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

ECG782: Multidimensional Digital Signal Processing

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

Lecture 07
Image Pre-Processing
13/02/11

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

Outline

- Integral Image
- Image Structures
- Image Pyramids

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

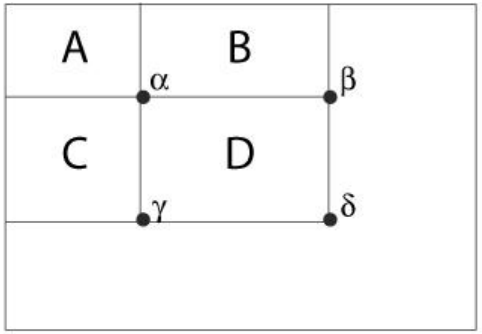


Figure 4.1: Calculation of rectangle features from an integral image. The sum of pixels within rectangle D can be obtained using four array references. $D_{sum} = ii(\delta) + ii(\alpha) - (ii(\beta) + ii(\gamma))$, where $ii(\alpha)$ is the value of the integral image at point α (and similarly for β,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

Haar-Like Features

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

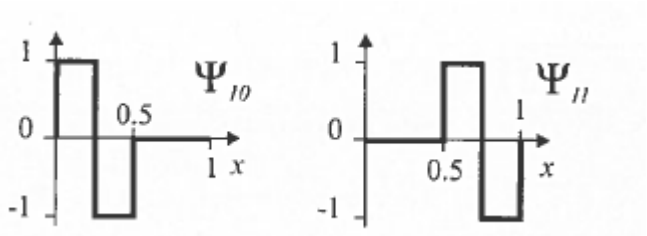


Figure 3.15: Haar wavelets Ψ_{11}, Ψ_{12}. © *Cengage Learning 2015.*

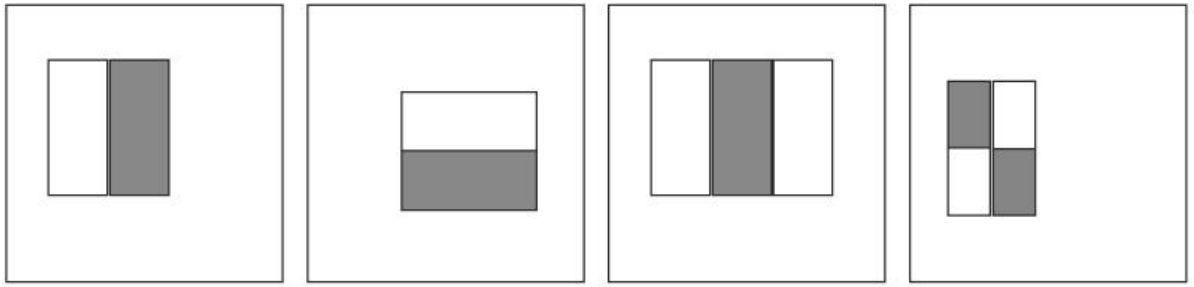


Figure 4.2: Rectangle-based features may be calculated from an integral image by subtraction of the sum of the shaded rectangle(s) from the non-shaded rectangle(s). The figure shows (a,b) two-rectangle, (c) three-rectangle, and (d) four-rectangle features. Sizes of the individual rectangles can be varied 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

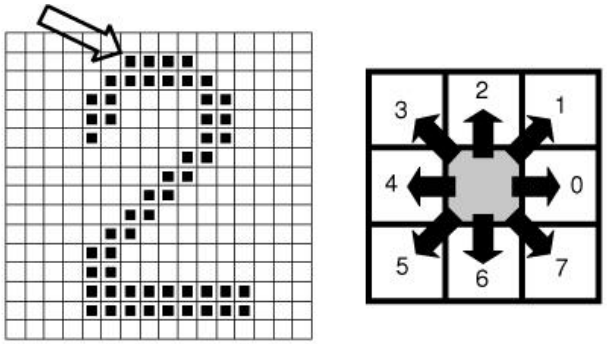


Figure 4.3: An example chain code; the reference pixel starting the chain is marked by an arrow: 00077665555556600000006444444442221111112234445652211. © *Cengage Learning 2015.*

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

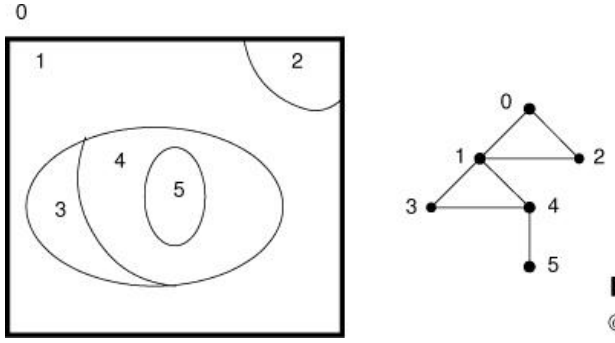


Figure 4.5: An example region adjacency graph.
© *Cengage Learning 2015.*

Relational Structures

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

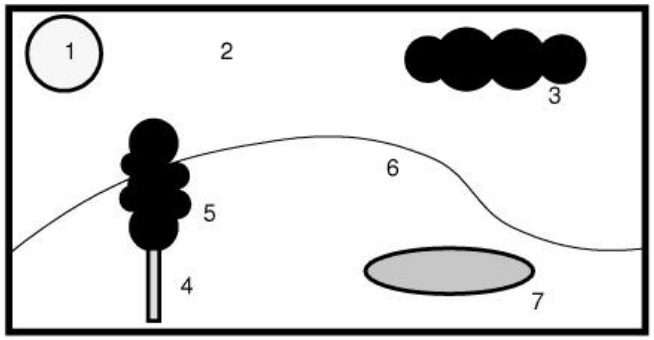


Figure 4.7: Description of objects using relational structure. © *Cengage L*

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 $1.33N^2$

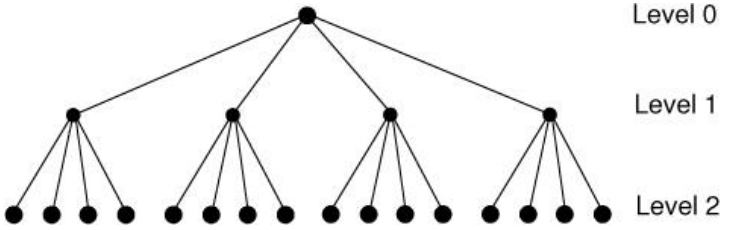


Figure 4.8: T-pyramid.
© Cengage Learning 2015.

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

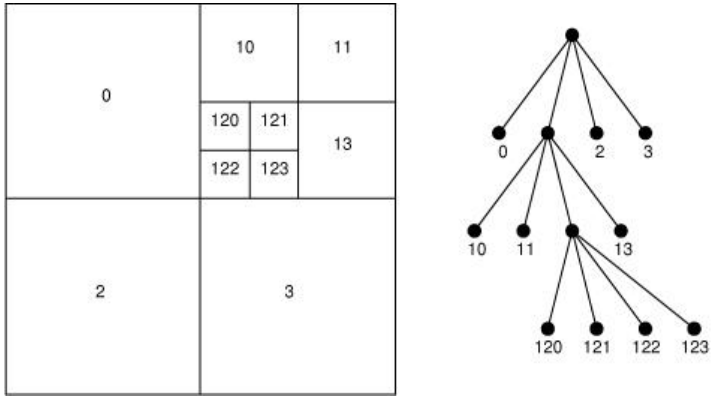


Figure 4.9: Quadtree. © *Cengage Learning 2015.*

Image Pre-Processing

- Low level operations
  - Lowest-level of abstraction
  - Image-to-image transformations
- Does not increase image information content
  - Actually decreases entropy
  - However, it can suppress irrelevant info
    - Not needed for analysis task
- Improve image by suppressing unwanted distortions and enhancing important image features
  - Note: geometric transforms also considered
  - Utilizes information redundancy
    - Large number of similar pixels for statistical characterization

Pixel Brightness Correction

- Modify pixel brightness with regard to position
- Systematic imaging degradation can be suppressed
  - E.g. CCD sensitivity on borders
- Multiplicative error model
  - $f(i,j) = e(i,j)g(i,j)$
  - f – degraded image
  - g – reference (“good”) image
  - e – multiplicative noise, error coefficient
- Recovery of good image
  - $g(i,j) = \frac{f(i,j)}{e(i,j)}$
  - Estimate error by imaging a known constant value

Gray-Scale Transformation

- Change pixel brightness without regard for position in image
  - E.g. histogram equalization
- Define a mapping between one gray level to another
  - Represented as a lookup table
  - Generalizes to multi-spectral images
    - Color conversion ta
- Typically used for hum observation
  - Contrast is needed

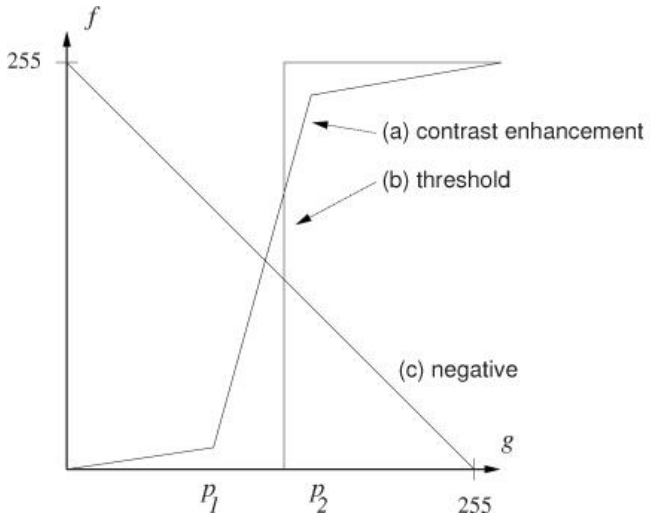
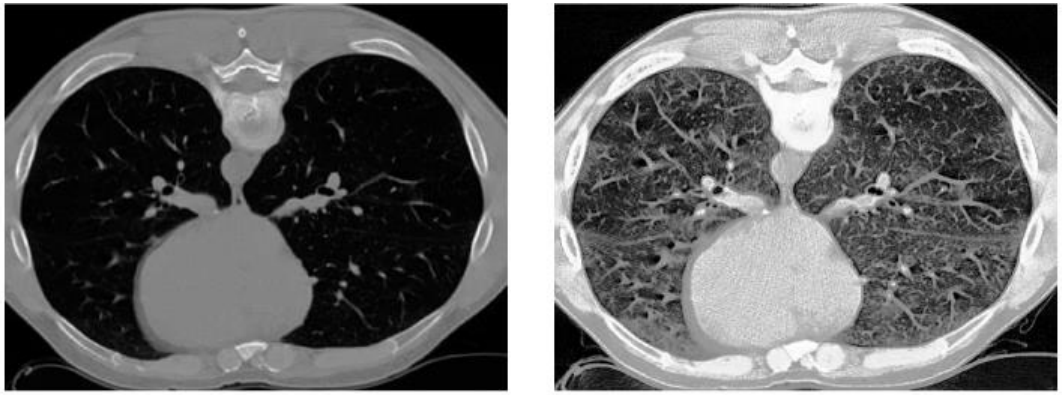


Figure 5.1: Some gray-scale transformations. © *Cengage Learning 2015.*



(a) (b)

Figure 5.3: Histogram equalization. (a) Original image. (b) Equalized image. © *Cengage Learning 2015.*

Local Pre-Processing

- Smoothing
  - Suppress noise and other small fluctuations
  - Equivalent to suppression of high frequency content
  - Blurs sharp edges
    - May lose information content
- (Sharpening) Gradient operators
  - Based on local derivatives of image
  - Suppress low frequency content
    - Accentuate edges
  - Increases noise level
- Linear transformations
  - Output value is a linear combination of local neighborhood values
  - $f(i,j) = \sum\sum_{(m,n)\in O} h(i-m, j-n)g(m,n)$
  - Discrete convolution(correlation) definition
    - Use rectangular neighborhoods with odd dimensions
- Non-linear transforms
  - Non-linear relationship between neighborhood
  - More computationally expensive
  - No strict frequency representation

Smoothing

- Want to edge-preserving smoothing
  - Remove noise but leave edges
- Averaging filter
  - Noise should be smaller in size than smallest object of interest
  - Significant edge blurring
  - $h = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$
- Gaussian approximation
  - Put more weight in center
  - $h = \frac{1}{10}\begin{bmatrix}1 & 1 & 1\\ 1 & 2 & 1\\ 1 & 1 & 1\end{bmatrix}$

- Separable filters
  - Used to significantly speed up convolution neighborhood operation
  - Kernel can be factorized into the product of two 1D vectors
    - Separate convolution summations
- 2D binomial kernel (Gaussian approximation)
- $h(x,y) = 4^{-(N-1)}\binom{N-1}{x}\binom{N-1}{y}$
  - $N = 3$
  - $h(x,y) = \frac{1}{16}\begin{bmatrix}1 & 2 & 1\\ 2 & 4 & 2\\ 1 & 2 & 1\end{bmatrix}$
  - $h(x,y) = \left(\frac{1}{4}\right)^2\begin{bmatrix}1\\ 2\\ 1\end{bmatrix}\begin{bmatrix}1 & 2 & 1\end{bmatrix}$
  - Elements from Pascal's triangle

Averaging with Limited Data Validity

- Avoid blurring by averaging only pixels that meet a criterion function
  - Try to avoid including pixels from separate features
    - E.g. two sides of edge
- Define an invalid data interval $[min, max]$
- $h(i,j) = \begin{cases} 1 & for\ g(m+i, n+j) \notin [min, max] \\ 0 & else \end{cases}$
  - Convolution mask defined for each neighborhood
    - Only changes invalid data
    - Uses only valid data for averaging
- Define a brightness interval around central pixel
- Use gradient strength to only average those pixels with low gradients

(a)

(b)

Figure 5.10: Averaging with limited data validity. (a) Original corrupted image. (b) Result of corruption removal. © *Cengage Learning 2015.*

Rotating Mask Averaging

- Non-linear smoothing technique
  - Also sharpens image
- Idea is to determine a good neighborhood for averaging
- Calculate average over different mask rotations

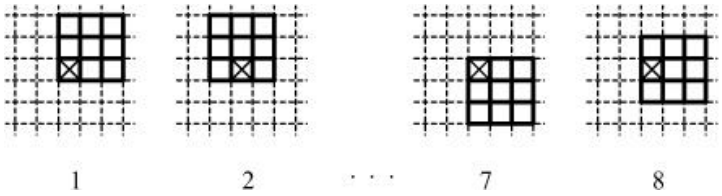


Figure 5.11: Eight possible rotated 3×3 masks. © *Cengage Learning 2015.*

- Homogeneity of region measured by a brightness dispersion

$$\sigma^2 = \frac{1}{n} \sum_{(i,j) \in R} \left(g(i,j) - \frac{1}{n} \sum_{(i,j) \in R} g(i,j) \right)^2 . \qquad (5.31)$$

Smoothing with Rotating Mask

Algorithm 5.2: Smoothing using a rotating mask

1. Consider each image pixel (i, j).
2. Calculate dispersion for all possible mask rotations about pixel (i, j) according to equation (5.31).
3. Choose the mask with minimum dispersion.
4. Assign to the pixel $f(i, j)$ in the output image f the average brightness in the chosen mask.

- Only use "best" mask for pixel replacement
  - Looking for "stable" average
- Iterative solution convergence depends on mask size and shape
  - Smaller mask has smaller changes and more iterations
  - Large mask suppresses noise faster and has more sharpening
    - Small detail is lost