Image Processsing and Analysis

Lecture 3、Image Enhancement in the Spatial Domain(I)

Weiqiang Wang
School of Computer Science and Technology, UCAS

September 19, 2023

Preview

- The principal objective of enhancement is to process an image so that the result is more suitable than the original image for a specific application
  - "Specific" means the techniques are very much problem oriented.
- Image enhancement approaches fall into two broad categories, spatial domain methods and frequency domain method.
  - spatial domain methods are based on direct manipulation of pixel in an image.
  - frequency domain techniques are based on modifying the Fourier transform of an image.
- There is no general theory of image enhancement.
  - When an image is processed for visual interpretation, the viewer is the ultimate judge of how well a particular method works, which makes it difficult to compare the performance of different methods.
  - For machine perception, the evaluation task is somewhat easier, e.g., character recognition task.

Outline

Background

- A mathematical representation of spatial domain processing:

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

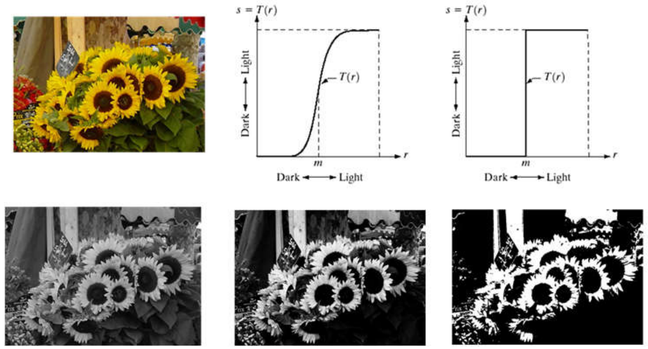
where $f(x, y)$: the input image
$g(x, y)$: the processed image
T: an operator on f, defined over some neighborhood of (x, y)

  - T can operate on a set of images, e.g., noise reduction
  - Square and rectangular neighborhood are by far the most popular due to their ease of implementation, although circle is also used.
- The simplest form of T is when the neighborhood is of size 1×1. In this case, g depends only on the value of f at (x, y), i.e.,

$$s = T(r),$$

that is called Intensity Transformation

Gray-level Transformation



Power-Law Transformation

- $s = cr^{\gamma}$ where c, γ: positive constants

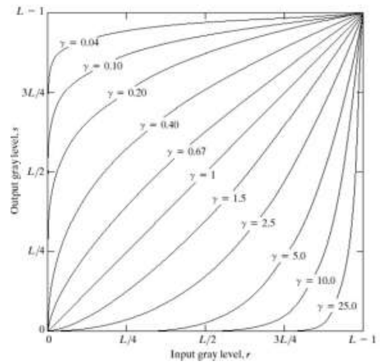
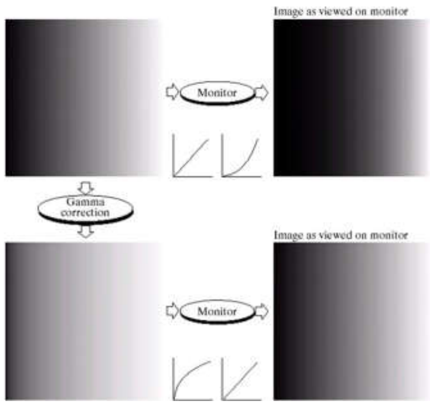


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

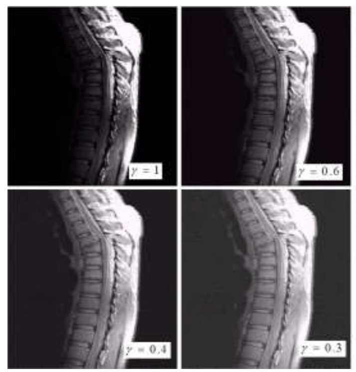
Power-Law Transformation Example 1: Gamma Correction

a b
c d

FIGURE 3.7
(a) Linear-wedge gray-scale image.
(b) Response of monitor to linear wedge.
(c) Gamma-corrected wedge.
(d) Output of monitor.



Power-Law Transform Example 2: Contrast manipulation



a b
c d

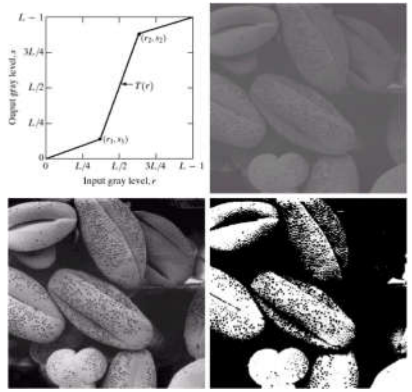
FIGURE 3.8
(a) Magnetic resonance (MR) image of a fractured human spine.
(b)–(d) Results of applying the transformation in Eq. (3.2-3) with $c = 1$ and $\gamma = 0.6, 0.4$, and 0.3, respectively. (Original image for this example courtesy of Dr. David R. Pickens, Department of Radiology and Radiological Sciences, Vanderbilt University Medical Center.)

Power-Law Transform Example 3: Contrast manipulation

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.)

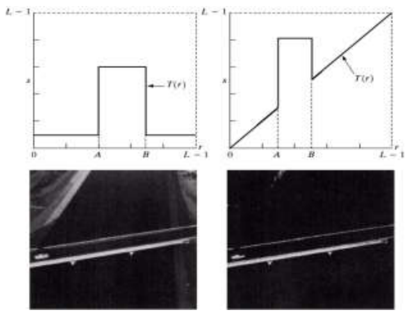
Piecewise-Linear Transformation Functions
Case 1: Contrast Stretching



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.)

Piecewise-Linear Transformation Functions
Case 2: Gray-level Slicing



a b
c d

FIGURE 3.11 (a) This transformation highlights range $[A, B]$ of gray levels and reduces all others to a constant level. (b) This transformation highlights range $[A, B]$ but preserves all other levels. (c) An image. (d) Result of using the transformation in (a).

Piecewise-Linear Transformation Functions
Case 3: Gray-level Slicing

- Bit-plane slicing:
  - It can highlight the contribution made to total image appearance by specific bits.
  - Each pixel in an image represented by 8 bits.
  - Image is composed of eight 1-bit planes, ranging from bit-plane 0 for the least significant bit to bit plane 7 for the most significant bit.

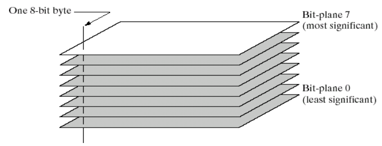


FIGURE 3.12 Bit-plane representation of an 8-bit image.

Piecewise-Linear Transformation Functions
Bit-plane Slicing: A Fractal Image

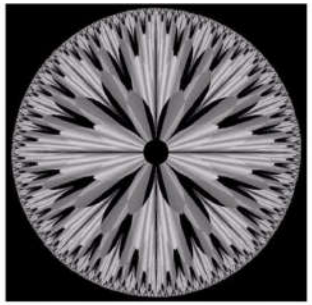


FIGURE 3.13 An 8-bit fractal image. (A fractal is an image generated from mathematical expressions). (Courtesy of Ms. Melissa D. Binde, Swarthmore College, Swarthmore, PA.)

Piecewise-Linear Transformation Functions
Bit-plane Slicing: A Fractal Image

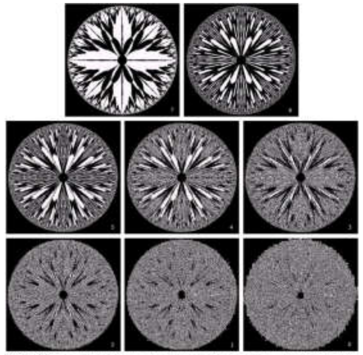
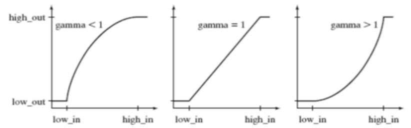


FIGURE 3.14 The eight bit planes of the image in Fig. 3.13. The number at the bottom, right of each image identifies the bit plane.

Function imadjust in IPT

- $g = imadjust(f, [low\_in, high\_in], [low\_out, high\_out], gamma)$
- This function maps the intensity values in image f to new values in g.
- Such that values between low_in and $high\_in$ map to values between low_out and $high\_out$.
- Values below low_in and above $high\_in$ are clipped; that is, values below low_in map to low_out, and those above $high\_in$ map to $high\_out$.

Function imadjust in IPT



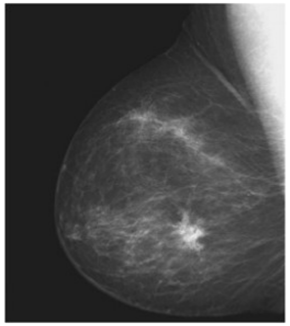
a b c

FIGURE 3.2 The various mappings available in function imadjust.

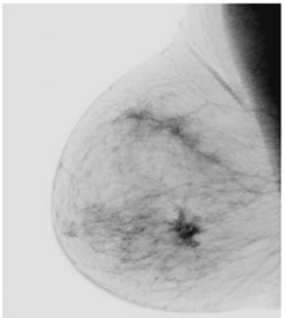
- Parameter $gamma$ specifies the shape of the curve that maps the intensity values in f to create g.
- If $gamma$ is less than 1, the mapping is weighted toward higher $brighter$ output values, as Fig. 3.2(a) shows.
- If $gamma$ is greater than 1,the mapping is weighted toward lower $darker$ output values.
- If it is omitted from the function argument, $gamma$ defaults to 1 $linearmapping$.

An Example

- I=imread('Fig0303(a)(breast).tif');
- imshow(I)



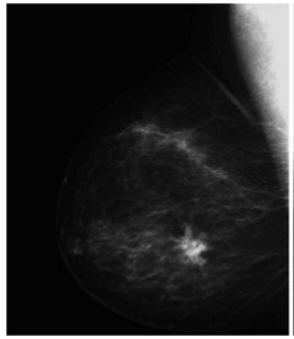
- G=imadjust(I,[0 1],[1 0]);
- imshow(G)



An Example

- G=imadjust(I,[0.5 0.75],[0 1]);
- imshow(G)

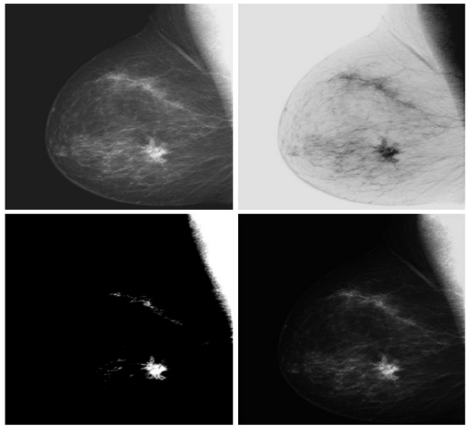
- G=imadjust(I,[],[],2);
- imshow(G)



Function imadjust

a b
c d

FIGURE 3.3 (a) Original digital mammogram. (b) Negative image. (c) Result of expanding the intensity range [0.5, 0.75]. (d) Result of enhancing the image with gamma = 2. (Original image courtesy of G. E. Medical Systems.)



Logarithmic Transformation

- Logarithmic and contrast-stretching transformations are basic tools for dynamic range manipulation.
- Logarithm transformations are implemented using the expression

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

  - where c is a constant.
  - The shape of this transformation is similar to the $gamma$ curve with the low values set at 0 and the high values set to 1 on both scales.
  - Note, however, that the shape of the $gamma$ curve is variable,whereas the shape of the log function is fixed.

Logarithmic Transformation

- One of the principal uses of the log transformation is to compress dynamic range.
  - For example, it is not unusual to have a Fourier spectrum (Chapter 4) with values in the range $[0, 10^6]$ or higher. When displayed on a monitor that is scaled linearly to 8 bits, the high values dominate the display, resulting in lost visual detail for the lower intensity values in the spectrum. By computing the log, a dynamic range on the order of, for example, 10^6 is reduced to approximately 14, which is much more manageable.
- When performing a logarithmic transformation, it is often desirable to bring the resulting compressed values back to the full range of the display. For 8 bits, the easiest way to do this in MATLAB is with the statement

$$gs = im2uint8(mat2gray(g));$$

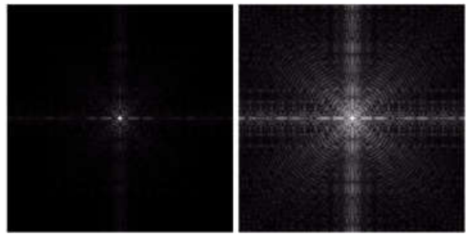
  - Use of $mat2gray$ brings the values to the range [0, 1] and $im2uint8$ brings them to the range [0, 255].

Logarithmic Transformation

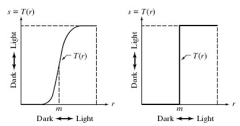
- Figure 3.5(a) is a Fourier spectrum with values in the range 0 to displayed on a linearly scaled, 8-bit system. Figure 3.5(b) shows the result obtained using the commands

```
>> g = im2uint8(mat2gray(log(1 + double(f))));
>> imshow(g)
```

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



Contrast-Stretching Transformation



a b
FIGURE 3.4 (a) Contrast-stretching transformation. (b) Thresholding transformation.

- A contrast-stretching transformation function can be defined as

$$s = T(r) = \frac{1}{1 + (m/r)^E}$$

  - it compresses the input levels lower than m into a narrow range of dark levels in the output image;
  - similarly, it compresses the values above m into a narrow band of light levels in the output;
  - where r represents the intensities of the input image, s the corresponding intensity values in the output image, and E controls the slope of the function.
- This equation is implemented in MATLAB for an entire image as

```
g = 1./(1 + (m./(double(f) + eps)).^E)
```

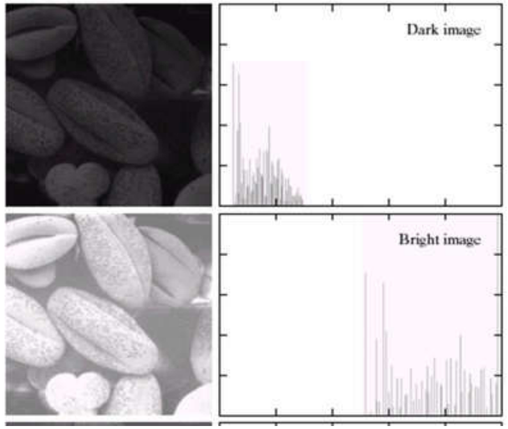

Histogram Processing

- Intensity transformation are based on information extracted from image intensity histograms, and histogram plays a basic role in image processing, in areas such as enhancement, compression, segmentation, and description.
- The histogram of a digital image with L total possible intensity levels in the range $[0,G]$ is defined as the discrete function

$$h(r_k) = n_k$$

  - where r_k is the kth intensity level in the interval $[0, G]$ and n_k is the number of pixels in the image whose intensity level is r_k.The value of G is 255 for images of class uint8, 65535 for images of class uint16, and 1.0 for images of class double.

Generating and Plotting Image Histograms



Normalized Histogram

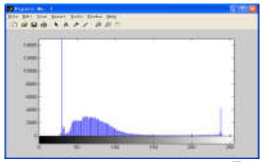
- Through dividing all elements of $h(r_k)$ by the total number of pixels in the image, a normalized histogram can be obtained, i.e.,

$$p(r_k) = \frac{h(r_k)}{N} = \frac{n_k}{N}, N = \sum_{k=1}^{L} n_k, k = 1,2,\dots,L$$

- From basic probability, we know $p(r_k)$ can be considered as an estimate of the probability of occurrence of intensity level r_k.

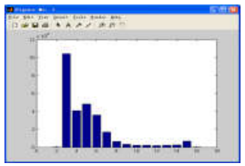
Generating Image Histograms

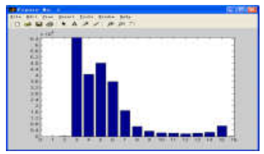
- h=imhist(f,b)
  - Where f is the input image, h is its histogram, and b is the number of bins used in forming the histogram.
  - If b is not included in the argument, b = 256 is used by default.
  - A bin is simply a subdivision of the intensity scale.
- We obtain the normalized histogram simply by using the expression:
  p=imhist(f,b)/numel(f)
- An example
  - f=imread('Fig3_8_a.tif');
  - imhist(f)



Plotting Image Histograms

- Histograms often are plotted using bar graphs. For this purpose we can use the function bar(horz, v, width)
  - Where v is a row vector containing the points to be plotted.
  - horz is a vector of the same dimension as v that contains the increments of the horizontal scale. If horz is omitted, the horizontal axis is divided in units from 0 to length(v).
  - width is a number between 0 and 1.
- s=imread('Fig0303(a)(breast).tif');
- h1=imhist(s,16);
- horz=1:16;
- bar(horz,h1,0.8)
- axis([0 16 0 65000]);
- set(gca,'xtick',0:1:16);
- set(gca,'ytick',0:4000:65000);





Plotting Image Histograms using stem

- A stem graph is similar to a bar graph. The syntax is

  stem(horz, v, 'color_linestyle_marker', 'fill')

  - where v is row vector containing the points to be plotted, and horz is as described for bar.
  - *color_linestyle_marker* is a triplet of values from Table below.

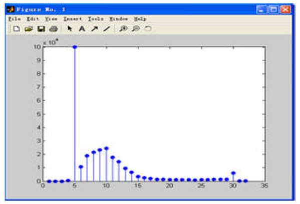
Symbol	Color	Symbol	Line Style	Symbol	Marker
k	Black	–	Solid	+	Plus sign
w	White	--	Dashed	o	Circle
r	Red	:	Dotted	*	Asterisk
g	Green	-.	Dash-dot	.	Point
b	Blue	none	No line	x	Cross
c	Cyan			s	Square
y	Yellow			d	Diamond
m	Magenta			none	No marker

TABLE 3.1 Attributes for functions `stem` and `plot`. The none attribute is applicable only to function `plot`, and must be specified individually. See the syntax for function `plot` below.

Some examples using stem

- stem(v, 'rs')
  produces a stem plot where the lines and markers are red, the lines are dashed, and the markers are squares.
- If *fill* is used, and the *marker* is a circle, square, or diamond, the *marker* is filled with the color specified in color.
- The default *color* is black, the *linestyle* default is solid, and the default *marker* is a circle.

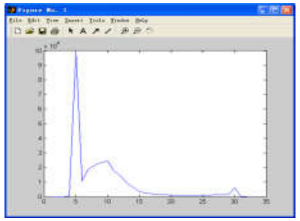
- s=imread('Fig0303(a)(breast).tif');
- hi=imhist(s,32);
- stem(hi,'b-o','fill');



Plotting Image Histograms using plot

- Function *plot* plots a set of points by linking them with straight lines. The syntax is plot(horz, v, 'color_linestyle_marker')
- where the arguments are as defined previously for stem plots.
- Function *plot* is used frequently to display transformation functions.

- s=imread('Fig0303(a)(breast).tif');
- hi=imhist(s,32);
- plot(hi);



Histogram Equalization

- Histogram equalization:
  - To improve the contrast of an image.
  - To transform an image in such a way that the transformed image has a nearly uniform distribution of pixel values.
- Transformation:
  - Assume r has been normalized to the interval [0,1], with $r = 0$ representing black and $r = 1$ representing white

$$s = T(r), 0 \le r \le 1$$

  - The transformation function satisfies the following conditions:
    - $T(r)$ is single-valued and monotonically increasing in the interval $0 \le r \le 1$
    - $0 \le T(r) \le 1$ for $0 \le r \le 1$

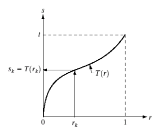


FIGURE 3.16 A gray-level transformation function that is both single valued and monotonically increasing.

Histogram Equalization

- Histogram equalization is based on a transformation of the probability density function of a random variable.
- Let $p_r(r)$ and $p_s(s)$ denote the probability density function of random variable r and s, respectively.
- If $p_r(r)$ and $T(r)$ are known, then the probability density function $p_s(s)$ of the transformed variable s can be obtained

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right|$$

- Define a transformation function

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

  - where w is a dummy variable of integration and the right side of this equation is the cumulative distribution function of random variable r.

Histogram Equalization

- Given transformation function $T(r)$

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

$$\frac{ds}{dr} = \frac{dT(r)}{dr} = \frac{d\left[\int_0^r p_r(w)dw\right]}{dr} = p_r(r)$$

$$p_s(s) = p_r(r)\left|\frac{dr}{ds}\right| = p_r(r)\frac{1}{p_r(r)} = 1, 0 \le s \le 1$$

- $p_s(s)$ now is a uniform probability density function.
- $T(r)$ depends on $p_r(r)$, but the resulting $p_s(s)$ always is uniform.

Histogram Equalization

- In discrete version:
  - The probability of occurrence of gray level r_k in an image is

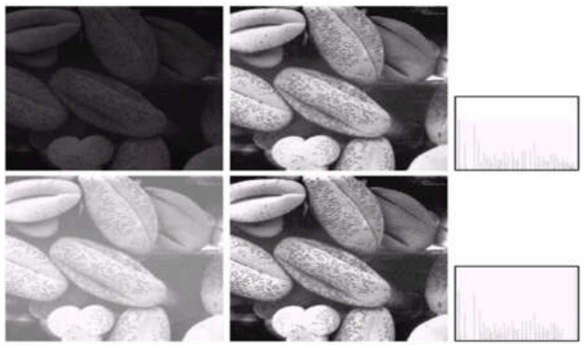
$$p_r(r_k) = \frac{n_k}{N}, k = 1,2,\ldots,L$$

  - The transformation function is

$$s = T(r_k) = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{N}$$

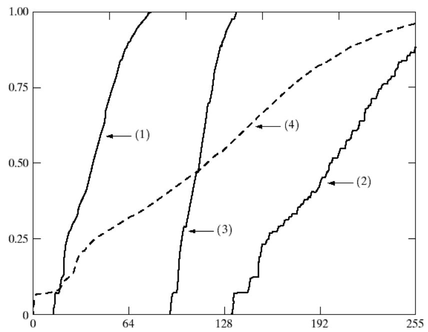
  - Thus, an output image is obtained by mapping each pixel with level r_k in the input image into a corresponding pixel with level s_k.

Histogram Equalization



Histogram Equalization

FIGURE 3.18 Transformation functions (1) through (4) were obtained from the histograms of the images in Fig.3.17(a), using Eq. (3.3-8).



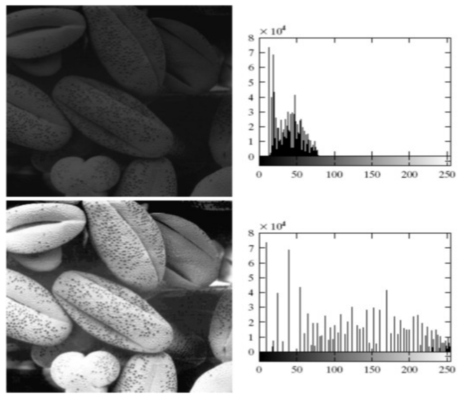
Histogram Equalization

- Histogram equalization is implemented in the toolbox by function histeq, which has the syntax
  g = histeq(f, nlev)
  - where f is the input image and nlev is the number of intensity levels specified for the output image.
  - If nlev is equal to L (the total number of possible levels in the input image), then histeq implements the transformation function directly. If nlev is less than L, then histeq attempts to distribute the levels so that they will approximate a flat histogram.

Histogram Equalization in Matlab

- >> f = imread('Fig0308(a)(pollen).tif');
- >> imshow(f);
- >> figure,imhist(f);
- >> ylim('auto');
- >> g = histeq(f,256);
- >> figure, imshow(g);
- >> figure, imhist(g);
- >> ylim('auto');

Histogram Equalization in Matlab

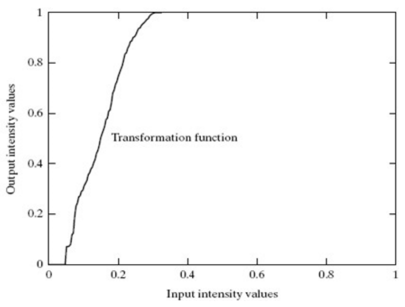


a b
c d

FIGURE 3.8 Illustration of histogram equalization. (a) Input image, and (b) its histogram. (c) Histogram-equalized image, and (d) its histogram. The improvement between (a) and (c) is quite visible. (Original image courtesy of Dr. Roger Heady, Research School of Biological Sciences, Australian National University, Canberra.)

Histogram Equalization

FIGURE 3.9 Transformation function used to map the intensity values from the input image in Fig. 3.8(a) to the values of the output image in Fig. 3.8(c).



Histogram Matching

- Histogram matching is similar to histogram equalization, except that instead of trying to make the output image have a flat histogram, we would like it to have a histogram of a specified shape.
- Consider for a moment continuous levels that are normalized to the interval [0, 1], and let r and z denote the intensity levels of the input and output images. The input levels have probability density function $p_r(r)$ and the output levels have the specified probability density function $p_z(z)$.

Histogram Matching

- We know from the discussion in the previous section that the transformation:

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

result in a ideal equalized histogram $p_s(s)$.
- Suppose now we define a variable z with the property

$$H(z) = \int_0^z p_z(w)dw = s$$

- From the preceding two equations, it follows that

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

- We can find $T(r)$ from the input image (this is the histogram equalization transformation discussed in the previous section), so it follows that we can use the preceding equation to find the transformed levels z whose PDF is the specified $p_z(z)$ as long as we can find H^{-1}.

Histogram Matching

- The toolbox implements histogram matching using the following syntax in histeq:

  g = histeq(f, hspec)

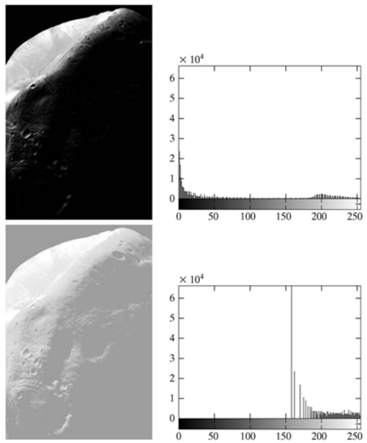
  - where f is the input image, hspec is the specified histogram (a row vector of specified values), and g is the output image, whose histogram approximates the specified histogram, hspec.
- This vector should contain integer counts corresponding to equally spaced bins. A property of histeq is that the histogram of g generally better matches hspec when length (hspec) is much smaller than the number of intensity levels in f.

Histogram Matching

- f=imread('Fig0310(a)(Moon Phobos).tif');
- f1=histeq(f,256);
- imshow(f1);

- It shows that histogram equalization in fact did not produce a particularly good result in this case.
- The reason for this can be seen by studying the histogram of the equalized image.

Histogram Matching



a b
c d

FIGURE 3.10 (a) Image of the Mars moon Phobos. (b) Histogram. (c) Histogram-equalized image. (d) Histogram of (c). (Original image courtesy of NASA).

Histogram Matching

- One possibility for remedying this situation is to use histogram matching, with the desired histogram having a lesser concentration of components in the low end of the gray scale, and maintaining the general shape of the histogram of the original image.
- We note that the histogram of original image is basically bimodal, with one large mode at the origin, and another, smaller, mode at the high end of the gray scale. These types of histograms can be modeled, for example, by using multimodal Gaussian functions.

Histogram Matching

- The following M-function computes a bimodal Gaussian function normalized to unit area, so it can be used as a specified histogram.
  - Function twomodegauss:

$$p(x) = k + \frac{A_1}{\sqrt{2\pi}\sigma_1}\exp\left(-\frac{(x-m_1)^2}{2\sigma_1^2}\right) + \frac{A_2}{\sqrt{2\pi}\sigma_2}\exp\left(-\frac{(x-m_2)^2}{2\sigma_2^2}\right)$$

- The following interactive function accepts inputs from a keyboard and plots the resulting Gaussian function. Refer to Section 2.10.5 for an explanation of the functions input and str2num. Note how the limits of the plots are set.
  - Function manualhist

Histogram Matching

- Since the problem with histogram equalization in this example is due primarily to a large concentration of pixels in the original image with levels near 0,a reasonable approach is to modify the histogram of that image so that it does not have this property.
- Figure 3.11(a) shows a plot of a function that preserves the general shape of the original histogram, but has a smoother transition of levels in the dark region of the intensity scale. The output of the program, p, consists of 256 equally spaced points from this function and is the desired specified histogram.

Histogram Matching

- An image with the specified histogram was generated using the command

  >> g = histeq(f, p);

  all commands are:

  >> f=imread('Fig0310(a)(Moon Phobos).tif');

  >> g=histeq(f,manualhist);

  Enter m1, sig1, m2, sig2, A1, A2, k OR x to quit:x;

  >> imshow(g);

Histogram Matching

a b
c

FIGURE 3.11 (a) Specified histogram. (b) Result of enhancement by histogram matching. (c) Histogram of (b).