EE795: Computer Vision and Intelligent Systems

Spring 2012
TTh 17:30-18:45 WRI C225

Lecture 04
130131

http://www.ee.unlv.edu/~b1morris/ecg795/

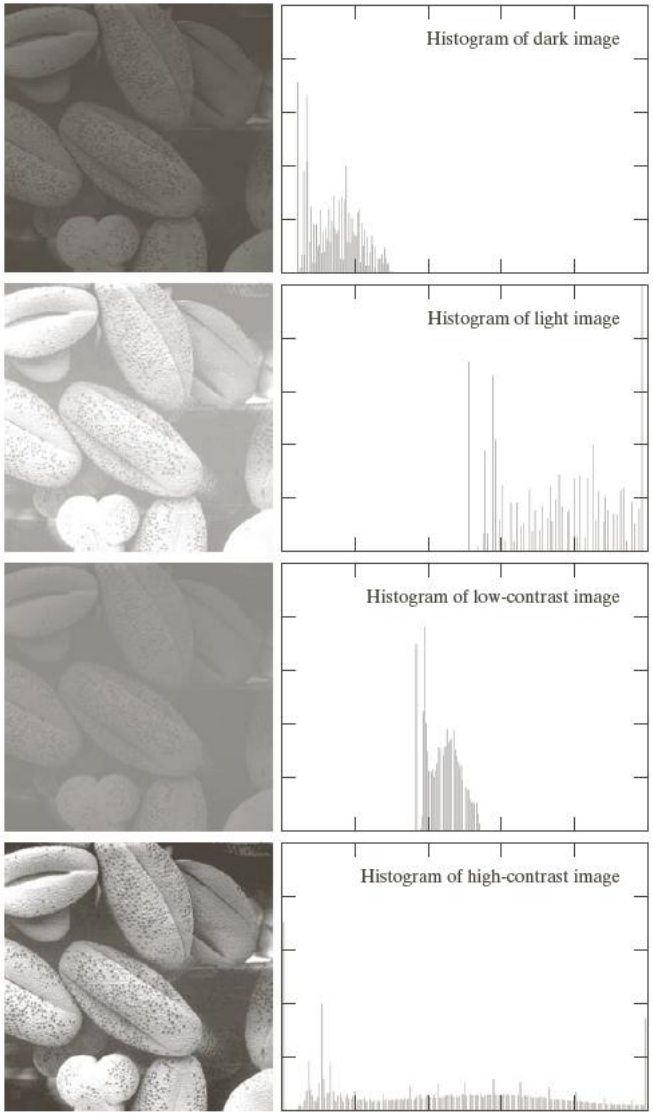
Outline

- Review
  - Histogram Equalization
- Image Filtering
- Linear Filtering
- Morphology
- Connected Components

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 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

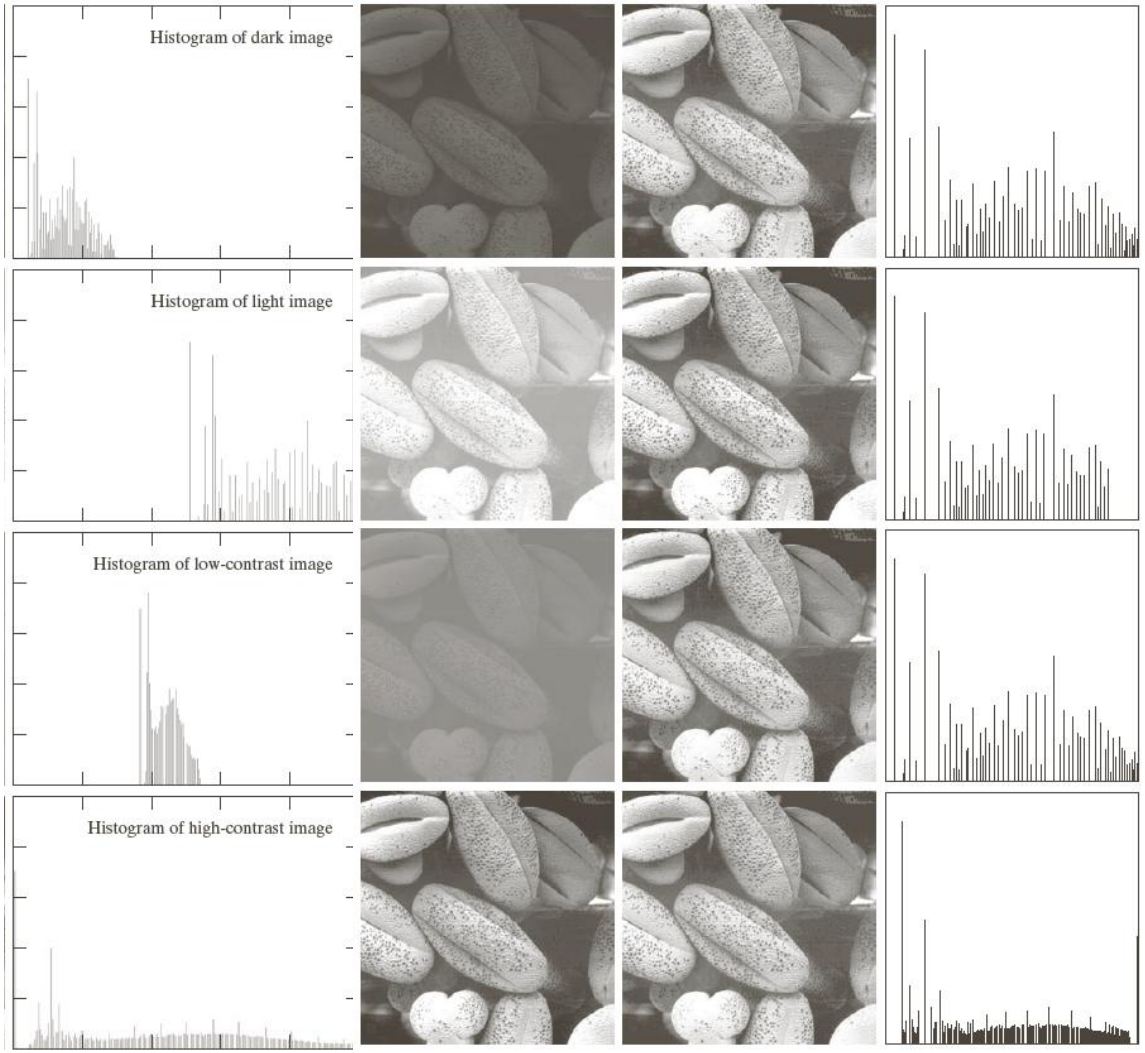
Histogram Equalization

- 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



Original histogram | original image | histogram equalized | equalized image

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

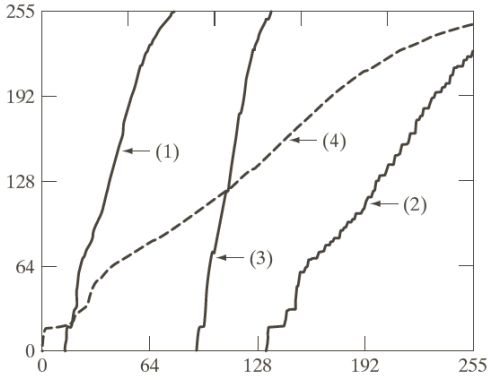


FIGURE 3.21 Transformation functions for histogram equalization. Transformations (1) through (4) were obtained from the histograms of the images (from top to bottom) in the left column of Fig. 3.20 using Eq. (3.3-8).

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

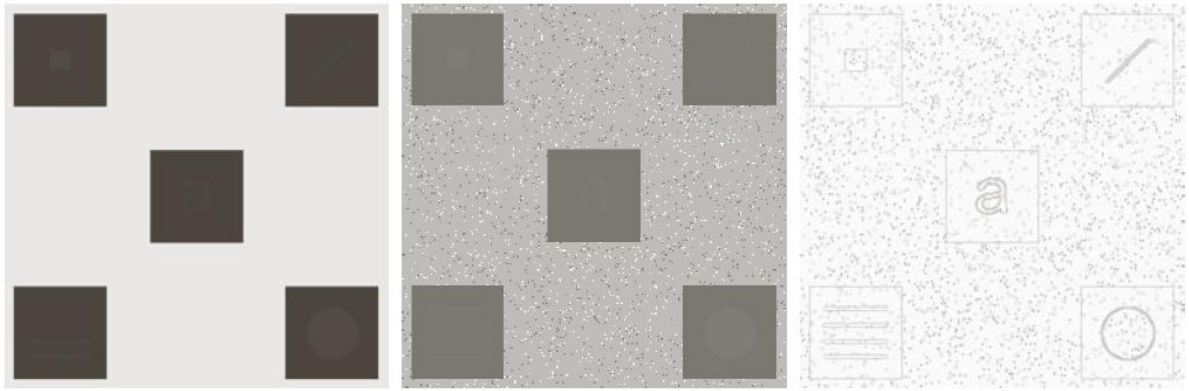
- Original image

- 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



a b c

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

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
  - Replace a pixel by the average value in a neighborhood

- 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

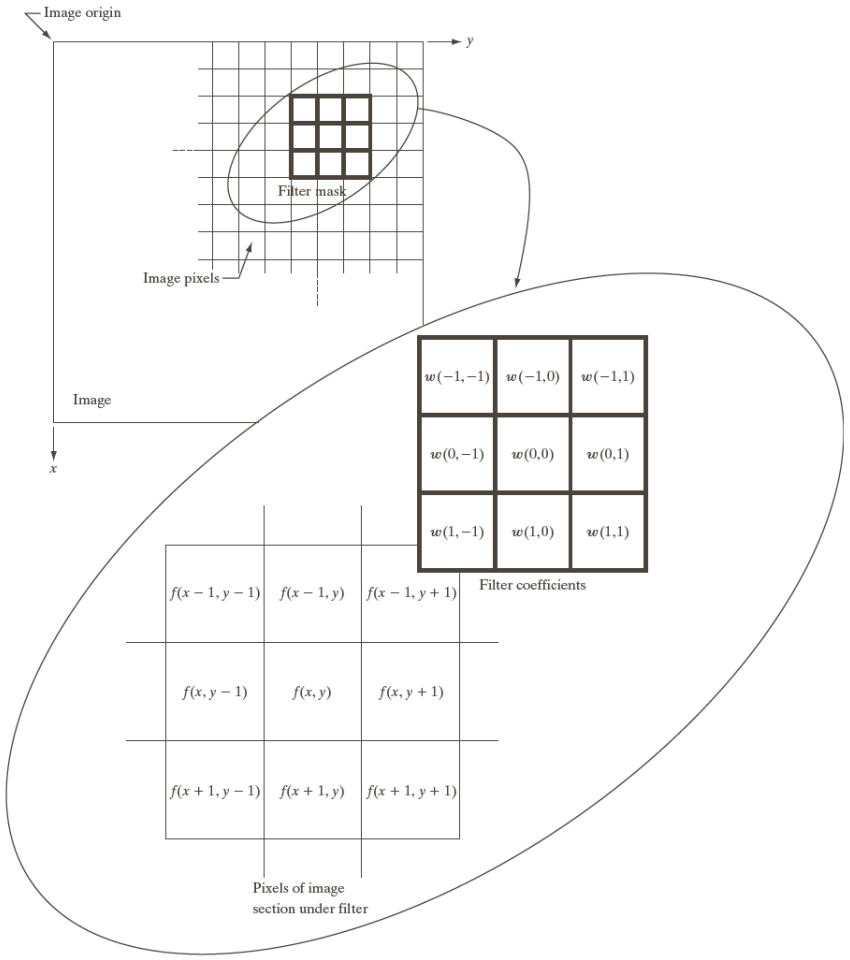
Adapted from S. Seitz

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
  - $g = f \otimes w$
- Linear filtering (correlation) is the same as convolution from signal processing classes
  - Convolution you flip your kernel but in correlation there is no flip

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 Process

$$g(x,y) = \sum_{k,l} f(x+k, y+l)w(k,l)$$

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f(x,y)$

$g(x,y)$

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

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
  - This is a separable filter

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

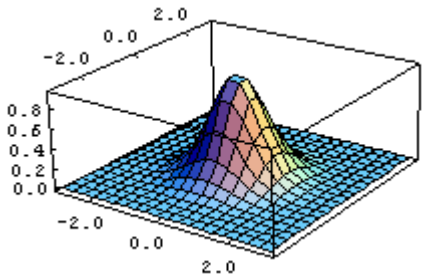
$$\frac{1}{K}\begin{bmatrix} 1 & 1 & \cdots & 1 \end{bmatrix}$$

- 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"

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



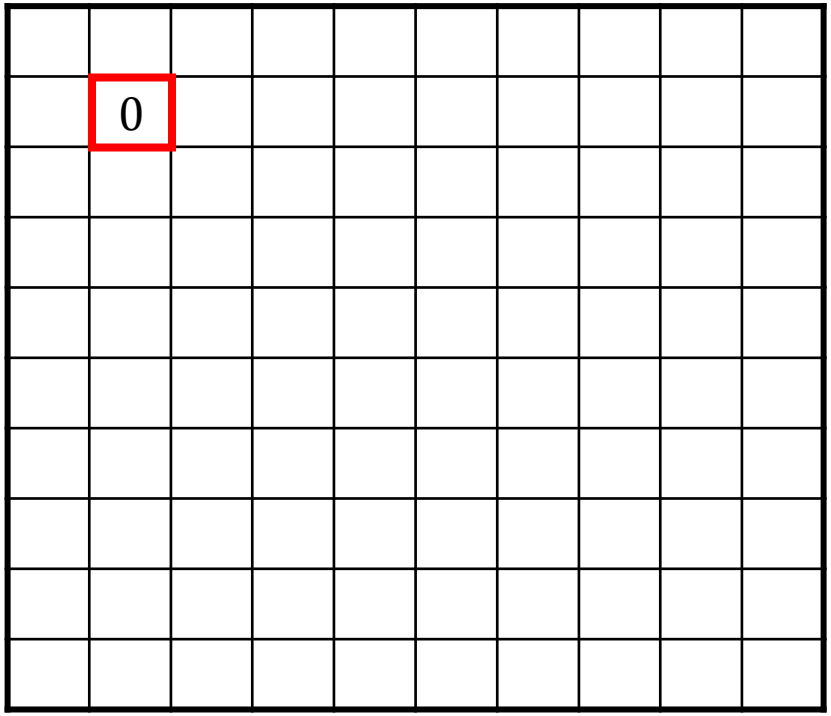
Mean Filtering

$$g(x,y) = \sum_{k,l} f(x+k, y+l)w(k,l)$$

$w(x,y)$- mean of 3×3 box

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f(x,y)$



$g(x,y)$

Mean Filtering

$$g(x,y) = \sum_{k,l} f(x+k, y+l)w(k,l)$$

$w(x,y)$- mean of 3×3 box

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f(x,y)$

	0	10							

$g(x,y)$

Mean Filtering

$$g(x,y) = \sum_{k,l} f(x+k, y+l)w(k,l)$$

$w(x,y)$- mean of 3×3 box

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f(x,y)$

	0	10							
				80					

$g(x,y)$

Mean Filtering

$$g(x,y) = \sum_{k,l} f(x+k, y+l)w(k,l)$$

$w(x,y)$- mean of 3×3 box

0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f(x,y)$

	0	10							
				80					
		10							

$g(x,y)$

Mean Filtering

$$g(x,y) = \sum_{k,l} f(x+k, y+l)w(k,l)$$

$w(x,y)$- mean of 3×3 box

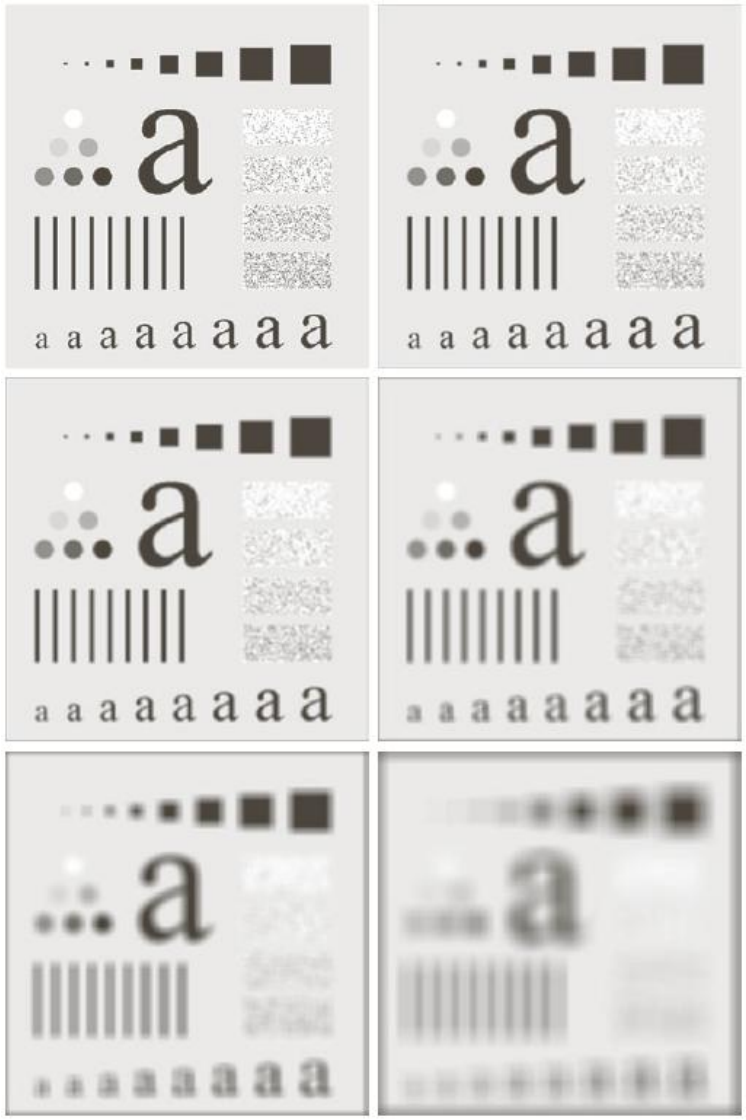
0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

$f(x,y)$

	0	10	20	30	30	30	20	10	
	0	20	40	60	60	60	40	20	
	0	30	60	90	90	90	60	30	
	0	30	50	80	80	90	60	30	
	0	30	50	80	80	90	60	30	
	0	20	30	50	50	60	40	20	
	10	20	30	30	30	30	20	10	
	10	10	10	0	0	0	0	0	

$g(x,y)$

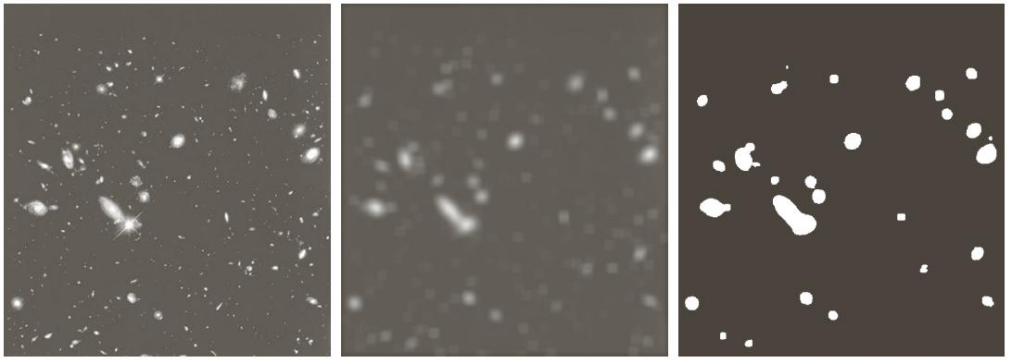
Smoothing Examples



a b
c d
e f

FIGURE 3.33 (a) Original image, of size 500 × 500 pixels. (b)–(f) Results of smoothing with square averaging filter masks of sizes $m = 3, 5, 9, 15$, and 35, respectively. The black squares at the top are of sizes 3, 5, 9, 15, 25, 35, 45, and 55 pixels, respectively; their borders are 25 pixels apart. The letters at the bottom range in size from 10 to 24 points, in increments of 2 points; the large letter at the top is 60 points. The vertical bars are 5 pixels wide and 100 pixels high; their separation is 20 pixels. The diameter of the circles is 25 pixels, and their borders are 15 pixels apart; their intensity levels range from 0% to 100% black in increments of 20%. The background of the image is 10% black. The noisy rectangles are of size 50 × 120 pixels.

Object detection

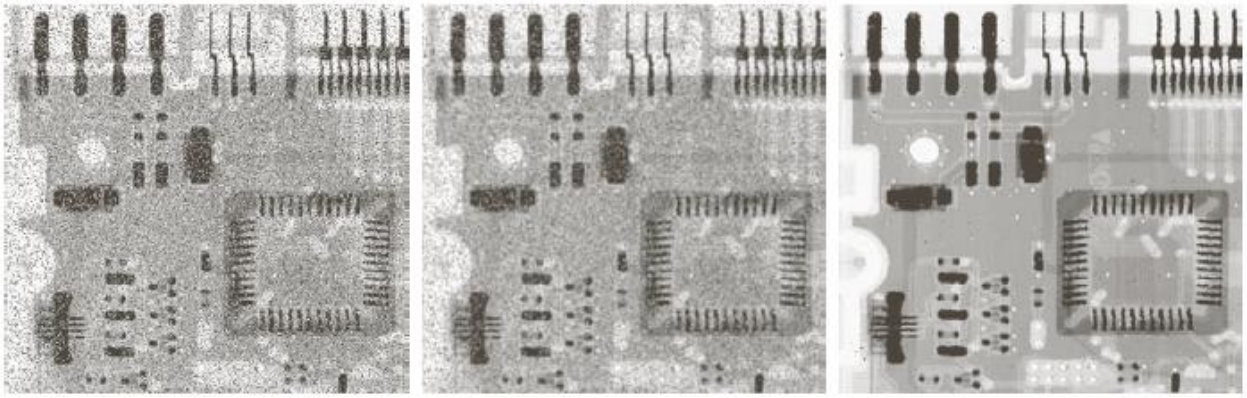


a b c

FIGURE 3.34 (a) Image of size 528 × 485 pixels from the Hubble Space Telescope. (b) Image filtered with a 15 × 15 averaging mask. (c) Result of thresholding (b). (Original image courtesy of NASA.)

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.)

Bilateral Filtering

- Combine the idea of a weighted filter kernel with a better version of outlier rejection
  - α-trimmed mean calculates average in neighborhood excluding the α fraction that are smallest or largest
- $w(i,j,k,l) = d(i,j,k,l) \times r(i,j,k,l)$
  - $d(i,j,k,l)$ - domain kernel specifies "distance" similarity between pixels (usually Gauassian)
  - $r(i,j,k,l)$ – range kernel specifies "appearance (intensity)" similarity between pixels

Bilateral Filtering Example

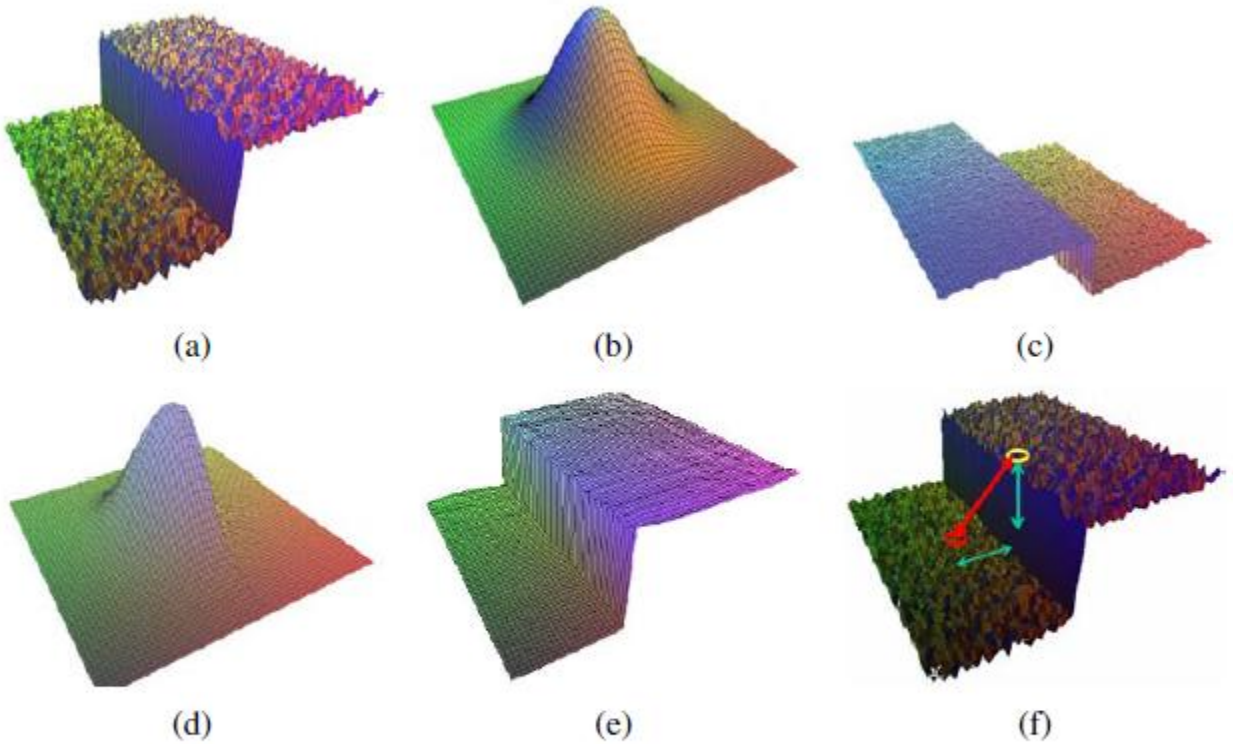


Figure 3.20 Bilateral filtering (Durand and Dorsey 2002) © 2002 ACM: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.

Sharpening Filters

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

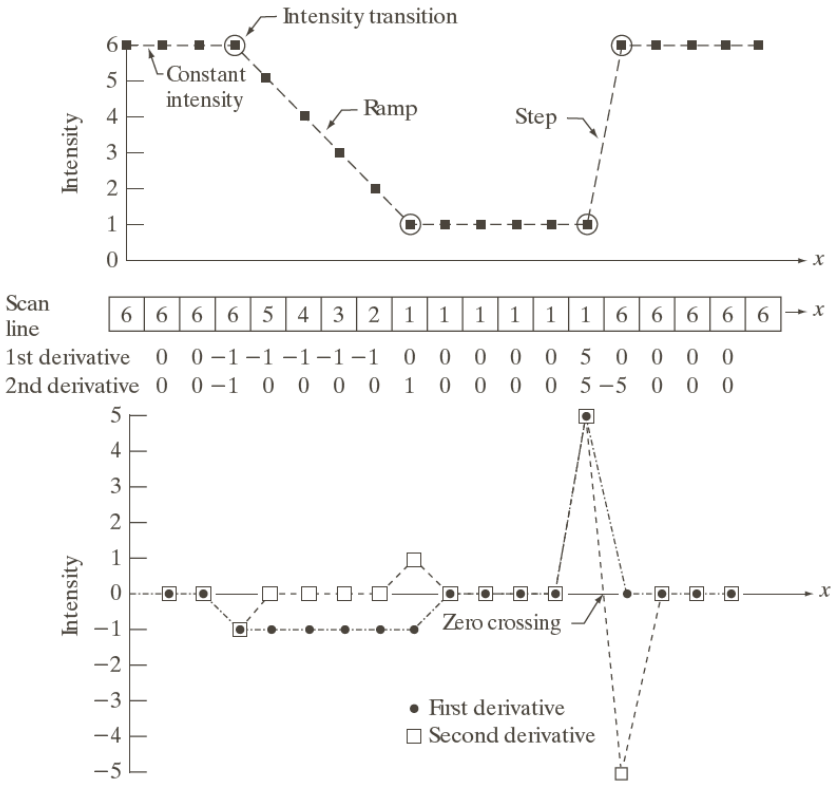
- Smoothing we saw was averaging
  - This is analogous to integration
- Since sharpening is the dual operation to smoothing, it can be accomplished through differentiation

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
  - $\frac{\partial^2 f}{\partial x^2} = f(x+1, y) + f(x-1, y) - 2f(x, y)$
  - $\frac{\partial^2 f}{\partial x^2} = f(x, y+1) + f(x, y-1) - 2f(x, y)$
  - $\nabla^2 f = [f(x+1, y) + f(x-1, y) + f(x, y+1) + f(x, y-1)] - 4f(x, y)$

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

0	1	0
1	−4	1
0	1	0

1	1	1
1	−8	1
1	1	1

0	−1	0
−1	4	−1
0	−1	0

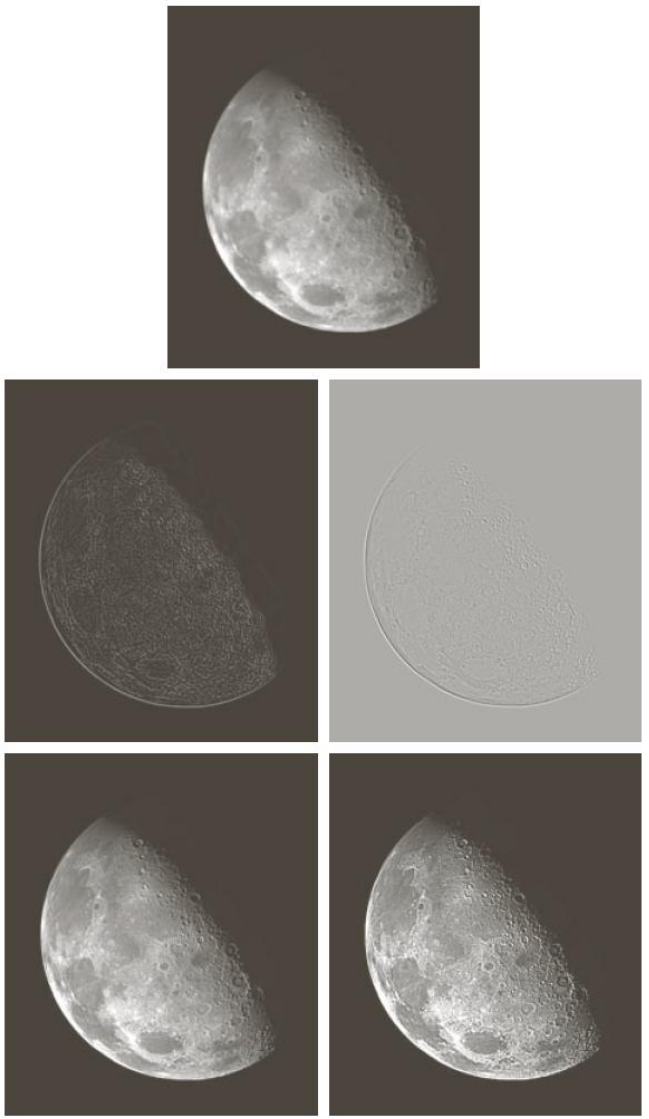
−1	−1	−1
−1	8	−1
−1	−1	−1

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

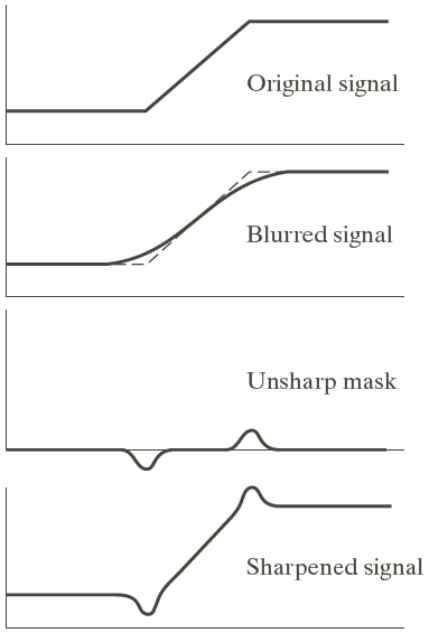


a
b c
d e

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

Morphological Image Processing

- Filtering done on binary images
  - Images with two values [0,1], [0, 255], [black,white]
  - Typically, this image will be obtained by thresholding
    - $g(x,y) = \begin{cases} 1 & f(x,y) > T \\ 0 & f(x,y) \leq T \end{cases}$
- Morphology is concerned with the structure and shape
- In morphology, a binary image is convolved with a structuring element s and results in a binary image

- See Chapter 9 of Gonzalez and Woods for a more complete treatment

Mathematical Morphology

- Tool for extracting image components that are useful in the representation and description of region shape
  - Boundaries, skeletons, convex hull, etc.
- The language of mathematical morphology is set theory
  - A set represents an object in an image
- This is often useful in video processing because of the simplicity of processing and emphasis on "objects"
  - Handy tool for "clean up" of a thresholded image

Morphological Operations

- Threshold operation
  - $\theta(f,t) = \begin{cases} 1 & f \geq t \\ 0 & \text{else} \end{cases}$
- Structuring element
  - s – e.g. 3 x 3 box filter (1's indicate included pixels in the mask)
  - S – number of "on" pixels in s
- Count of 1s in a structuring element
  - $c = f \otimes s$
  - Correlation (filter) raster scan procedure
- Basic morphological operations can be extended to grayscale images
- Dilation
  - $\text{dilate}(f,s) = \theta(c,1)$
  - Grows (thickens) 1 locations
- Erosion
  - $\text{erode}(f,s) = \theta(c,S)$
  - Shrink (thins) 1 locations
- Opening
  - $\text{open}(f,s) = \text{dilate}(\text{erode}(f,s),s)$
  - Generally smooth the contour of an object, breaks narrow isthmuses, and eliminates thin protrusions
- Closing
  - $\text{close}(f,s) = \text{erode}(\text{dilate}(f,s),s)$
  - Generally smooth the contour of an object, fuses narrow breaks/separations, eliminates small holes, and fills gaps in a contour

Morphology Example

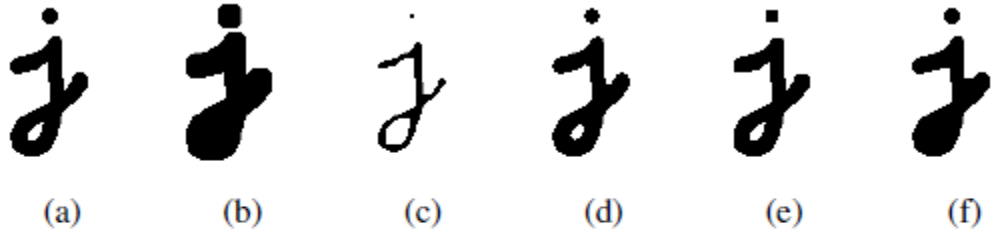


Figure 3.21 Binary image morphology: (a) original image; (b) dilation; (c) erosion; (d) majority; (e) opening; (f) closing. The structuring element for all examples is a 5×5 square. The effects of majority are a subtle rounding of sharp corners. Opening fails to eliminate the dot, since it is not wide enough.

- Dilation - grows (thickens) 1 locations
- Erosion - shrink (thins) 1 locations
- Opening - generally smooth the contour of an object, breaks narrow isthmuses, and eliminates thin protrusions
- Closing - generally smooth the contour of an object, fuses narrow breaks/separations, eliminates small holes, and fills gaps in a contour

Connected Components

- Semi-global image operation to provide consistent labels to similar regions
  - Based on adjacency concept
- Most efficient algorithms compute in two passes

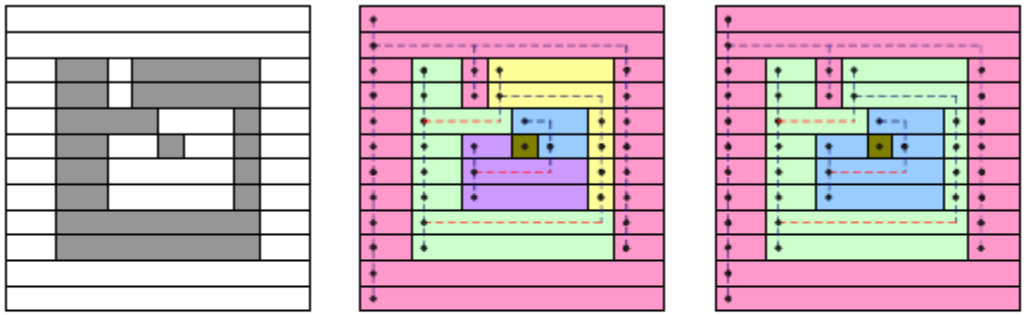


Figure 3.23 Connected component computation: (a) original grayscale image; (b) horizontal runs (nodes) connected by vertical (graph) edges (dashed blue)—runs are pseudocolored with unique colors inherited from parent nodes; (c) re-coloring after merging adjacent segments.

- More computational formulations (iterative) exist from morphology
  - $X_k = (X_{k-1} \oplus B) \cap A$

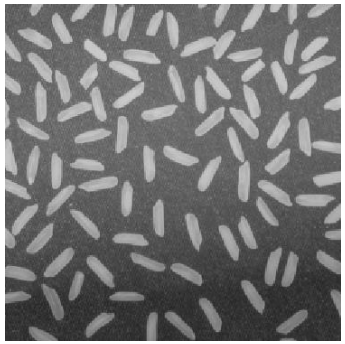
Connected component ↑ Structuring element ↑ Set ←

More Connected Components

- Typically, only the “white” pixels will be considered objects
  - Dark pixels are background and do not get counted
- After labeling connected components, statistics from each region can be computed
  - Statistics describe the region – e.g. area, centroid, perimeter, etc.

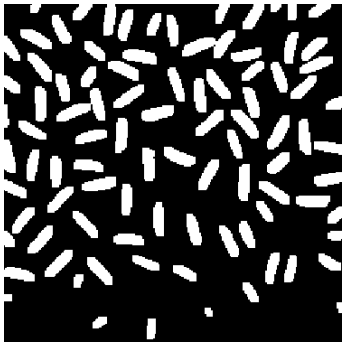
- Matlab functions
  - `bwconncomp.m, labelmatrix.m (bwlabel.m)`- label image
  - `label2rgb.m` – color components for viewing
  - `regionprops.m` – calculate region statistics

Connected Component Example

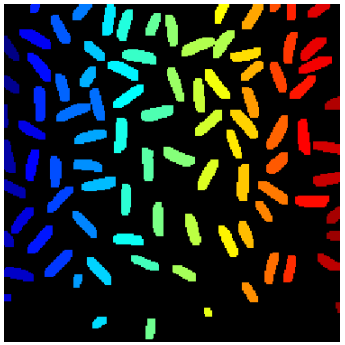


Grayscale image

Threshold image



Opened Image



Labeled image – 91 grains of rice