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

EE482: Digital Signal Processing Applications

Spring 2014
TTh 14:30-15:45 CBC C222

Lecture 15
Image Processing
14/04/15

http://www.ee.unlv.edu/~b1morris/ee482/

Outline

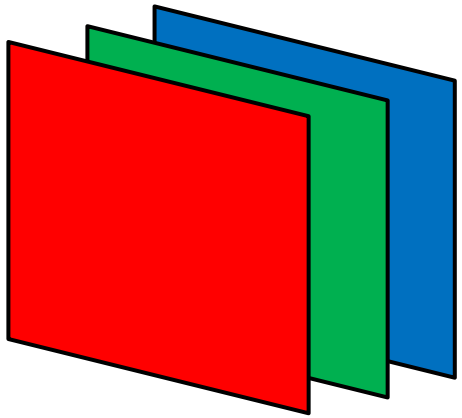
- Digital Images
- Color
- Histogram Equalization
- Image Filtering

Digital Image Processing

- Extension of 1D signal processing to 2D signal
  - E.g. vector valued signal domain or 2D range
  - Many common principles and ideas
- Many specific concepts arise from images
  - Large signals (e.g. 10 M pixel image) → video
  - Need for very efficient and optimized processing
  - Use of hardware accelerators (e.g. graphic processing units)

Digital Images

- Set of data samples mapped onto a 2D grid of points
  - $x(m,n) = v$; $M \times N$ image
    - $m = 0, \ldots, M-1$; column (width) index
    - $n = 0, \ldots, N-1$; row (height) index
    - Be aware: this is not the same notation as Matlab
      - Row, column indexing beginning with 1 index
  - Each sample is known as a pixel
- Image resolution
  - Ability to distinguish spatial details (dots/pixels per inch)
  - Analogous to sampling frequency
- Image value
  - Grayscale – $v = [0,255]$ (8-bit byte)
    - 0 – black, 255 – white
  - Color - $v = [R,G,B]$ (24-bit value)
    - Mixing of primary Red, Green, and Blue colors
    - Typically thought of as color "channels"



Color

- Color comes from underlying physical properties

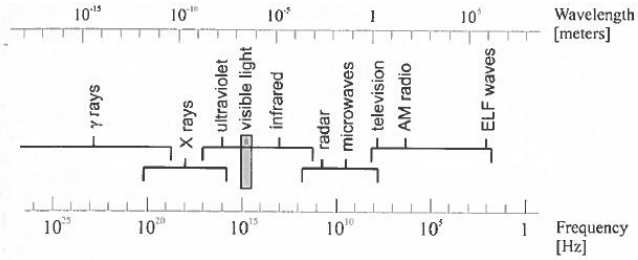


Figure 2.23: Division of the electromagnetic spectrum (ELF is Extremely Low Frequencies). © *Cengage Learning 2015.*

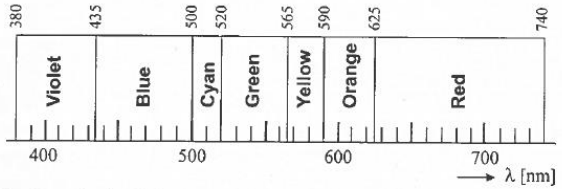


Figure 2.24: Wavelength λ of the spectrum visible to humans. © *Cengage Learning 2015.*

- However, humans do not perceive color in the same physical process
  - There is some subjectivity (e.g. color similarity)

- Cones in human retina are sensitive to color
  - In the center of eye
  - 3 different types for different EM frequency sensitivity
    - RGB mixing to build all colors
- Rods are monochromatic
  - On outside of the eye and good for low lighting and motion sensing

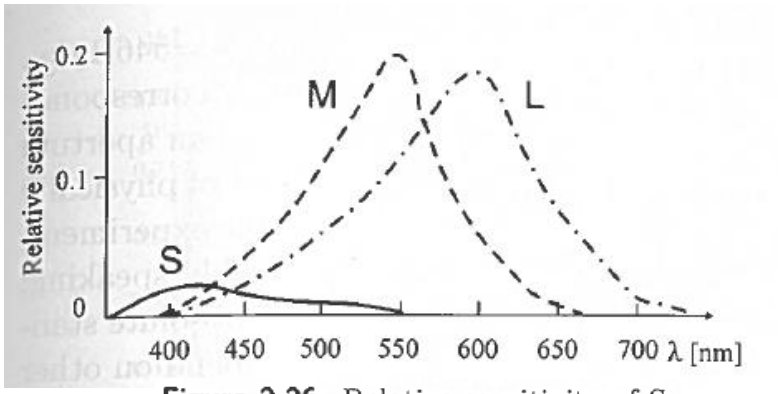


Figure 2.26: Relative sensitivity of S, M, L cones of the human eye to wavelength. © *Cengage Learning 2015.*

Colorspaces

- Uniform method for defining colors
- Can transform from one to another
  - Want to take advantage of properties and color gamut
- XYZ
  - International absolute color standard
  - No negative mixing
- RGB
  - Additive color mixing for red, green, and blue
  - Widely used in computers
- CMYK
  - Cyan, magenta, yellow, black
  - Used for printers and based off of reflectivity
- HSV
  - Hue, saturation, and value = color, amount, brightness
  - Closer to human perception

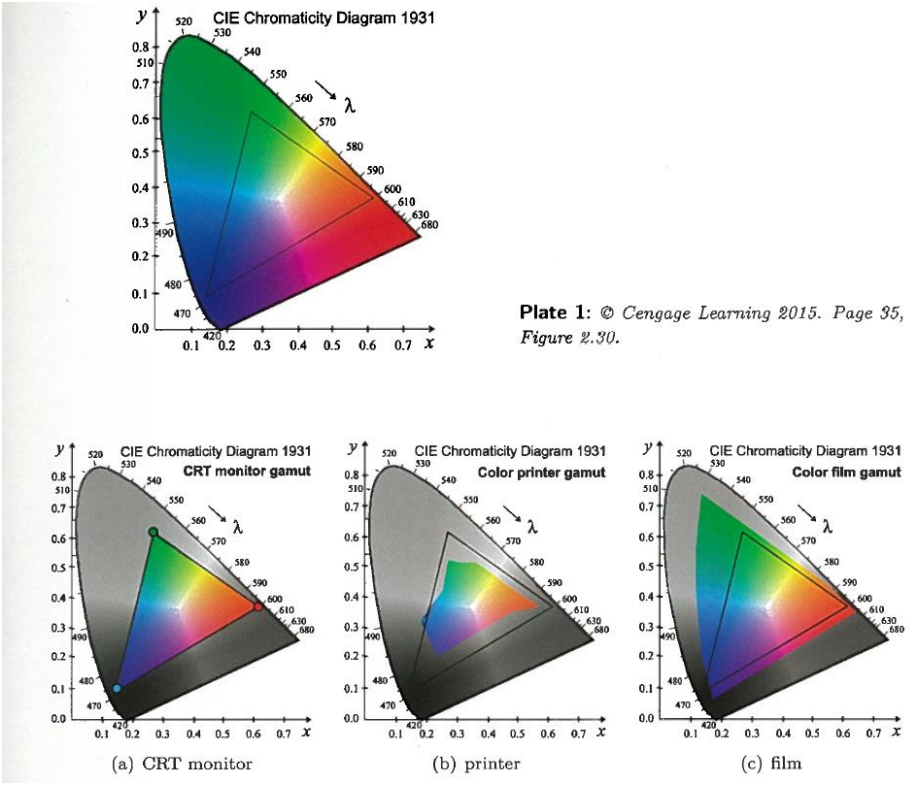
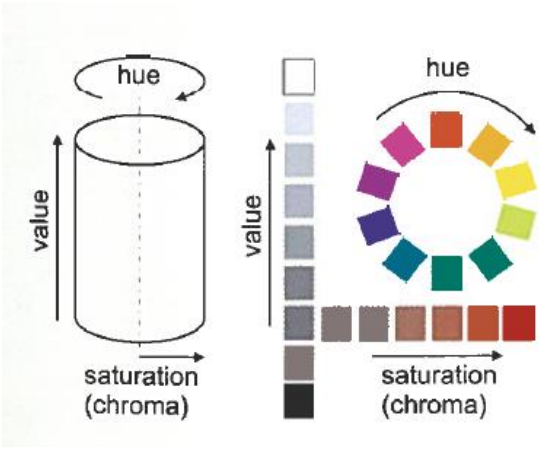


Plate 1: © *Cengage Learning 2015. Page 35, Figure 2.30.*

(a) CRT monitor (b) printer (c) film



Perceptual Colorspace Examples

- YUV – composite color video standard (analog)
- Separate brightness from chrominance (color)
  - More perceptually meaningful colorspace
    - Humans perceive brightness changes more than color
- $\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$

- YCbCr – digital color standard
- Separate brightness from chrominance
  - Used in JPEG
- $\begin{bmatrix} Y \\ U \\ V \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.147 & -0.289 & 0.436 \\ 0.615 & -0.515 & -0.100 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} + \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix}$
- Matlab – `rgb2ycbcr.m`
- Efficient representation using subsampling color space
  - Can reduce chrominance bits

Table 11.1 Number of bits used for four YC_bC_r sub-sampling schemes

720 × 480 pixels	Bits for Y	Bits for C_b	Bits for C_r	Total bits
YC_rC_b4:4:4	720 × 480 × 8	720 × 480 × 8	720 × 480 × 8	8 294 400
YC_rC_b4:2:2	720 × 480 × 8	360 × 480 × 8	360 × 480 × 8	5 529 600
YC_rC_b4:2:0	720 × 480 × 8	360 × 240 × 8	360 × 240 × 8	4 147 200
YC_rC_b4:1:1	720 × 480 × 8	180 × 480 × 8	180 × 480 × 8	4 147 200

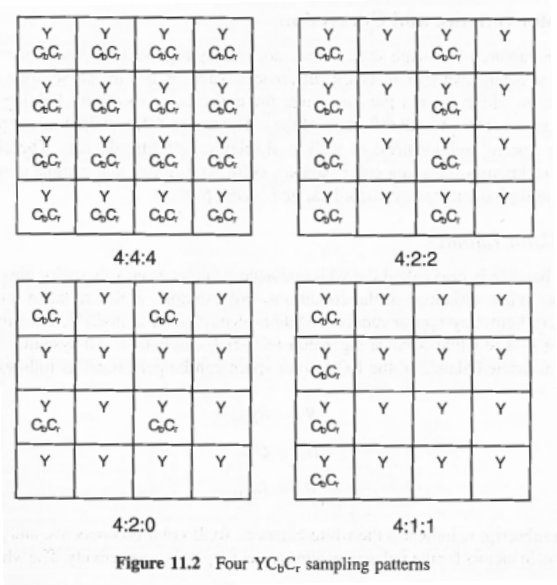


Figure 11.2 Four YC_bC_r sampling patterns

Example Color Spaces

RGB Image

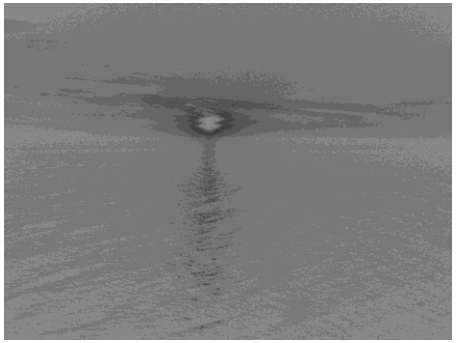
R Channel

G Channel

B Channel

Y Channel (Intensity)

Cb Channel



Cr Channel

YCbYr Image

Color Balance

- Correct color bias caused by lighting and other variations
  - Also known as white balance
- Adjust image color to more closely depict human visual system
- White balance algorithm
  - $R_w = R g_R$
  - $G_w = G g_G$
  - $B_w = B g_B$
    - Apply gain to each color channel
    - Normalize to green color channel

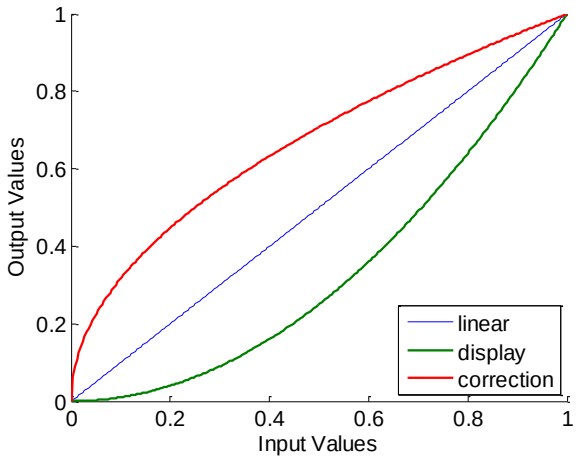
- Example 11.2
  - Color balance an image

Color Correction

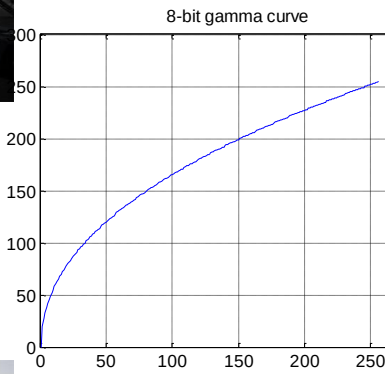
- RGB from digital camera may not match color perceived by humans
- Color correction adjusts RGB values to correspond better to human vision
  - Also known as chromatic or saturation correction
- Apply correction to white-balanced RGB image
  - $$\begin{bmatrix} R_c \\ G_c \\ G_c \end{bmatrix} = \begin{bmatrix} c_{11} & c_{12} & c_{13} \\ c_{21} & c_{22} & c_{23} \\ c_{31} & c_{32} & c_{33} \end{bmatrix} \begin{bmatrix} R_w \\ G_w \\ B_w \end{bmatrix}$$
- The coefficients are selected to minimize mean-square error between a reference color chart
  - $\min \left\{ \sum_{n=1}^{3} \sum_{m=1}^{3} \left[c_{nm} x_w(m,n) - x_{ref}(m,n) \right]^2 \right\}, n \neq m$
  - $c_{nm} = 1, n = m$

Gamma Correction

- Used to compensate for nonlinearity in display device
  - $R_w = gR_c^{1/\gamma}$
  - $G_w = gG_c^{1/\gamma}$
  - $B_w = gB_c^{1/\gamma}$
    - γ – gamma value represents non-linearity of display
    - g – is a correction factor



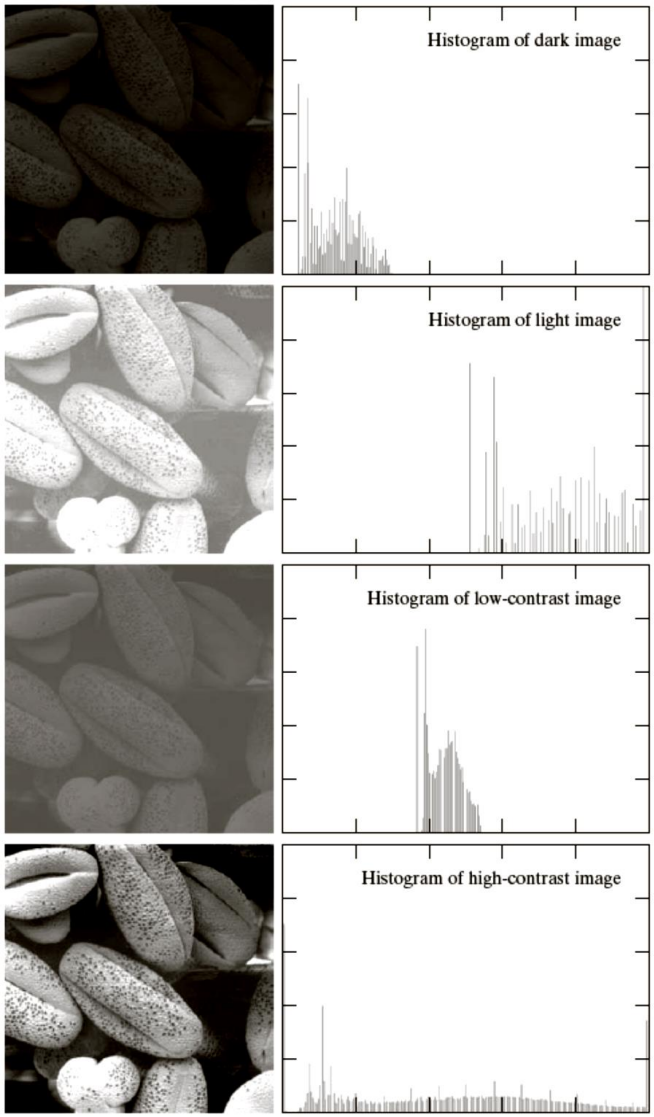
- Example 11.3



Histogram Processing

- Digital image histogram is the count of pixels in an image having a particular value in range $[0, L-1]$
  - $h(r_k) = n_k$
    - r_k - the kth gray level value
      - Set of r_k are known as the bins of the histogram
    - n_k - the numbers of pixels with kth gray level
- Empirical probability of gray level occurrence is obtained by normalizing the histogram
  - $p(r_k) = n_k/n$
    - n – total number of pixels
  - Histogram is viewed as the probability that a pixel will take a given intensity value in an image

Histogram Example



- x-axis – intensity value
  - Bins [0, 255]
- y-axis – count of pixels

- Dark image
  - Concentration in lower values
- Bright image
  - Concentration in higher values
- Low-contrast image
  - Narrow band of values
- High-contrast image
  - Intensity values in wide band

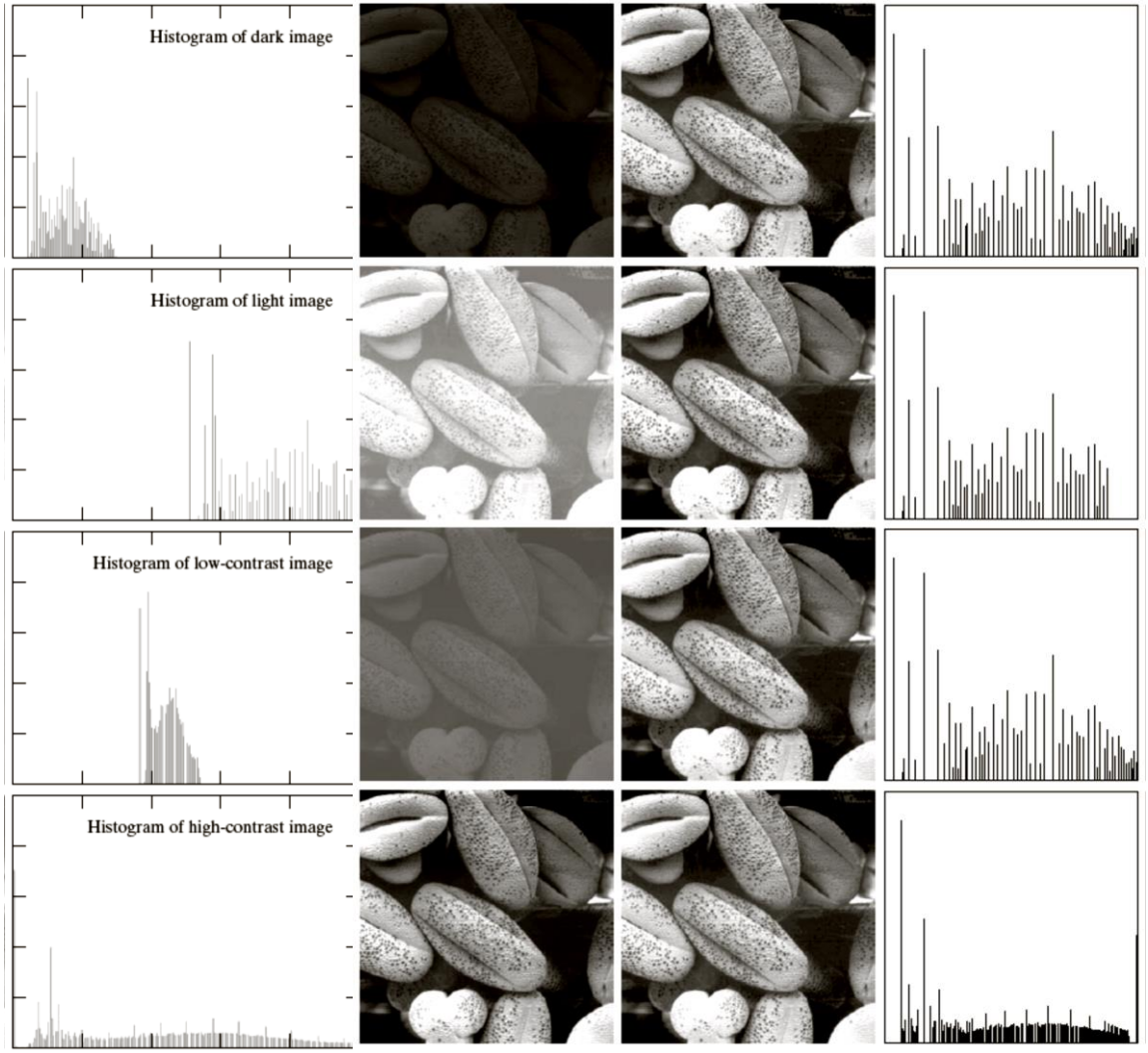
Histogram Equalization

- Assume continuous functions (rather than discrete images)
- Define a transformation of the intensity values to "equalize" each pixel in the image
  - $s = T(r) \quad 0 \leq r \leq 1$
  - Notice: intensity values are normalized between 0 and 1
- The inverse transformation is given as
  - $r = T^{-1}(s) \quad 0 \leq s \leq 1$
- Viewing the gray level of an image as a random variable
  - $p_s(s) = p_r(r) \left| \frac{dr}{ds} \right|$
- Let s by the cumulative distribution function (CDF)
  - $s = T(r) = \int_0^r p_r(w) dw$
- Then
  - $\frac{ds}{dr} = p_r(r)$
- Which results in a uniform PDF for the output intensity
  - $p_s(s) = 1$
- Hence, using the CDF of a histogram will "equalize" an image
  - Make the resulting histogram flat across all intensity levels

Discrete Histogram Equalization

- The probability density is approximated by the normalized histogram
  - $p_r(r_k) = \frac{n_k}{n} \quad k = 0, \ldots, L-1$
- The discrete CDF transformation is
  - $s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j)$
  - $s_k = \sum_{j=0}^{k} \frac{n_k}{n}$
- This transformation does not guarantee a uniform histogram in the discrete case
  - It has the tendency to spread the intensity values to span a larger range

Histogram Equalization Example



- Equalized histograms have wider spread of intensity levels
- Notice the equalized images all have similar visual appearance
  - Even though histograms are different
  - Contrast enhancement

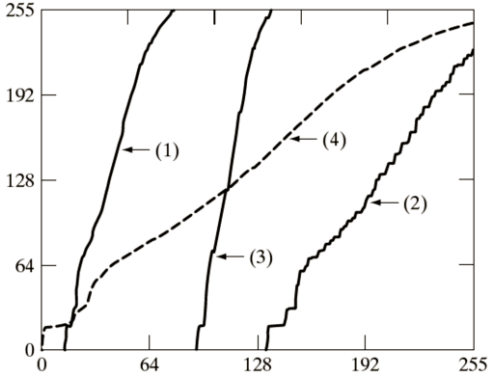
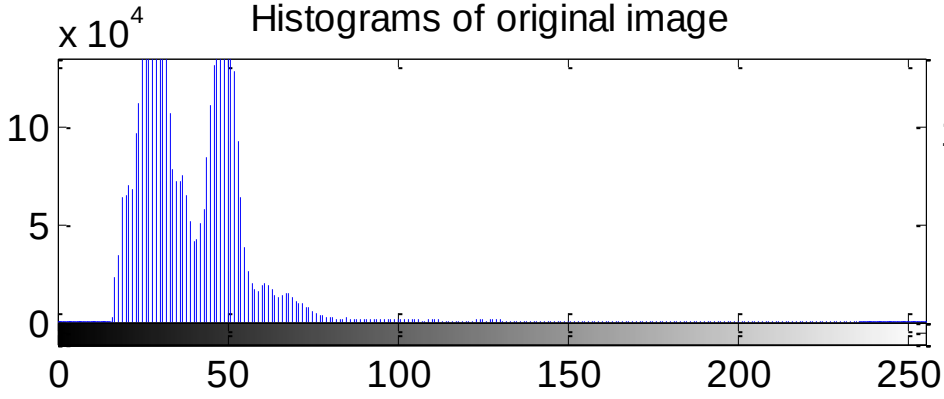
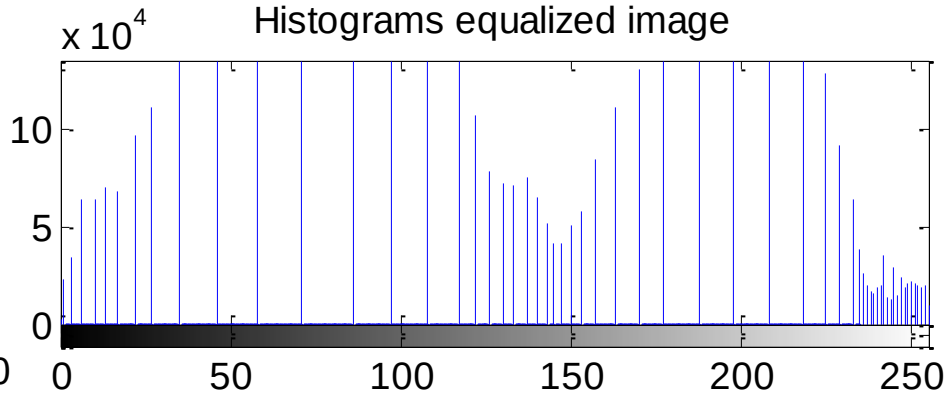


FIGURE 3.21 Transformation functions for histogram equalization. Transformations (1) through (4) were obtained from the histograms of the images (from top to bottom) in the left column of Fig. 3.20 using Eq. (3.3-8).

Example 11.4

- Histogram equalization of a dark image





Local Histogram Enhancement

- Global methods (like histogram equalization as presented) may not always make sense
  - What happens when properties of image regions are different?
- Compute histogram over smaller windows
  - Break image into “blocks”
  - Process each block separately

- Original image

- Block histogram equalization

- Notice the blocking effects that cause noticeable boundary effects

Local Enhancement

- Compute histogram over a block (neighborhood) for every pixel in a moving window

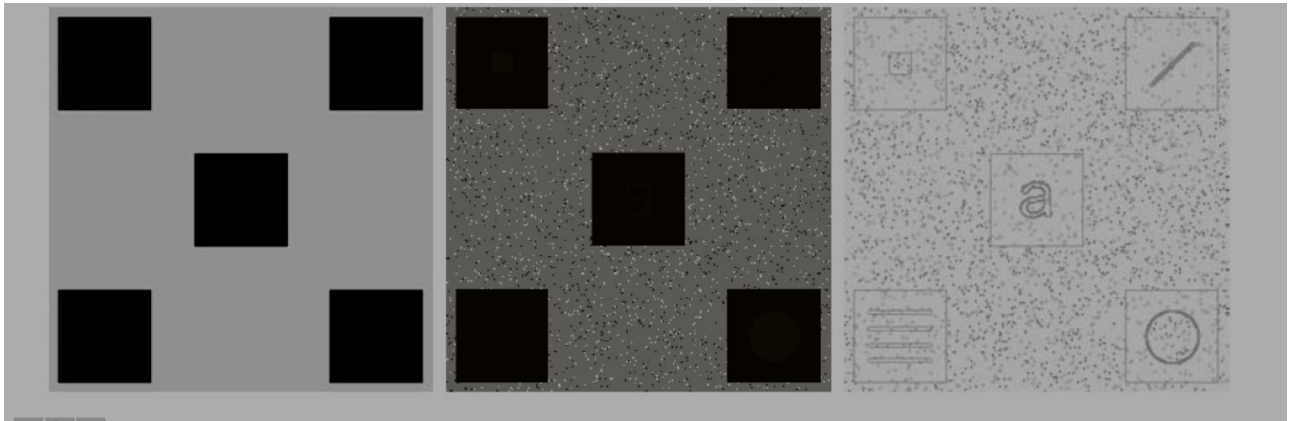


FIGURE 3.26 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization applied to (a), using a neighborhood of size 3 × 3.

- Adaptive histogram equalization (AHE) is a computationally efficient method to combine block based computations through interpolation (`adapthisteq.m`)

Figure 3.8 Locally adaptive histogram equalization: (a) original image; (b) block histogram equalization; (c) full locally adaptive equalization.

Image Processing Motivation

- Image processing is useful for the reduction of noise

- Common types of noise
  - Salt and pepper – random occurrences of black and white pixels
  - Impulse – random occurrences of white pixels
  - Gaussian – variations in intensity drawn from normal distribution

Original

Salt and pepper noise

Impulse noise

Gaussian noise

Adapted from S. Seitz

Ideal Noise Reduction

- How can we reduce noise given a single camera and a still scene?
  - Take lots of images and average them

- What about if you only have a single image?

Adapted from S. Seitz

Image Filtering

- Filtering is a neighborhood operation
  - Use the pixels values in the vicinity of a given pixel to determine its final output value

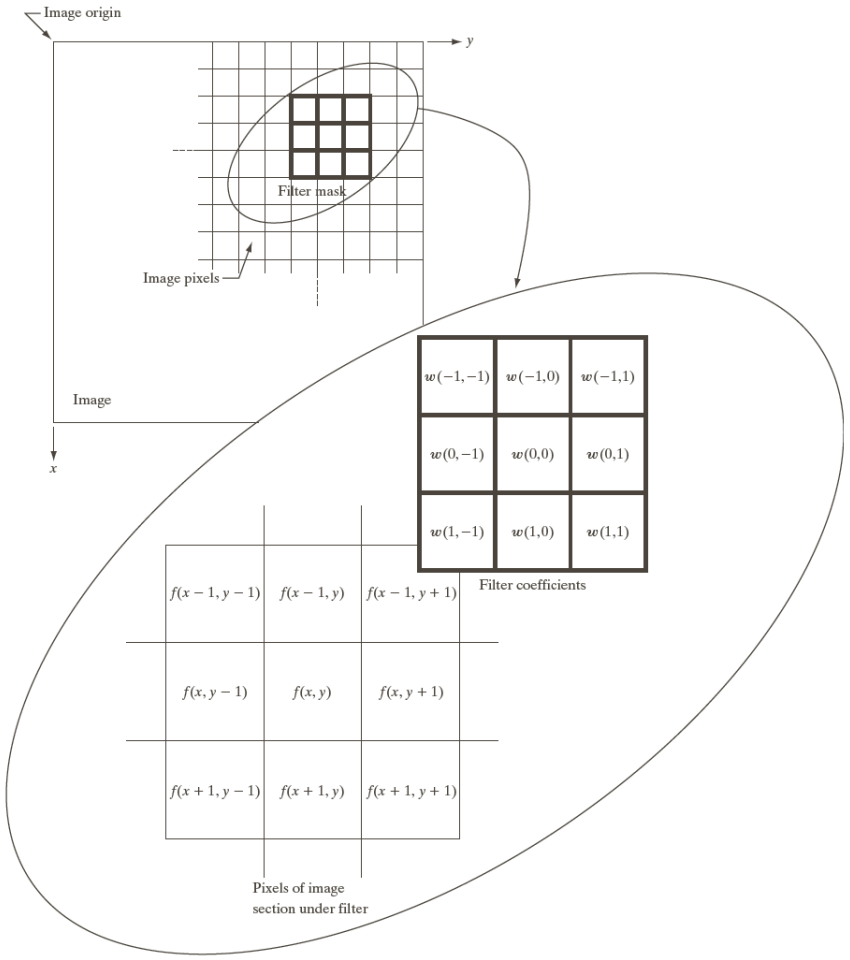
- Motivation: noise reduction
  - Replace a pixel by the average value in a neighborhood
  - Assumptions:
    - Expect pixels to be similar to their neighbors (local consistency)
    - Expect noise processes to be independent from pixel to pixel (i.i.d.)

Linear Filtering

- Most common type of neighborhood operator
- Output pixel is determined as a weighted sum of input pixel values
  - $g(x,y) = \sum_{k,l} f(x+k, y+l)w(k,l)$
    - w – is known as the kernel, mask, filter, template, or window
      - $w(k,l)$ – entry is known as a kernel weight or filter coefficient
- This is also known as the correlation operator
  - $g = f \otimes w$

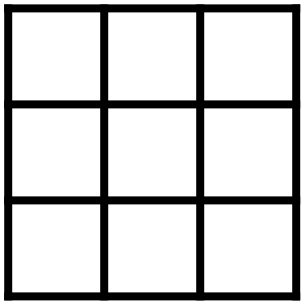
Filtering Operation

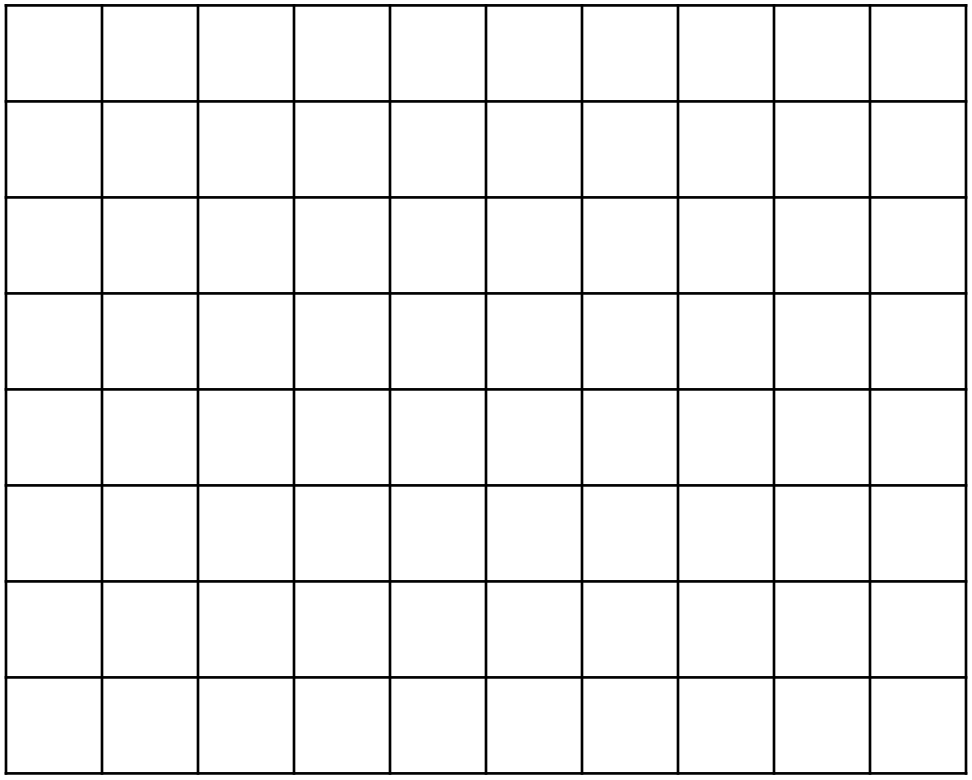
- $g(x,y) = \sum_{k,l} f(x+k, y+l)w(k,l)$
- The filter mask is moved from point to point in an image
  - The response is computed based on the sum of products of the mask coefficients and image
- Notice the mask is centered at $w(0,0)$
  - Usually we use odd sized masks so that the computation is symmetrically defined
- Matlab commands
  - `imfilter.m, filter2.m, conv2.m`



Filtering Raster Scan

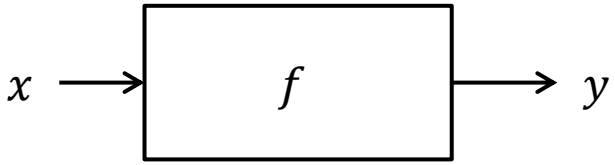
- Zig-zag scan through of image
  - Process image row-wise



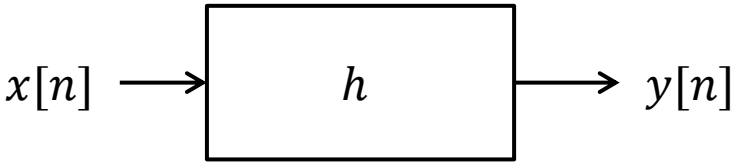


Connection to Signal Processing

- General system notation

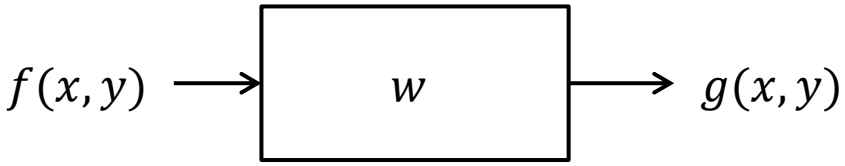


- LTI system
  - Convolution relationship
- Discrete 1D LTI system



$$y[n] = \sum_{k=-\infty}^{\infty} x[k]h[n-k]$$

- Discrete 2D LTI system



$$g(x,y) = \sum_{s=-\infty}^{\infty} \sum_{t=-\infty}^{\infty} f(s,t)w(x-s, y-t)$$

  - Linear filtering is the same as convolution without flipping

Image Filters

- Can be used for noise reduction, edge enhancement, sharpening, blurring, etc.
  - Generally like to use linear filtering (simple)
    - Advanced photoshopping uses more complex non-linear filters
- Lowpass filters - remove high frequency (noise) components
  - Smoothing filter
    - Blurs edges
- Highpass filters - remove low frequency components
  - Edge enhancement filter
- Generally, kernels are symmetric in both horizontal and vertical directions
- Filtering is computationally expensive
  - Use small 3×3 or 5×5 kernels for real-time application

Smoothing Filters

- Smoothing filters are used for blurring and noise reduction
  - Blurring is useful for small detail removal (object detection), bridging small gaps in lines, etc.
- These filters are known as lowpass filters
  - Higher frequencies are attenuated
  - What happens to edges?

Linear Smoothing Filter

- The simplest smoothing filter is the moving average or box filter
  - Computes the average over a constant neighborhood

$$\frac{1}{K^2}\begin{bmatrix} 1 & 1 & \cdots & 1 \\ 1 & 1 & \cdots & 1 \\ \vdots & \vdots & 1 & \vdots \\ 1 & 1 & \cdots & 1 \end{bmatrix}$$

- This is a separable filter
  - Horizontal 1D filter
  - Remember your square wave from DSP
    - $h[n] = \begin{cases} 1 & 0 \le n \le M \\ 0 & \text{else} \end{cases}$
    - Fourier transform is a sinc function

$$\frac{1}{K}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$$

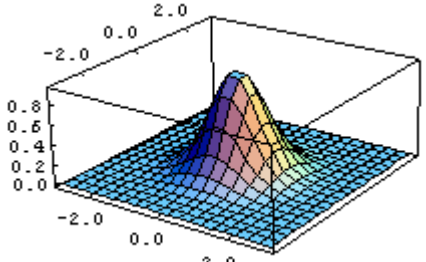
More Linear Smoothing Filters

- More interesting filters can be readily obtained
- Weighted average kernel (bilinear) - places more emphasis on closer pixels
  - More local consistency

$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

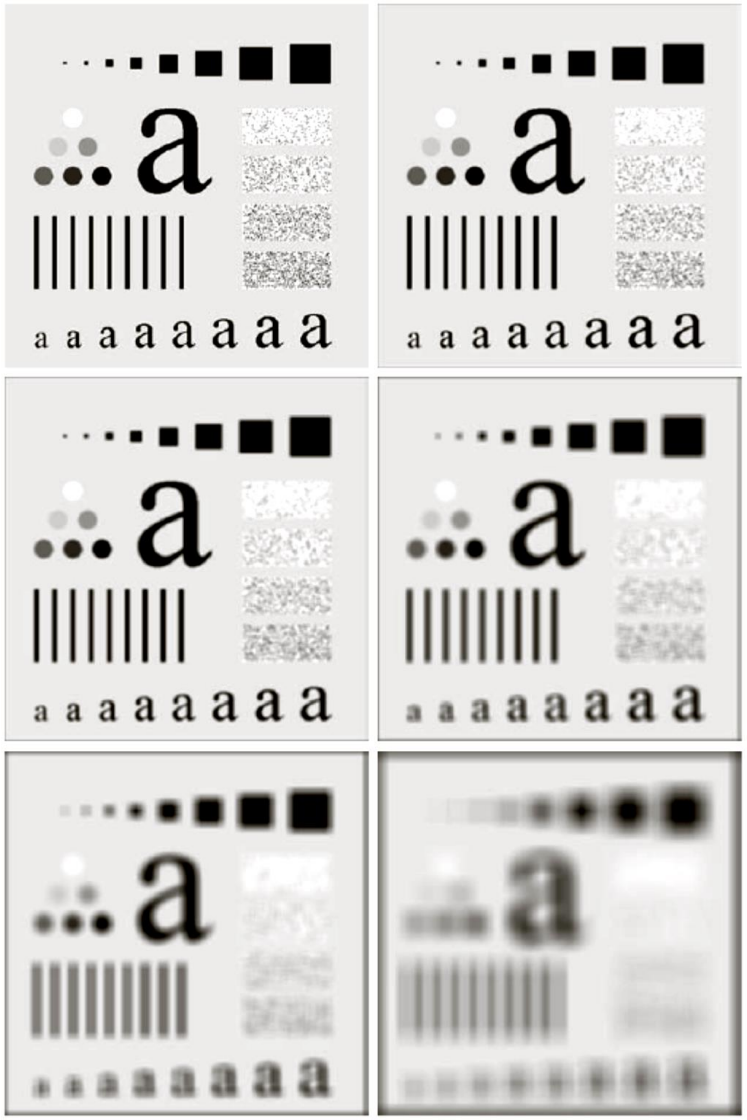
- Gaussian kernel - an approximation of a Gaussian function
  - Has variance parameter to control the kernel "width"
  - `fspecial.m`

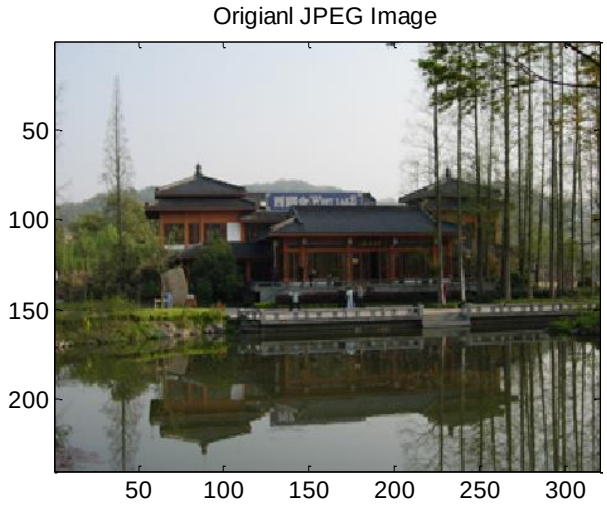
$$h(u,v) = \frac{1}{2\pi\sigma^2} e^{-\frac{u^2+v^2}{\sigma^2}}$$

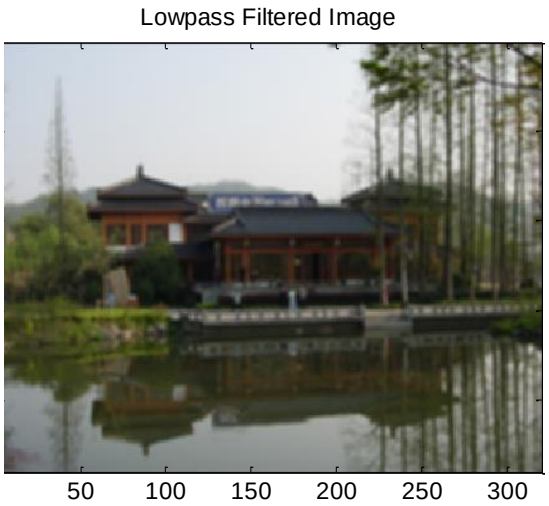


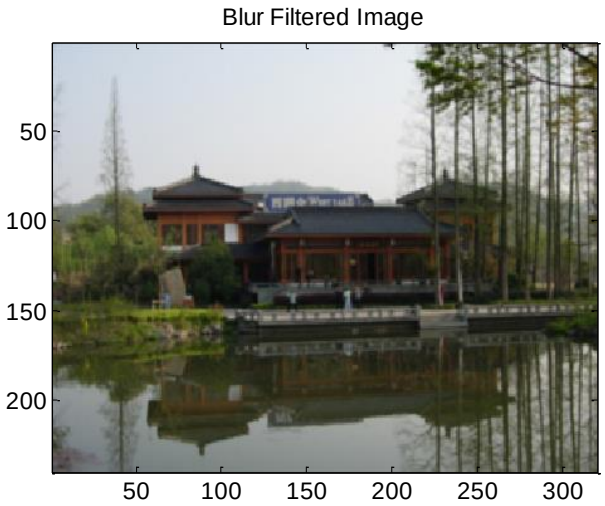
Adapted from S. Seitz

Lowpass Examples



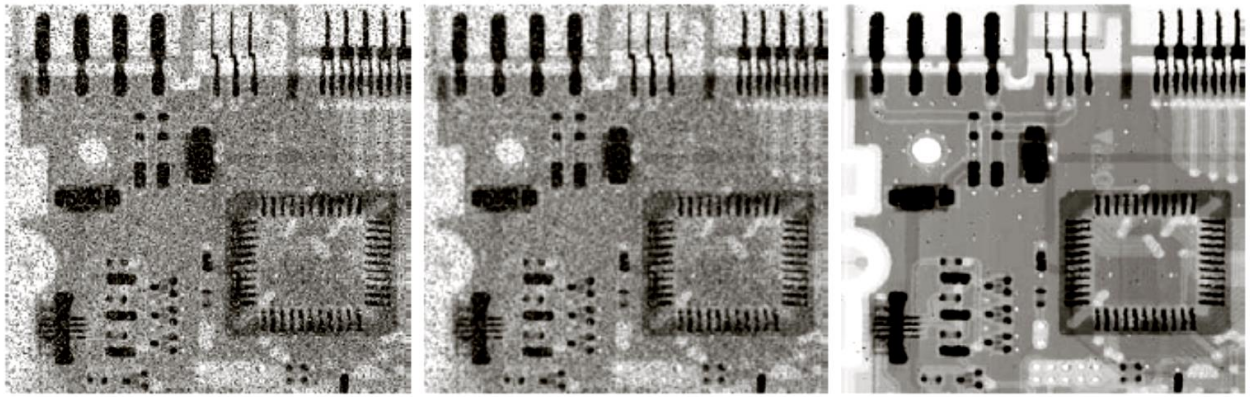






Median Filtering

- Sometimes linear filtering is not sufficient
  - Non-linear neighborhood operations are required
- Median filter – replaces the center pixel in a mask by the median of its neighbors
  - Non-linear operation, computationally more expensive
  - Provides excellent noise-reduction with less blurring than smoothing filters of similar size (edge preserving)
    - For impulse and salt-and-pepper noise



a b c

FIGURE 3.35 (a) X-ray image of circuit board corrupted by salt-and-pepper noise. (b) Noise reduction with a 3×3 averaging mask. (c) Noise reduction with a 3×3 median filter. (Original image courtesy of Mr. Joseph E. Pascente, Lixi, Inc.)

Sharpening Filters

- Sharpening filters are used to highlight fine detail or enhance blurred detail

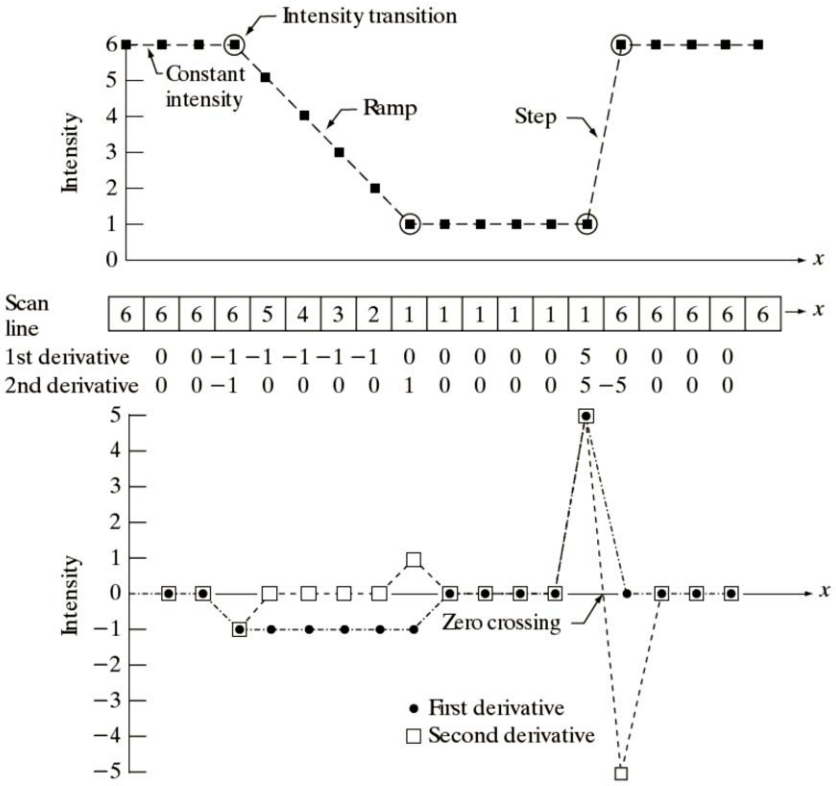
- Smoothing we saw was averaging
  - This is analogous to integration
- Since sharpening is the dual operation to smoothing, it can be accomplished through differentiation

Digital Derivatives

- Derivatives of digital functions are defined in terms of differences
  - Various computational approaches
- Discrete approximation of a derivative
  - $\frac{\partial f}{\partial x} = f(x+1) - f(x)$
  - $\frac{\partial f}{\partial x} = f(x+1) - f(x-1)$
    - Center symmetric
- Second-order derivative
  - $\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x)$

Difference Properties

- 1st derivative
  - Zero in constant segments
  - Non-zero at intensity transition
  - Non-zero along ramps
- 2nd derivative
  - Zero in constant areas
  - Non-zero at intensity transition
  - Zero along ramps
- 2nd order filter is more aggressive at enhancing sharp edges
  - Outputs different at ramps
    - 1st order produces thick edges
    - 2nd order produces thin edges
  - Notice: the step gets both a negative and positive response in a double line

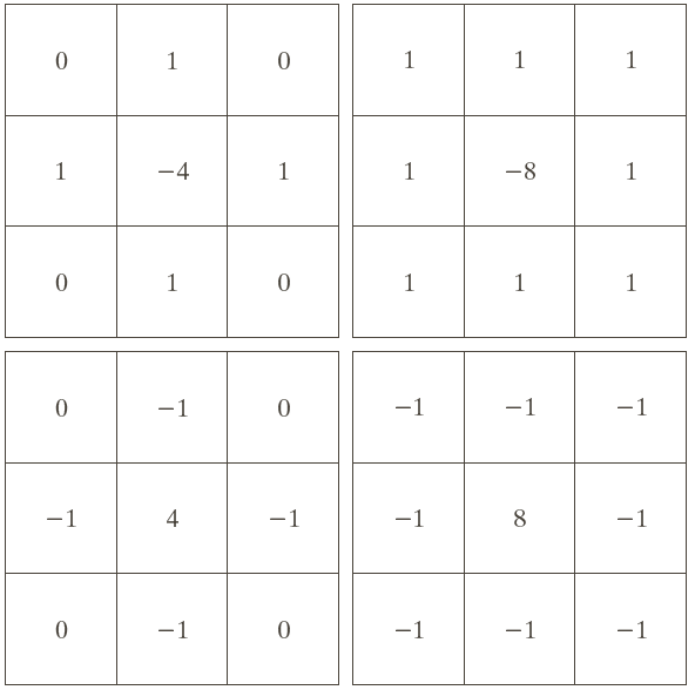


The Laplacian

- 2nd derivatives are generally better for image enhancement because of sensitivity to fine detail
- The Laplacian is simplest isotropic derivative operator
  - $\nabla^2 f = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$
  - Isotropic – rotation invariant
- Discrete implementation using the 2nd derivative previously defined
  - $\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$
  - $\frac{\partial^2 f}{\partial y^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$
  - $\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)$

Discrete Laplacian

- Zeros in corners give isotropic results for rotations of 90°
- Non-zeros corners give isotropic results for rotations of 45°
  - Include diagonal derivatives in Laplacian definition
- Center pixel sign indicates light-to-dark or dark-to-light transitions
  - Make sure you know which



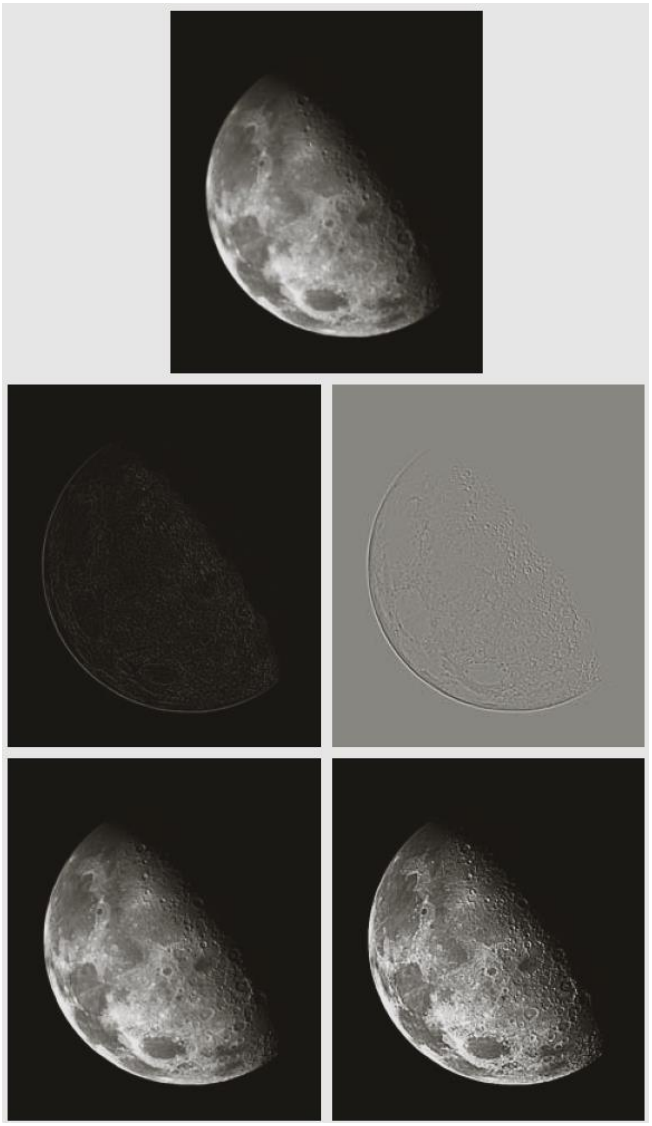
0	1	0	1	1	1
1	−4	1	1	−8	1
0	1	0	1	1	1
0	−1	0	−1	−1	−1
−1	4	−1	−1	8	−1
0	−1	0	−1	−1	−1

a b
c d

FIGURE 3.37
(a) Filter mask used to implement Eq. (3.6-6). (b) Mask used to implement an extension of this equation that includes the diagonal terms. (c) and (d) Two other implementations of the Laplacian found frequently in practice.

Sharpening Images

- Sharpened image created by addition of Laplacian
  - $g(x,y) = \begin{cases} f(x,y) - \nabla^2 f(x,y) & w(0,0) < 0 \\ f(x,y) + \nabla^2 f(x,y) & w(0,0) > 0 \end{cases}$
- Notice: the use of diagonal entries creates much sharper output image
- How can we compute $g(x,y)$ in one filter pass without the image addition?
  - Think of a linear system

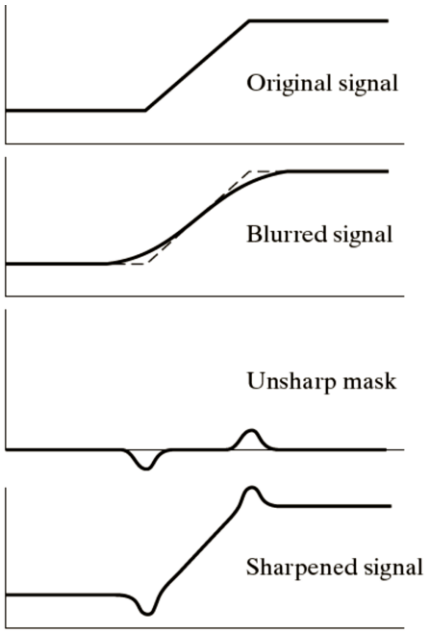


a
b c
d e

FIGURE 3.38 (a) Blurred image of the North Pole of the moon. (b) Laplacian without scaling. (c) Laplacian with scaling. (d) Image sharpened using the mask in Fig. 3.37(a). (e) Result of using the mask in Fig. 3.37(b). (Original image courtesy of NASA.)

Unsharp Masking

- Edges can be obtained by subtracting a blurred version of an image
  - $f_{us}(x,y) = f(x,y) - \bar{f}(x,y)$
  - Blurred image
    - $\bar{f}(x,y) = h_{\text{blur}} * f(x,y)$
- Sharpened image
  - $f_s(x,y) = f(x,y) + \gamma f_{us}(x,y)$



a
b
c
d

FIGURE 3.39 1-D illustration of the mechanics of unsharp masking. (a) Original signal. (b) Blurred signal with original shown dashed for reference. (c) Unsharp mask. (d) Sharpened signal, obtained by adding (c) to (a).

a
b
c
d
e

FIGURE 3.40 (a) Original image. (b) Result of blurring with a Gaussian filter. (c) Unsharp mask. (d) Result of using unsharp masking. (e) Result of using highboost filtering.

The Gradient

- 1st derivatives can be useful for enhancement of edges
  - Useful preprocessing before edge extraction and interest point detection
- The gradient is a vector indicating edge direction
  - $\nabla \mathbf{f} = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$
- The gradient magnitude can be approximated as
  - $\nabla f \approx |G_x| + |G_y|$
  - This give isotropic results for rotations of 90°

- Sobel operators
  - Have directional sensitivity
  - Coefficients sum to zero
    - Zero response in constant intensity region

−1	−2	−1
0	0	0
1	2	1

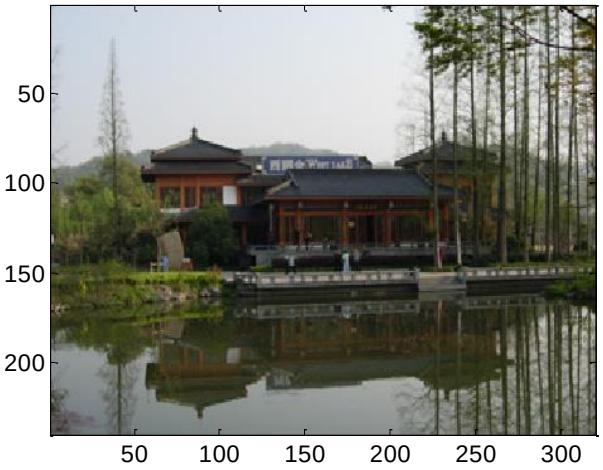
G_y

−1	0	1
−2	0	2
−1	0	1

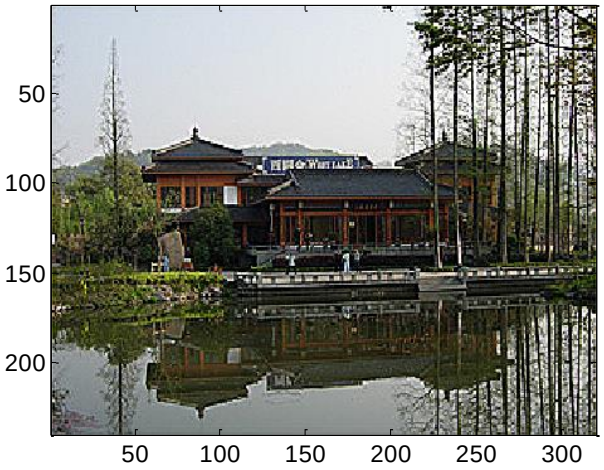
G_x

Highpass Examples

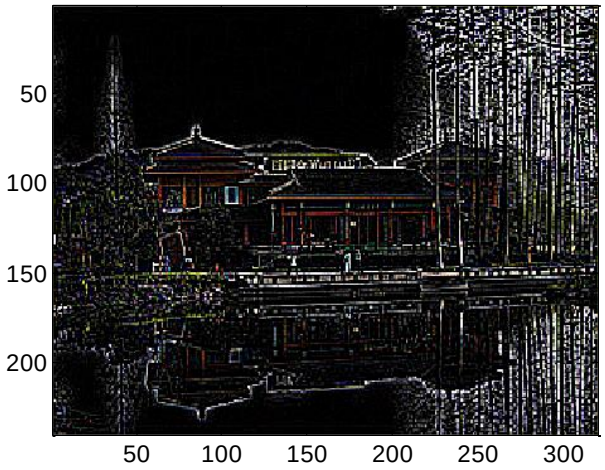
Origianl JPEG Image



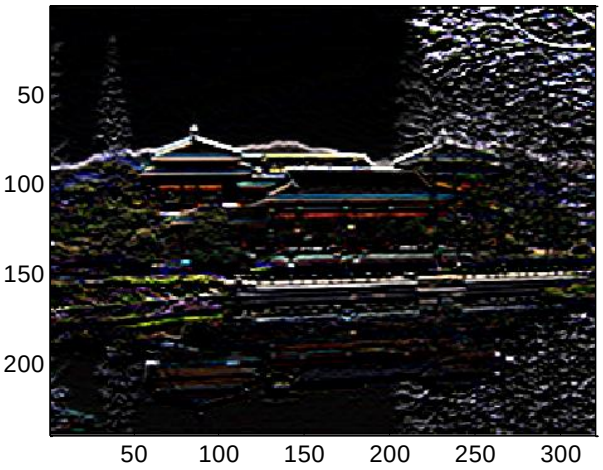
Highpass Filtered Image



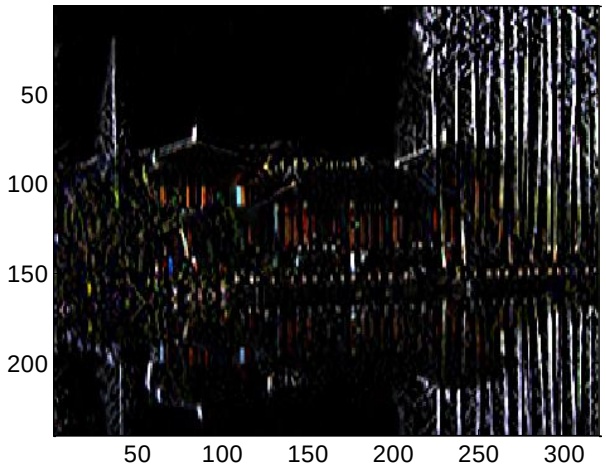
Edge Filtered Image



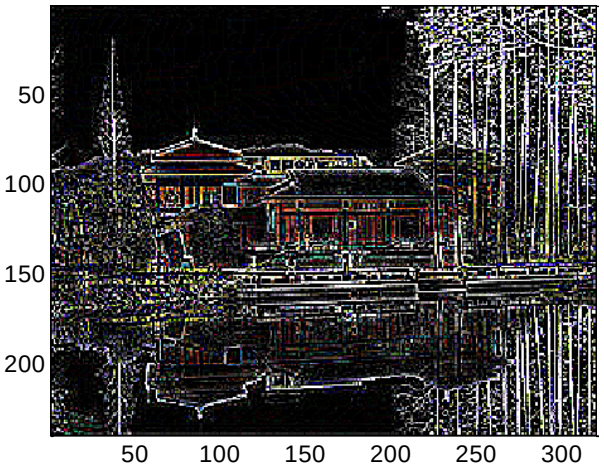
Sobel H-Filtered Image



Prewitt V-Filtered Image



Laplacian Filtered Image



Border Effects

- The filtering process suffers from boundary effects
  - What should happen at the edge of an image?
  - No values exist outside of image
- Padding extends image values outside of the image to “fill” the kernel at the borders
  - Zero – set pixels to 0 value
    - Will cause a darkening of the edges of the image
  - Constant – set border pixels to fixed value
  - Clamp – repeat edge pixel value
  - Mirror – reflect pixels across image edge

Computational Requirements

- Convolution requires K^2 operations per pixel for a $K \times K$ size filter
- Total operations on an image is $M \times N \times K^2$

- This can be computationally expensive for large K
- Cost can be greatly improved if the kernel is separable
  - First do 1D horizontal convolution
  - Follow with 2D vertical convolution

- Separable kernel
  - $w = vh^T$
    - v – vertical kernel
    - h - horizontal kernel
  - Defined by outer product

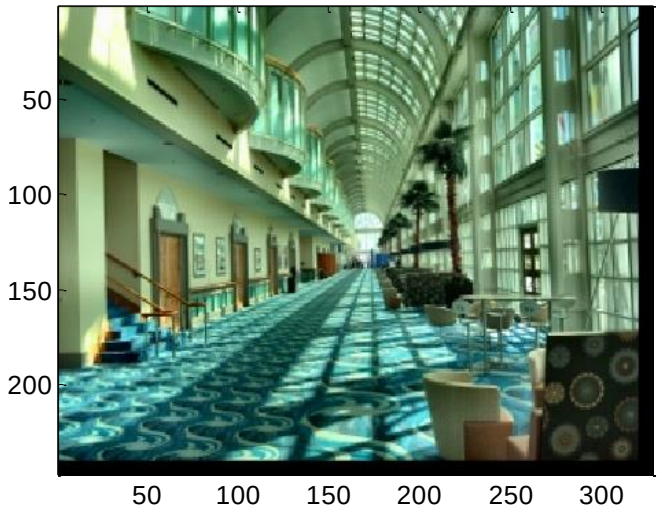
- Can approximate a separable kernel using singular value decomposition (SVD)
  - Truly separable kernels will only have one non-zero singular value

Fast Convolution

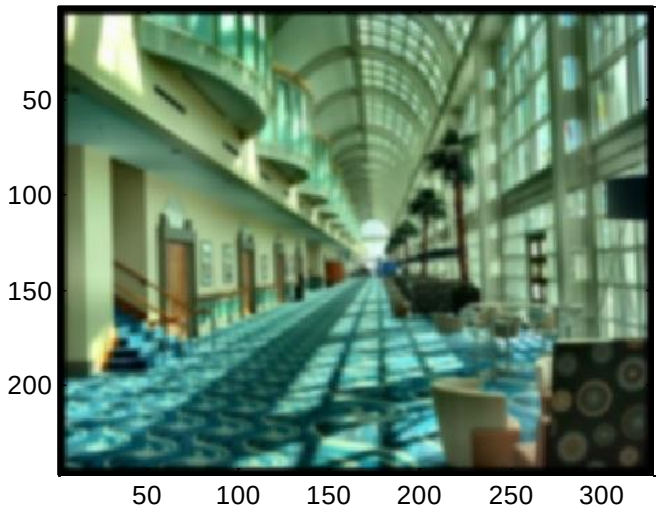
- Computationally efficient linear filtering by using the 2D FFT for large kernels
  - Avoid large nested loops – instead only have multiplication in frequency domain
    - $O(\log_2 NJ)$ instead of $O(NJ)$ term
  - Use `fft2.m` and `ifft2.m`
- Steps:
  - Pad both image and kernel with zeros to same size
    - Image + kernel size
  - Compute 2D FFT of both image and kernel
  - Multiply element-wise
  - Inverse FFT for result
    - Crop to get usable image

Fast Convolution Examples

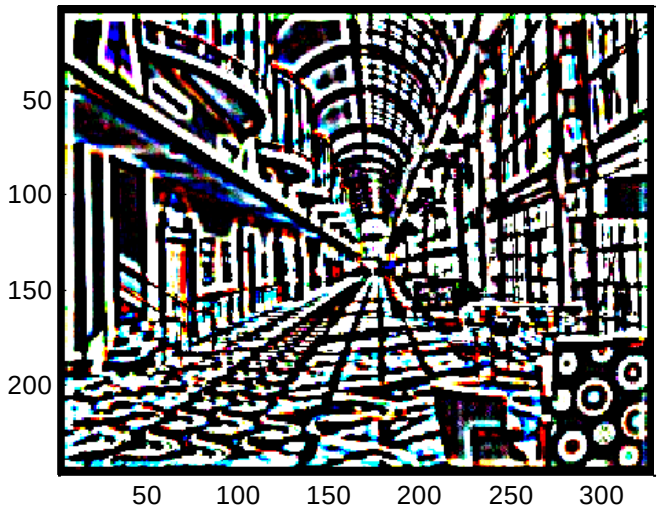
Origianl JPEG Image



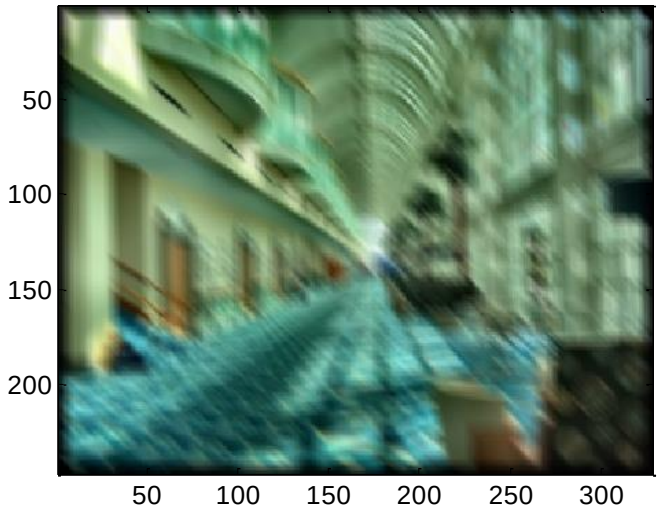
Gaussian Filtered Image



edge Filtered Image



Motion Filtered Image



Discrete Cosine Transform for Coding

- DCT is widely used in image compression
  - Part of JPEG standard

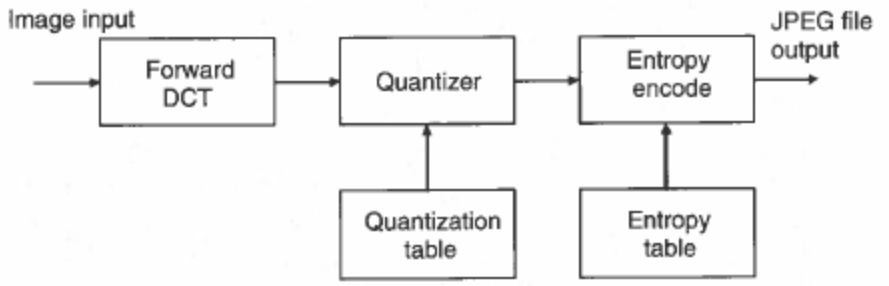


Figure 11.10 Block diagram of baseline JPEG encoder

  - Process image in 8 × 8 blocks
- JPEG2000 improves compression and removes block artifacts using wavelet transform
  - Never really caught on

- DCT definitions

$$X(k,l) = \frac{2}{N} C(k)C(l) \sum_{m=0}^{N-1} \sum_{n=0}^{N-1} x(m,n) \cos\left[\frac{(2n+1)l\pi}{2N}\right] \cos\left[\frac{(2m+1)k\pi}{2N}\right],$$

$$x(m,n) = \frac{2}{N} \sum_{k=0}^{N-1} \sum_{l=0}^{N-1} C(k)C(l)X(k,l) \cos\left[\frac{(2n+1)l\pi}{2N}\right] \cos\left[\frac{(2m+1)k\pi}{2N}\right],$$

$$C(k) = C(l) = \begin{cases} \sqrt{2}/2, & \text{if } k = l = 0 \\ 1, & \text{otherwise.} \end{cases}$$

- DCT is separable
  - Horizontal (column-wise) and vertical (row-wise)

$$X(k) = \frac{1}{2} C(k) \sum_{m=0}^{7} x(m) \cos\left[\frac{(2m+1)k\pi}{16}\right],$$

$$x(m) = \frac{1}{2} \sum_{k=0}^{7} C(k)X(k) \cos\left[\frac{(2m+1)k\pi}{16}\right],$$

  - Significant computation reduction (1D operations)

JPEG Coding Example

- DCT coefficients are ordered in zig-zag fashion
  - DC component first (only code difference between blocks)
  - AC coefficients have lower weight in higher-order
    - Compaction property (only code non-zero coefficients)

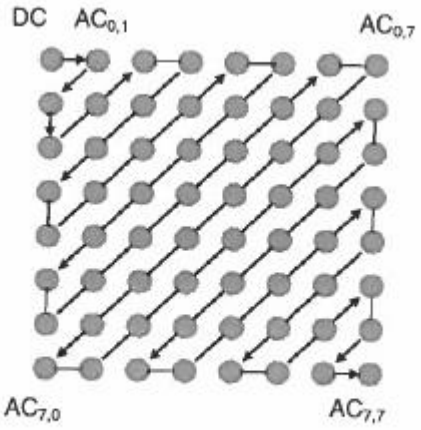


Figure 11.11 Ordering of DCT coefficients in zigzag fashion

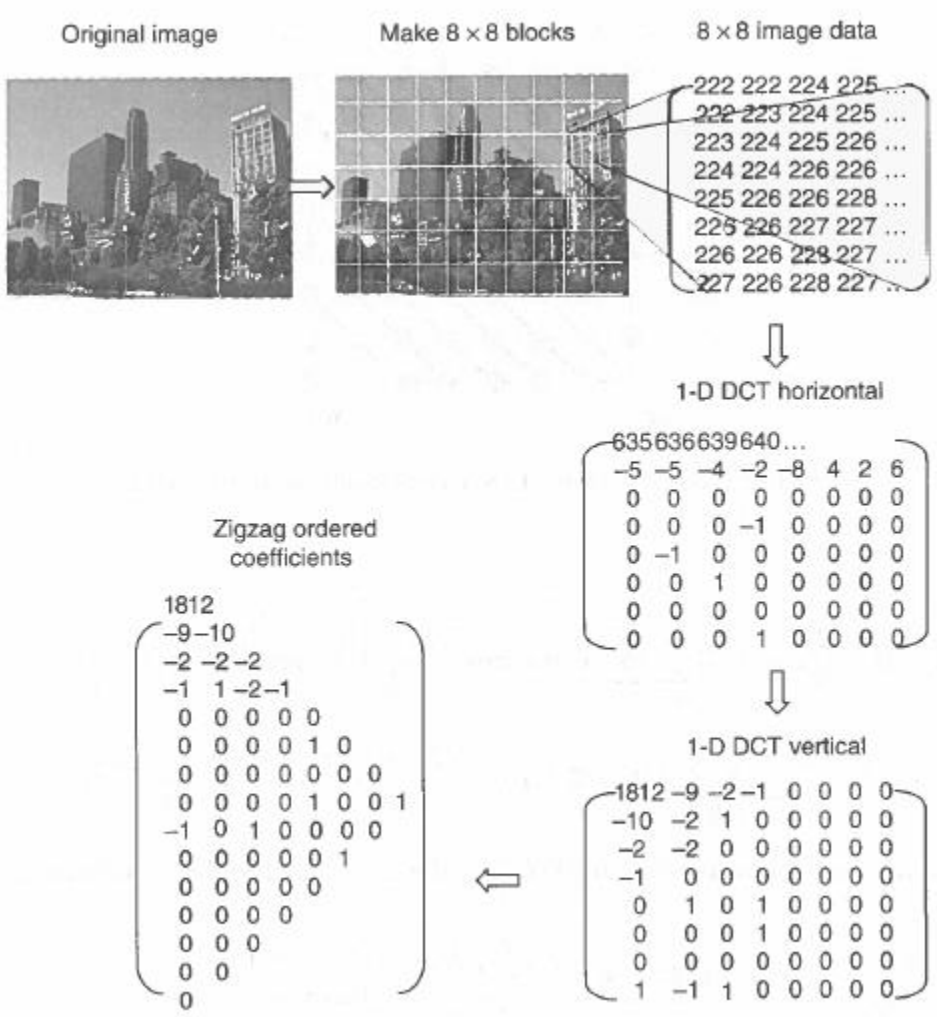


Figure 11.12 DCT block transform and reorder in JPEG image coding process