DIGITAL IMAGE PROCESSING

Unit 5: Image Enhancement in the Spatial Domain

IMAGE ENHANCEMENT DEFINITION

- **Image Enhancement:** is the process that improves the quality of the image for a specific application

The reasons for doing this include:

- Highlighting interesting detail in images
- Removing noise from images
- Making images more visually appealing

IMAGE ENHANCEMENT METHODS

- **Spatial Domain Methods (Image Plane)**

  Techniques are based on direct manipulation of pixels in an image
- **Frequency Domain Methods**

  Techniques are based on modifying the Fourier transform of the image.
- **Combination Methods**

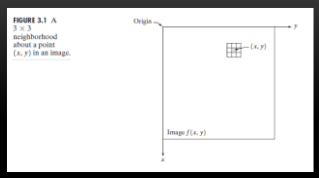
  There are some enhancement techniques based on various combinations of methods from the first two categories

SPATIAL DOMAIN METHODS

- As indicated previously, the term *spatial domain* refers to the aggregate of pixels composing an image.
- Spatial domain methods are procedures that operate directly on these pixels.
- Spatial domain processes will be denoted by the expression:

  $g(x,y) = \boldsymbol{T}[f(x,y)]$

  Where f(x,y) in the input image, g(x,y) is the processed image and ***T*** is an operator on f, defined over some neighborhood of (x,y)
- In addition, ***T*** can operate on a set of input images.

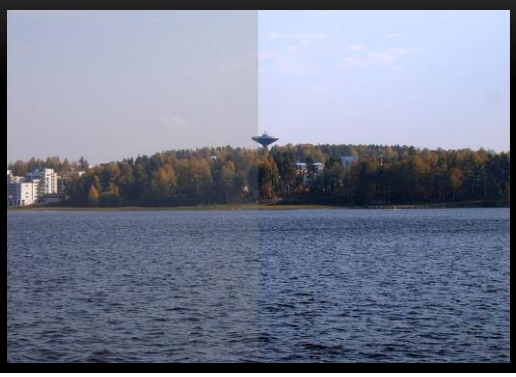


- The simplest form of ***T***, is when the neighborhood of size 1X1 (that is a single pixel).
- In this case, g depends only on the value of f at (x,y), and ***T*** becomes a *grey-level* (also called *intensity* or *mapping*) *transformation function* of the form:

  $s = T(r)$

Where, for simplicity in notation, r and s are variables denoting, respectively, the grey level of f(x,y) and g(x,y) at any point (x,y)

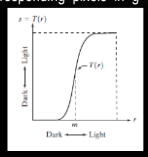
CONTRAST



EXAMPLES OF ENHANCEMENT TECHNIQUES

- **Contrast Stretching:**

If T(r) has the form as shown in the figure below, the effect of applying the transformation to every pixel of f to generate the corresponding pixels in g would:



Produce higher contrast than the original image, by:

- Darkening the levels below m in the original image
- Brightening the levels above m in the original image

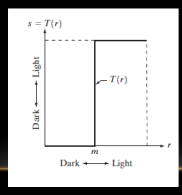
Contrast Stretching…:

- So, Contrast Stretching: is a simple image enhancement technique that improves the contrast in an image by 'stretching' the range of intensity values it contains to span a desired range of values.
- Typically, it uses a linear function

EXAMPLES OF ENHANCEMENT TECHNIQUES

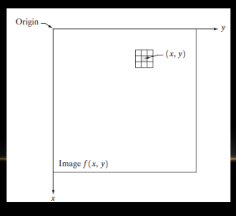
Thresholding

- Is a limited case of contrast stretching, it produces a two-level (binary) image.
- Some fairly simple, yet powerful, processing approaches can be formulated with grey-level transformations.
- Because enhancement at any point in an image depends only on the gray level at that point, techniques in this category often are referred to as *point processing*.



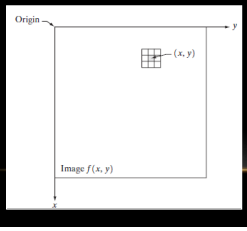
EXAMPLES OF ENHANCEMENT TECHNIQUES

- Larger neighborhoods allow considerably more flexibility.
- The general approach is to use a function of the values of *f* in a predefined neighborhood of (x,y) to determine the value of *g* at (x,y).
- One of the principal approaches in this formulation is based on the use of so-called *masks* (also referred to as *filters)*



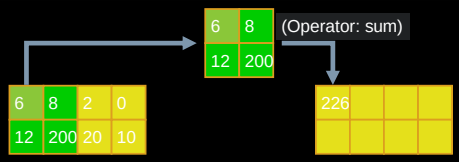
EXAMPLES OF ENHANCEMENT TECHNIQUES

- So, a mask/filter: is a small (say 3X3) 2-D array, such as the one shown in the figure, in which the values of the mask coefficients determine the nature of the process, such as image sharpening.
- Enhancement techniques based on this type of approach often are referred to as mask processing or filtering.



Examples of Enhancement Techniques

Operation on the set of 'neighbourhoods' N(x,y) of each pixel



SOME BASIC INTENSITY (GRAY-LEVEL) TRANSFORMATION FUNCTIONS

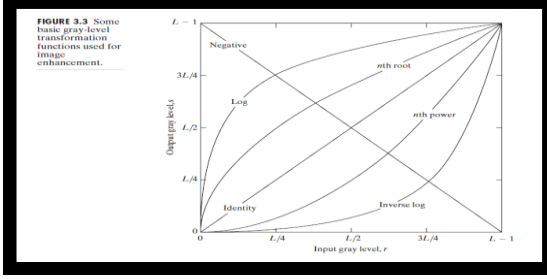
- Grey-level transformation functions (also called, intensity functions), are considered the simplest of all image enhancement techniques.
- The value of pixels, before an after processing, will be denoted by *r* and *s*, respectively. These values are related by the expression of the form:

  $s = T(r)$

  where T is a transformation that maps a pixel value r into a pixel value s.

SOME BASIC INTENSITY (GRAY-LEVEL) TRANSFORMATION FUNCTIONS

Consider the following figure, which shows three basic types of functions used frequently for image enhancement:



SOME BASIC INTENSITY (GRAY-LEVEL) TRANSFORMATION FUNCTIONS

- The three basic types of functions used frequently for image enhancement:
  - Linear Functions:
    - Negative Transformation
    - Identity Transformation
  - Logarithmic Functions:
    - Log Transformation
    - Inverse-log Transformation
  - Power-Law Functions:
    - n^{th} power transformation
    - n^{th} root transformation

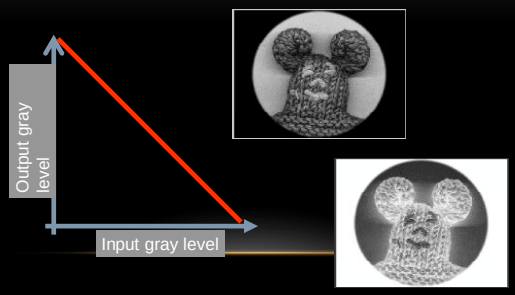
LINEAR FUNCTIONS

- **Identity Function**
  - Output intensities are identical to input intensities
  - This function doesn't have an effect on an image, it was included in the graph only for completeness
  - Its expression:

    $s = r$

LINEAR FUNCTIONS

Image Negatives: $T(f) = L-f$



LINEAR FUNCTIONS

- **Image Negatives (Negative Transformation)**
  - The negative of an image with gray level in the range [0, L-1], where L = Largest value in an image, is obtained by using the negative transformation's expression:

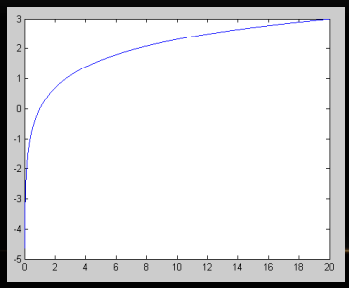
$$s = L - 1 - r$$

Which reverses the intensity levels of an input image, in this manner produces the equivalent of a photographic negative.

  - The negative transformation is suitable for enhancing white or gray detail embedded in dark regions of an image, especially when the black area are dominant in size

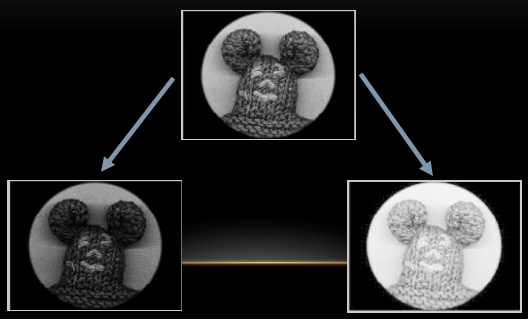
LINEAR FUNCTIONS

Log Transformations:
T(f) = c * log (1+ f)



LINEAR FUNCTIONS

Log Transformations



LOGARITHMIC TRANSFORMATIONS

- **Log Transformation**

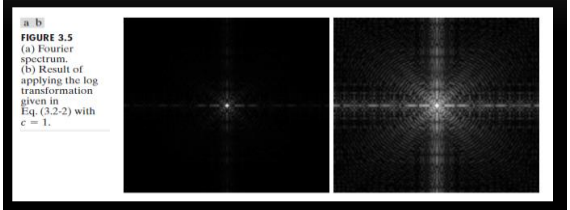
The general form of the log transformation:

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

Where c is a constant, and $r \geq 0$

  - Log curve maps a narrow range of low gray-level values in the input image into a wider range of the output levels.
  - Used to expand the values of dark pixels in an image while compressing the higher-level values.
  - It compresses the dynamic range of images with large variations in pixel values.

LOGARITHMIC TRANSFORMATIONS



a b

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

LOGARITHMIC TRANSFORMATIONS

- **Inverse Logarithm Transformation**
  - Do opposite to the log transformations
  - Used to expand the values of high pixels in an image while compressing the darker-level values.

POWER-LAW TRANSFORMATIONS

- Power-law transformations have the basic form of:

  $s = c.r^{\gamma}$

  Where c and γ are positive constants

POWER-LAW TRANSFORMATIONS

- Different transformation curves are obtained by varying γ (gamma)

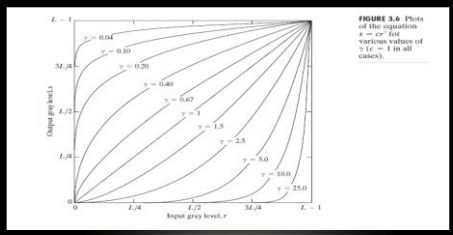


FIGURE 3.6 Plots of the equation $s = cr^{\gamma}$ for various values of γ ($c = 1$ in all cases).

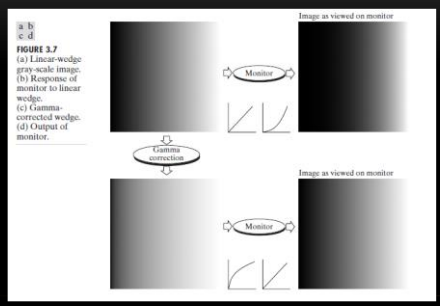
POWER-LAW TRANSFORMATIONS

- Variety of devices used for image capture, printing and display respond according to a power law.
- The process used to correct this power-law response phenomena is called ***gamma correction***.

  For example, cathode ray tube (CRT) devices have an intensity-to-voltage response that is a power function, with exponents varying from approximately 1.8 to 2.5.

  With reference to the curve for g=2.5 in Fig. 3.6, we see that such display systems would tend to produce images that are darker than intended. This effect is illustrated in Fig. 3.7.

POWER-LAW TRANSFORMATION



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 TRANSFORMATIONS

Figure 3.7(a) shows a simple gray-scale linear wedge input into a CRT monitor.

As expected, the output of the monitor appears darker than the input, as shown in Fig. 3.7(b).

Gamma correction in this case is straightforward.

All we need to do is preprocess the input image before inputting it into the monitor by performing the transformation.

The result is shown in Fig. 3.7(c).

When input into the same monitor, this gamma-corrected input produces an output that is close in appearance to the original image, as shown in Fig. 3.7(d).

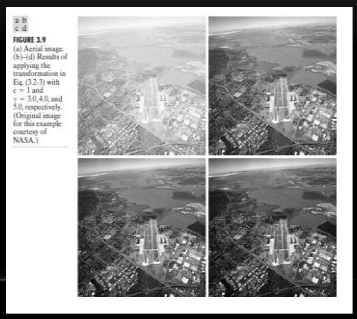
POWER-LAW TRANSFORMATION

- In addition to gamma correction, power-law transformations are useful for general-purpose 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 TRANSFORMATION

- Another illustration of Power-law transformation



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.)

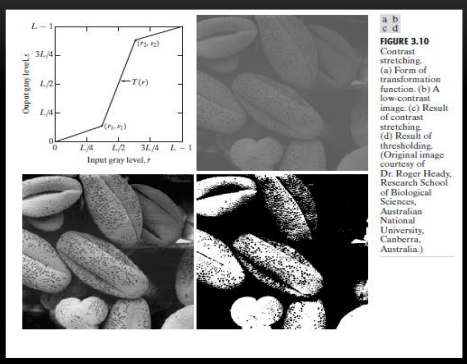
PIECEWISE-LINEAR TRANSFORMATION FUNCTIONS

- **Principle Advantage:** Some important transformations can be formulated only as a piecewise function.
- **Principle Disadvantage:** Their specification requires more user input than previous transformations
- **Types of Piecewise transformations are:**
  - Contrast Stretching
  - Gray-level Slicing
  - Bit-plane slicing

CONTRAST STRETCHING

- One of the simplest piecewise linear functions is a contrast-stretching transformation, which is used to enhance the low contrast images.
- Low contrast images may result from:
  - Poor illumination
  - Wrong setting of lens aperture during image acquisition.

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.)

CONTRAST STRETCHING

- Figure 3.10(a) shows a typical transformation used for contrast stretching. The locations of points (r1, s1) and (r2, s2) control the shape of the transformation function.
- If r1 = s1 and r2 = s2, the transformation is a linear function that produces no changes in gray levels.
- If r1 = r2, s1 = 0 and s2 = L-1, the transformation becomes a *thresholding function* that creates a binary image. As shown previously in slide 7.
- Intermediate values of (r1, s1) and (r2, s2) produce various degrees of spread in the gray levels of the output image, thus affecting its contrast.
- In general, $r1 \leq r2$ and $s1 \leq s2$ is assumed, so the function is always increasing.

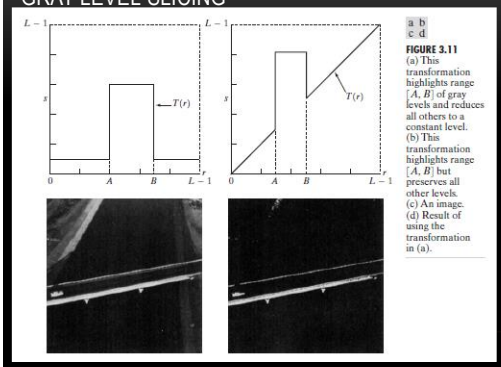
CONTRAST STRETCHING

- Figure 3.10(b) shows an 8-bit image with low contrast.
- Fig. 3.10(c) shows the result of contrast stretching, obtained by setting (r1, s1) = (r_{min}, 0) and (r2, s2) = (r_{max},L-1) where r_{min} and r_{max} denote the minimum and maximum gray levels in the image, respectively. Thus, the transformation function stretched the levels linearly from their original range to the full range [0, L-1].
- Finally, Fig. 3.10(d) shows the result of using the *thresholding function* defined previously, with r1=r2=m, the mean gray level in the image.

GRAY-LEVEL SLICING

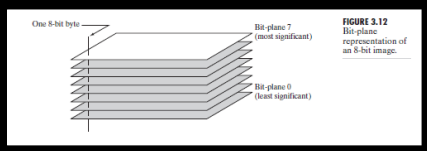
- This technique is used to highlight a specific range of gray levels in a given image.
- It can be implemented in several ways, but the two basic themes are:
  - One approach is to display a high value for all gray levels in the range of interest and a low value for all other gray levels.
    - This transformation, shown in Fig 3.11 (a), produces a binary image.
  - The second approach, based on the transformation shown in Fig 3.11 (b), brightens the desired range of gray levels but preserves gray levels unchanged.
  - Fig 3.11 (c) shows a gray scale image, and fig 3.11 (d) shows the result of using the transformation in Fig 3.11 (a).

GRAY-LEVEL SLICING



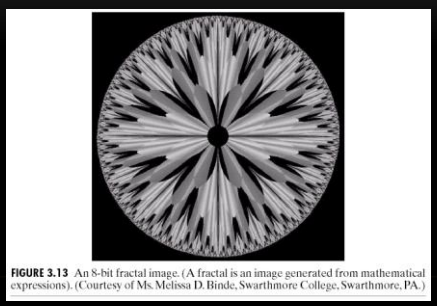
BIT-PLANE SLICING

- Pixels are digital numbers, each one composed of bits. Instead of highlighting gray-level range, we could highlight the contribution made by each bit.
- This method is useful and used in image compression.

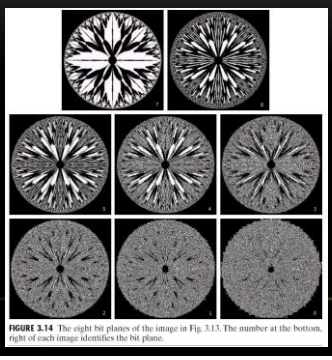


- Most significant bits contain the majority of visually significant data.

BIT-PLANE SLICING…



BIT-PLANE SLICING…



ASSIGNMENT

- Write an M-function, that
  - inputs an image, a low value, a high value and a range of gray levels of interest (low, high, from, to)
  - Outputs a two images (g, h)
  - Applies the two approaches of Gray-level slicing on image f, and produces images g and h respectively.
  - Deadline: Next Class…!!!

WHAT IS A HISTOGRAM?

- In Statistics, **Histogram** is a graphical representation showing a visual impression of the distribution of data.
- An **Image Histogram** is a type of histogram that acts as a graphical representation of the lightness/color distribution in a digital image.
  - It plots the number of pixels for each value.
- Histogram provides a global description of the appearance of the image.

WHY HISTOGRAM?

- Histograms are the basis for numerous spatial domain processing techniques
- Histogram manipulation can be used effectively for image enhancement
- Histograms can be used to provide useful image statistics
- Information derived from histograms are quite useful in other image processing applications, such as image compression and segmentation.

HISTOGRAM PROCESSING

- The histogram of a digital image with gray levels in the range [0, L-1] is a discrete function

$$h(r_k) = n_k$$

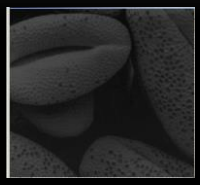
where r_k is the k^{th} gray level and n_k is the number of pixels in the image having gray level r_k.

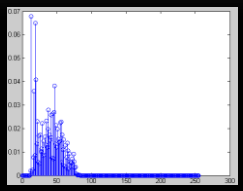
HISTOGRAM PROCESSING

- It is common practice to normalize a histogram by dividing each of its values by the total number of pixels in the image, denoted by n.
- Thus, a normalized histogram is given by $p(r_k) = n_k / n$, for k = 0, 1, …, L -1.
- Thus, $p(r_k)$ gives an estimate of the probability of occurrence of gray level r_k. Note that the sum of all components of a normalized histogram is equal to 1.

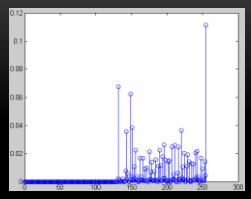
Some Typical Histograms

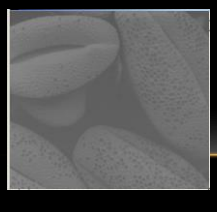
- The shape of a histogram provides useful information for contrast enhancement.

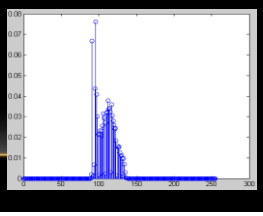


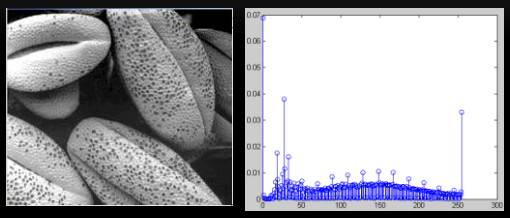


```
>> clear;
[ix,map]=imread('Fig3_15a.jpg');
imshow(ix)
figure;
ix=double(ix);
h=histogram(ix);
figure
stem(0:255,h);
```









INTRODUCTORY EXAMPLE OF HISTOGRAMS…

- We note in the dark Image that the components of the histogram are concentrated on the low (dark) side of the gray scale.
- Similarly, the components of the histogram of the bright image are biased toward the high side of the gray scale.
- An image with low contrast has a histogram that will be narrow and will be centered toward the middle of the gray scale.
- For a monochrome image this implies a dull, washed-out gray look.
- Finally, we see that the components of the histogram in the high-contrast image cover a broad range of the gray scale and, further, that the distribution of pixels is not too far from uniform, with very few vertical lines being much higher than the others.

INTRODUCTORY EXAMPLE OF HISTOGRAMS… CONT.

- Intuitively, it is reasonable to conclude that an image whose pixels tend to occupy the entire range of possible gray levels and, in addition, tend to be distributed uniformly, will have an appearance of high contrast and will exhibit a large variety of gray tones.

Histogram Equalization

- What is the histogram equalization?
- The histogram equalization is an approach to enhance a given image.
- The approach is to design a transformation $T(.)$ such that the gray values in the output is uniformly distributed in [0, 1].

• Let us assume for the moment that the input image to be enhanced has continuous gray values, with $r = 0$ representing black and $r = 1$ representing white.

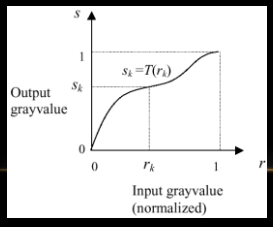
• We need to design a gray value transformation $s = T(r)$, based on the histogram of the input image, which will enhance the image.

Histogram Equalization

- As before, we assume that:

(1) $T(r)$ is a monotonically increasing function for $0 \le r \le 1$ (preserves order from black to white).

(2) $T(r)$ maps [0,1] into [0,1] (preserves the range of allowed Gray values).



Histogram Equalization

- Let us denote the inverse transformation by $r = T^{-1}(s)$.
- We assume that the inverse transformation also satisfies the above two conditions.

• We consider the gray values in the input image and output image as random variables in the interval [0, 1].

• Let $p_{in}(r)$ and $p_{out}(s)$ denote the probability density of the Gray values in the input and output images.

Histogram Equalization

- If $p_{in}(r)$ and T(r) are known, and $r = T^{-1}(s)$ satisfies condition 1, we can write (result from probability theory):

$$p_{out}(s) = \left[p_{in}(r) \frac{dr}{ds} \right]_{r=T^{-1}(s)}$$

- One way to enhance the image is to design a transformation $T(.)$ such that the gray values in the output is uniformly distributed in [0, 1], i.e. $p_{out}(s) = 1, \quad 0 \le s \le 1$

- In terms of histograms, the output image will have all gray values in "equal proportion" .

- This technique is called **histogram equalization**.

Next we derive the gray values in the output is uniformly distributed in [0, 1].

- Consider the transformation

$$s = T(r) = \int_0^r p_{in}(w)dw, \qquad 0 \le r \le 1$$

- Note that this is the cumulative distribution function (CDF) of $p_{in}(r)$ and satisfies the previous two conditions.

- From the previous equation and using the fundamental theorem of calculus,

$$\frac{ds}{dr} = p_{in}(r)$$

- Therefore, the output histogram is given by

$$p_{out}(s) = \left[p_{in}(r) \cdot \frac{1}{p_{in}(r)} \right]_{r=T^{-1}(s)} = [1]_{r=T^{-1}(s)} = 1, \qquad 0 \le s \le 1$$

- The output probability density function is uniform, regardless of the input.

- Thus, using a transformation function equal to the CDF of input gray values r, we can obtain an image with uniform gray values.

- This usually results in an enhanced image, with an increase in the dynamic range of pixel values.

How to implement histogram equalization?

Step 1:For images with discrete gray values, compute:

$$p_{in}(r_k) = \frac{n_k}{n} \qquad 0 \le r_k \le 1 \qquad 0 \le k \le L-1$$

L: Total number of gray levels

n_k: Number of pixels with gray value r_k

n: Total number of pixels in the image

Step 2: Based on CDF, compute the discrete version of the previous transformation :

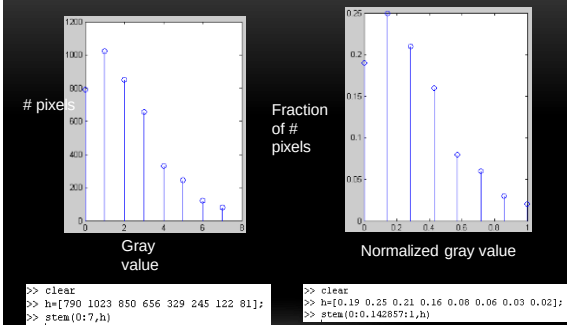
$$s_k = T(r_k) = \sum_{j=0}^{k} p_{in}(r_j) \qquad 0 \le k \le L-1$$

Example:

- Consider an 8-level 64 x 64 image with gray values (0, 1, …, 7). The normalized gray values are (0, 1/7, 2/7, …, 1). The normalized histogram is given below:

k	r_k	n_k	$p(r_k) = n_k/n$
0	0	790	0.19
1	1/7	1023	0.25
2	2/7	850	0.21
3	3/7	656	0.16
4	4/7	329	0.08
5	5/7	245	0.06
6	6/7	122	0.03
7	1	81	0.02

NB: The gray values in output are also (0, 1/7, 2/7, …, 1).



• Applying the transformation, $s_k = T(r_k) = \sum_{j=0}^{k} p_{in}(r_j)$ we have

$$s_0 = T(r_0) = \sum_{j=0}^{0} p_{in}(r_j) = p_{in}(r_0) = 0.19 \rightarrow 1/7$$

$$s_1 = T(r_1) = \sum_{j=0}^{1} p_{in}(r_j) = p_{in}(r_0) + p_{in}(r_1) = 0.44 \rightarrow 3/7$$

$$s_2 = T(r_2) = \sum_{j=0}^{2} p_{in}(r_j) = p_{in}(r_0) + p_{in}(r_1) + p_{in}(r_2) = 0.65 \rightarrow 5/7$$

$$s_3 = T(r_3) = \sum_{j=0}^{3} p_{in}(r_j) = p_{in}(r_0) + p_{in}(r_1) + \cdots + p_{in}(r_3) = 0.81 \rightarrow 6/7$$

$$s_4 = T(r_4) = \sum_{j=0}^{4} p_{in}(r_j) = p_{in}(r_0) + p_{in}(r_1) + \cdots + p_{in}(r_4) = 0.89 \rightarrow 6/7$$

$$s_5 = T(r_5) = \sum_{j=0}^{5} p_{in}(r_j) = p_{in}(r_0) + p_{in}(r_1) + \cdots + p_{in}(r_5) = 0.95 \rightarrow 1$$

$$s_6 = T(r_6) = \sum_{j=0}^{6} p_{in}(r_j) = p_{in}(r_0) + p_{in}(r_1) + \cdots + p_{in}(r_6) = 0.98 \rightarrow 1$$

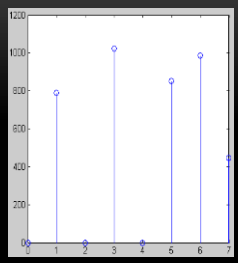
$$s_7 = T(r_7) = \sum_{j=0}^{7} p_{in}(r_j) = p_{in}(r_0) + p_{in}(r_1) + \cdots + p_{in}(r_7) = 1.00 \rightarrow 1$$

• Notice that there are only five distinct gray levels --- (1/7, 3/7, 5/7, 6/7, 1) in the output image. We will relabel them as (s_0, s_1, …, s_4).

• With this transformation, the output image will have histogram

k	s_k	n_k	$p(s_k) = n_k/n$
0	1/7	790	0.19
1	3/7	1023	0.25
2	5/7	850	0.21
3	6/7	985	0.24
4	1	448	0.11

pixels



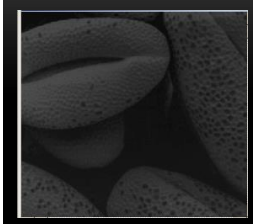
Gray values

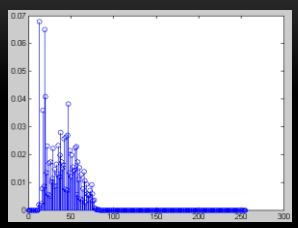
```
>> clear
>> h=[0 790 0 1023 0 850 985 448];
>> stem(0:7,h)
```

• Note that the histogram of output image is only approximately, and not exactly, uniform. This should not be surprising, since there is no result that claims uniformity in the **discrete** case.

Example

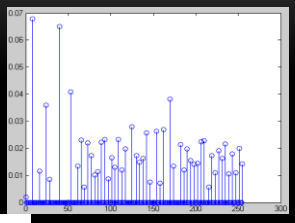
Original image and its histogram





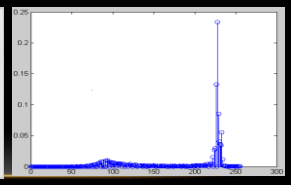

```
>> clear;
[ix,map]=imread('Fig3_15a.jpg');
imshow(ix)
figure;
ix=double(ix);
h=histogram(ix);
stem(0:255,h);
```

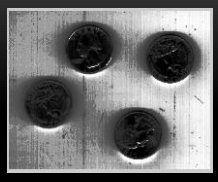
Histogram equalized image and its histogram

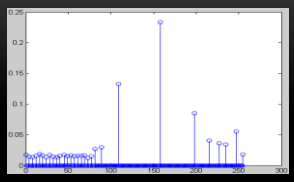



```
>> clear
>> [ix,map]=imread('Fig3_15a.jpg');
>> imshow(ix);
>> iy=histeq(ix);
>> figure
>> imshow(iy);
>> iy=double(iy);
>> hy=histogram(iy);
>> figure
>> stem(0:255,hy);
```

• Comments:
Histogram equalization may not always produce desirable results, particularly if the given histogram is very narrow. It can produce false edges and regions. It can also increase image “graininess” and “patchiness.”







```
>> clear
>> ix=imread('eight.tif');
>> imshow(ix);
>> iy=histeq(ix);
>> figure
>> imshow(iy);
>> ix=double(ix);
>> figure
>> hx=histogram(ix);
>> stem(0:255,hx);
>> iy=double(iy);
>> figure
>> hy=histogram(iy);
>> stem(0:255,hy);
```

Histogram Specification (Histogram Matching)

- Histogram equalization yields an image whose pixels are (in theory) uniformly distributed among all gray levels.
- Sometimes, this may not be desirable. Instead, we may want a transformation that yields an output image with a pre-specified histogram.
- This technique is called **histogram specificatio**n.

- Given Information

  (1) Input image from which we can compute its histogram .

  (2) Desired histogram.

- Goal

Derive a point operation, $H(r)$, that maps the input image into an output image that has the user-specified histogram.

- Again, we will assume, for the moment, continuous-gray values.

Approach of derivation

$z=H(r) \quad = G^{-1}(v=s=T(r))$

- Suppose, the input image has probability density in $p(r)$. We want to find a transformation $z = H(r)$, such that the probability density of the new image obtained by this transformation is $p_{out}(z)$, which is not necessarily uniform.

- First apply the transformation

$$s = T(r) = \int_0^r p_{in}(w)dw, \qquad 0 \le r \le 1 \qquad (*)$$

This gives an image with a uniform probability density.

- If the desired output image were available, then the following transformation would generate an image with uniform density:

$$V = G(z) = \int_0^z p_{out}(w)dw, \qquad 0 \le z \le 1 \qquad (**)$$

- From the gray values v we can obtain the gray values z by using the inverse transformation, $z = G^{-1}(v)$

- If instead of using the gray values v obtained from (**), we use the gray values s obtained from (*) above (both are uniformly distributed !), then the point transformation

$$Z=H(r)= G^{-1}[\ v=s =T(r)]$$

will generate an image with the specified density out $p(z)$, from an input image with density in $p(r)$!

- For discrete gray levels, we have

$$s_k = T(r_k) = \sum_{i=0}^{k} p_{in}(r_j) \qquad 0 \le k \le L-1$$

$$v_k = G(z_k) = \sum_{j=0}^{k} p_{out}(z_j) = s_k \qquad 0 \le k \le L-1$$

- If the transformation $z_k \rightarrow G(z_k)$ is one-to-one, the inverse transformation $s_k \rightarrow G^{-1}(s_k)$, can be easily determined, since we are dealing with a small set of discrete gray values.

- In practice, this is not usually the case (i.e.,) $z_k \rightarrow G(z_k)$ is not one-to-one) and we assign gray values to match the given histogram, as closely as possible.

Algorithm for histogram specification:

(1) Equalize input image to get an image with uniform gray values, using the discrete equation:

$$s_k = T(r_k) = \sum_{j=0}^{k} p_{in}(r_j) \qquad 0 \le k \le L-1$$

(2) Based on desired histogram to get an image with uniform gray values, using the discrete equation:

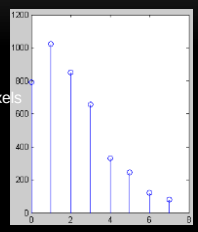
$$v_k = G(z_k) = \sum_{j=0}^{k} p_{out}(z_j) = s_k \qquad 0 \le k \le L-1$$

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

Example:

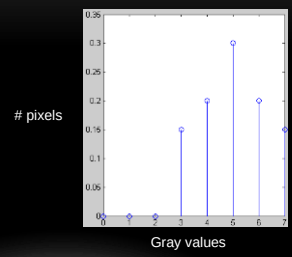
- Consider an 8-level 64 x 64 previous image.

k	r_k	n_k	$p(r_k) = n_k/n$
0	0	790	0.19
1	1/7	1023	0.25
2	2/7	850	0.21
3	3/7	656	0.16
4	4/7	329	0.08
5	5/7	245	0.06
6	6/7	122	0.03
7	1	81	0.02



- It is desired to transform this image into a new image, using a transformation $Z=H(r)= G^{-1}[T(r)]$, with histogram as specified below:

k	z_k	$p_{out}(z_k)$
0	0	0.00
1	1/7	0.00
2	2/7	0.00
3	3/7	0.15
4	4/7	0.20
5	5/7	0.30
6	6/7	0.20
7	1	0.15



- The transformation $T(r)$ was obtained earlier (reproduced below):

$r_j \rightarrow s_k$	n_k	$p(s_k)$
$r_0 \rightarrow s_0 = 1/7$	790	0.19
$r_1 \rightarrow s_1 = 3/7$	1023	0.25
$r_2 \rightarrow s_2 = 5/7$	850	0.21
$r_3, r_4 \rightarrow s_3 = 6/7$	985	0.24
$r_5, r_6, r_7 \rightarrow s_4 = 1$	448	0.11

- Now we compute the transformation G as before.

$$v_0 = G(z_0) = \sum_{j=0}^{0} p_{out}(z_j) = p_{out}(z_0) = 0.00 \rightarrow 0$$

$$v_1 = G(z_1) = \sum_{j=0}^{1} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) = 0.00 \rightarrow 0$$

$$v_2 = G(z_2) = \sum_{j=0}^{2} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + p_{out}(z_2) = 0.00 \rightarrow 0$$

$$v_3 = G(z_3) = \sum_{j=0}^{3} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + \cdots + p_{out}(z_3) = 0.15 \rightarrow 1/7$$

$$v_4 = G(z_4) = \sum_{j=0}^{4} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + \cdots + p_{out}(z_4) = 0.35 \rightarrow 2/7$$

$$v_5 = G(z_5) = \sum_{j=0}^{5} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + \cdots + p_{out}(z_5) = 0.65 \rightarrow 5/7$$

$$v_6 = G(z_6) = \sum_{j=0}^{6} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + \cdots + p_{out}(z_6) = 0.85 \rightarrow 6/7$$

$$v_7 = G(z_7) = \sum_{j=0}^{7} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + \cdots + p_{out}(z_7) = 1.00 \rightarrow 1$$

- Computer $z=G^{-1}(s)$, Notice that G is not invertible.

$G^{-1}(0) = ?$

$G^{-1}(1/7) = 3/7$

$G^{-1}(2/7) = 4/7$

$G^{-1}(3/7) = ?$

$G^{-1}(4/7) = ?$

$G^{-1}(5/7) = 5/7$

$G^{-1}(6/7) = 6/7$

$G^{-1}(1) = 1$

$$v_0 = G(z_0) = \sum_{j=0}^{0} p_{out}(z_j) = p_{out}(z_0) = 0.00 \rightarrow 0$$

$$v_1 = G(z_1) = \sum_{j=0}^{1} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) = 0.00 \rightarrow 0$$

$$v_2 = G(z_2) = \sum_{j=0}^{2} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + p_{out}(z_2) = 0.00 \rightarrow 0$$

$$v_3 = G(z_3) = \sum_{j=0}^{3} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + \cdots + p_{out}(z_3) = 0.15 \rightarrow 1/7$$

$$v_4 = G(z_4) = \sum_{j=0}^{4} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + \cdots + p_{out}(z_4) = 0.35 \rightarrow 2/7$$

$$v_5 = G(z_5) = \sum_{j=0}^{5} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + \cdots + p_{out}(z_5) = 0.65 \rightarrow 5/7$$

$$v_6 = G(z_6) = \sum_{j=0}^{6} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + \cdots + p_{out}(z_6) = 0.85 \rightarrow 6/7$$

$$v_7 = G(z_7) = \sum_{j=0}^{7} p_{out}(z_j) = p_{out}(z_0) + p_{out}(z_1) + \cdots + p_{out}(z_7) = 1.00 \rightarrow 1$$

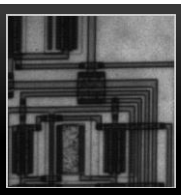
- Combining the two transformation T and G^{-1}, compute $z=H(r)= G^{-1}[v=s=T(r)]$

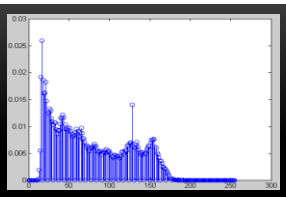
$r \rightarrow T(r)=s$	$s \rightarrow G^{-1}(s)=z$	$r \rightarrow z=H(r)= G^{-1}[T(r)]$
$r_0 = 0 \rightarrow 1/7$	$s_0 = 0 \rightarrow ?$	$r_0 = 0 \rightarrow G^{-1}[1/7]=3/7$
$r_1 = 1/7 \rightarrow 3/7$	$s_1 = 1/7 \rightarrow 3/7$	$r_1 = 1/7 \rightarrow G^{-1}[3/7]=? 4/7$
$r_2 = 2/7 \rightarrow 5/7$	$s_2 = 2/7 \rightarrow 4/7$	$r_2 = 2/7 \rightarrow G^{-1}[5/7]=5/7$
$r_3 = 3/7 \rightarrow 6/7$	$s_3 = 3/7 \rightarrow ?$	$r_3 = 3/7 \rightarrow G^{-1}[6/7]=6/7$
$r_4 = 4/7 \rightarrow 6/7$	$s_4 = 4/7 \rightarrow ?$	$r_4 = 4/7 \rightarrow G^{-1}[6/7]=6/7$
$r_5 = 5/7 \rightarrow 1$	$s_5 = 5/7 \rightarrow 5/7$	$r_5 = 5/7 \rightarrow G^{-1}[1]=1$
$r_6 = 6/7 \rightarrow 1$	$s_6 = 6/7 \rightarrow 6/7$	$r_6 = 6/7 \rightarrow G^{-1}[1]=1$
$r_7 = 1 \rightarrow 1$	$s_7 = 1 \rightarrow 1$	$r_7 = 1 \rightarrow G^{-1}[1]=1$

- Applying the transformation H to the original image yields an image with histogram as below:

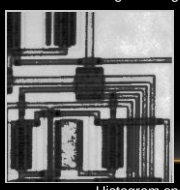
k	z_k	n_k	n_k/n (actual hist.)	$p_{out}(z_k)$ (specified hist.)
0	0	0	0.00	0.00
1	1/7	0	0.00	0.00
2	2/7	0	0.00	0.00
3	3/7	790	0.19	0.15
4	4/7	1023	0.25	0.20
5	5/7	850	0.21	0.30
6	6/7	985	0.24	0.20
7	1	448	0.11	0.15

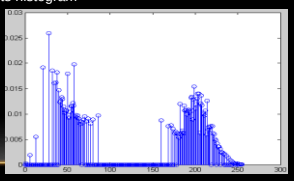
- Again, the actual histogram of the output image does not exactly but only approximately matches with the specified histogram. This is because we are dealing with discrete histograms.



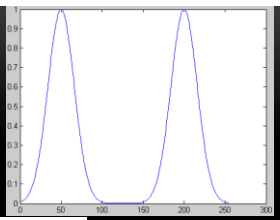


Original image and its histogram





Histogram specified image and its histogram



Desired histogram

```
>> clear
>> x=imread('circuit.tif');
>> imshow(x);
>> hx=histogram(double(x));
>> figure
>> stem(0:255,hx);
>> D=0:255;
>> h = exp(-(D-50).^2/(2*16^2)) + exp(-(D-200).^2/(2*16^2));
>> y=histeq(x,h);
>> figure
>> imshow(y);
>> hy=histogram(double(y));
>> figure
>> stem(0:255,hy);
>> figure
>> plot(0:255,h)
```