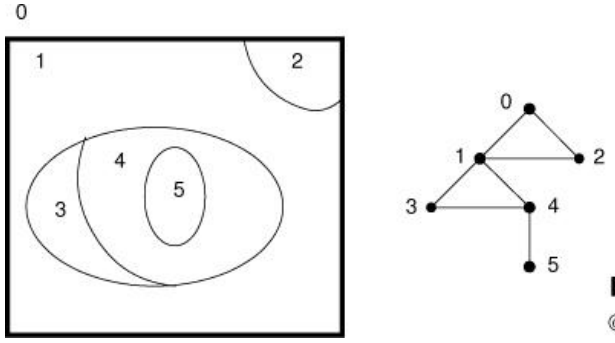


Figure 4.5: An example region adjacency graph.

Relational Structures

- Relations between image objects (segmentations) are stored in a table
- Useful for higher levels of image understanding

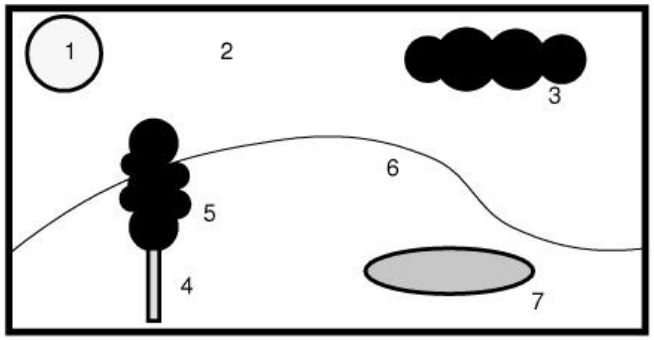


Figure 4.7: Description of objects using relational structure. © *Cengage L*

No.	Object name	Color	Min. row	Min. col.	Inside
1	sun	white	5	40	2
2	sky	blue	0	0	–
3	cloud	gray	20	180	2
4	tree trunk	brown	95	75	6
5	tree crown	green	53	63	–
6	hill	light green	97	0	–
7	pond	blue	100	160	6

Table 4.1: Relational table. © *Cengage Learning 2015.*

- Relational databases are popular (MySQL, MyMaria, PostgreSQL, Oracle)
  - Efficient search with keys

Hierarchical Data Structures

- Computer vision is a difficult and requires computational power
  - Can't always use brute force
- Hierarchical data structures give rise to algorithms that operate more efficiently
  - Use smaller subset of the data first
  - Go to full resolution processing only when required

Image Pyramids

- Matrix-pyramid
  - Sequence of images of different resolution
  - $\{M_L, M_{L-1}, \dots, M_0\}$
    - Each M_{l+1} is half the resolution of M_l
  - Allows operations at different resolutions (scale)
    - Half-size is ¼ pixels and 4 times speed up
- Tree-pyramid
  - Use several resolutions simultaneously
  - At the bottom of the tree (the highest level) are the original pixels values
  - Each lower level contains a mapping from 4 higher resolution "pixels"
    - Each level is a lower resolution image
- The memory for storing all images in a pyramid is only $1.33N^2$

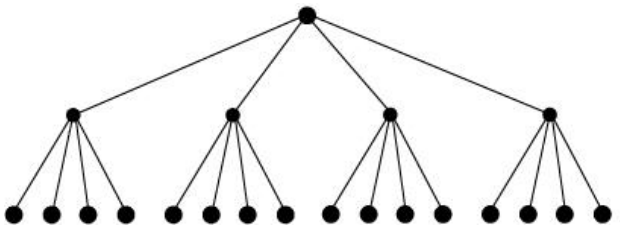


Figure 4.8: T-pyramid.

Quadtrees

- Modification of T-pyramid
  - Less expensive representations
  - Do not need to keep all 4 children nodes unless necessary
    - No need to store 4 children with same value
- Advantages: simple algorithms for addition of images, object areas, statistical moments
- Disadvantages: dependence on position, orientation, size of objects
  - Normalized shape quadtree
    - Build quadtree for each object
- Have become popular in GIS mapping for layered data

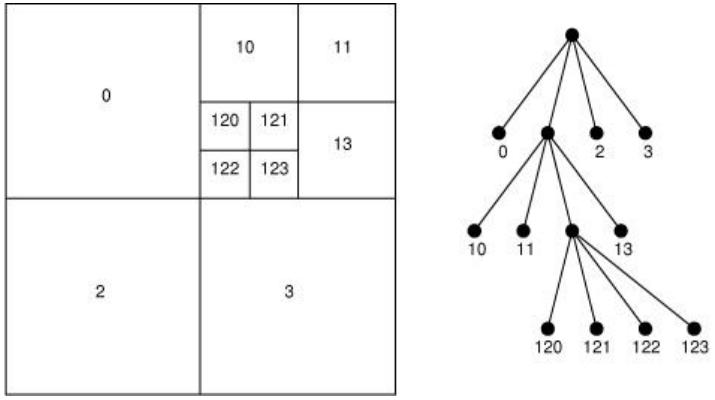


Figure 4.9: Quadtree. © *Cengage Learning 2015.*