Digital Image Processing

Chap 3: Intensity Transformations & Spatial Filtering

清華大學電機系林嘉文
cwlin@ee.nthu.edu.tw
Tel: 03-5731152

Background

- Spatial domain process on image can be described as

$$g(x,y) = T[f(x,y)]$$

where $f(x,y)$ is the **input image**, $g(x,y)$ is the **output image**, T is an **operator**

- T operates on the neighbors of (x,y)(a square or rectangular sub-image centered at (x,y) to yield the output $g(x,y)$).

Background

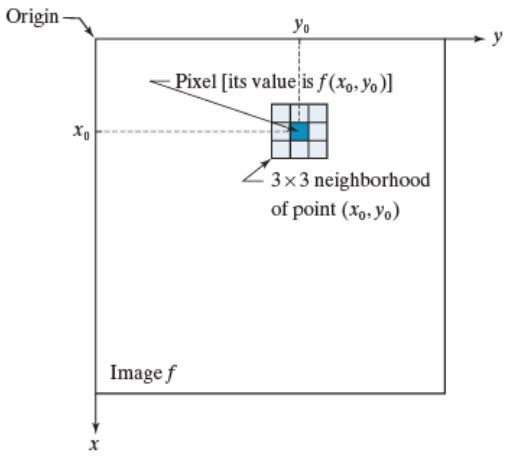


FIGURE 3.1 A 3×3 neighborhood about a point (x_0, y_0) in an image. The neighborhood is moved from pixel to pixel in the image to generate an output image. Recall from Chapter 2 that the value of a pixel at location (x_0, y_0) is $f(x_0, y_0)$, the value of the image at that location.

Background

- The simplest form of T is the neighborhood of size 1×1. g depends on the value of f at (x, y) which is a **gray level transformation** as $s = T(r)$

  where r and s are the gray-level of $f(x, y)$ and $g(x, y)$ at any point (x, y).

  Fig. 3.2(a) shows **Contrast stretching**

  Fig. 3.2(b) shows **Thresholding**.
- Enhancement of any point depends on that point only
  - **Point processing**
- Larger neighborhood provides more flexibility
  - **Mask processing or filtering**

Transformation for Contrast Enhancement

a b

FIGURE 3.2 Intensity transformation functions. (a) Contrast-stretching function. (b) Thresholding function.

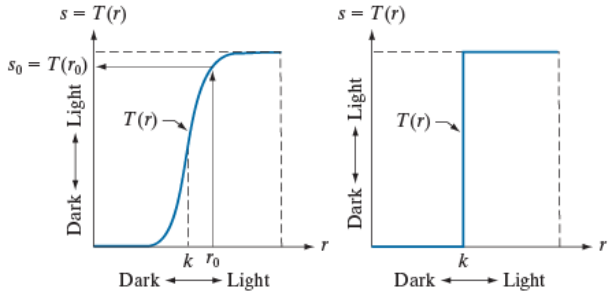


Image Enhancement

- Three basic functions used in image enhancement
  - Linear (negative and identity transformation)

    $s = L - 1 - r$
  - Logarithmic (log and inverse log)

    $s = c\log(1 + r)$
  - Power law (nth power and nth root transformation)

    $s = cr^{\gamma}$ or $s = c(r + \varepsilon)^{\gamma}$

Image Enhancement

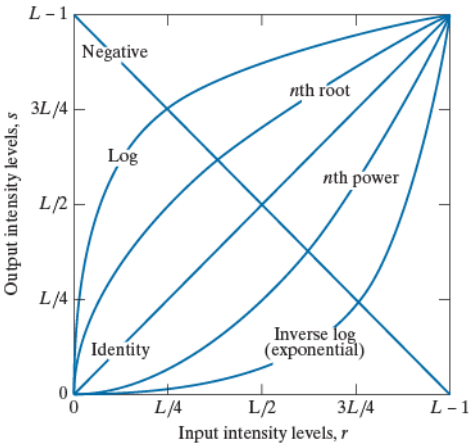
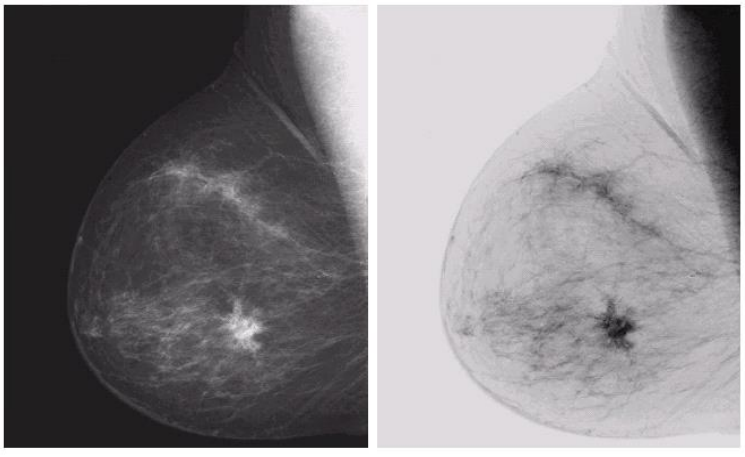


FIGURE 3.3 Some basic intensity transformation functions. Each curve was scaled *independently* so that all curves would fit in the same graph. Our interest here is on the *shapes* of the curves, not on their relative values.

Image Negatives

$$s = L–1– r$$



a b

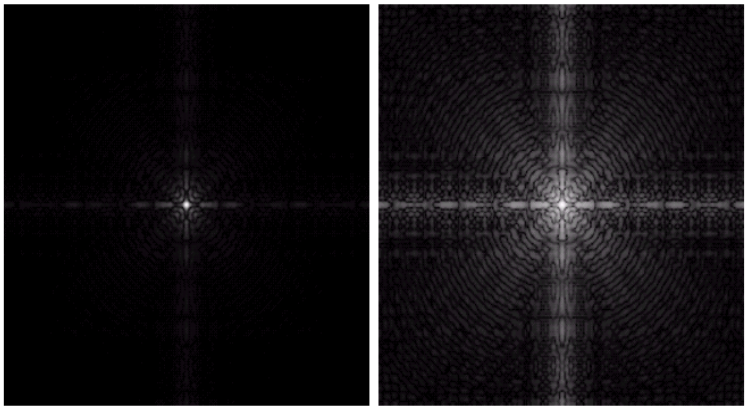
FIGURE 3.4 (a) Original digital mammogram. (b) Negative image obtained using the negative transformation in Eq. (3.2-1). (Courtesy of G.E. Medical Systems.)

Image Enhancement

$$s = c\log(1 + r)$$

a b

FIGURE 3.5
(a) Fourier spectrum.
(b) Result of applying the log transformation given in Eq. (3.2-2) with $c = 1$.



Power Law (Gamma) Transformation

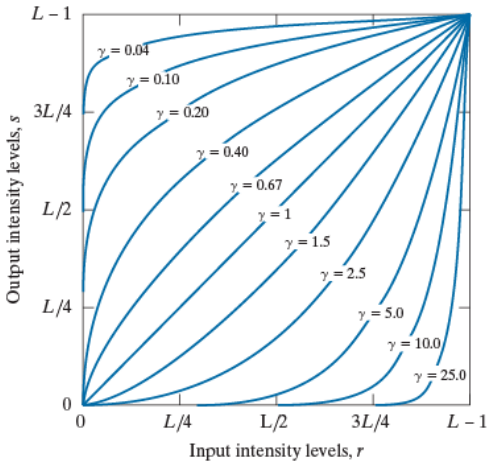


FIGURE 3.6
Plots of the gamma equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases). Each curve was scaled *independently* so that all curves would fit in the same graph. Our interest here is on the *shapes* of the curves, not on their relative values.

$$s = cr^{\gamma}$$

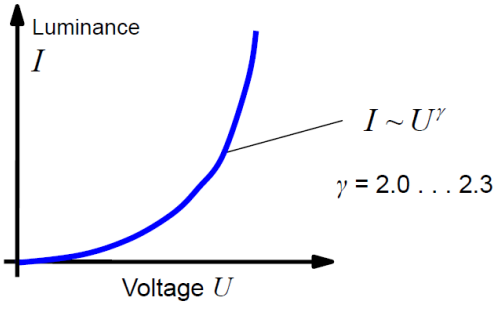
Gamma Correction

- **γ correction**
  - The CRT devices have an intensity-to-voltage response which is a **power function**.
  - γ ranges from 1.8 to 2.5
  - Without γ correction, the monitor output will become darker than the original input
  - Prepare the input image before inputting it into the CRT monitor by performing the transforming

$$s = r^{1/\gamma} = r^{1/2.5} = r^{0.4}$$

Gamma Correction

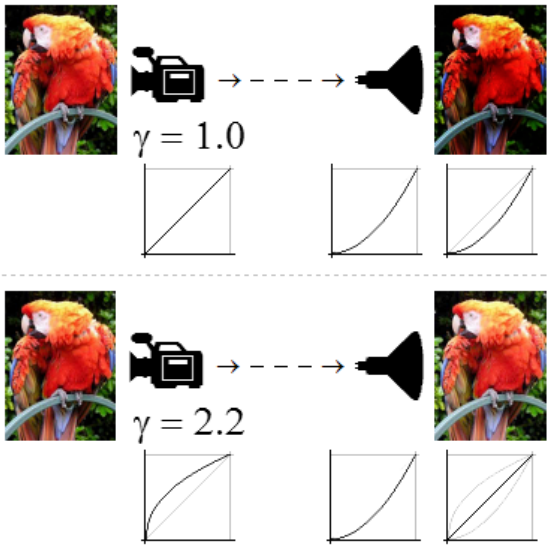
- CRTs are non-linear



- Cameras contain γ–predistortion circuit

$$U \sim I^{1/\gamma}$$

Gamma Correction



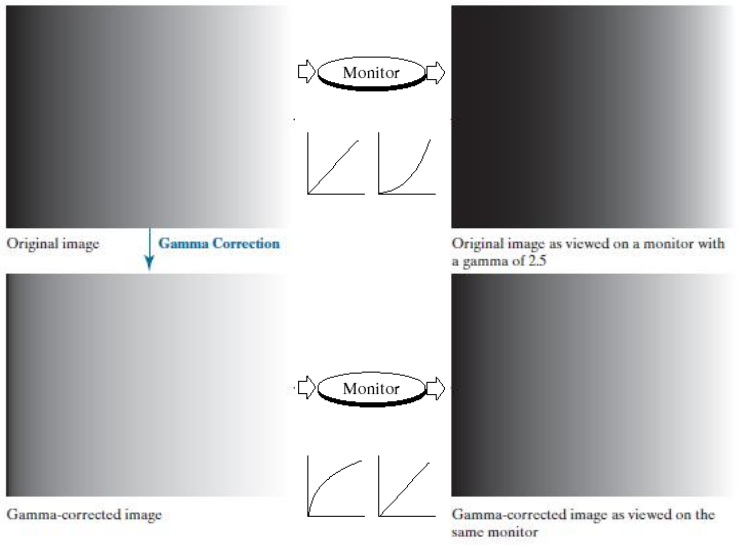
Linear

Gamma Corrected

Gamma Correction

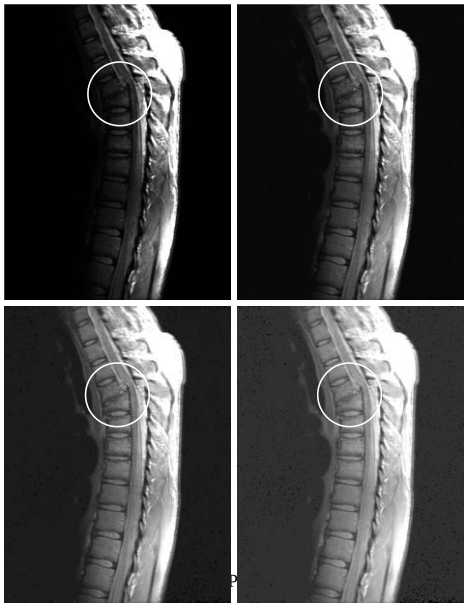
a b
c d

FIGURE 3.7
(a) Intensity ramp image. (b) Image as viewed on a simulated monitor with a gamma of 2.5. (c) Gamma-corrected image. (d) Corrected image as viewed on the same monitor. Compare (d) and (a).



Gamma Correction

$s = cr^{\gamma}$



a b
c d

FIGURE 3.8
(a) Magnetic resonance image (MRI) of a fractured human spine (the region of the fracture is enclosed by the circle). (b)–(d) Results of applying the transformation in Eq. (3-5) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3, respectively. (Original image courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

15

a b
c d

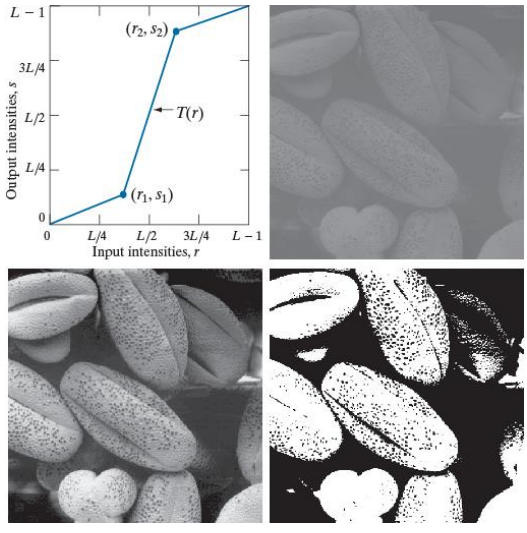
FIGURE 3.9
(a) Aerial image. (b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 3.0, 4.0$, and 5.0, respectively. (Original image for this example courtesy of NASA.)

$s = cr^{\gamma}$

Gray-Level Transformations

- Piecewise-Linear Transformation Functions
  - Contrast stretching
  - Gray-level slicing
  - Bit-plane slicing

Contrast Stretching



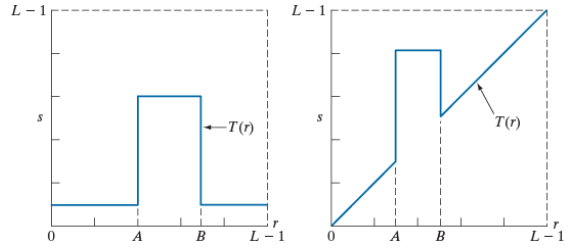
a b
c d

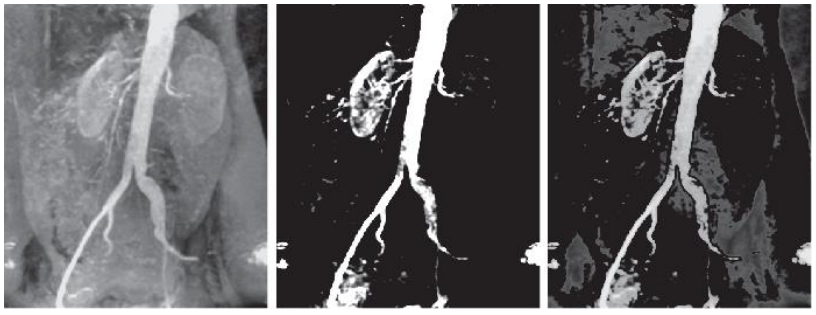
FIGURE 3.10 Contrast stretching. (a) Piecewise linear transformation function. (b) A low-contrast electron microscope image of pollen, magnified 700 times. (c) Result of contrast stretching. (d) Result of thresholding. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra, Australia.)

Intensity-Level Slicing

a b

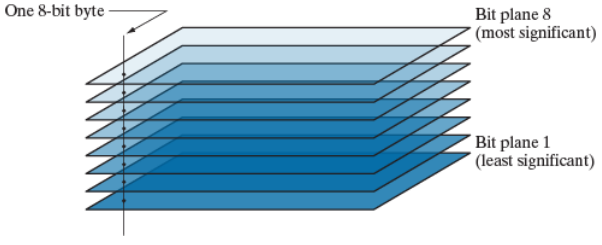
FIGURE 3.11
(a) This transformation function highlights range $[A, B]$ and reduces all other intensities to a lower level. (b) This function highlights range $[A, B]$ and leaves other intensities unchanged.



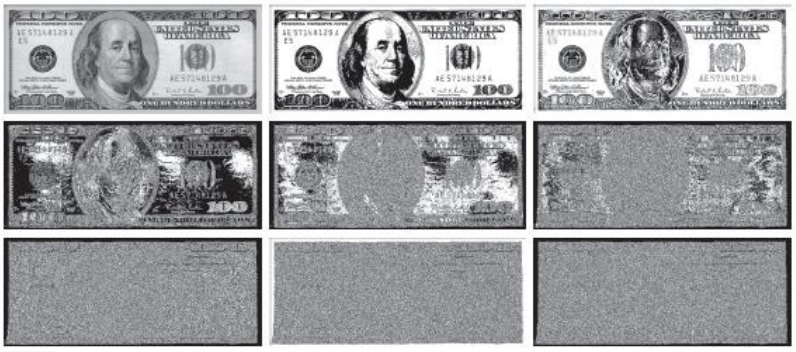


Bit-Plane Representation

FIGURE 3.13
Bit-planes of an 8-bit image.



Bit-Plane Slicing



a b c
d e f
g h i

FIGURE 3.14 (a) An 8-bit gray-scale image of size 550×1192 pixels. (b) through (i) Bit planes 8 through 1, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image..

a b c **FIGURE 3.15** Image reconstructed from bit planes: (a) 8 and 7; (b) 8, 7, and 6; (c) 8, 7, 6, and 5.

Histogram Processing

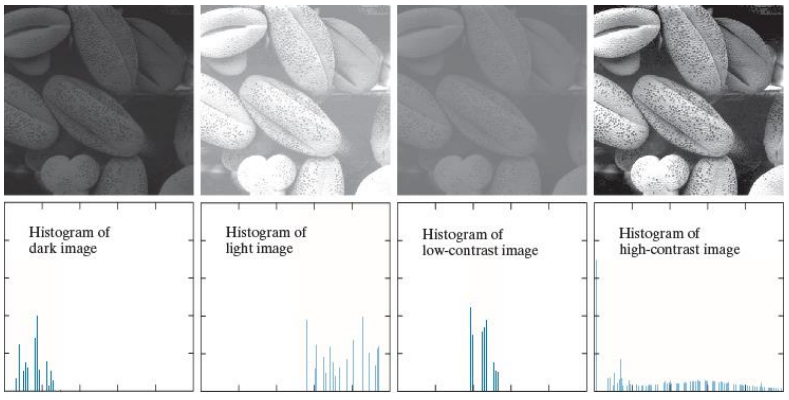
- The **histogram** of a digital image with gray-levels in the range [0, $L-1$] is a discrete function $h(r_k) = n_k$ or $p(r_k) = n_k/MN$,

  where r_k is the kth level and n_k is the number of pixels having the gray-level r_k.

- A **normalized histogram**, $p(r_k) = n_k/NM$,

  NM is the total number of pixels in the image.

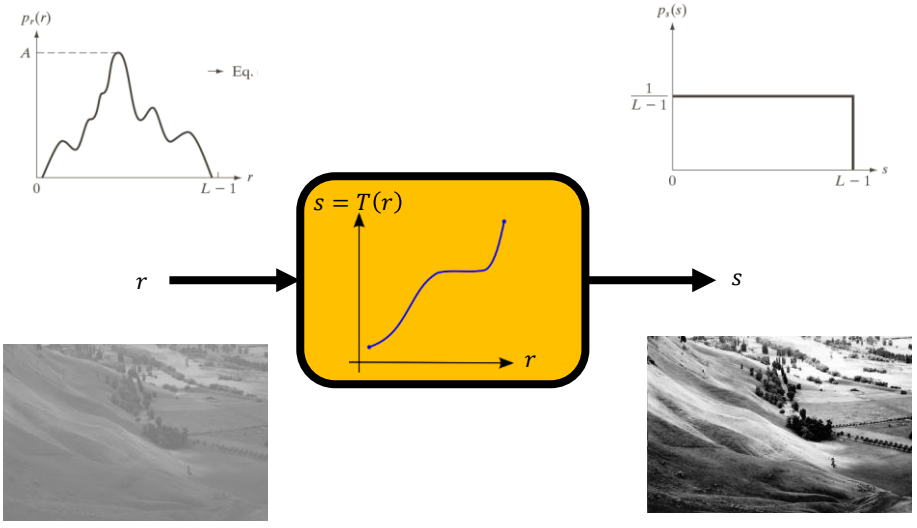
Histogram Processing



a b c d

FIGURE 3.16 Four image types and their corresponding histograms. (a) dark; (b) light; (c) low contrast; (d) high contrast. The horizontal axis of the histograms are values of r_k and the vertical axis are values of $p(r_k)$.

Histogram Equalization



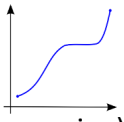
Histogram Equalization

Histogram equalization is to find a transformation $s = T(r)$, $0 \le r \le L-1$ that satisfies the following conditions:

- $T(r)$ is **(strictly) monotonically increasing** in the interval $0 \le r \le L-1$
- $0 \le T(r) \le L-1$ for $0 \le r \le L-1$
- $T(r)$ is one-to-one (if strictly monotonically increasing) so that its inverse function exists.
- The inverse transform from s to r is denoted as

$$r = T^{-1}(s) = Q(s), 0 \le s \le L-1$$

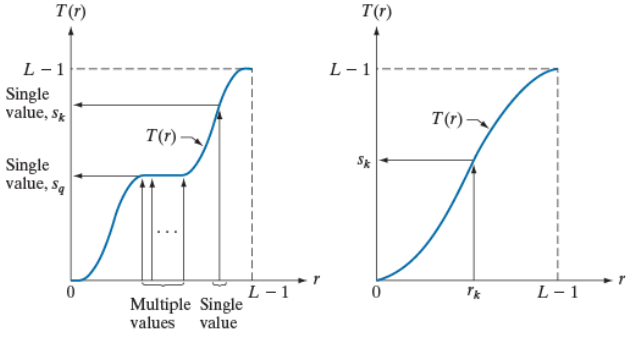
- If strict monotonicity is not satisfied, $s = Q^{-1}(r)$ may be ambiguous



Histogram Equalization

a b

FIGURE 3.17
(a) Monotonic increasing function, showing how multiple values can map to a single value. (b) Strictly monotonic increasing function. This is a one-to-one mapping, both ways.



Histogram Equalization

- The gray-level in an image may be viewed as a random variable, so we let $p_r(r)$ and $p_s(s)$ denote the probability density functions of random variables r and s.
- If $p_r(r)$ and $T(r)$ are known and $T(r)$ is continuous and differentiable then

$$p_s(s) = p_r(r)|dr/ds|$$

- If we set the transformation function as

$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$

where w is a dummy variable,
$s = T(r)$ is a **cumulative distribution function** (CDF) of the random variable r.

Histogram Equalization

- Given transformation $T(r)$, we may find $p_s(s)$ as

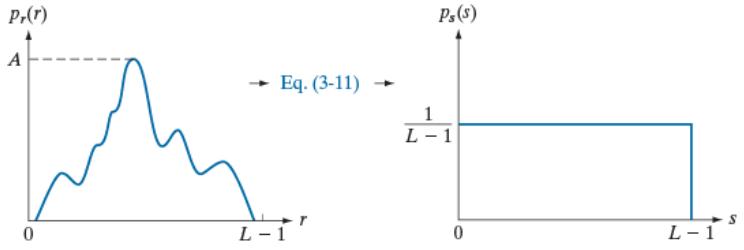
$$ds/dr = dT(r)/dr = (L-1)\ d\left[\int_0^r p_r(w)dw\right]/dr$$
$$= (L-1)p_r(r) \quad \text{(Leibniz's rule)}$$

and then $p_s(s) = p_r(r)|dr/ds|$
$= p_r(r)|1/(L-1)p_r(r))|$
$= 1/(L-1)$ for $0 \le s \le L-1$

- $p_s(s)$ is a **uniform prob. distribution**
- $T(r)$ depends on $p_r(r)$

Histogram Equalization

- $p_r(r) \rightarrow p_s(s)$



a b

FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying Eq. (3-11) to the input PDF. The resulting PDF is always uniform, independently of the shape of the input.

Histogram Equalization

- For discrete case (image $N \times M$)

$$p_r(r_k) = n_k / NM \text{ for } k = 0,1,\dots,L-1$$

- The discrete version of the transformation function is

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{L-1}{NM}\sum_{j=0}^{k} n_j$$

- The above mapping is called **histogram equalization**
- The inverse transform

$$r_k = T^{-1}(s_k) \text{ for } k = 0,1,\dots,L-1$$

It exists if none of the levels, r_k, $k = 0,1,\dots,L-1$ are missing from the input images

Histogram Equalization

Example 3.5

3-bit Gray Level, L= 8

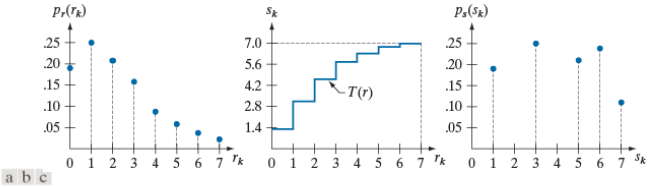
$$s_k = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j$$

s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
1.33	3.08	4.55	5.67	6.23	6.65	6.86	7.00
0→1	1→3	2→5	3→6	4→6	5→7	6→7	7→7

r_k	n_k	$p_r(r_k) = n_k/MN$
$r_0 = 0$	790	0.19
$r_1 = 1$	1023	0.25
$r_2 = 2$	850	0.21
$r_3 = 3$	656	0.16
$r_4 = 4$	329	0.08
$r_5 = 5$	245	0.06
$r_6 = 6$	122	0.03
$r_7 = 7$	81	0.02

TABLE 3.1 Intensity distribution and histogram values for a 3-bit, 64 × 64 digital image.

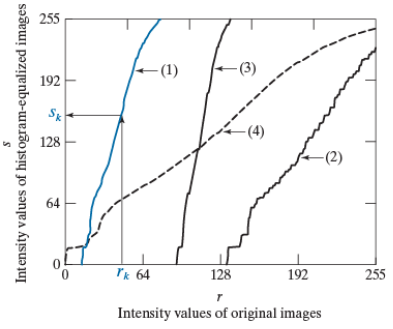
$r_k \rightarrow s_k$



a b c

FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

Histogram Equalization



Transformation functions

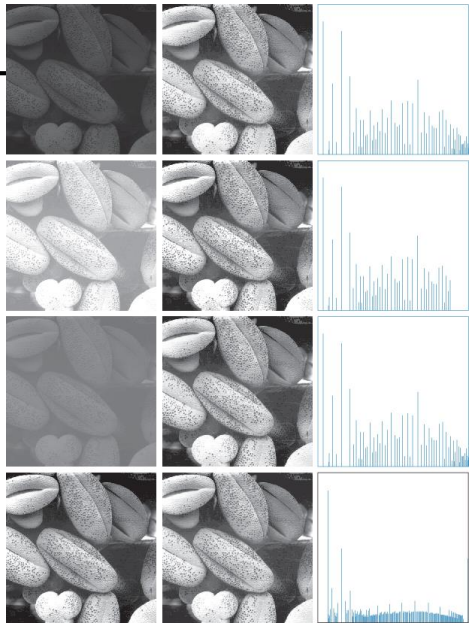
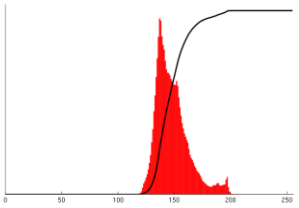
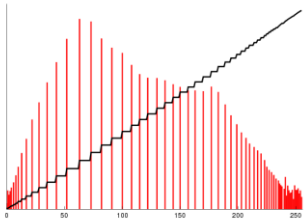


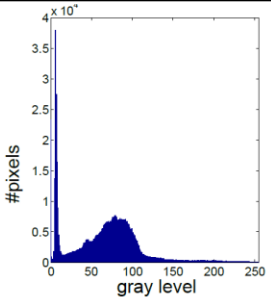
FIGURE 3.20 Left column: Images from Fig. 3.16. Center column: Corresponding histogram-equalized images. Right column: histograms of the images in the center column (compare with the histograms in Fig. 3.16).

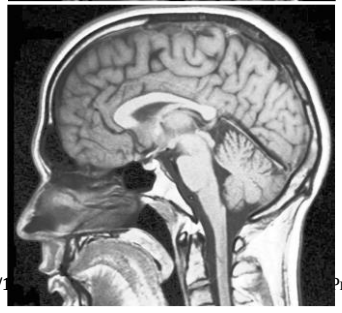
Histogram Equalization

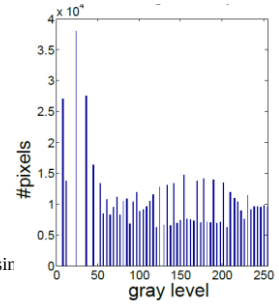




Histogram Equalization







Histogram Matching

- Given an input image with $p_r(r)$, and the specific output image with $p_z(z)$, find the transfer function between r & z.
- Let $s = T(r) = (L-1)\int_0^r p_r(w)dw$

  where w is a dummy variable
- Define a random variable z with the property

  $G(z) = (L-1)\int_0^z p_z(t)dt = s$

  where t is a dummy variable
- From the above equations $G(z) = T(r)$ we have

  $z = G^{-1}(s) = G^{-1}[T(r)]$

Histogram Matching

- **For discrete case:**
- From given histogram $p_r(r_k)$, $k = 0, 1, \ldots, L-1$

  $$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_r(r_j) = \frac{(L-1)}{MN}\sum_{j=0}^{k} n_j$$
- From given histogram $p_z(z_i)$, $i = 0, 1, \ldots, L-1$

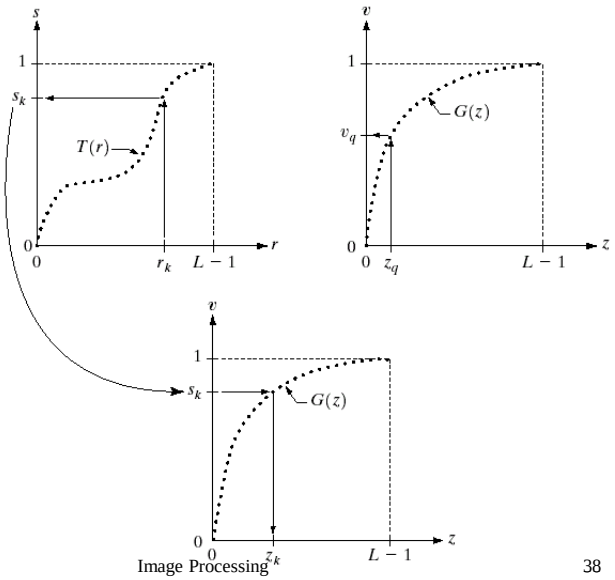
  $$v_q = G(z_q) = (L-1)\sum_{i=0}^{q} p_z(z_i)$$
- Finally, we have $G(z_q) = T(r_k)$ (i.e., $v_q = s_k$)

  and $z_q = G^{-1}(s_k) = G^{-1}[T(r_k)]$
- Notes: $r \xrightarrow{T} s$ $z \xrightarrow{G} v$ $s = v$, hence $r \xrightarrow{T} s = v \xrightarrow{G^{-1}} z$

Histogram Matching

1. Obtain the histogram of each given image
2. Pre-compute a mapped s_k for each r_k, i.e., $s_k = T(r_k)$
3. Obtain the transformation function G from given $p(z)$ using $v_k = G(z_k) = (L-1)\sum_{i=0}^{k} p_z(z_i) = s_k$
4. Precompute z_k for each value s_k using **iterative scheme** as follows:
   - To find $z_k = G^{-1}(s_k) = G^{-1}(v_k)$, however, there may not exist such z_k.
   - Since we are dealing with integer, we find the closest z_k we can get to satisfy $G(z_k) - s_k = 0$
5. For each pixel in the original image, if the value of that pixel is r_k, map this value to its corresponding levels s_k; then map level s_k into the final level z_k.

Histogram Matching

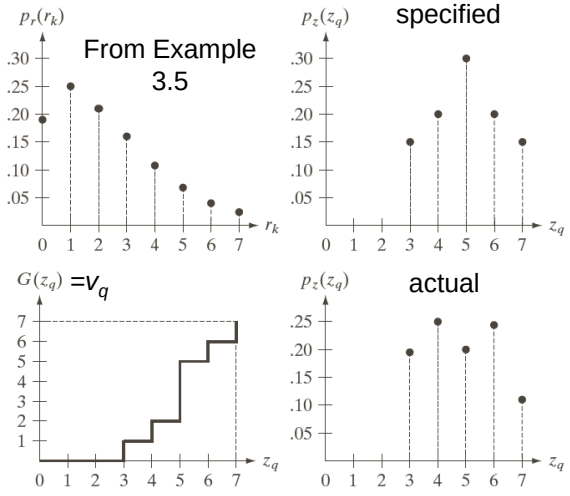


a b c

FIGURE 3.19 (a) Graphical interpretation of mapping from r_k to s_k via $T(r)$. (b) Mapping of z_q to its corresponding value v_q via $G(z)$. (c) Inverse mapping from s_k to its corresponding value of z_k.

Histogram Matching

$r_k \rightarrow s_k = v_q \rightarrow z_q$



a b
c d

FIGURE 3.22
(a) Histogram of a 3-bit image. (b) Specified histogram. (c) Transformation function obtained from the specified histogram. (d) Result of performing histogram specification. Compare (b) and (d).

Histogram Matching

z_q	Specified $p_z(z_q)$	Actual $p_z(z_k)$
$z_0 = 0$	0.00	0.00
$z_1 = 1$	0.00	0.00
$z_2 = 2$	0.00	0.00
$z_3 = 3$	0.15	0.19
$z_4 = 4$	0.20	0.25
$z_5 = 5$	0.30	0.21
$z_6 = 6$	0.20	0.24
$z_7 = 7$	0.15	0.11

TABLE 3.2
Specified and actual histograms (the values in the third column are from the computations performed in the body of Example 3.8).

Mapping : $z_q \rightarrow v_q$

z_q	$G(z_q)$
$z_0 = 0$	0
$z_1 = 1$	0
$z_2 = 2$	0
$z_3 = 3$	1
$z_4 = 4$	2
$z_5 = 5$	5
$z_6 = 6$	6
$z_7 = 7$	7

TABLE 3.3
All possible values of the transformation function G scaled, rounded, and ordered with respect to z.

$r_k \rightarrow s_k = v_q \rightarrow z_q$

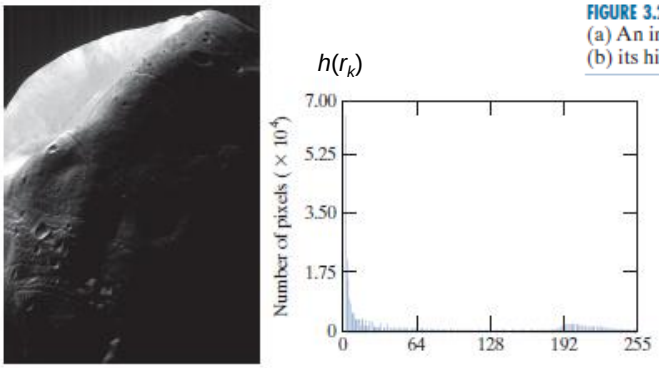
s_k	→	z_q
1	→	3
3	→	4
5	→	5
6	→	6
7	→	7

TABLE 3.4
Mappings of all the values of s_k into corresponding values of z_q.

Mapping : $r_k \rightarrow s_k$

s_0	s_1	s_2	s_3	s_4	s_5	s_6	s_7
1.33	3.08	4.55	5.67	6.23	6.65	6.86	7.00
0→1	1→3	2→5	3→6	4→6	5→7	6→7	7→7

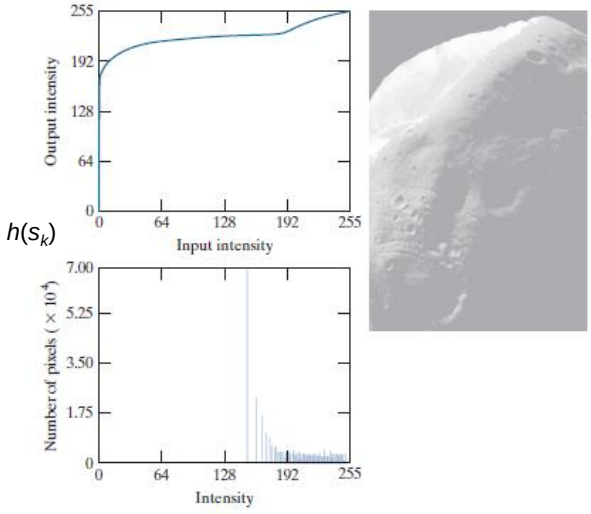
Histogram Matching



a b

FIGURE 3.23
(a) An image, and (b) its histogram.

Histogram Matching



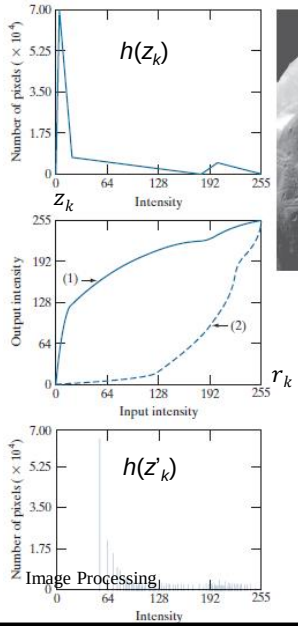
a b
c

FIGURE 3.24
(a) Histogram equalization transformation obtained using the histogram in Fig. 3.23(b). (b) Histogram equalized image. (c) Histogram of equalized image.

Histogram Matching

a c
b
d

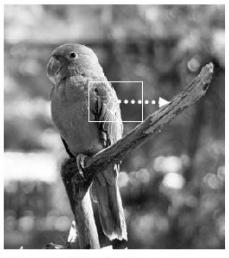
FIGURE 3.25
Histogram specification.
(a) Specified histogram.
(b) Transformation $G(z_q)$, labeled (1), and $G^{-1}(s_k)$, labeled (2).
(c) Result of histogram specification.
(d) Histogram of image (c).



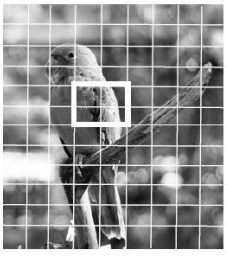
1. $r \xrightarrow{T} s \quad v \xrightarrow{G^{-1}} z \quad s = v$
2. Curve 1: $r_k = T^{-1}(G(z_k))$
3. Curve 2: $z_k = G^{-1}(T(r_k))$

Local Histogram Equalization

- Histogram equalization based on a portion of the image



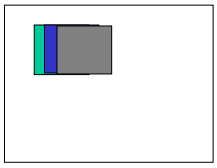
Sliding window approach:
different histogram (and mapping) for every pixel



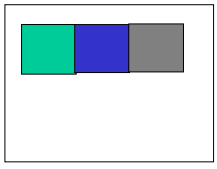
Tiling approach:
subdivide into overlapping regions, mitigate blocking effect by smooth blending between neighboring tiles

Local Histogram Equalization

- Transformation function based on **gray-level distribution** in the **neighborhood** of every pixel in the image.
- At each location, the histogram of points in the neighborhood (or inside a **region**) is computed and histogram equalization is applied.
- Enhancement applied for **overlapped regions** is better than **non-overlapped regions**.



Over-lapped



Non-Overlapped

Local Histogram Equalization

Original image
Parrot

Global histogram equalization

Adaptive histogram equalization, 8x8 tiles

Adaptive histogram equalization, 16x16 tiles

Local Enhancement Using Histogram Statistics

- Let $p(r_i)$ be an estimate of the probability of occurrence of gray-level r_i.
- The nth moment of r about its mean is

$$\mu_n(r) = \sum_{i=0}^{L-1}(r_i - m)^n p(r_i)$$

where m is the mean value of r

$$m = \sum_{i=0}^{L-1} r_i p(r_i)$$

- The second moment is given by

$$\sigma^2 = \mu_2 = \sum_{i=0}^{L-1}(r_i - m)^2 p(r_i)$$

Local Enhancement Using Histogram Statistics

- Let S_{xy} denote a neighborhood of a specified size centered on (x, y) , the mean of pixels in S_{xy} is

$$m_{S_{xy}} = \sum_{i=0}^{L-1} r_i p_{S_{xy}}(r_i)$$

where is the histogram of the pixels in S_{xy}

- The variance of pixels in S_{xy} is

$$\sigma_{S_{xy}}^2 = \sum_{i=0}^{L-1}\left(r_i - m_{S_{xy}}\right)^2 p_{S_{xy}}(r_i)$$

Local Enhancement Using Histogram Statistics

- Example image (Fig. 3.27): To enhance the **dark areas** while leaving the **light area** as unchanged.
- Consider pixel (x, y) for enhancement by measuring whether an area centered on (x, y) is relatively light or dark by comparing the **local average gray level** $m_{S_{xy}}$ to the **global mean** m_G.

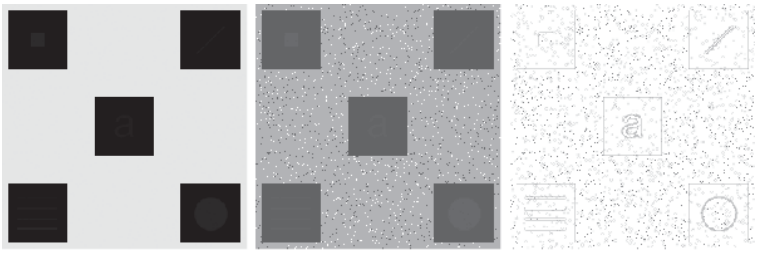
$$k_0 m_G \leq m_{S_{xy}} \leq k_1 m_G \text{ where } 0 \leq k_0 < k_1$$

and the **local standard deviation** $\sigma_{S_{xy}}$ to the **global standard deviation** σ_G.

$$k_2 \sigma_G \leq \sigma_{S_{xy}} \leq k_3 \sigma_G \text{ where } 0 \leq k_2 < k_3.$$

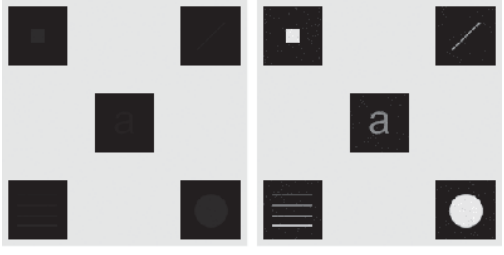
$$g(x,y) = \begin{cases} Cf(x,y) & \text{if } k_0 m_G \leq m_{S_{xy}} \leq k_1 m_G \text{ AND } k_2 \sigma_G \leq \sigma_{S_{xy}} \leq k_3 \sigma_G \\ f(x,y) & \text{otherwise} \end{cases}$$

Local Histogram Equalization



a b c

FIGURE 3.26 (a) Original image. (b) Result of global histogram equalization. (c) Result of local histogram equalization.



a b

FIGURE 3.27 (a) Original image. (b) Result of local enhancement based on local histogram statistics. Compare (b) with Fig. 3.26(c).

Image Processing Using Spatial Filtering

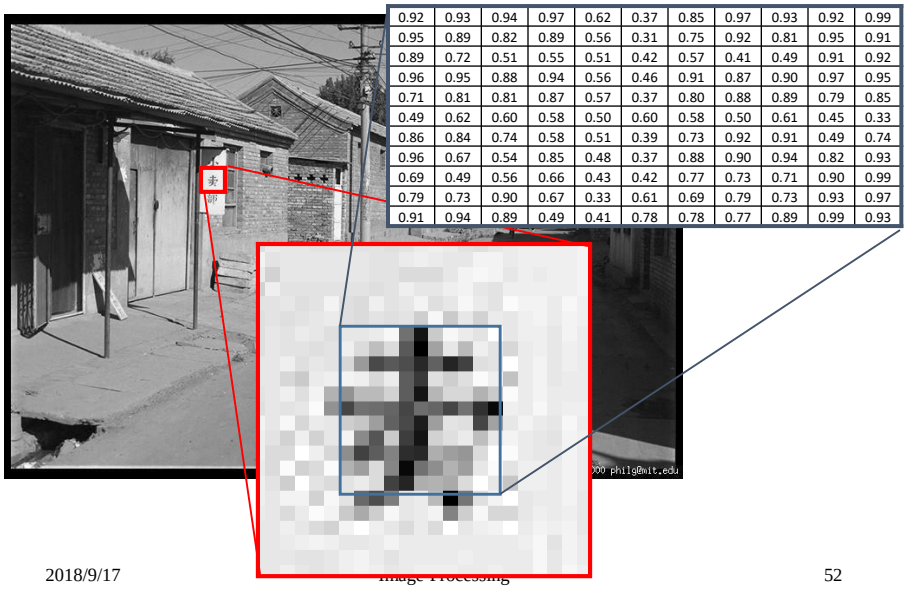
Original Image

Smoothing

Sharpening

Smoothing with edge-preserving

The Raster Image (Pixel Matrix)



0.92	0.93	0.94	0.97	0.62	0.37	0.85	0.97	0.93	0.92	0.99
0.95	0.89	0.82	0.89	0.56	0.31	0.75	0.92	0.81	0.95	0.91
0.89	0.72	0.51	0.55	0.51	0.42	0.57	0.41	0.49	0.91	0.92
0.96	0.95	0.88	0.94	0.56	0.46	0.91	0.87	0.90	0.97	0.95
0.71	0.81	0.81	0.87	0.57	0.37	0.80	0.88	0.89	0.79	0.85
0.49	0.62	0.60	0.58	0.50	0.60	0.58	0.50	0.61	0.45	0.33
0.86	0.84	0.74	0.58	0.51	0.39	0.73	0.92	0.91	0.49	0.74
0.96	0.67	0.54	0.85	0.48	0.37	0.88	0.90	0.94	0.82	0.93
0.69	0.49	0.56	0.66	0.43	0.42	0.77	0.73	0.71	0.90	0.99
0.79	0.73	0.90	0.67	0.33	0.61	0.69	0.79	0.73	0.93	0.97
0.91	0.94	0.89	0.49	0.41	0.78	0.78	0.77	0.89	0.99	0.93

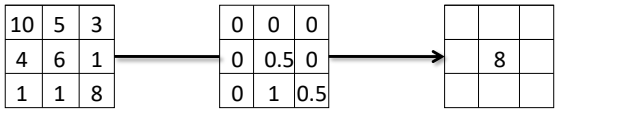
Image Filtering

- Image filtering: for each pixel, compute function of local neighborhood and output a new value
  - Same function applied at each position
  - Output and input image are typically the same size

Local image data Modified image data

Image Filtering

- Linear filtering
  - function is a weighted sum/difference of pixel values



Local image data kernel Modified image data

- Really important!
  - Enhance images
    - Denoise, smooth, increase contrast, etc.
  - Extract information from images
    - Texture, edges, distinctive points, etc.
  - Detect patterns
    - Template matching

Image Filtering

- Given a camera and a still scene, how can you reduce noise?

Take lots of images and average them!
What's the next best thing?

First Attempt at a Solution

- Let's replace each pixel with an average of all the values in its neighborhood
- Assumptions:
  - Expect pixels to be like their neighbors
  - Expect noise processes to be independent from pixel to pixel

Mean Kernel

What's the kernel for a 3x3 mean filter?

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

$F[x, y]$

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

$H[u, v]$

Image Filtering: Mean Filter

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

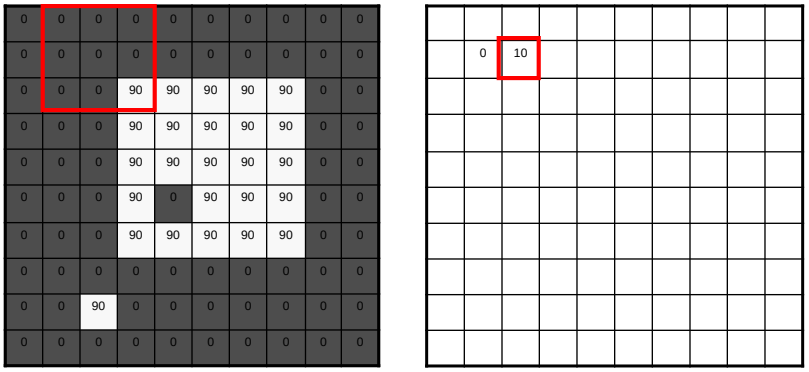
$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k, n+l]$$

Image Filtering: Mean Filter

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$ $h[.,.]$



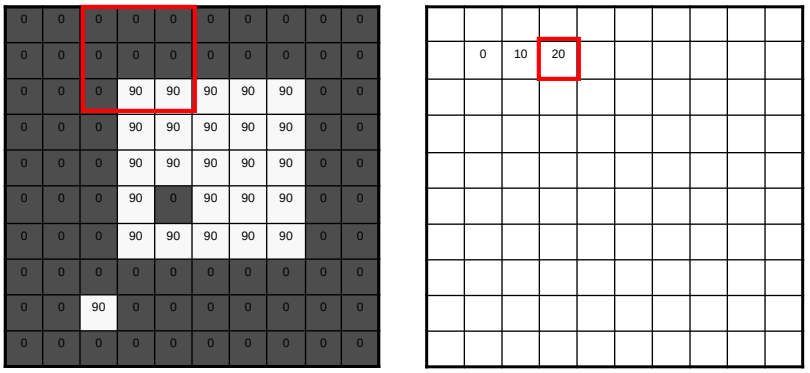
$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Image Filtering: Mean Filter

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$ $h[.,.]$



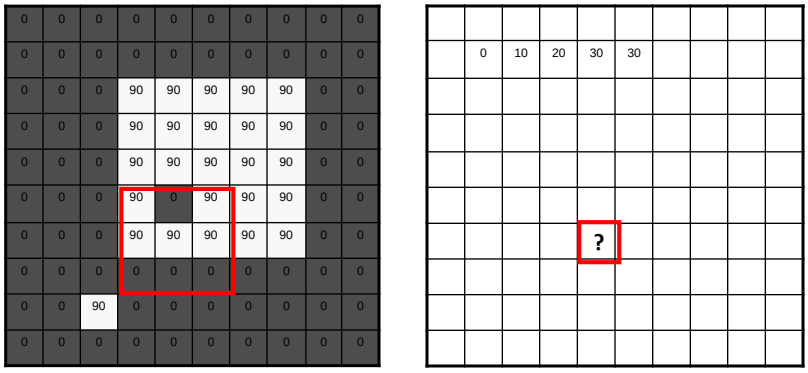
$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Image Filtering: Mean Filter

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$ $h[.,.]$



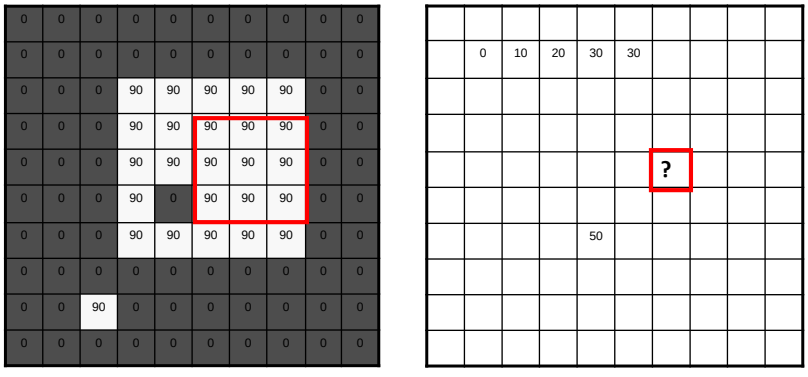
$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Image Filtering: Mean Filter

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$ $h[.,.]$



$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Image Filtering: Mean Filter

$g[\cdot,\cdot]$ $\frac{1}{9}$

1	1	1
1	1	1
1	1	1

$f[.,.]$

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

$h[.,.]$

	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	

$$h[m,n] = \sum_{k,l} g[k,l]\ f[m+k,n+l]$$

Mean (Box) Filter

What does it do?

- Replaces each pixel with an average of its neighborhood
- Achieve smoothing effect (remove sharp features)

$g[\cdot,\cdot]$

$\frac{1}{9}$

1	1	1
1	1	1
1	1	1

Smoothing with Mean Filter

Correlation Filtering

Say the averaging window size is 2k+1 x 2k+1:

$$G[i,j] = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Attribute uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k}\sum_{v=-k}^{k} F[i+u, j+v]}_{\text{Loop over all pixels in neighborhood around image pixel F[i,j]}}$$

Now generalize to allow different weights depending on neighboring pixel's relative position:

$$G[i,j] = \sum_{u=-k}^{k}\sum_{v=-k}^{k} \underbrace{H[u,v]}_{\text{Non-uniform weights}} F[i+u, j+v]$$

Correlation Filtering

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

This is called cross-correlation, denoted $G = H \otimes F$

Filtering an image: replace each pixel with a linear combination of its neighbors.

The filter "kernel" or "mask" *H*[*u,v*] is the prescription for the weights in the linear combination.

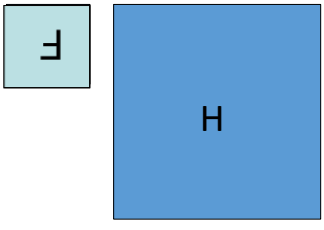
Convolution

- Convolution:
  - Flip the filter in both dimensions (bottom to top, right to left)
  - Then apply cross-correlation

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

Notation for convolution operator



Correlation vs. Convolution

Convolution

```
G=filter2(H,F); or
G=imfilter(F,H);
```

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i-u, j-v]$$

$$G = H \star F$$

Cross-correlation

```
G=conv2(H,F);
```

$$G[i,j] = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H[u,v]F[i+u, j+v]$$

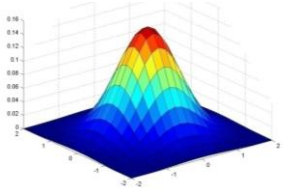
$$G = H \otimes F$$

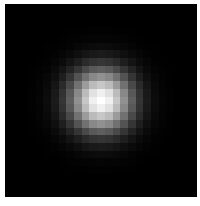
For a Gaussian or box filter, how will the outputs differ?

If the input is an impulse signal, how will the outputs differ?

Gaussian Filter

- Spatially-weighted average



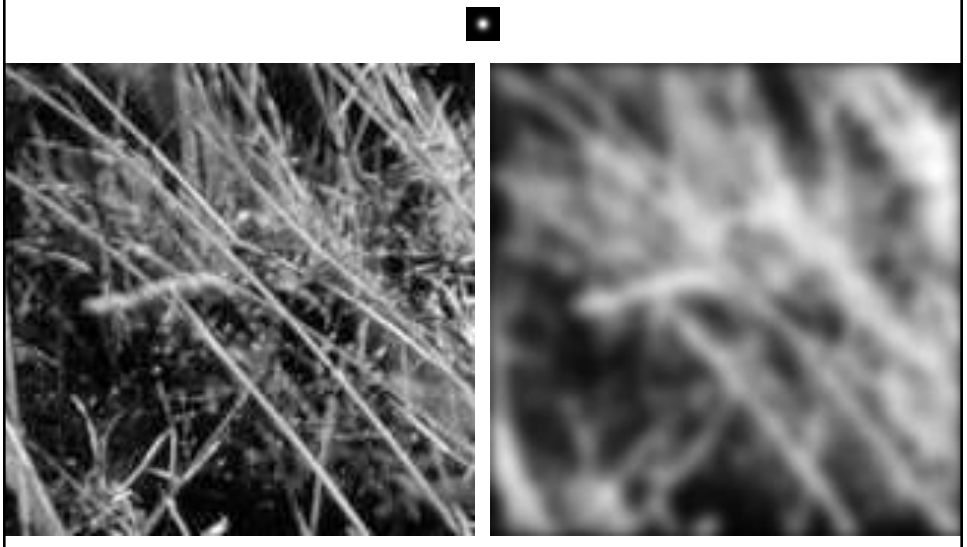


0.003	0.013	0.022	0.013	0.003
0.013	0.059	0.097	0.059	0.013
0.022	0.097	0.159	0.097	0.022
0.013	0.059	0.097	0.059	0.013
0.003	0.013	0.022	0.013	0.003

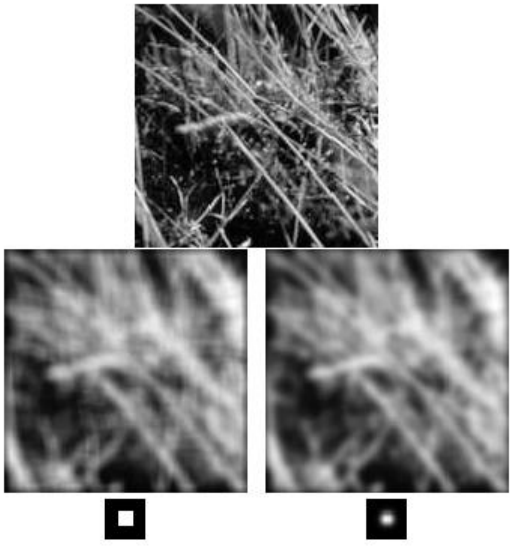
5 x 5, σ = 1

$$G_\sigma = \frac{1}{2\pi\sigma^2} e^{-\frac{(x^2+y^2)}{2\sigma^2}}$$

Smoothing with Gaussian Filter

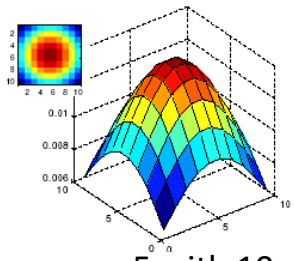


Mean vs. Gaussian Kernel

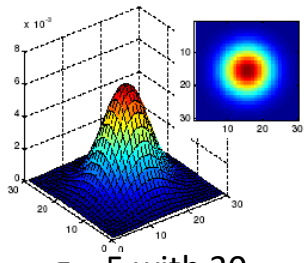


Gaussian Filters

- What parameters matter here?
- **Size** of kernel or mask
  - Note, Gaussian function has infinite support, but discrete filters use finite kernels



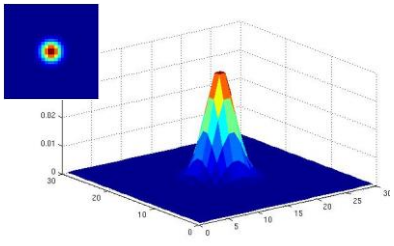
σ = 5 with 10 x 10 kernel



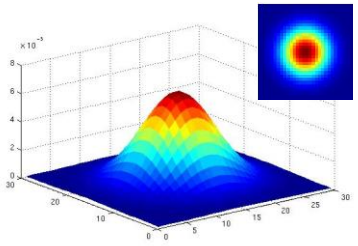
σ = 5 with 30 x 30 kernel

Gaussian Filters

- What parameters matter here?
- **Variance** of Gaussian: determines extent of smoothing

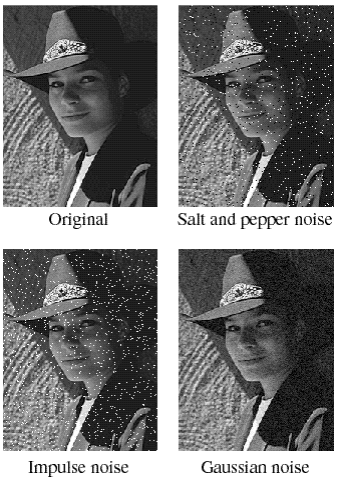


σ = 2 with 30 x 30 kernel



σ = 5 with 30 x 30 kernel

Noise in an Image

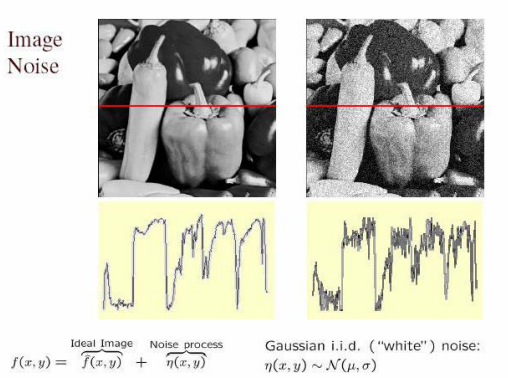


Common types of noise:

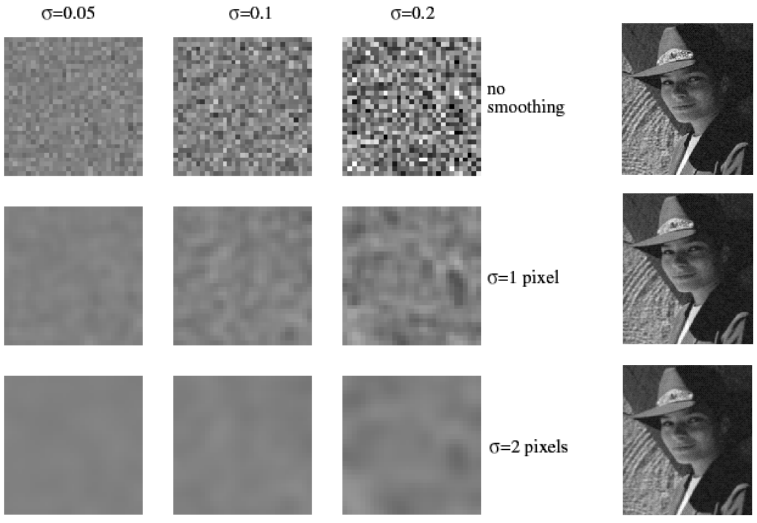
- **Salt and pepper noise**: contains random occurrences of black and white pixels
- **Impulse noise:** contains random occurrences of white pixels
- **Gaussian noise**: variations in intensity drawn from a Gaussian normal distribution

Gaussian Noise

- Mathematical model: sum of many independent factors
- Good for small standard deviations
- Assumption: independent, zero-mean noise



Reducing Gaussian Noise



Smoothing with larger standard deviations suppresses noise, but also blurs the image

Reducing Salt-and-Pepper Noise

3x3 5x5 7x7

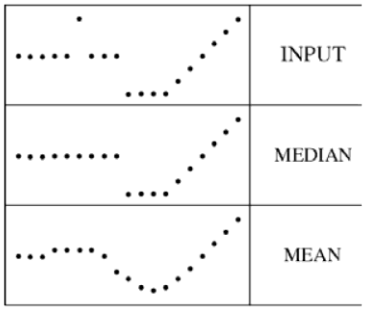
What's wrong with the results?

Median (Order-Statistics) Filter

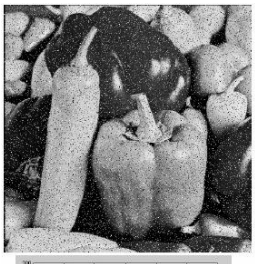
- A **median filter** operates over a $k \times k$ window by returning the median pixel value in that window.
- Median filters create a smoothing effect, and large mask sizes create a painted (and blurred) look
- With a large mask the median filter takes a long time to process, so in practice a fast algorithm or a pseudo-median filter may be used

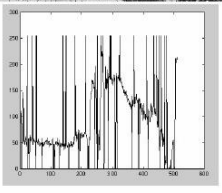
Median Filter

filters have width 5 :

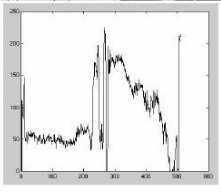


Salt-and-pepper noise





Median filtered



Gaussian vs. Median Filtering

Other Non-linear Filters

- Weighted median (pixels further from center count less)
- Clipped mean (average, ignoring few brightest and darkest pixels)
- Bilateral filtering (weight by spatial distance *and* intensity difference)

Bilateral filtering

Bilateral Filter

- Edge preserving: weights similar pixels more

(a) (b) (c)

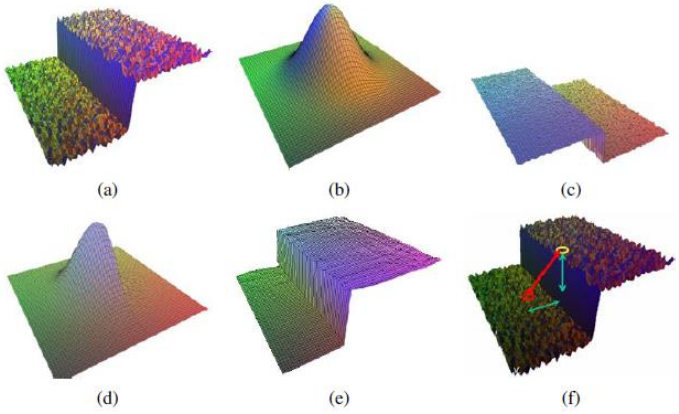
spatial similarity (e.g., intensity)

$$I_{\mathbf{p}}^{\mathrm{b}} = \frac{1}{W_{\mathbf{p}}^{\mathrm{b}}} \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

$$\text{with} \quad W_{\mathbf{p}}^{\mathrm{b}} = \sum_{\mathbf{q}\in\mathcal{S}} G_{\sigma_{\mathrm{s}}}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_{\mathrm{r}}}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)$$

Carlo Tomasi, Roberto Manduchi, Bilateral Filtering for Gray and Color Images, ICCV, 1998.

Bilateral Filter



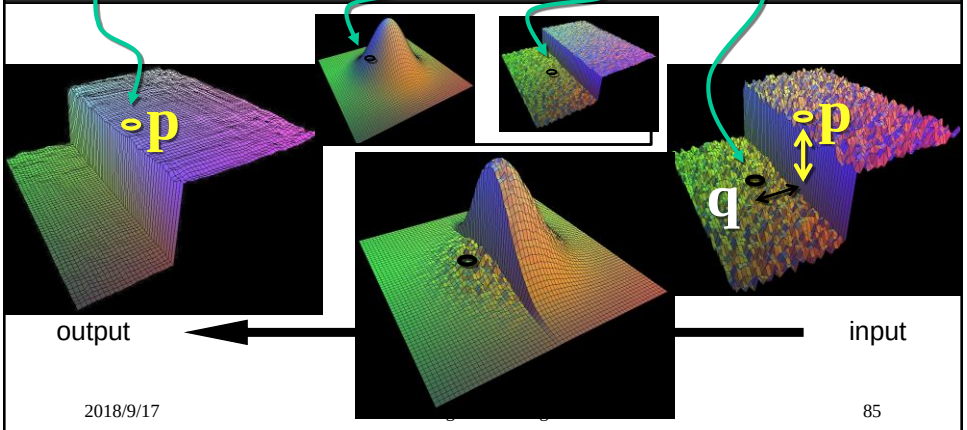
(a) (b) (c)

(d) (e) (f)

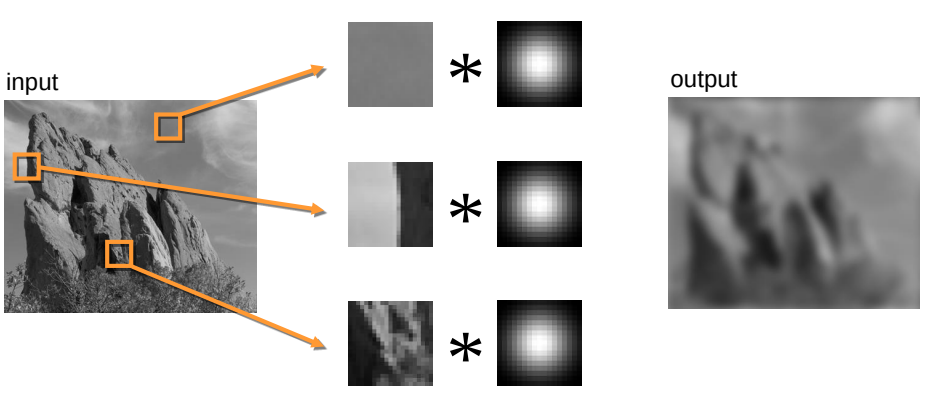
Figure 3.20 Bilateral filtering (Durand and Dorsey 2002) © 2002 ACM: (a) noisy step edge input; (b) domain filter (Gaussian); (c) range filter (similarity to center pixel value); (d) bilateral filter; (e) filtered step edge output; (f) 3D distance between pixels.

Bilateral Filter

$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}(\|\mathbf{p}-\mathbf{q}\|)\, G_{\sigma_r}(|I_{\mathbf{p}}-I_{\mathbf{q}}|)\, I_{\mathbf{q}}$$

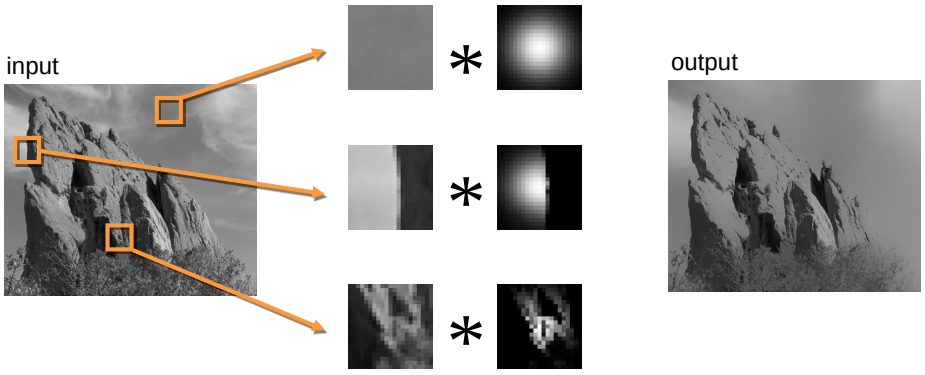


Gaussian Filter



Same Gaussian kernel everywhere.

Bilateral Filter



The kernel shape depends on the image content.

Bilateral Filter vs. Gaussian Filter

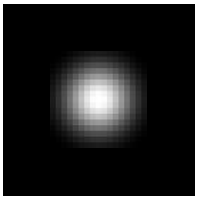
$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q} \in S} G_{\sigma_s}\left(\|\mathbf{p}-\mathbf{q}\|\right) G_{\sigma_r}\left(|I_{\mathbf{p}}-I_{\mathbf{q}}|\right) I_{\mathbf{q}}$$

new — not new — new

normalization factor

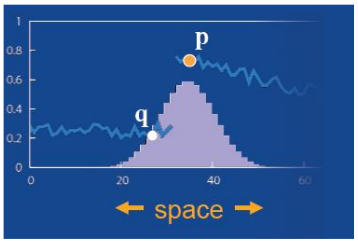
space weight

range weight



Bilateral Filter vs. Gaussian Filter

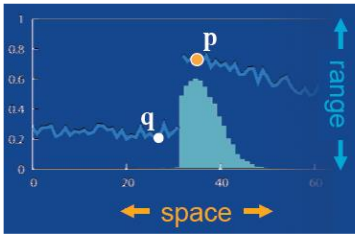
Gaussian filter



$$GB[I]_{\mathbf{p}} = \sum_{\mathbf{q}\in S} G_{\sigma}\left(\|\mathbf{p}-\mathbf{q}\|\right) I_{\mathbf{q}}$$

space

Bilateral filter

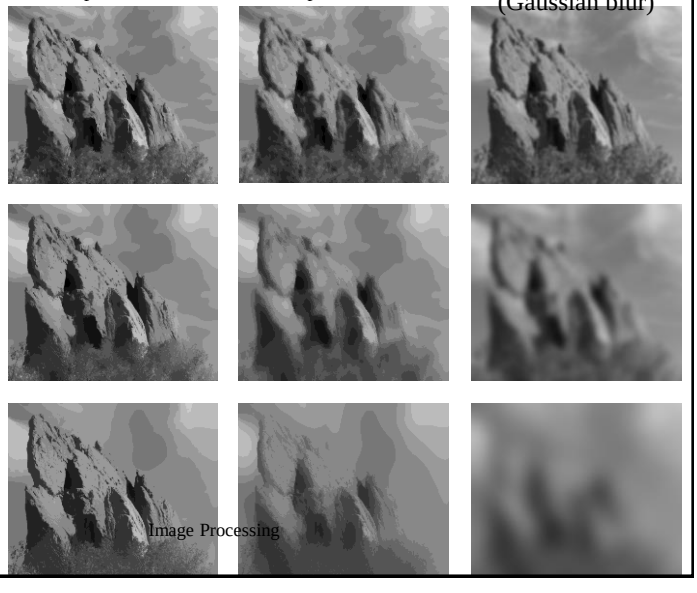


$$BF[I]_{\mathbf{p}} = \frac{1}{W_{\mathbf{p}}} \sum_{\mathbf{q}\in S} G_{\sigma_s}\left(\|\mathbf{p}-\mathbf{q}\|\right) G_{\sigma_r}\left(|I_{\mathbf{p}}-I_{\mathbf{q}}|\right) I_{\mathbf{q}}$$

normalization　space　range

Bilateral Filter: Parameters

input



Comparison of Smoothing Filters

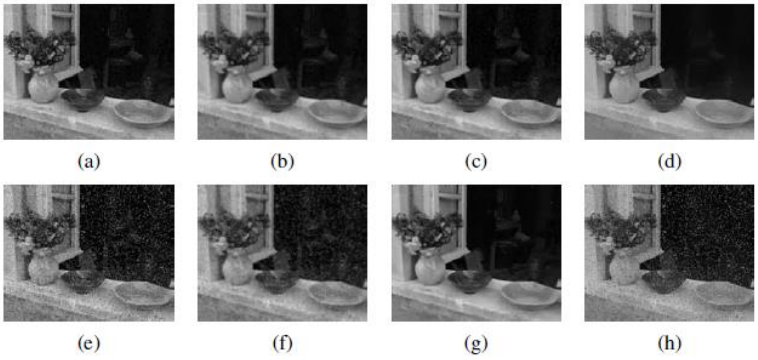


Figure 3.18 Median and bilateral filtering: (a) original image with Gaussian noise; (b) Gaussian filtered; (c) median filtered; (d) bilaterally filtered; (e) original image with shot noise; (f) Gaussian filtered; (g) median filtered; (h) bilaterally filtered. Note that the bilateral filter fails to remove the shot noise because the noisy pixels are too different from their neighbors.

Image Sharpening

- Image sharpening deals with enhancing detailed information in an image, typically edges and, in general, correspond to image features that are small spatially
- Most of the techniques include some form of highpass filtering
- The information is visually important, because it delineates object and feature boundaries, and is important for textures in objects

Linear Filters

Original

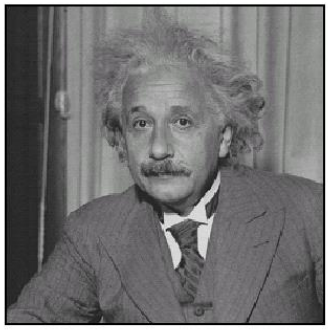
$$\begin{bmatrix} 0 & 0 & 0 \\ 0 & 2 & 0 \\ 0 & 0 & 0 \end{bmatrix} - \frac{1}{9}\begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

(Note that filter sums to 1)

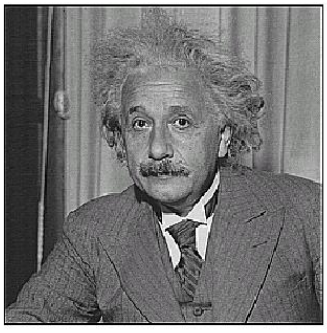
Sharpening filter

- Accentuates differences with local average

Sharpening



before



after

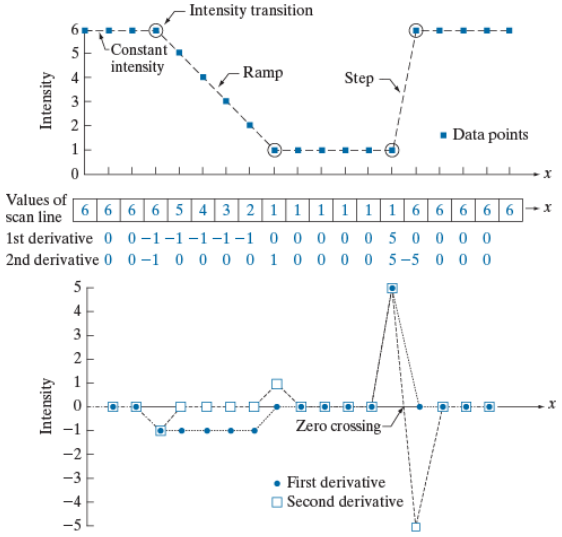
Sharpening Spatial Filters

- Objective of sharpening: to highlight fine details in an image or to enhance details that have been blurred
- Image differentiation enhances edges and other discontinuities (including noises) and deemphasizes areas with slowly varying gray-level values

$$\frac{\partial f}{\partial x} \approx f(x+1, y) - f(x, y), \quad \frac{\partial f}{\partial y} \approx f(x, y+1) - f(x, y),$$

$$\frac{\partial^2 f}{\partial x^2} \approx f(x+1, y) + f(x-1, y) - 2f(x, y)$$

Sharpening Spatial Filters



a
b
c

FIGURE 3.44
(a) A section of a horizontal scan line from an image, showing ramp and step edges, as well as constant segments.
(b) Values of the scan line and its derivatives.
(c) Plot of the derivatives, showing a zero crossing. In (a) and (c) points were joined by dashed lines as a visual aid.

Laplacina Kernel

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2}$$

$$\approx f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y)$$

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

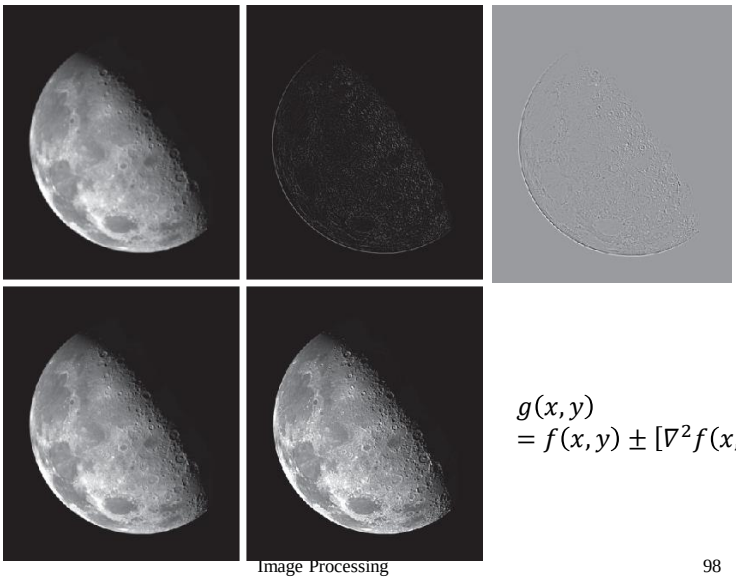
a b c d

FIGURE 3.45 (a) Laplacian kernel used to implement Eq. (3-53). (b) Kernel used to implement an extension of this equation that includes the diagonal terms. (c) and (d) Two other Laplacian kernels.

Image Sharpening with Laplacian

a b
c d

FIGURE 3.46 (a) Blurred image of the North Pole of the moon. (b) Laplacian image obtained using the kernel in Fig. 3.45(a). (c) Image sharpened using Eq. (3-54) with $c = -1$. (d) Image sharpened using the same procedure, but with the kernel in Fig. 3.45(b). (Original image courtesy of NASA.)



$$g(x,y) = f(x,y) \pm [\nabla^2 f(x,y)]$$

Unsharp Masking & High-Boost Filtering

- **Unsharp masking**

$$g_{\text{mask}}(x,y) = f(x,y) - \bar{f}(x,y)$$

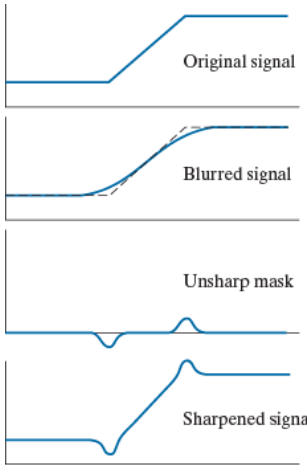
where $\bar{f}(x,y)$ is a blurred image

$$f_{\text{um}}(x,y) = f(x,y) + k g_{\text{mask}}(x,y),\ k = 1$$

- **High boost filtering**

$$f_{\text{hb}}(x,y) = f(x,y) + k g_{\text{mask}}(x,y),\ k > 1$$

Unsharp Masking & High-Boost Filtering



a b c
d e

FIGURE 3.49 (a) Original image of size 600 × 259 pixels. (b) Image blurred using a 31 × 31 Gaussian lowpass filter with $\sigma = 5$. (c) Mask. (d) Result of unsharp masking using Eq. (3-56) with $k = 1$. (e) Result of highboost filtering with $k = 4.5$.

Gradient-Based Image Enhancement

- Image gradient $\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$
- Gradient magnitude

$$\|\nabla f\| = \sqrt{\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2}, \qquad \|\nabla f\| = \left|\frac{\partial f}{\partial x}\right| + \left|\frac{\partial f}{\partial y}\right|$$

- Roberts cross-gradient operator

$$\partial f/\partial x = z_9 - z_5$$
$$\partial f/\partial y = z_8 - z_6$$

$$M(x,y) = [(z_9 - z_5)^2 + (z_8 - z_6)^2]^{\frac{1}{2}} \text{ or} \approx |z_9 - z_5| + |z_8 - z_6|$$

- Sobel operator

$$M(x,y) = |(z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)| + |(z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)|$$

1st-Order Derivative for Enhancement

a
b c
d e

FIGURE 3.50
(a) A 3×3 region of an image, where the zs are intensity values. (b)–(c) Roberts cross-gradient operators. (d)–(e) Sobel operators. All the kernel coefficients sum to zero, as expected of a derivative operator.

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

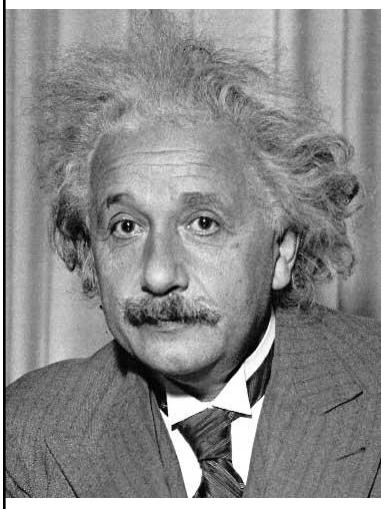
−1	0
0	1

0	−1
1	0

−1	−2	−1
0	0	0
1	2	1

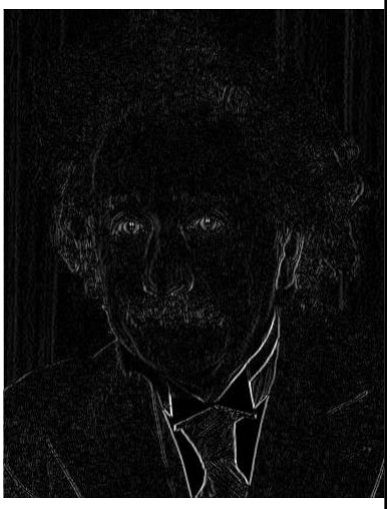
−1	0	1
−2	0	2
−1	0	1

Edge Detection



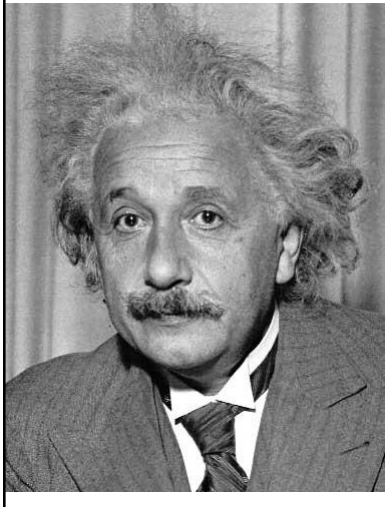
1	0	-1
2	0	-2
1	0	-1

Sobel



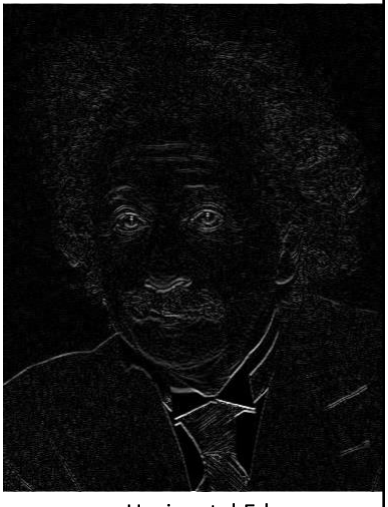
Vertical Edge
(absolute value)

Edge Detection



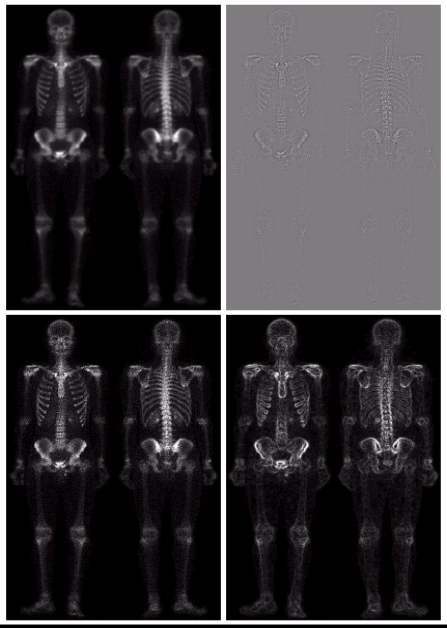
1	2	1
0	0	0
-1	-2	-1

Sobel



Horizontal Edge
(absolute value)

Combining Spatial Enhancement Methods

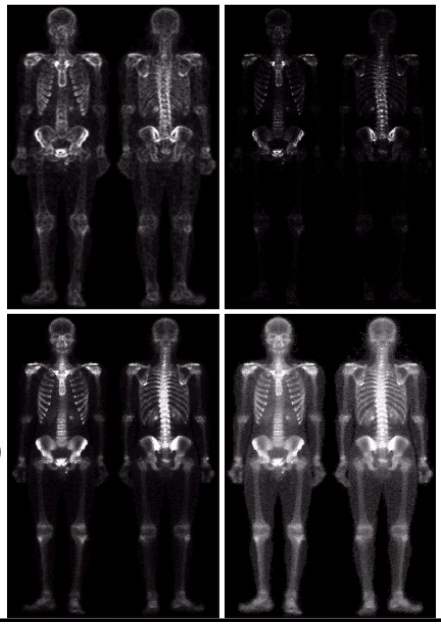


(c)=(a)+(b)

e f
g h

FIGURE 3.57
(Continued)
(e) Sobel image smoothed with a 5×5 box filter.
(f) Mask image formed by the product of (b) and (e).
(g) Sharpened image obtained by the adding images (a) and (f).
(h) Final result obtained by applying a power-law transformation to (g). Compare images (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)

Combining Spatial Enhancement Methods



(f)=(c)(e)

(g)=(a)+(f)

Power-law Transform of (g)

e f
g h

FIGURE 3.57
(Continued)
(e) Sobel image smoothed with a 5×5 box filter.
(f) Mask image formed by the product of (b) and (e).
(g) Sharpened image obtained by the adding images (a) and (f).
(h) Final result obtained by applying a power-law transformation to (g). Compare images (g) and (h) with (a). (Original image courtesy of G.E. Medical Systems.)