Image Intensity Transformation and Image Enhancement

AJIT RAJWADE
CS 663

Intensity transformations

- Consider an image I(x,y).
- An **intensity transformation** refers to the transformation applied to individual image intensities to yield a transformed image of the form: J(x,y) = T(I(x,y)) where T(.) is a transformation operator.

- For example: $J(x,y) = [I(x,y)]^2$ or $J(x,y) = I(x,y)-I(x,y+1)$

- In the latter case, the transformation makes use of other pixel values, usually from the neighborhood of the pixel (x,y).

Image Enhancement

- Defined as the process of manipulating image intensities, so that the resulting image is **more visually suitable** for a specific application.

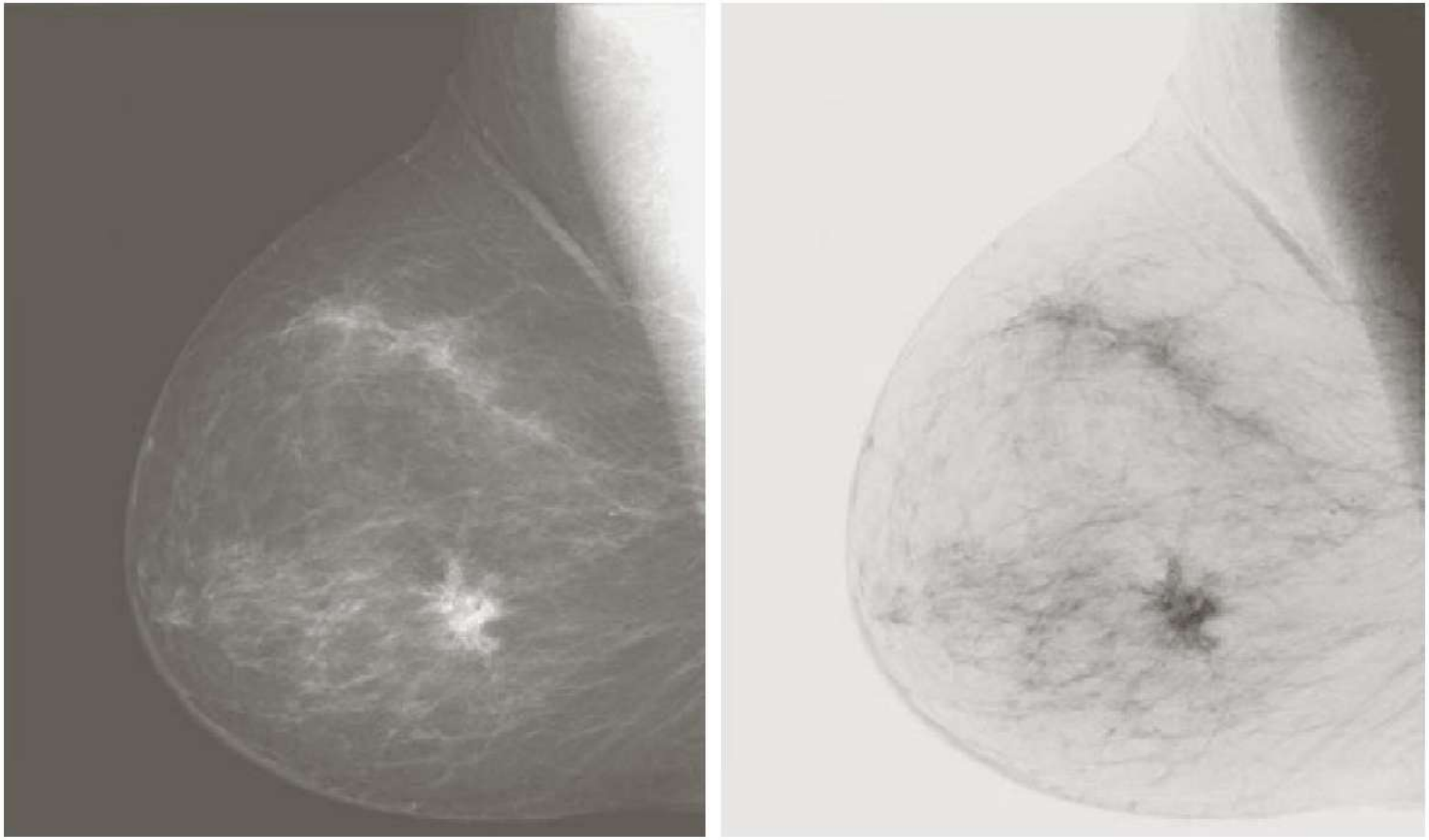
- The quality of an image enhancement algorithm is judged by the **perception** of the end-user.

Popular image transformations

- Image Negatives
- Logarithm and Power Law transformations
- Contrast Stretching
- Bit-plane slicing
- Histogram Equalization
- Histogram Specification

Image Negatives

- Let an image have intensity range $[0, L-1]$.
- The negative transformation has the form $s(r)=L-1-r$ where s,r = output, input intensity level.
- This reverses the intensity levels of an image – equivalent of a photographic negative
- This can produce an enhancing effect – to enhance grayish detail on dark background.



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.)

Logarithmic transformation

- Has the form $s(r) = c \log(1+r)$ where s,r = output, input intensity level assuming $r >= 0$ and c is some constant.
- It maps a narrow range of **low** intensity values to a wider range.
- It maps a range of **high** intensity values to a narrow range.
- This can be observed from the graphs on the next slide.

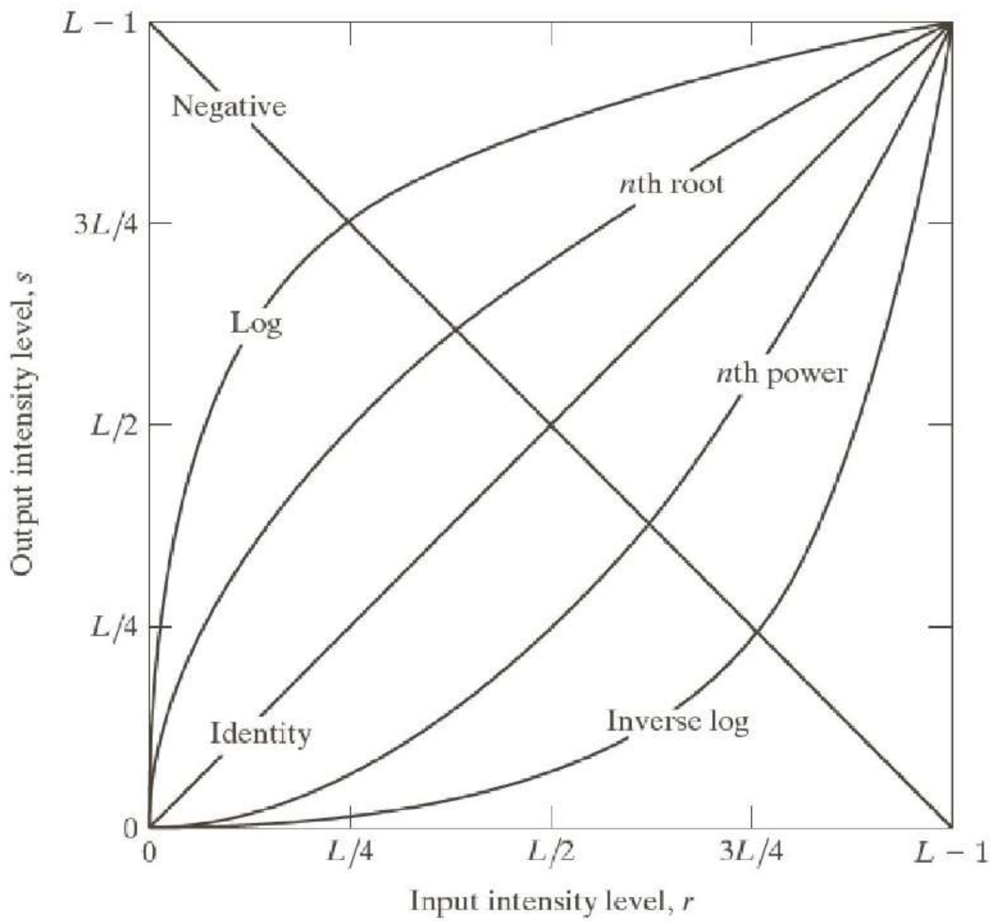
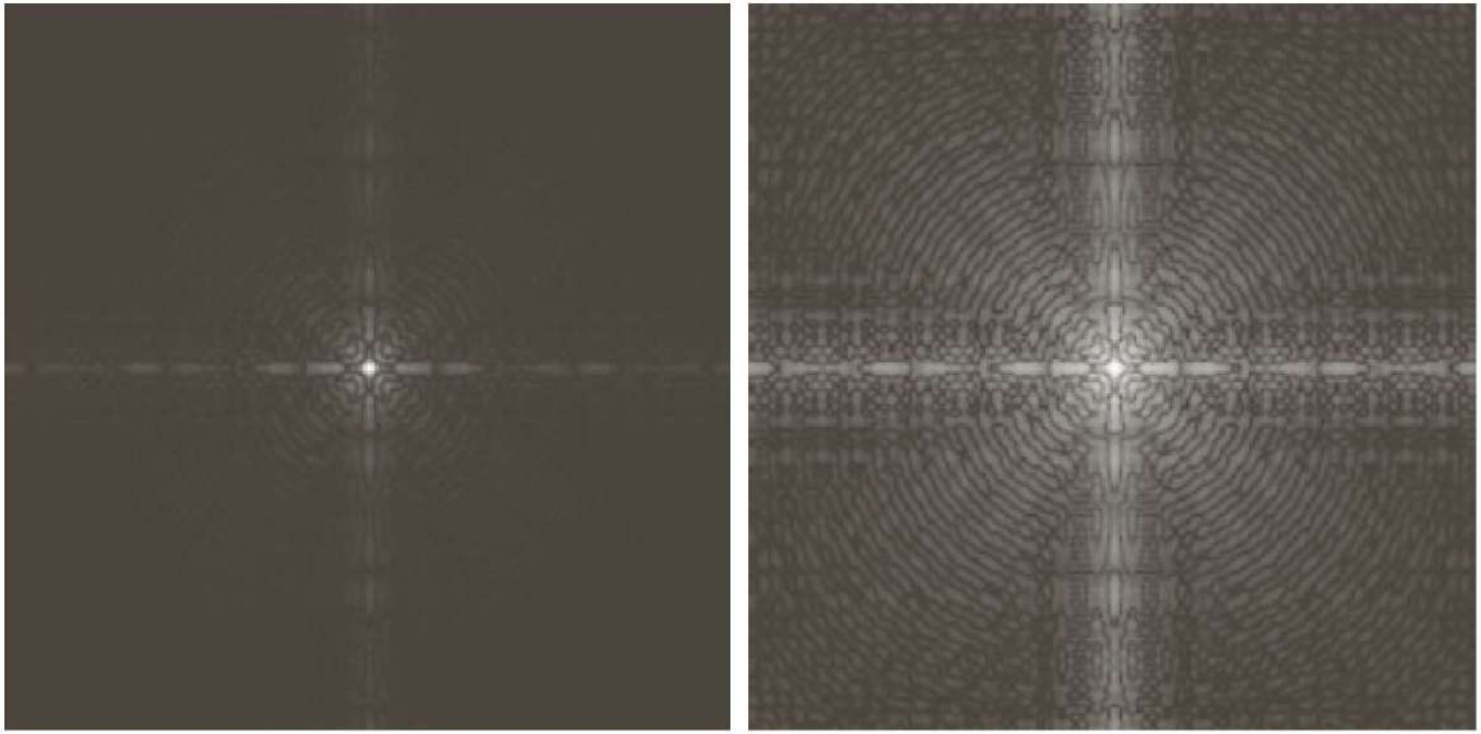


FIGURE 3.3 Some basic intensity transformation functions. All curves were scaled to fit in the range shown.

Logarithmic transformations

- A logarithmic transformation compresses the intensity range of images whose initial intensity range is very wide.
- For example, the Fourier spectrum of an image (we will study these later in class) often has intensities ranging from close to 0 to 10^7.
- Displaying the original intensities of the Fourier spectrum as is, causes a huge loss of detail.
- A log transformation of the intensities allows perception of a great deal of detail (on the same display system). See next slide.



a b

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

When we do Fourier transforms later in class, we will make heavy use of log(r+1) type transformations.

Note: the 1 in log(r+**1**) is used for stabilization since log 0 is undefined. In some applications, if there are intensity values which are very tiny which need to be preserved, the 1 should be replaced by some ε value which is several times smaller than the smallest non-zero value.

Power-Law (Gamma) Transformation

- This is a transformation of the form $s(r) = cr^{\gamma}$.
- Here, $c > 0$ is a constant, r is the input intensity and $\gamma > 0$ is a constant for the power law.
- For fractional values of γ (between 0 to 1), a narrow range of low intensity values gets mapped to a wider range, and a range of high intensity values gets mapped to a narrower range.
- The effect is similar in principle to the log transformation, but we have more flexibility here as we can tune γ per our liking.

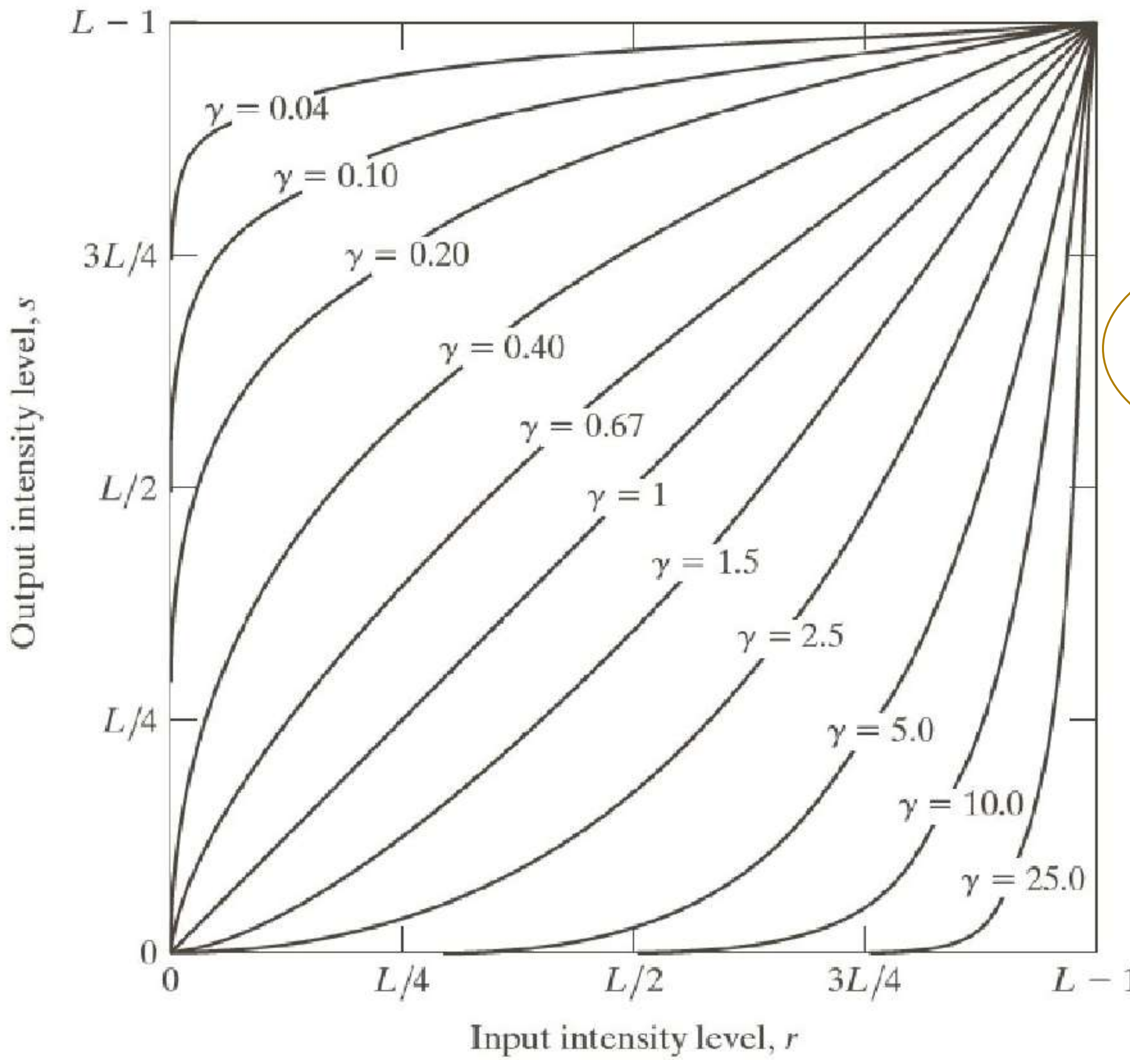
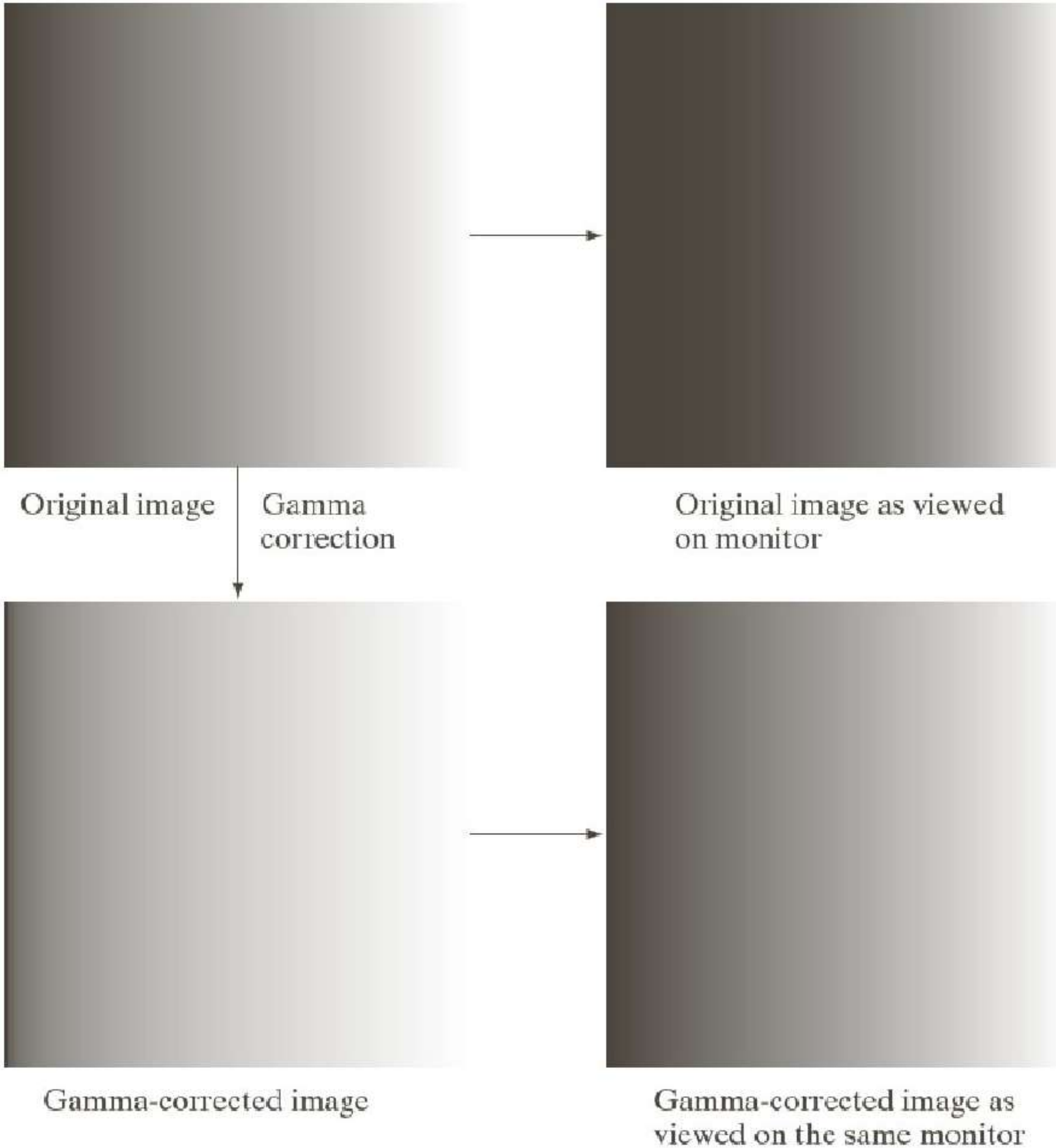


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases). All curves were scaled to fit in the range shown.

Note: The curves have been scaled to the range [0,L-1] after applying the gamma transformation. This has the effect of enhancing brighter intensities when gamma is more than 1.

Power-Law (Gamma) Transformation

- Many devices for display or printing work as per a power law.
- For example, a cathode ray tube (CRT) has an input intensity to output voltage mapping that uses a power law with γ between 1.8 to 2.5.
- Since the brightness of the pixel perceived on a CRT monitor depends on the voltage, it ends up producing images that are darker than intended to be (See next slide), i.e. the system displays cr^{γ} instead of r and hence brighter values dominate.
- Hence the image is pre-processed before inputting to the display system of the computer in the following manner: $s(r) = (r/c)^{1/\gamma}$ based on the gamma value (γ) of the display system.
- The output voltage values will be $cs^{\gamma} = c[(r/c)^{1/\gamma}]^{\gamma} = r$ as intended to be.

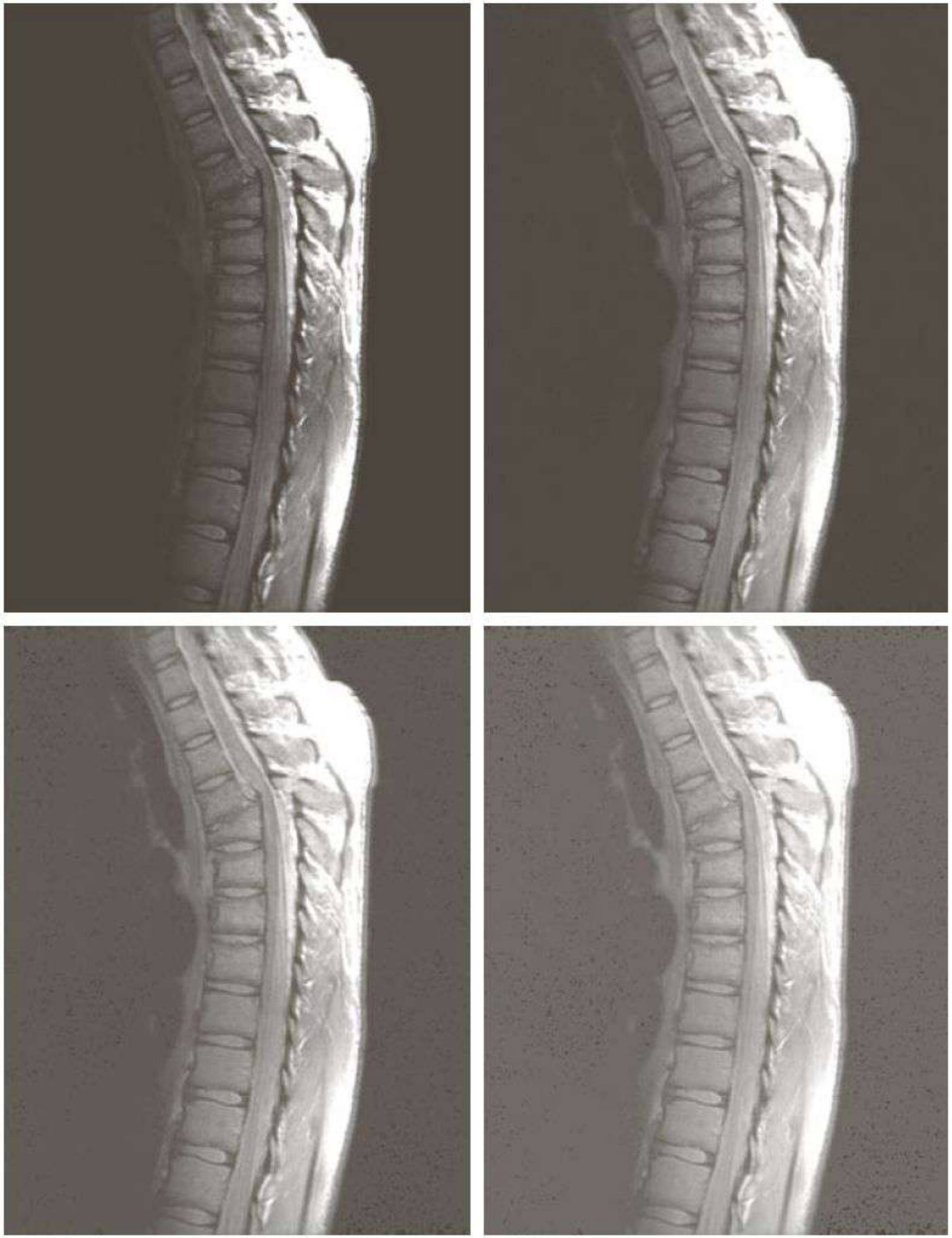


a b
c d

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).

Power-Law (Gamma) Transformation

- The aforementioned transformation is called **gamma correction.**
- It is performed internally by the display system (oblivious to the end-user).
- Power-law transformations are also used for general “contrast enhancment” apart from gamma correction.



a	b
c	d

FIGURE 3.8
(a) Magnetic resonance image (MRI) of a fractured human spine.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) 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.)

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.)

Contrast Stretching

- Low-contrast images occur due to poor lighting or low dynamic range of the camera sensor.

- Contrast stretching expands the intensity range in an image so that it spans the full intensity range of the display medium.

- For example suppose image I has range from $[r_{min}, r_{max}]$ whereas the range of the device is $[0,L]$.

- In the most basic form of contrast stretching, we perform: $s(r) = (L-1)(r- r_{min})/(r_{max} - r_{min})$ so that r_{min} is mapped to 0 and r_{max} is mapped to $L-1$.

a b
c d

FIGURE 3.10 Contrast stretching. (a) Form of transformation function. (b) A low-contrast image. (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.)

Ignore the yellowed out images

Bit-plane slicing

- Pixel values are usually 8-bit integers in photographic images.
- Thus at each pixel location, we have 8 bits defined.
- Such an image can be considered as being composed of eight 1-bit planes – each plane being a binary image.

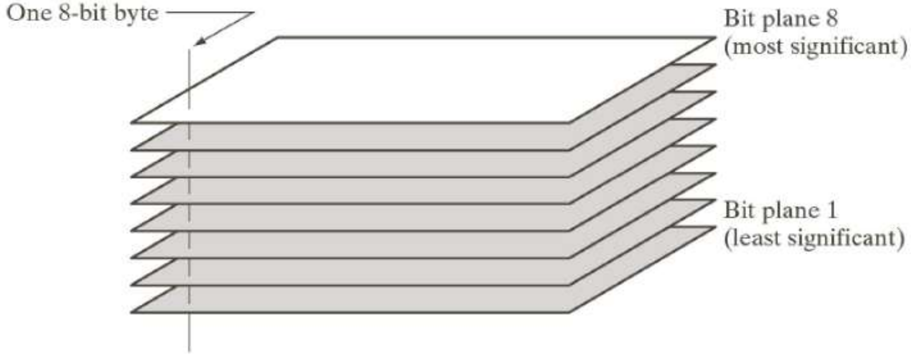


FIGURE 3.13 Bit-plane representation of an 8-bit image.

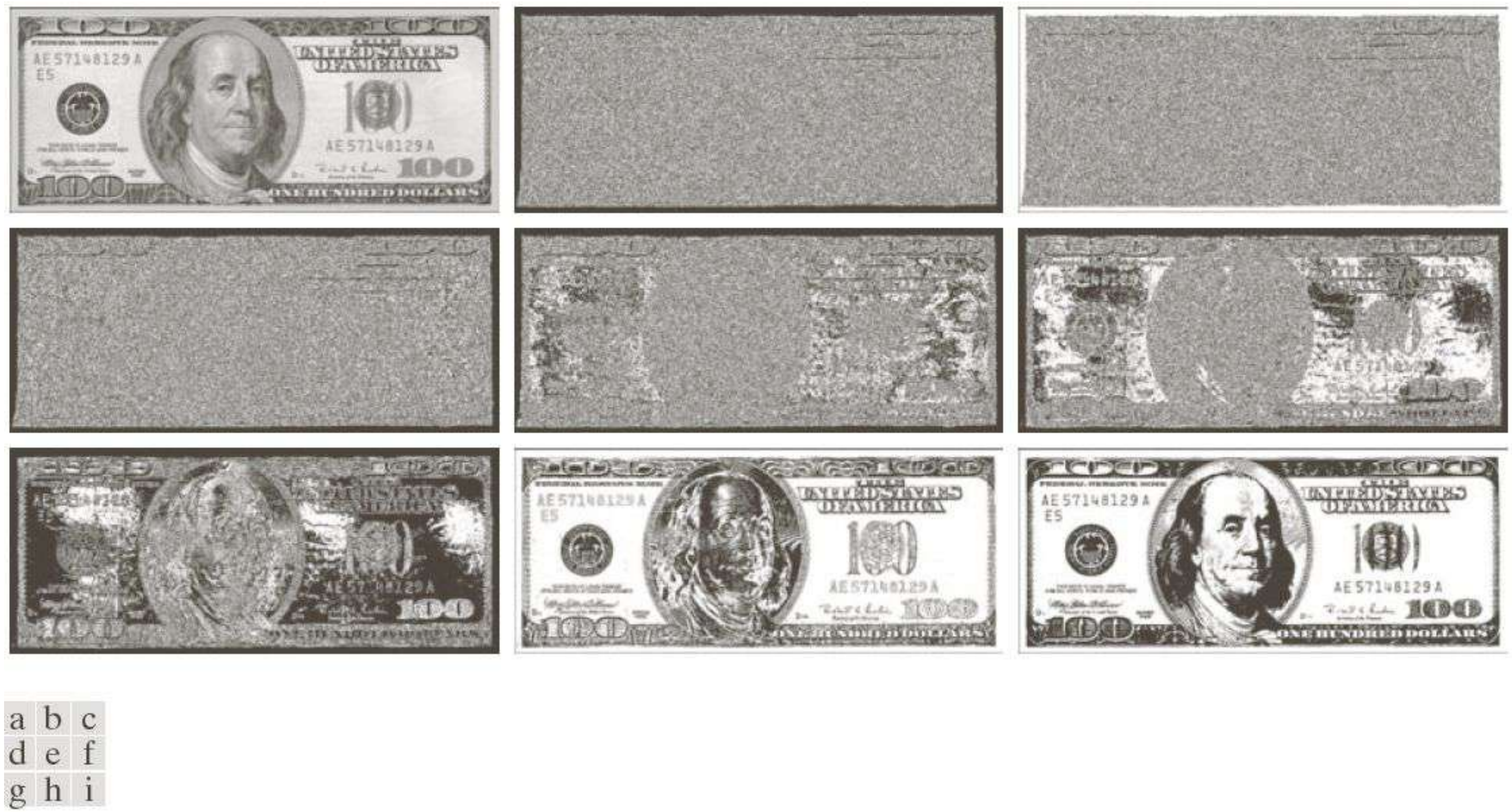


FIGURE 3.14 (a) An 8-bit gray-scale image of size 500 × 1192 pixels. (b) through (i) Bit planes 1 through 8, with bit plane 1 corresponding to the least significant bit. Each bit plane is a binary image.

Bit slicing

- From the previous example, we see that the first 4 most significant bits carry most of the image information.
- The 4 least significant bits carry less information and mostly subtle texture, even resembling noise.
- One can compress an image (with some loss!) by using the most significant bits from Q' to Q, as follows:

$$J(x,y) = \sum_{k=Q'}^{Q} 2^{k-1} I(x,y,k)$$

k-th bit-plane image

a b c

FIGURE 3.15 Images reconstructed using (a) bit planes 8 and 7; (b) bit planes 8, 7, and 6; and (c) bit planes 8, 7, 6, and 5. Compare (c) with Fig. 3.14(a).

Parameter selection in image enhancement

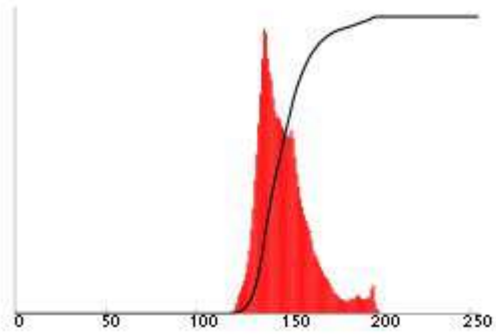
- All aforementioned techniques are intended to improve the (subjective) visual appeal of the image.
- The choice of various parameters in these techniques (eg: γin power-law) therefore depends on the user.
- It can be considered creative user choice.

Parameter selection in image enhancement

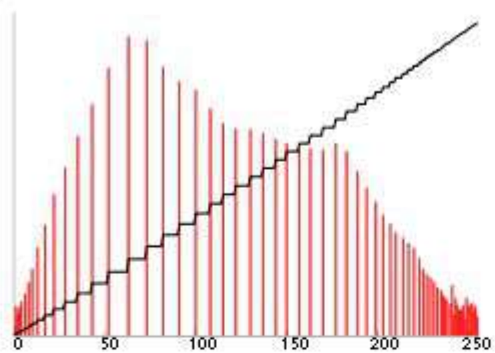
- Here image enhancement differs from image restoration (which we will study later).
- In the latter, we model a degradation phenomenon (eg: image blurring) and seek to inverts its effect.

Histogram equalization

- A method to improve image contrast



Low contrast image: histogram is narrow



High contrast image: histogram is more spread out!
Image has higher dynamic range – details more clearly visible

Histogram equalization

- Seeks to apply an intensity transformation to the pixel intensities of a low-contrast image so as to convert it into one with a (nearly) **uniform histogram**.

- A uniform histogram is a histogram with Q bins where the probability mass in each bin is 1/Q.

- Principle: images with uniform histogram (or more spread out histograms) will have better contrast.

- Consider $S = T(R)$, where R = random variable standing for intensity in the original image (assume r lies from 0 to L-1).

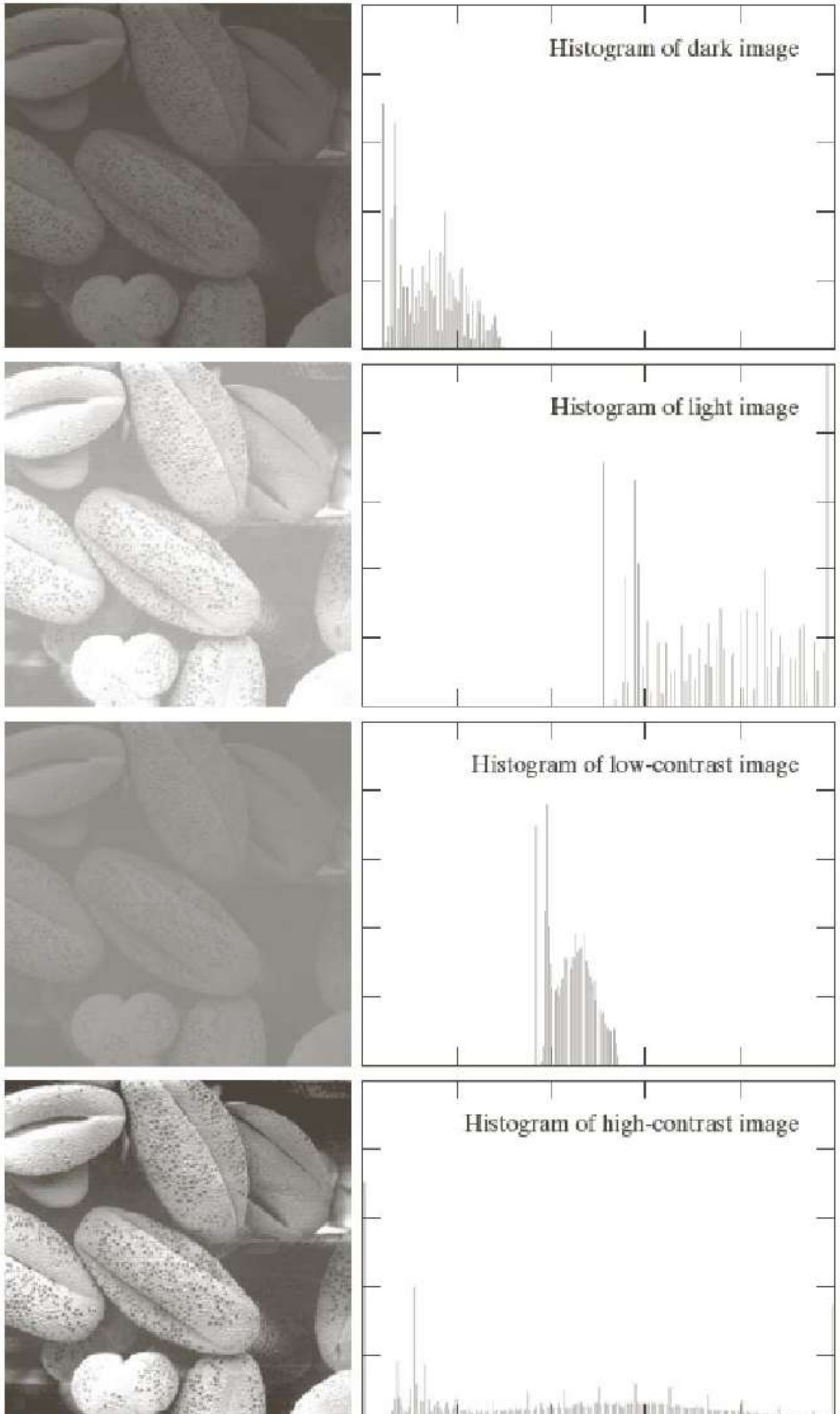


FIGURE 3.16 Four basic image types: dark, light, low contrast, high contrast, and their corresponding histograms.

Histogram equalization

- Consider $S = T(R)$, where R = random variable standing for intensity in the original image (assume R lies from 0 to L-1).
- S stands for the transformed random variable.
- The values that S and R acquire are denoted as s, r respectively.
- In particular, we are interested in T which satisfies the following two conditions:
- It should be strictly monotonically increasing
- If $0 <= r <= L-1$, then $0 <= T(r) <= L-1$ as well.

Segway: Probability refresher

- A random variable is the outcome of a randomized experiment.
- Eg: the number of heads appearing in N independent coin tosses, the time at which the first head appears in N independent coin tosses, etc.
- A random variable is denoted by upper case alphabets and its values with lower cases.
- A random variable may be discrete or continuous valued.
- It is characterized by its Probability density function (PDF) in continuous cases or its Probability mass function (PMF) in discrete cases.
- Denoted as $p_R(r)$ – acquires values between 0 and 1 and integrates to 1.

Histogram equalization

- The monotonicity condition prevents artifacts due to intensity reversals.

- The second condition prevents change of the intensity range of the image.

Histogram equalization

- If **R** has a probability density $p_R(r)$, then S has the following probability density:

$$p_S(s) = p_S(T(r)) = \frac{p_R(r)}{|T'(r)|}$$

Called as "Transformation of random variables" (eg: https://www.math.arizona.edu/~jwatkins/f-transform.pdf)

- Consider the following transformation:

$$s = T(r) = (L-1)\boxed{\int_0^r p_R(w)dw}$$

Cumulative distribution function (cdf) of R. It satisfies the monotonicity as well as range-based criteria mentioned earlier.

Histogram equalization

- Then, we can show that the probability density of S is given as:

$$p_S(s) = \frac{p_R(r)}{|T'(r)|} = \frac{p_R(r)}{(L-1)\,p_R(r)} = \frac{1}{(L-1)}$$

$$T'(r) = (L-1)\frac{d}{dr}\left(\int_0^r p_R(w)\,dw\right) = (L-1)\,p_R(r)$$

- Thus S has a uniform PDF, **independent** of the original PDF of R.

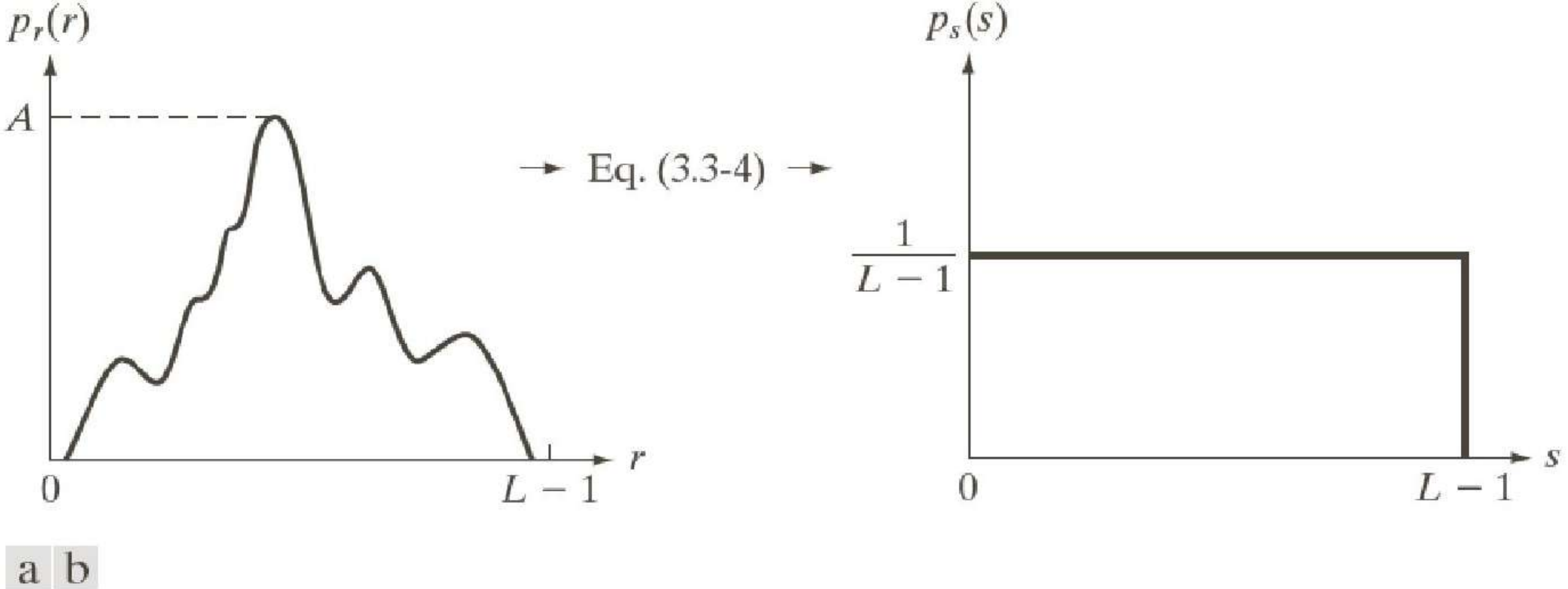
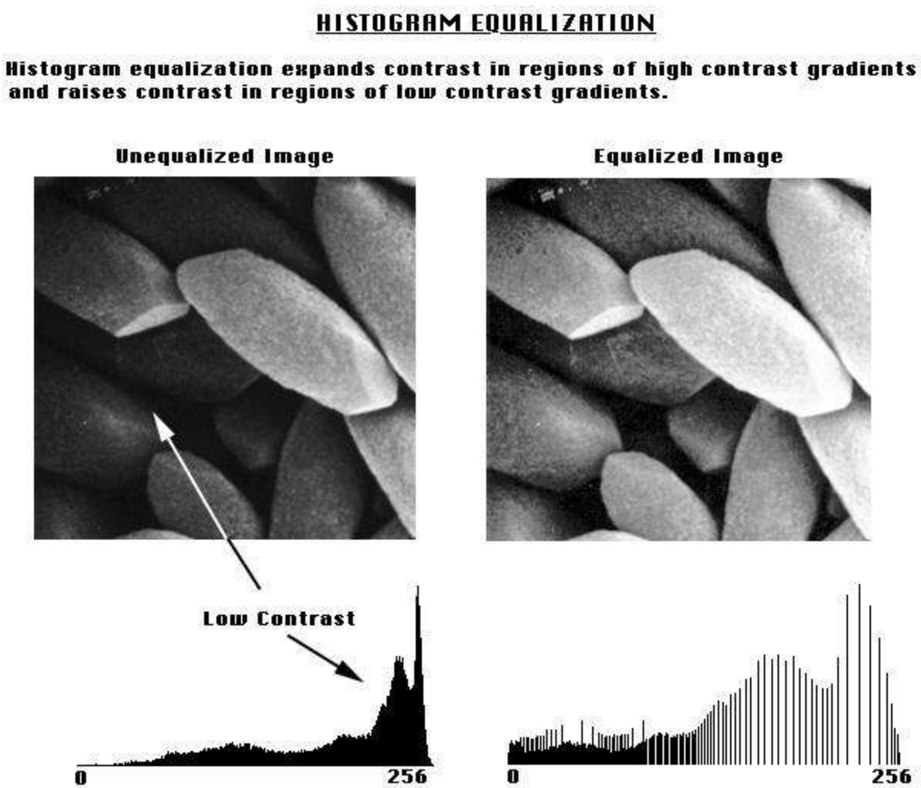


FIGURE 3.18 (a) An arbitrary PDF. (b) Result of applying the transformation in Eq. (3.3-4) to all intensity levels, r. The resulting intensities, s, have a uniform PDF, independently of the form of the PDF of the r's.

Histogram equalization

- Thus to perform this procedure, replace intensity value r in an image by the cumulative density (cdf) of r, times the maximum intensity level in the image minus 1!



http://www.udel.edu/biology/Wags/b617/digital/digital14.gif

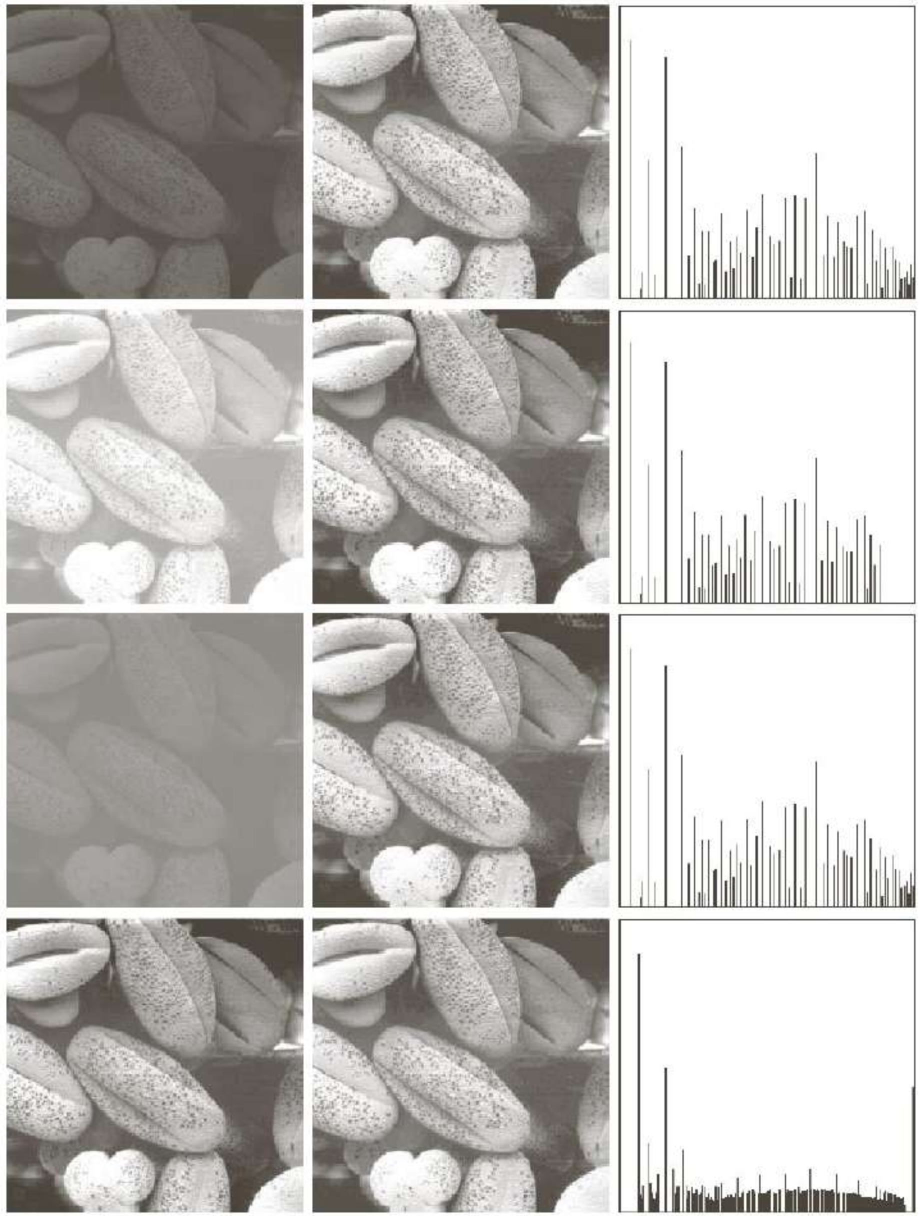


FIGURE 3.20 Left column: images from Fig. 3.16. Center column: corresponding histogram-equalized images. Right column: histograms of the images in the center column.

Histogram equalization

- When you implement this, we change the integrals to a discrete summation.
- Thus we have:

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

- Thus you go to each pixel with value r_k, replace it by s_k, and repeat this for every intensity value r_k.

A 3-bit 64x64 image has the following intensities:

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

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

Applying histogram equalization:

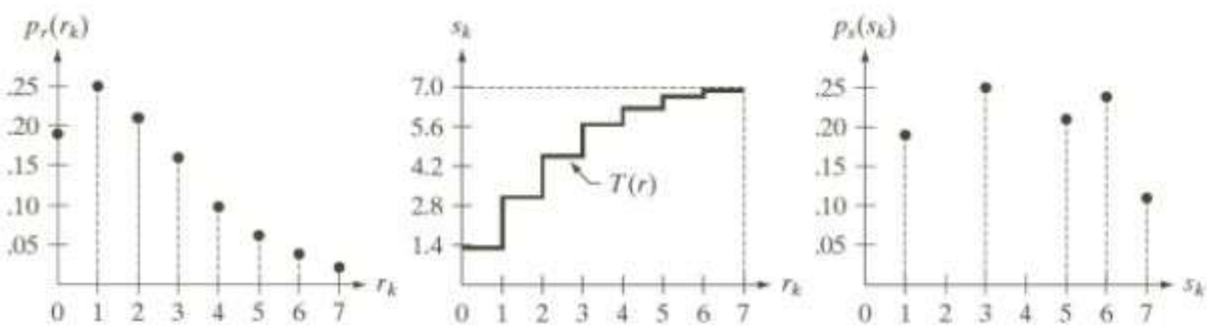
$$s_0 = T(r_0) = 7\sum_{j=0}^{0} p_r(r_j) = 7p_r(r_0) = 1.33$$

$$s_1 = T(r_1) = 7\sum_{j=0}^{1} p_r(r_j) = 7p_r(r_0) + 7p_r(r_1) = 3.08$$

Rounding to the nearest integer:

$$s_0 = 1.33 \rightarrow 1 \quad s_1 = 3.08 \rightarrow 3 \quad s_2 = 4.55 \rightarrow 5 \quad s_3 = 5.67 \rightarrow 6$$

$$s_4 = 6.23 \rightarrow 6 \quad s_5 = 6.65 \rightarrow 7 \quad s_6 = 6.86 \rightarrow 7 \quad s_7 = 7.00 \rightarrow 7$$



Histogram equalization

- The discrete histogram of an equalized image is more spread out. But it is not exactly uniform.
- Why?
  - Quantization (rounding off) of intensity values
  - Intensity range missing in the original image

https://towardsdatascience.com/histogram-matching-ee3a67b4cbc1

Histogram specification (also called matching)

- Apply an intensity transformation to an existing image so that the resultant image has **some** pre-specified histogram (perhaps corresponding to some other image).

- Equalization – special case of specification where the pre-specified histogram is the uniform histogram!

Histogram specification

- Let $p_R(r)$ = histogram of the original image.
- Let $p_Z(z)$ = pre-specified histogram.
- We apply the following transformation to r:

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

Consider random variable Z with the property that :

$$G(z) = (L-1)\int_0^z p_Z(v)dv$$

Now as $G(z) = T(r)$,

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

Histogram specification

- The pdf of the transformed image will be $p_Z(z)$.

- Problem: the specified histogram needs to be chosen carefully – by trial and error.

Proof

$R =$ intensity of image to be transformed

$Z =$ intensity of target image

$H =$ intensity of transformed image $= G^{-1}(T(R))$

$$\text{Recall}: T(r) = (L-1)\int_0^r p_R(w)dw; G(z) = (L-1)\int_0^z p_Z(w)dw$$

$$p_H(z) = \frac{p_R(r)}{d\left(G^{-1}(T(r))\right)/dr} = \frac{p_R(r)}{\left(G^{-1}\right)'(T(r))T'(r)} = \frac{p_R(r)}{\left(G^{-1}\right)'(T(r))(L-1)p_R(r)}$$

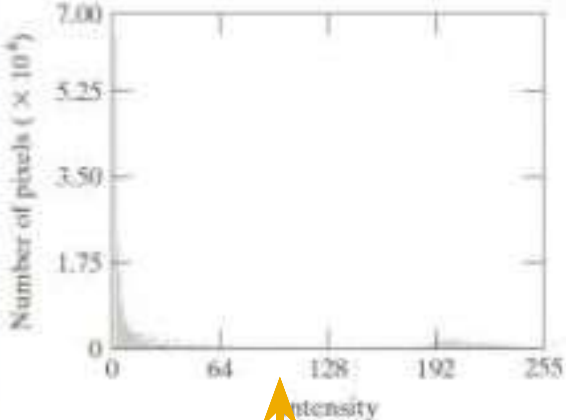
$$= \frac{1}{\left(G^{-1}\right)'(T(r))(L-1)} = \frac{G'(G^{-1}[T(r)])}{(L-1)} = \frac{(L-1)p_Z(z)}{(L-1)} = p_Z(z)$$

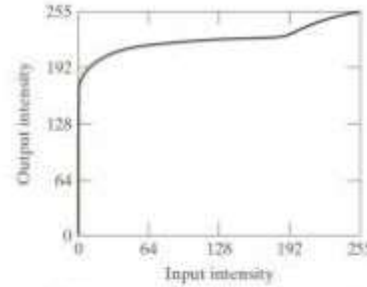
Inverse function theorem

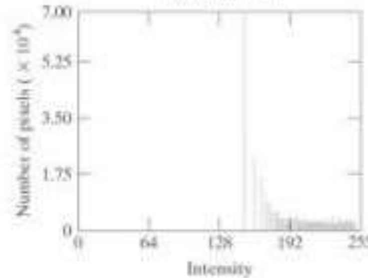
Chain rule

Histogram specification

- Let R be the image which needs to be transformed so that its histogram will now resemble the histogram of image Z.
- Compute $p_R(r)$ and $p_Z(z)$.
- Compute the functions $s = T(r)$ and $G(z)$. Then compute $G^{-1}(s)$.
- The histogram of the transformed image containing values $G^{-1}(s)$ will have PDF $p_Z(z)$.

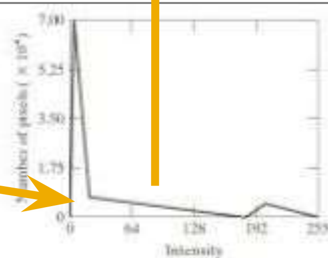




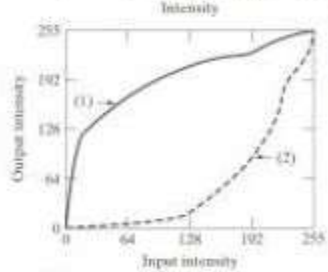


Equalization

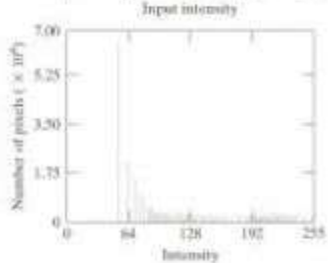
Specified histogram



Transformation function and its inverse



Resulting histogram



Histogram specification versus equalization

- Histogram equalization produces an image with a washed out appearance.
- Why?
- The original image histogram has a number of dark pixels, followed by an empty range, and a moderate number of bright pixels.
- HE maps a narrow interval of dark pixels to the brighter side of the grayscale range (see the associated transformation function on the previous slide), causing a washed out appearance.

Histogram specification versus equalization

- We want a target histogram which has a smoother transition from dark to light pixels.
- Hence the particular histogram chosen on the previous slides.
- Notice that this produces an image with improved appearance.
- Read example 3.9 of the textbook by Gonzalez.

https://towardsdatascience.com/histogram-matching-ee3a67b4cbc1

Example

$p_R(r) = 2r/(L-1)^2$ when $0 \le r \le L-1$ and 0 otherwise

Target histogram $p_Z(z) = 3z^2/(L-1)^3$ when $0 \le z \le L-1$ and 0 otherwise

$$s = T(r) = (L-1)\int_0^r p_R(w)dw = r^2/(L-1)$$

$$G(z) = (L-1)\int_0^z p_Z(w)dw = \frac{3}{(L-1)^2}\int_0^z w^2 dw = z^3/(L-1)^2$$

$$G(z) = T(r) \rightarrow z = \left((L-1)^2 s\right)^{1/3} = \left((L-1)^2 r^2/(L-1)\right)^{1/3}$$

$$= \left((L-1)r^2\right)^{1/3}$$

Discrete Representation

$$s_k = T(r_k) = (L-1)\sum_{j=0}^{k} p_R(r_j)$$

$$G(z_q) = (L-1)\sum_{i=0}^{q} p_Z(z_i) = s_k$$

$$z_q = G^{-1}(s_k), \tilde{z}_q = round(z_q)$$

The inverse of G is computed via a lookup table. In practice, one evaluates G at all integer values from 0 to L-1. We use the stored values of G to find the integer z_q such that $G(z_q)$ is closest to s_k for each s_k value.

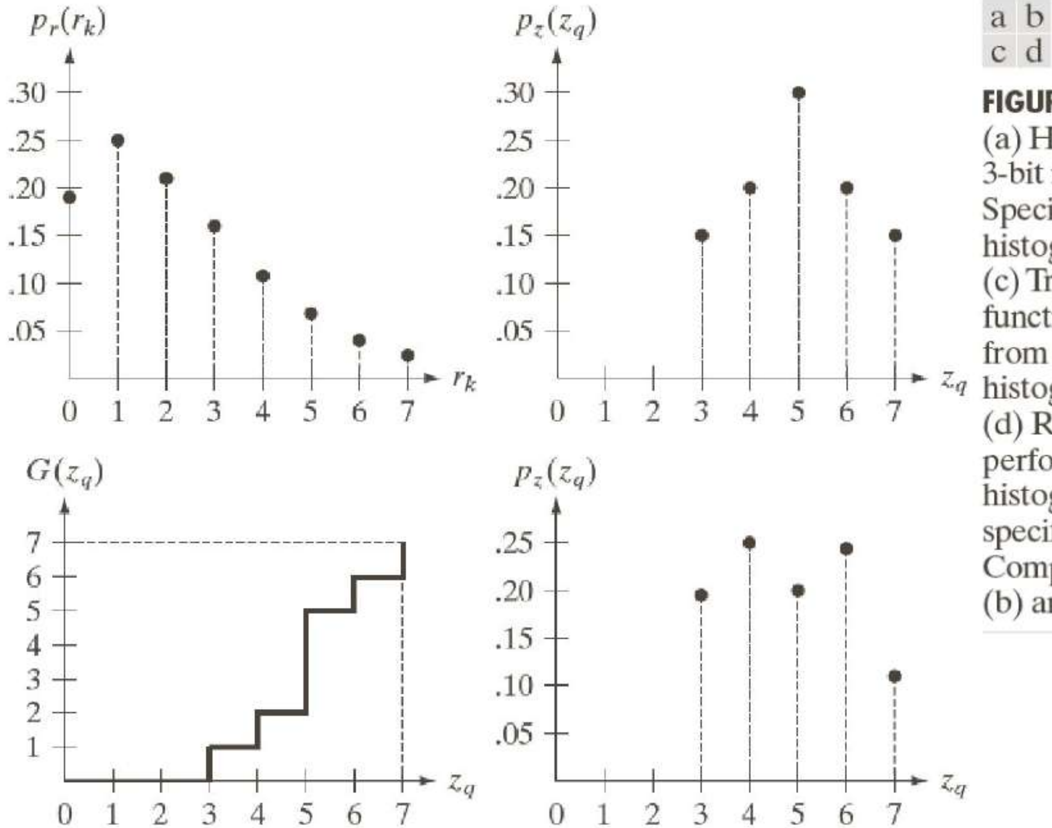


FIGURE 3.22 (a) Histogram of a 3-bit image. (b) Specified histogram. (c) Transformation function obtained from the specified histogram. (d) Result of performing histogram specification. Compare (b) and (d).

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).

Trial and error

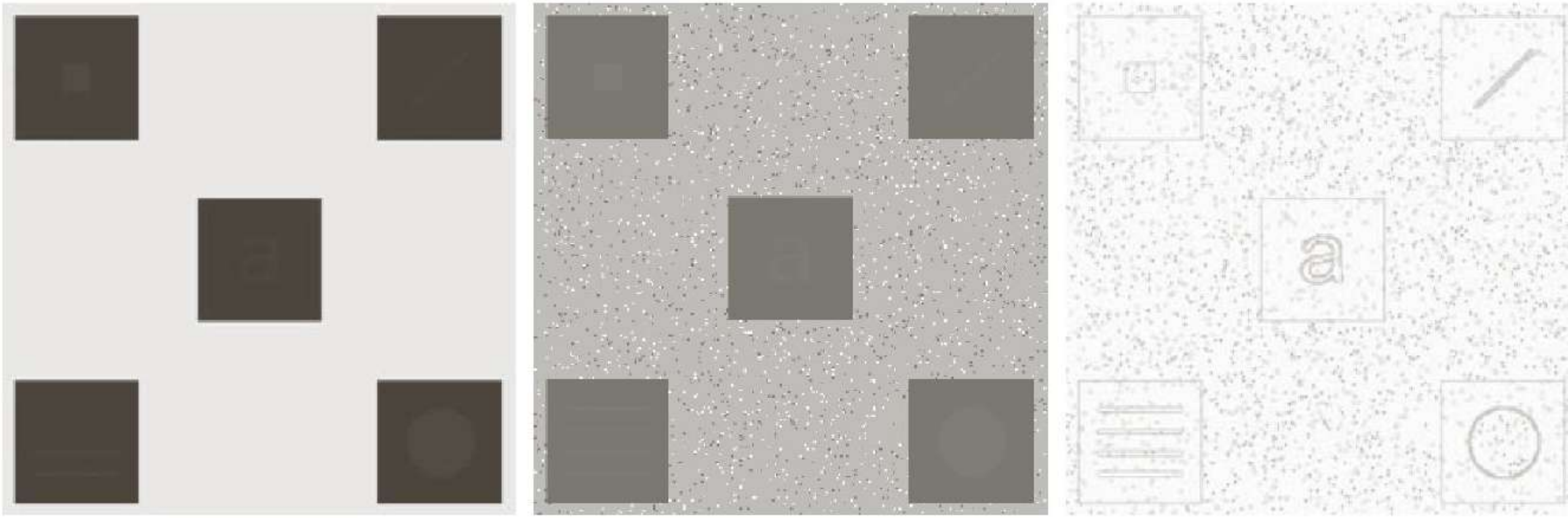
- HE and HS are both image enhancement methods.
- The choice of whether to use HE or HS, or which histogram to use for the specification in HS, are left to trial and error.
- There is no universal way of specifying a histogram for HS.
- In fact, HE or HS can be even be done at a “local” level – see next slide.

Local Histogram Equalization

- HE/HS – studied so far – are **global** methods, i.e. pixel intensities are modified by a transformation function affected by intensity values of the **entire** image.
- This approach sometimes does not enhance details in small areas of an image.
- Because the number of pixels in small image areas may have an insufficient influence on the global transformation.
- Solution: devise transformations based on local histograms.

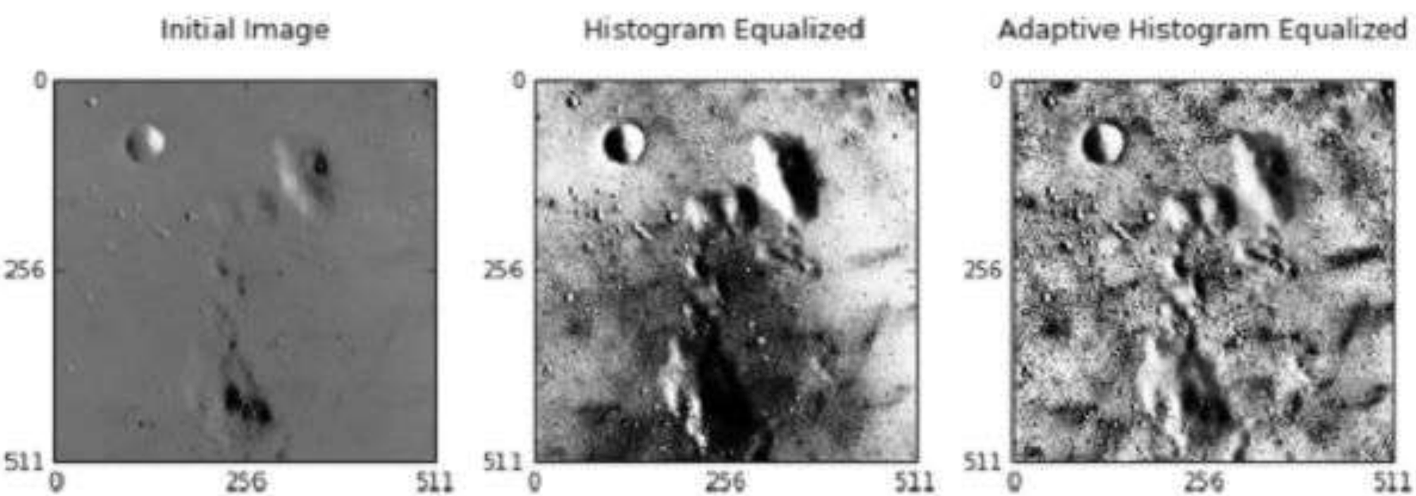
Local (also called Adaptive) Histogram Equalization

- Define a symmetrical neighborhood N(x,y) around any pixel (x,y).
- The neighborhood is typically rectangular and has size (say) K x K.
- At each (x,y), compute the histogram of intensity values confined to N(x,y).
- Compute a transformation based on this local histogram using either HE or HS.
- Replace the intensity value of only (x,y) by the transformed intensity.
- Repeat for the entire image.
- See example on next slide – where global HE (GHE) is unable to enhance the local details in the black squares, but local HE (LHE) with a window size of 3 x 3 is.



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 means same as local

https://towardsdatascience.com/histogram-equalization-5d1013626e64

Original Image

https://cromwell-intl.com/3d/histogram/

Global histogram-equalized image (see next slide for the results with local HE)

Local
histogram-equalized
image (size 100 x 100)

Local
histogram-equalized
image (size 50 x 50)

Ignore the fact that these are color images because color image HE is slightly different and we will study it later. Concentrate on the advantages of local over global.

Spatial Filters

Local Spatial Filters

- Change pixel values based on some weighted combination of pixels from a small (often rectangular) neighborhood around the pixel.
- This is represented as follows:

Output image

$$g(x,y)=\sum_{i=-a}^{a}\sum_{j=-a}^{a} w(i,j)\,f(x+i,y+j)$$

Input image

Weights

- The formula is applied for each pixel (x,y) in the pixel domain, using the input signal values.
- The output consists of a new "filtered" image g. In most cases, the new values do not replace the values in the original image f.
- The **weights** in w and the **neighborhood size** are both characteristic features of the filter.

Local Spatial Filters

- If the operation performed on the image pixels is linear, we call it a linear filter, otherwise we call it a non-linear filter.

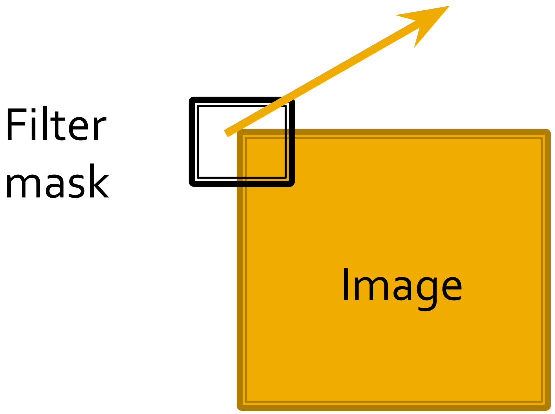
- The filtered image is generated as the center of the filter (w(0,0)) visits each pixel (x,y) of the input image f.

$$\text{Example}: a = b = 1$$

$$g(x,y) = w(-1,-1)f(x-1,y-1) + w(-1,0)f(x-1,y) + w(0,-1)f(x,y-1) +$$
$$w(0,0)f(x,y) + w(1,1)f(x+1,y+1) + w(1,0)f(x+1,y) + w(0,1)f(x,y+1)$$

Local Spatial Filters

- Note: in order to apply a filter to an image, it will usually require zero-padding to accommodate border areas.



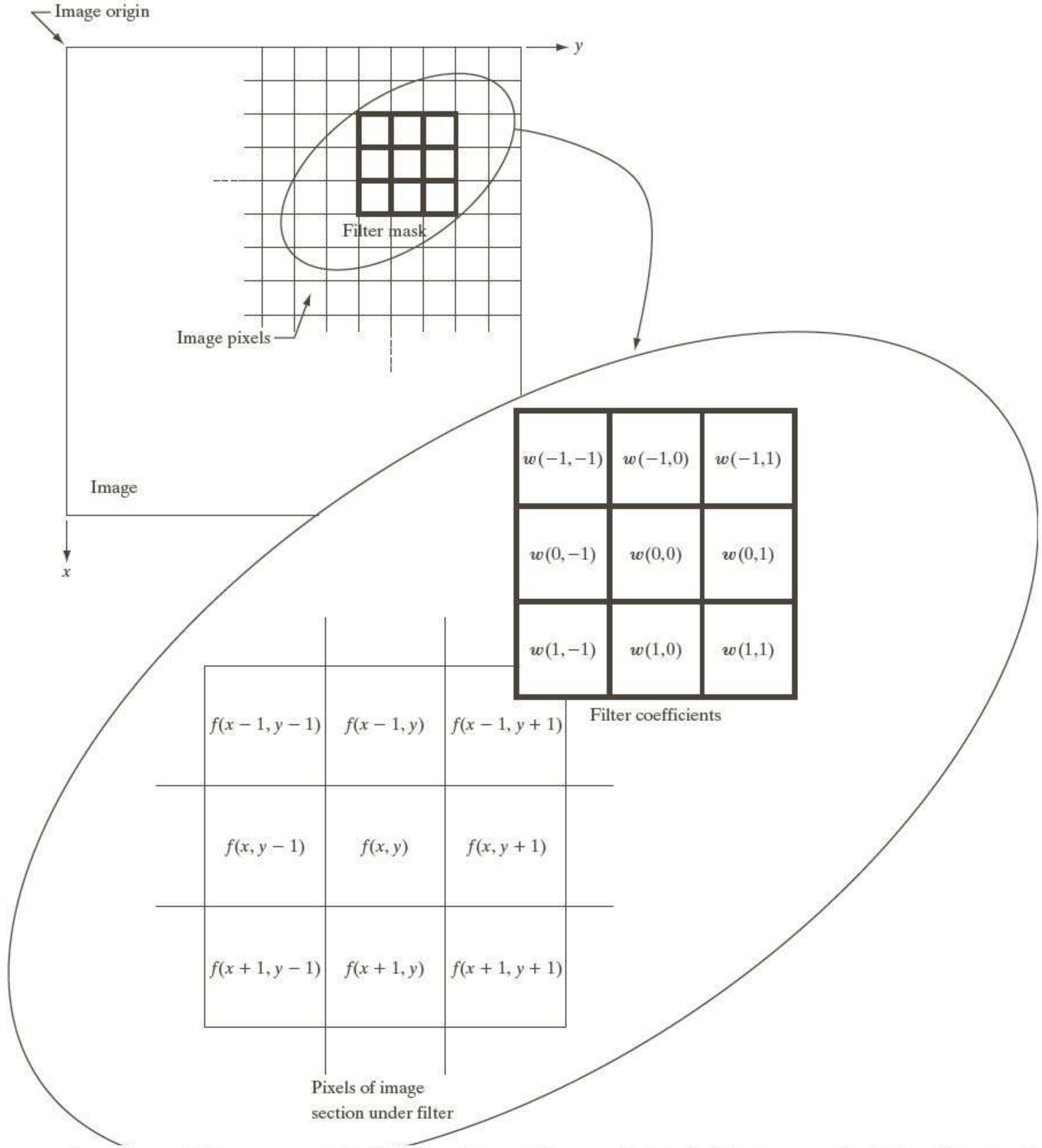


FIGURE 3.28 The mechanics of linear spatial filtering using a 3×3 filter mask. The form chosen to denote the coordinates of the filter mask coefficients simplifies writing expressions for linear filtering.

Correlation and convolution

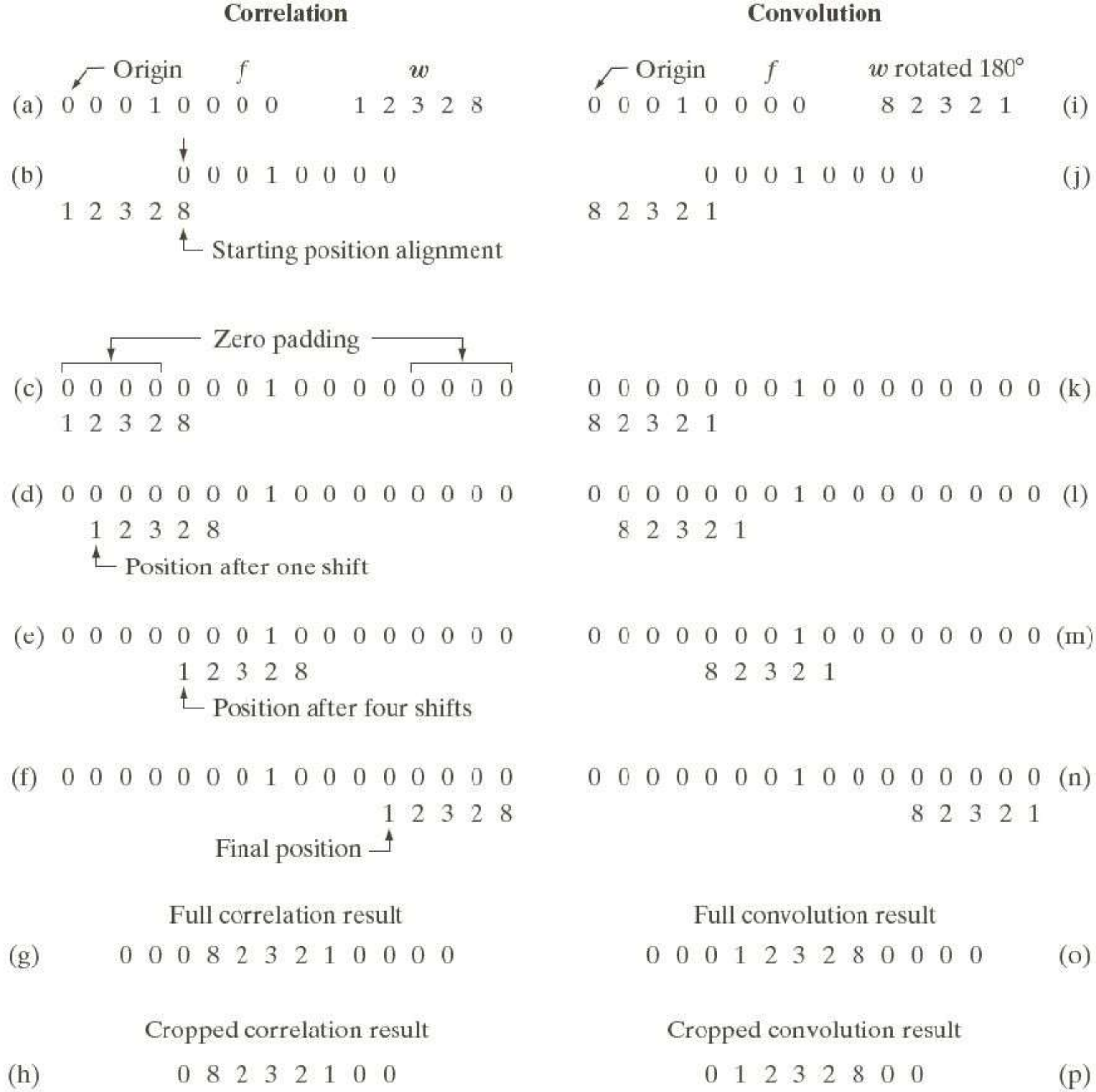
- Local spatial filters involve one of two operations: correlation and convolution

- **Correlation** = moving a filter mask (2D array consisting of filter weights) over the image and computing the sum of products at each location

- This is identical to the previous mathematical formula.

Correlation and convolution

- **Convolution** = identical to correlation except that the filter is **first rotated by 180 degrees** (in both X and Y directions if it is in 2D) before moving over the image and computing sum of products.



For a filter of size m (in 1D), pad with m-1 zeros on either side

FIGURE 3.29 Illustration of 1-D correlation and convolution of a filter with a discrete unit impulse. Note that correlation and convolution are functions of *displacement*.

Correlation and convolution

- In the preceding example, we consider a 1D image with all zeros except a single one - called a **unit impulse** (or a **Kronecker delta**) function.
- The result of correlation with a filter with mask *w* is a reverse copy of *w*, centered at the location of the impulse.
- The result of convolution with a filter with mask *w* is a copy of *w*, centered at the location of the impulse.
- We will see the same result on the next slide in 2D.

Origin $f(x, y)$

0	0	0	0	0
0	0	0	0	0
0	0	1	0	0
0	0	0	0	0
0	0	0	0	0

$w(x, y)$

1	2	3
4	5	6
7	8	9

(a)

Padded f

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(b)

Initial position for w

1	2	3	0	0	0	0	0	0
4	5	6	0	0	0	0	0	0
7	8	9	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(c)

Full correlation result

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	9	8	7	0	0	0
0	0	0	6	5	4	0	0	0
0	0	0	3	2	1	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(d)

Cropped correlation result

0	0	0	0	0
0	9	8	7	0
0	6	5	4	0
0	3	2	1	0
0	0	0	0	0

(e)

Rotated w

9	8	7	0	0	0	0	0	0
6	5	4	0	0	0	0	0	0
3	2	1	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	1	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(f)

Full convolution result

0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	1	2	3	0	0	0
0	0	0	4	5	6	0	0	0
0	0	0	7	8	9	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0

(g)

Cropped convolution result

0	0	0	0	0
0	1	2	3	0
0	4	5	6	0
0	7	8	9	0
0	0	0	0	0

(h)

FIGURE 3.30 Correlation (middle row) and convolution (last row) of a 2-D filter with a 2-D discrete, unit impulse. The 0s are shown in gray to simplify visual analysis.

Correlation and convolution

$$(w \otimes f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x+s, y+t)$$

Correlation (denoted by an empty asterisk in the book)

$$(w * f)(x,y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(s,t) f(x-s, y-t)$$

Convolution (denoted by a solid black asterisk in the book)

Notes:

- The negative signs in the formula for the convolution, which represents flipping by 180 degrees. You would get similar formulae in 1D as well.
- Intuitively: correlation between a mask and a unit impulse signal yields a reversed copy of the mask. If we pre-rotate the mask by 180 degrees, we will now get a copy of the mask.
- If w were symmetric, then convolution and correlation yield identical results.

Correlation and convolution

- Convolution is a commutative operation!

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

$$= \sum_{x'=\infty}^{-\infty}\sum_{y'=\infty}^{-\infty} w(x-x',y-y')f(x',y'); x'=x-s, y'=y-t$$

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

$$= (f * w)(x,y)$$

After zero-padding w and f so that they have infinite extent. Note this doesn't change the summation value.

- Repeat this exercise for correlation - it is [NOT commutative](#) (work out a counter-example).

Correlation and convolution

- Convolution is an associative operation!

$$((x * y) * z)(t) = [x * (y * z)](t)$$

- Correlation is not associative.

- The proofs of these results are much easier when using Fourier transforms (which we will study later).

Genesis of convolution

- Consider a system which takes in input signal $x(t)$ and produces an output $y(t)$ where $y(t) = T[x(t)]$.
- T is the transformation that is executed by the system.
- Consider that T satisfies two conditions: linearity and time-invariance.

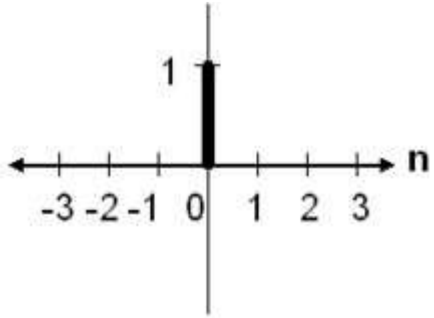
Genesis of convolution

- Consider that T satisfies two conditions: linearity and time-invariance.

$$\text{Linearity}: T[ax_1(t)+x_2(t)] = aT[x_1(t)]+T[x_2(t)]$$

$$\text{Time invariance}: \text{If } y(t)=T[x(t)], \text{ then } y(t-s)=T[x(t-s)]$$

- Let the signal x be a unit impulse δ(t) at 0 (Kronecker delta function).



$$\delta(t)=1 \text{ for } t=0, \text{ otherwise } 0$$

Genesis of convolution

- The output of the system for a unit impulse input is called its impulse response h(t).

$$h(t) = T[\delta(t)]$$

- It turns out that the response of a linear time-invariant (LTI) system to any input signal x is the convolution of the signal with the impulse response of the LTI system.

Genesis of convolution: Proof

$$h(t) = T[\delta(t)]$$

We can express $x(t) = \sum_{u=-\infty}^{\infty} \delta(t-u)x(u)$ often called sifting property of δ

$$\therefore \mathrm{T}[\,x(t)] = T\left[\sum_{u=-\infty}^{\infty} \delta(t-u)x(u)\right]$$

$$= \sum_{u=-\infty}^{\infty} T[\delta(t-u)x(u)] \text{ by linearity}$$

$$= \sum_{u=-\infty}^{\infty} T[\delta(t-u)]x(u) \text{ as } x(u) \text{ is independent of } h$$

$$= \sum_{u=-\infty}^{\infty} h(t-u)x(u) \text{ due to time invariance}$$

$$= (h * x)(t) = (x * h)(t)$$

Genesis of correlation

- Correlation gives the similarity between a signal w and (a shifted version) of a signal f.

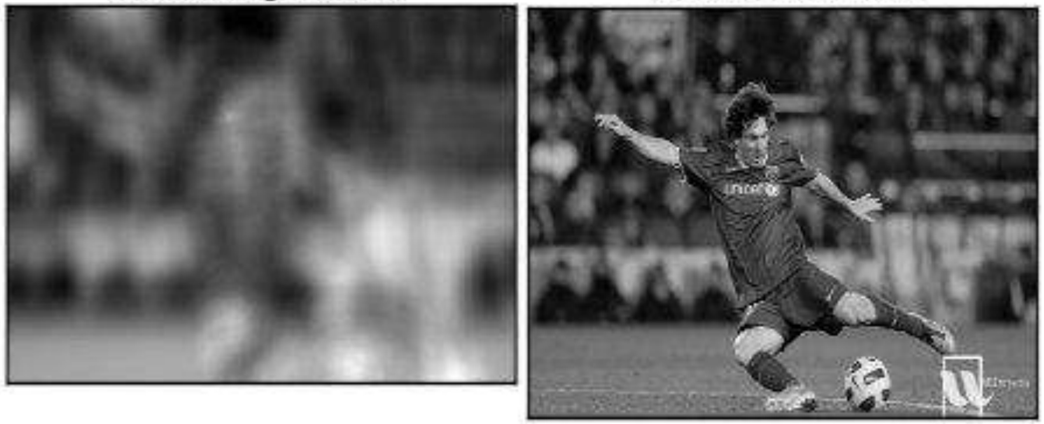
$$(w \otimes f)(x) = \sum_{t=-b}^{b} w(t) f(x+t) \text{ in 1D}$$

$$(w \otimes f)(x, y) = \sum_{s=-a}^{a} \sum_{t=-b}^{b} w(t, s) f(x+t, y+s) \text{ in 2D}$$

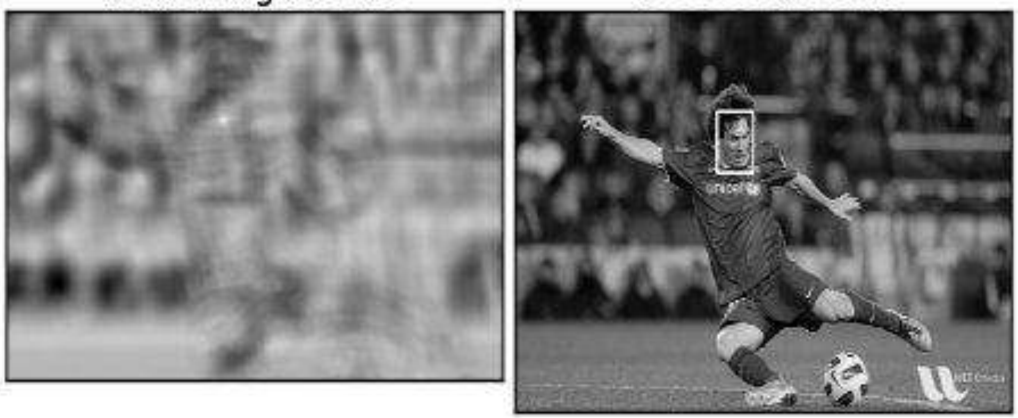
- Correlation is large for those shifts x for which w and shifted f have similar values, or even similar signs.

- It can be used for template matching.

Matching Result | Detected Point



Matching Result | Detected Point



Search for the template (Messi's face) inside the larger image in the middle.

First row: On the left you have the cross-correlation map

Second row: On the left you have the **normalized** cross-correlation map

Why is the latter superior to the former? Notice the false match in the top row and the correct match in the second one.

https://docs.opencv.org/4.5.2/d4/dc6/tutorial_py_template_matching.html

$$NCC(w,f)(x) = \frac{\sum_{t=-b}^{b} w(t) f(x+t)}{\sqrt{\sum_{t=-b}^{b} (w(t)-\mu_w)^2 \sum_{t=-b}^{b} (f(x+t)-\mu_f)^2}} \text{ in 1D}$$

Work it out in 2D

Types of image noise

- Gaussian noise: addition of perturbations to an image, the perturbations are random numbers from the Gaussian distribution.

$$f(x,y) = f_c(x,y) + \eta, \eta \sim N(0,\sigma)$$

Symbol for Gaussian distribution with mean 0, standard deviation σ

$$G(x;\mu,\sigma) = \frac{e^{-(x-\mu)^2/(2\sigma^2)}}{\sigma\sqrt{2\pi}}$$

Formula for Gaussian distribution with mean μ, standard deviation σ

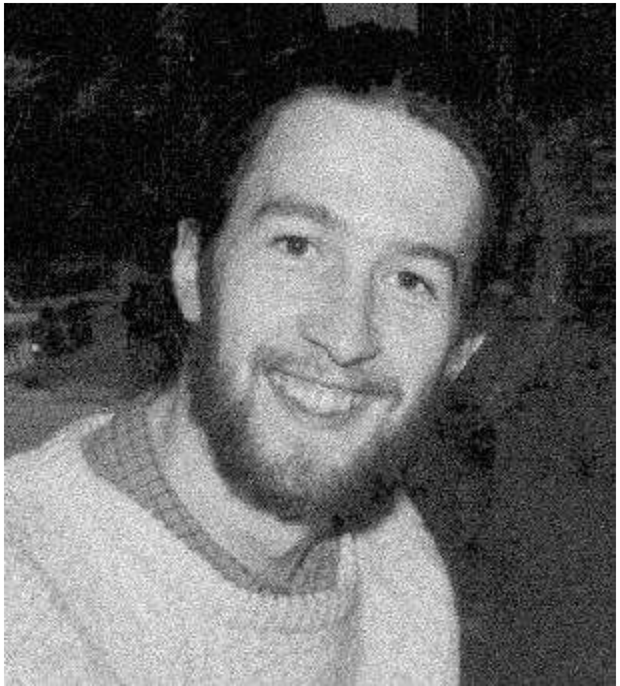
- Examples of Gaussian noise: Film grain noise, thermal noise in a camera

Types of image noise

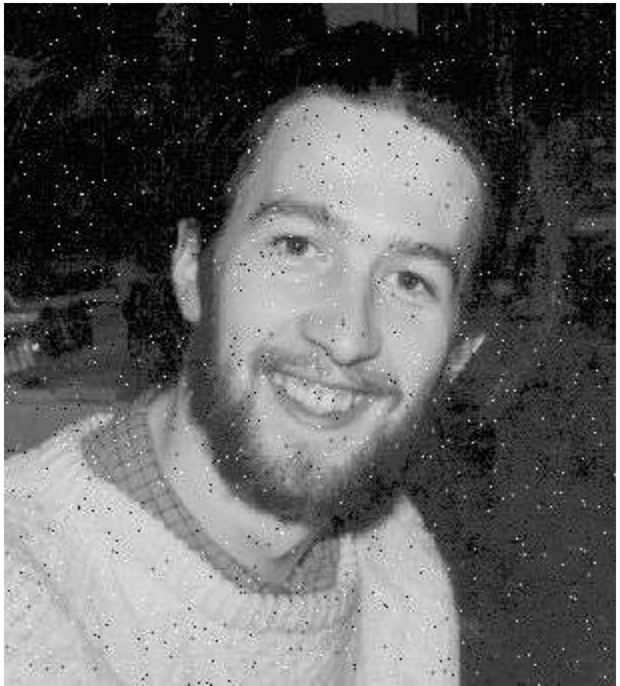
- Impulse noise: random **large magnitude** perturbations at a **few** pixels – example: flicker artifacts in old movies, or blotches in old photographs.
- Note: Gaussian noise typically affects **all** pixels but by a **smaller** magnitude.

Low σ, Gaussian

High σ, Gaussian



Impulse



Mean filter

- Replace the central pixel value by arithmetic mean of all surrounding pixel values.
- This acts as a **smoothing** filter.
- Image noise means random transition in the intensity values – these get attenuated by mean filter.

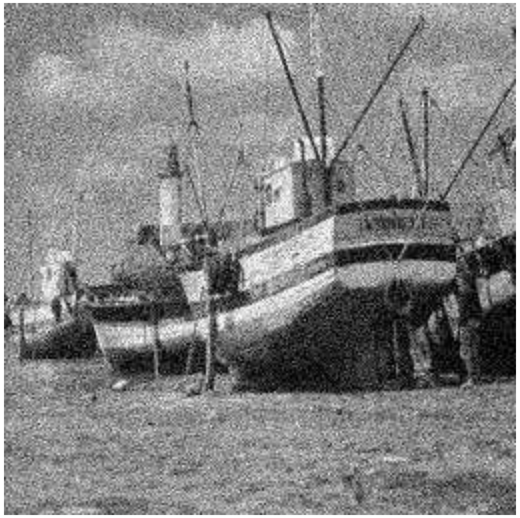
$$g(x,y) = \frac{1}{(2a+1)^2} \sum_{i=-a}^{a} \sum_{j=-a}^{a} f(x+j, y+i)$$

- But edges (boundaries between regions with different colors) get blurred as well ☹

Mean filter

- Implemented by convolving the image with a mask contains all values equal to $1/(2a+1)^2$.

Original image



Noisy image

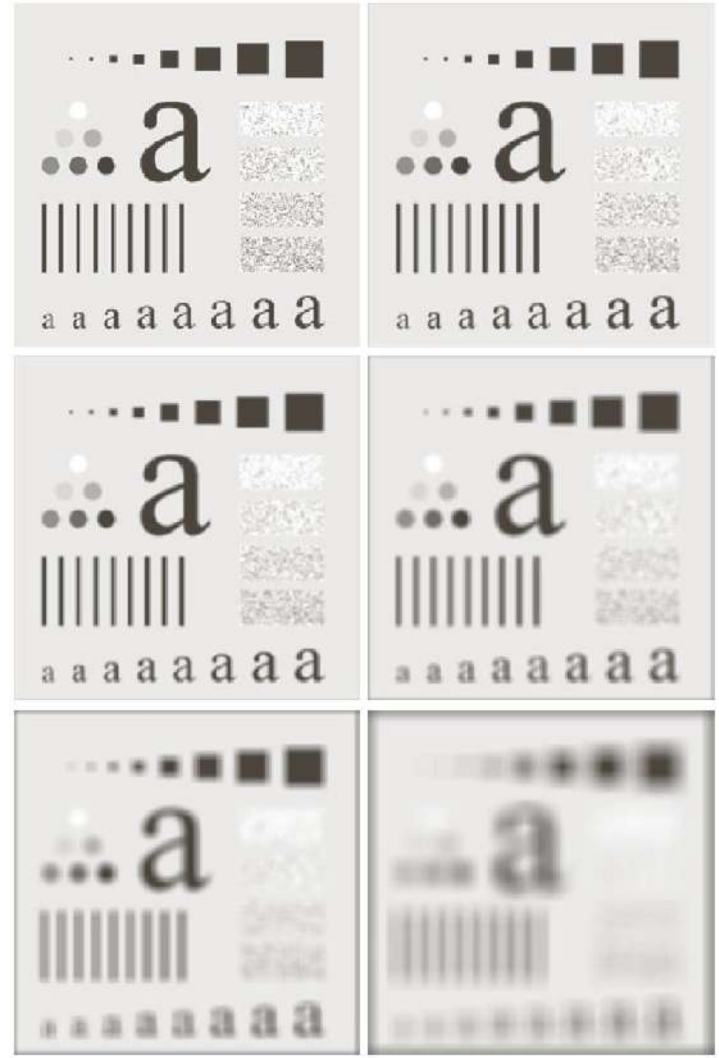
Filtered image

-Amount of smoothing is directly proportional to the width of the filter window.

-Repeated application of a mean filter will lead to a constant intensity image in the limit of infinite iterations

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.

a b
c d
e f



Weighted mean filter

- Assign different weights to each pixel: weigh pixels located closer to the center of the window more than those towards the edge of the window.

$$g(x,y)=\frac{\sum_{i=-a}^{a}\sum_{j=-a}^{a} f(x+j,y+i)e^{-(j^2+i^2)/(2\sigma^2)}}{\sum_{i=-a}^{a}\sum_{j=-a}^{a} e^{-(j^2+i^2)/(2\sigma^2)}}$$

Decay parameter

Gaussian weights

$$G(i,j)=\frac{e^{-(i^2+j^2)/(2\sigma^2)}}{\sigma\sqrt{2\pi}}$$

Called the 2D Gaussian function

Weighted mean filter

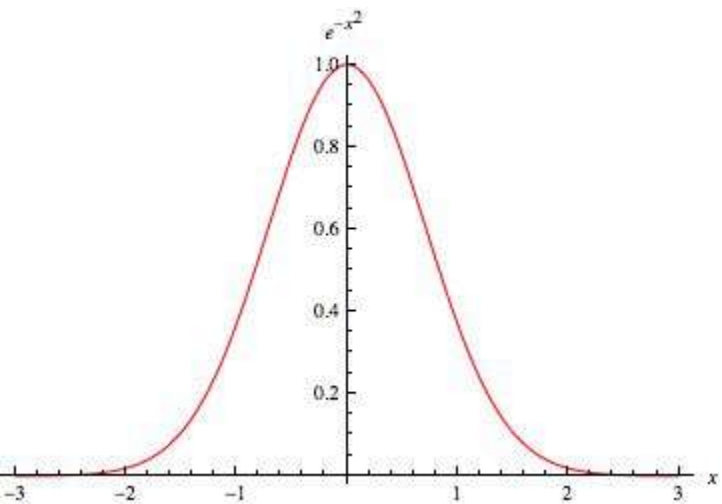
- Also does not preserve edges.
- Note: we want all the weights to sum up to 1, otherwise the dynamic range of the image will not be preserved!
- What's special about the Gaussian function? It gives more weights to image pixels near the center of the mask and lower weights to pixels farther away from the center.

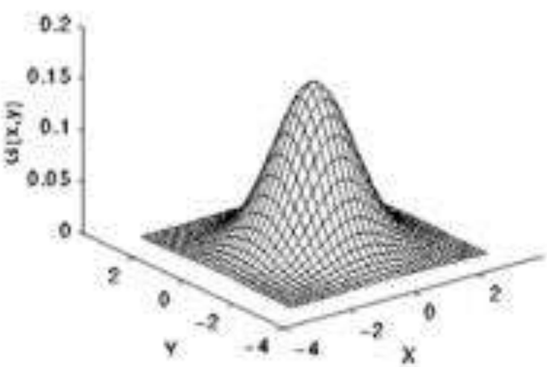
Weighted mean filter

- Implemented by convolving the image with a mask containing weights of the form

$$\frac{e^{-(j^2+i^2)/(2\sigma^2)}}{\sum_{i=-a}^{a}\sum_{j=-a}^{a} e^{-(j^2+i^2)/(2\sigma^2)}}$$

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





Median Filter

- When there are wild outliers (such as impulse noise, also called salt and pepper noise) in the image, the mean filter gives a poor response.
- Common when there are data transmission errors, common in old film recordings.
- Median: more robust to large outliers in a given dataset.
- Median filters – give better preservation of features under impulse noise than mean filter.
- Equation for median filter:

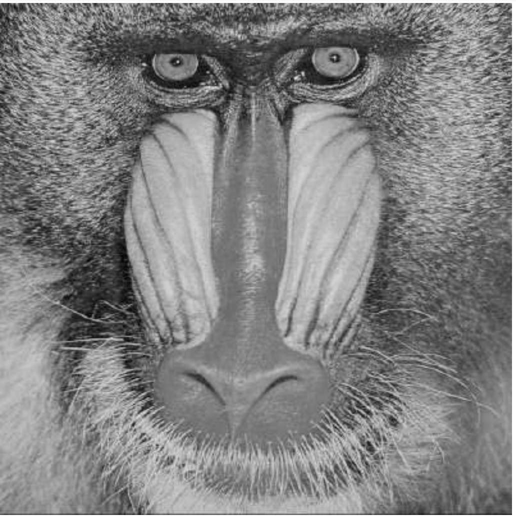
$$g(x, y) = \text{median of all values in the image } f \text{ from location}$$

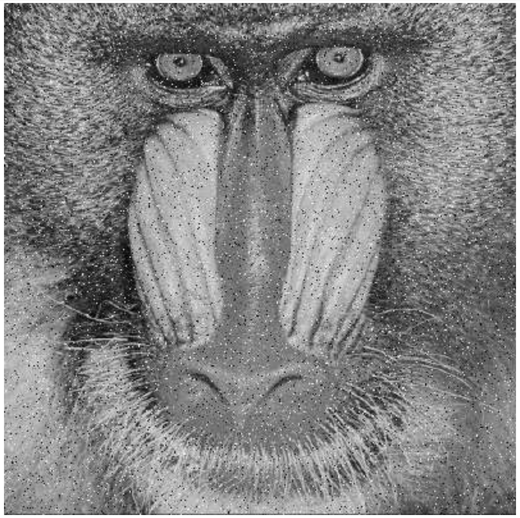
$$(x-a, y-a) \text{ to } (x+a, y+a)$$

Median versus mean: try out

- Compute mean and median of [1:100], i.e. an array of all integers from 1 to 100.
- Change one of the numbers in the array to a large value (like 10000)
- Re-compute the mean and median.
- The mean changes drastically, the median remains almost the same – as long as these drastic changes affect less than half the number of elements in the array.

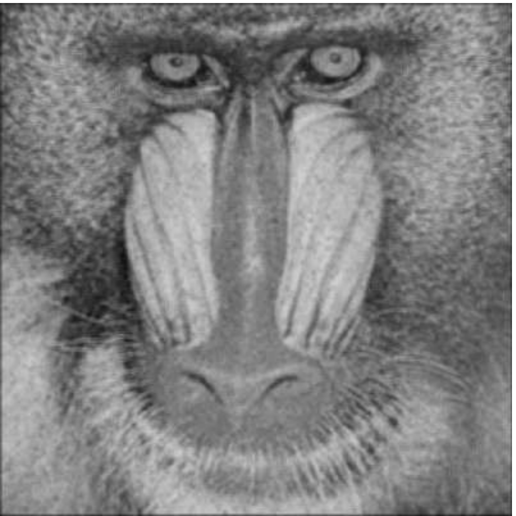
Median Filter

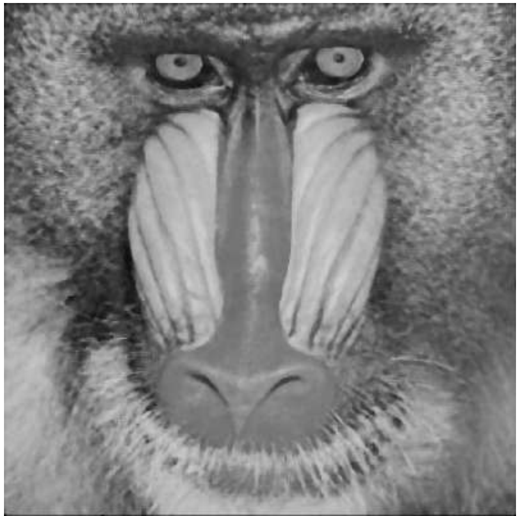




5 x 5 Mean filter

5 x 5 Median filter





Much better feature preservation with median filter

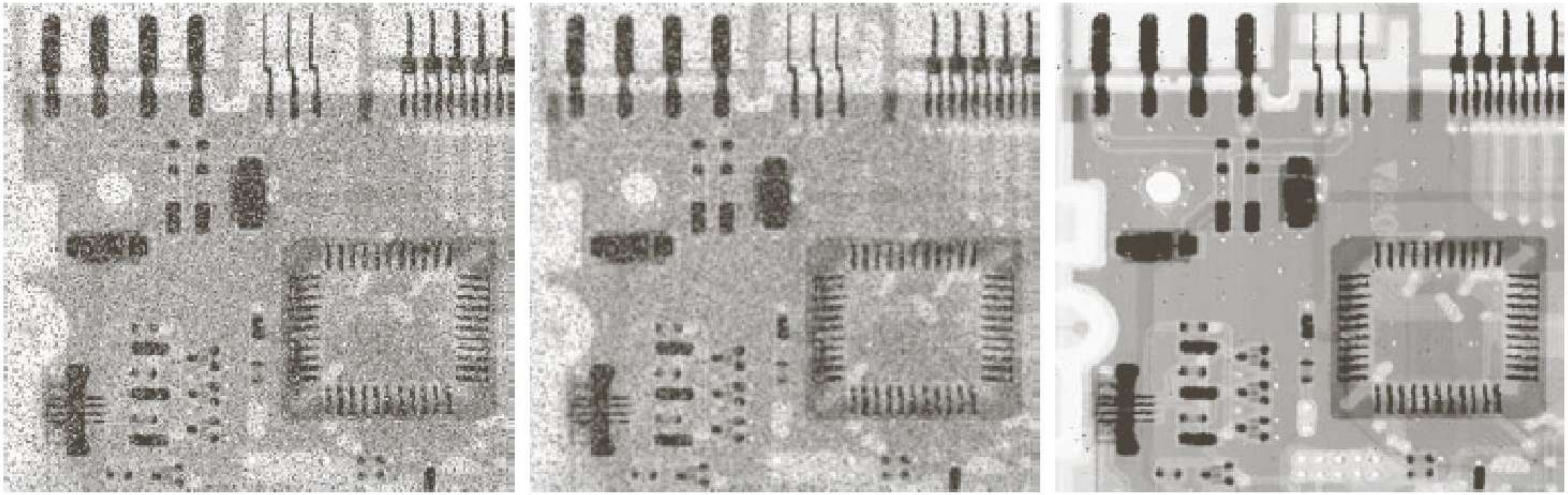
Comparing mean and median filter

- Median filter: MUCH better than mean filter for impulse noise.
- Preserves edges better than mean filter, but creates artifacts in smoother regions.

High σ, Gaussian

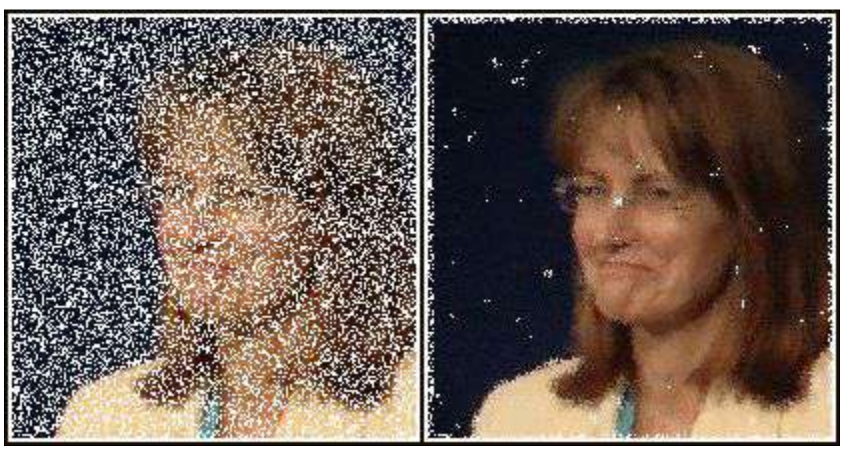
3 x 3 Median

3 x 3 Mean



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.)



Works quite well even with strong impulses, provided most neighborhood have no more than 50 percent of the pixels corrupted.

(a)

(b)

Original and noisy (impulse corrupted) images

(c) PSNR = 9.2898 dB

(d) PSNR = 12.4450 dB

Median filter outputs – given filters of size 3 x 3 and 5 x 5

http://what-when-how.com/embedded-image-processing-on-the-tms320c6000-dsp/non-linear-filtering-of-images-image-processing-part-1/

Linear and non-linear filters

- The mean filter is a linear filter:

$$\text{mean}(af + g) = a\,\text{mean}(f) + \text{mean}(g)$$

- It is also space invariant.
- So it can be implemented as a convolution.
- The median filter cannot be implemented using a convolution as it is nonlinear.

$$\text{median}(f + g) \neq \text{median}(f) + \text{median}(g)$$