

Digital Image Processing

Image Enhancement
(Point Processing)

What Is Image Enhancement?

Image enhancement (图像增强) is the process of making images more useful

The reasons for doing this include:

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

Image Enhancement Examples

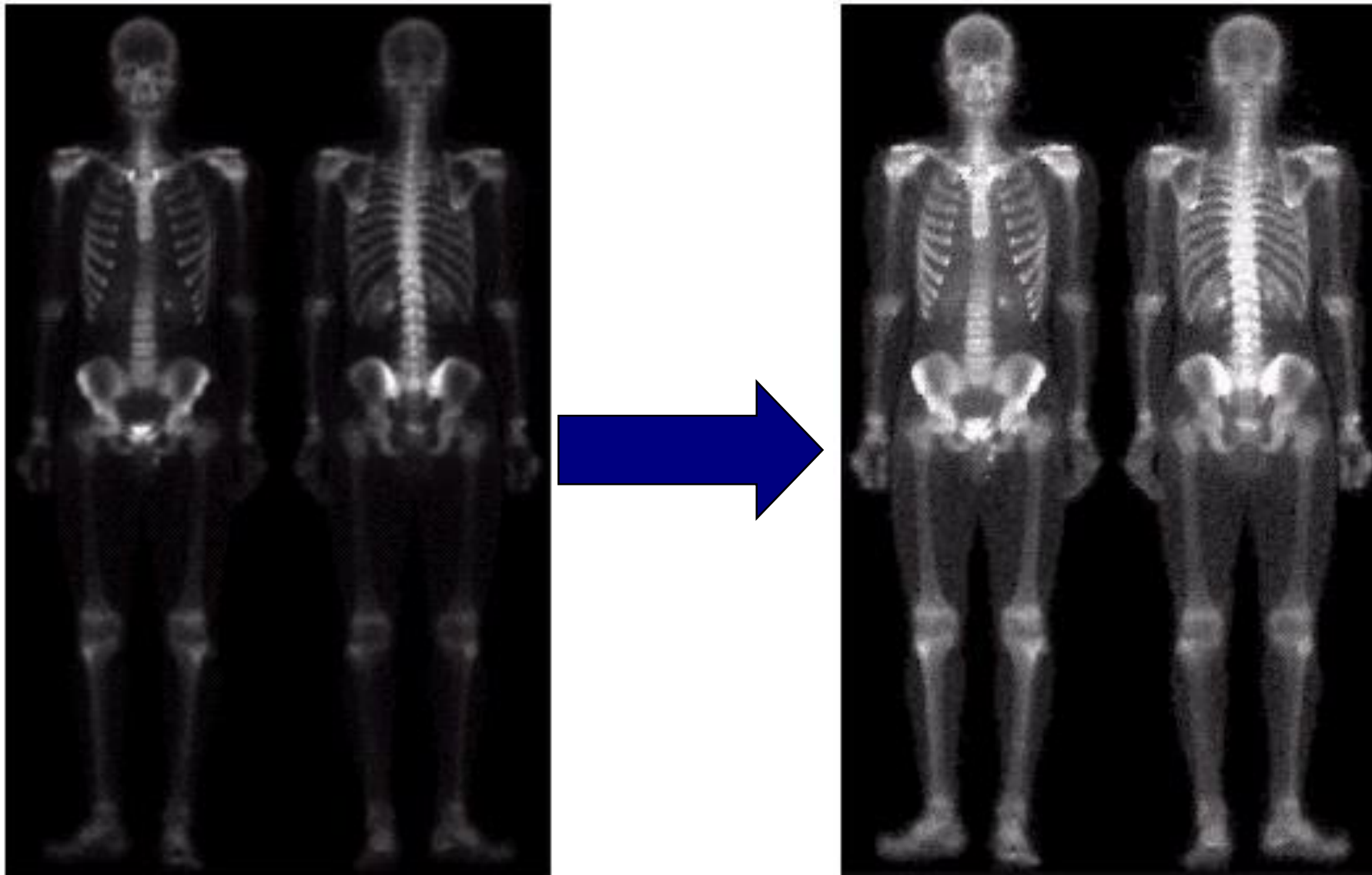
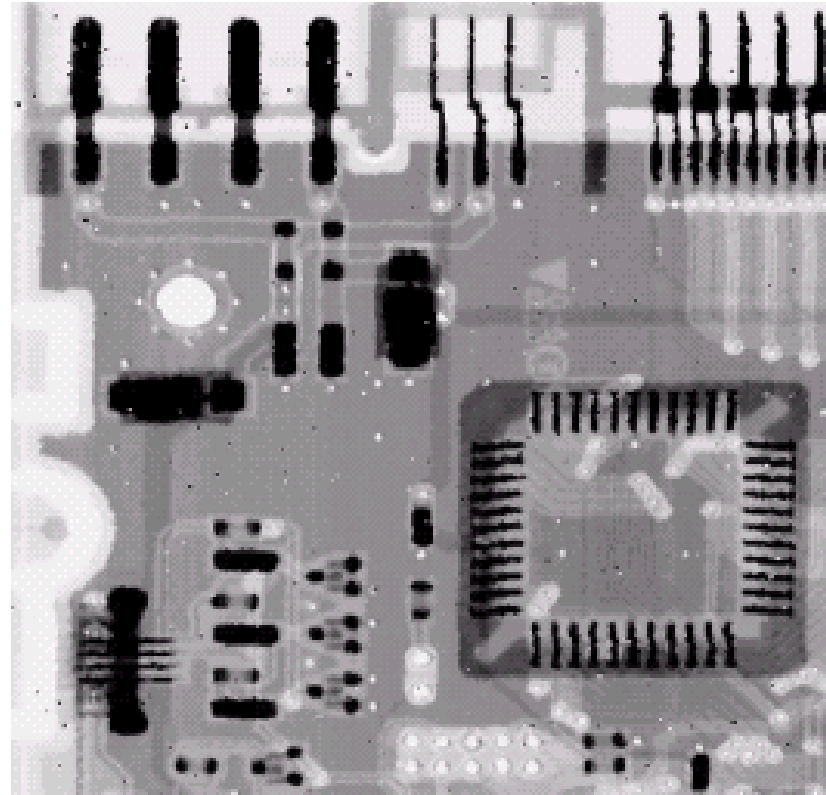
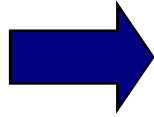
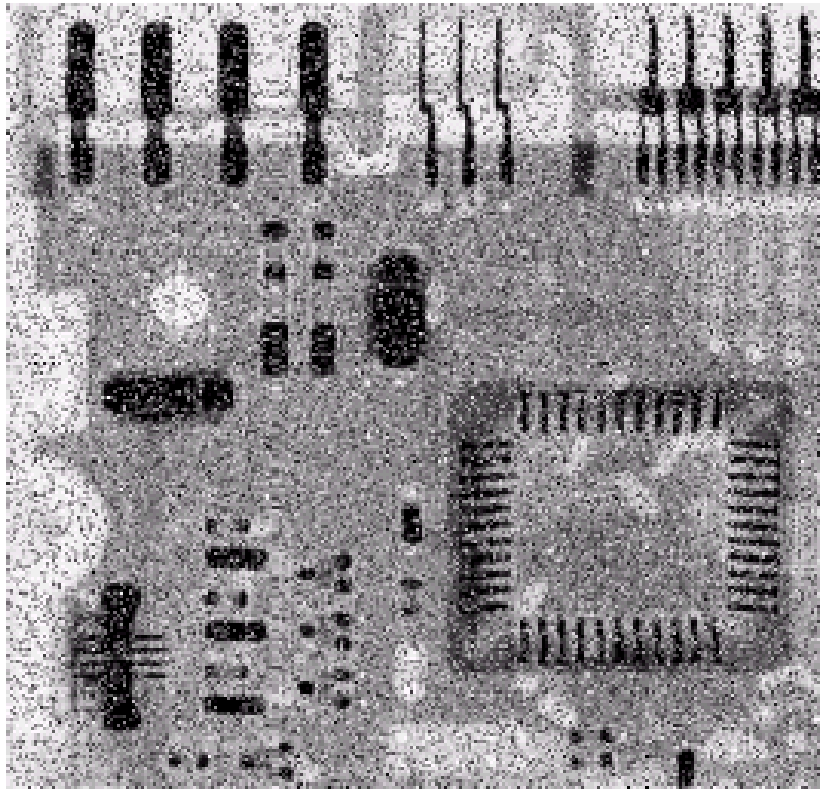


Image Enhancement Examples (cont...)





So far when we have spoken about image **gray level values** we have said they are in **the range [0, 255]**

- Where 0 is black and 255 is white

For many of the image processing operations in this lecture gray levels are assumed to be given in **the range [0.0, 1.0]**

可以简单的认为这是归一化

Two broad categories of image enhancement techniques

- Spatial domain techniques (空域)
 - Direct manipulation of image pixels
 - Point processing
 - Neighborhood operations
- Frequency domain techniques (频域)
 - Manipulation of Fourier transform or wavelet transform of an image

For the moment we will concentrate on techniques that operate in the spatial domain

Basic Spatial Domain Image Enhancement

Most spatial domain enhancement operations can be reduced to the form

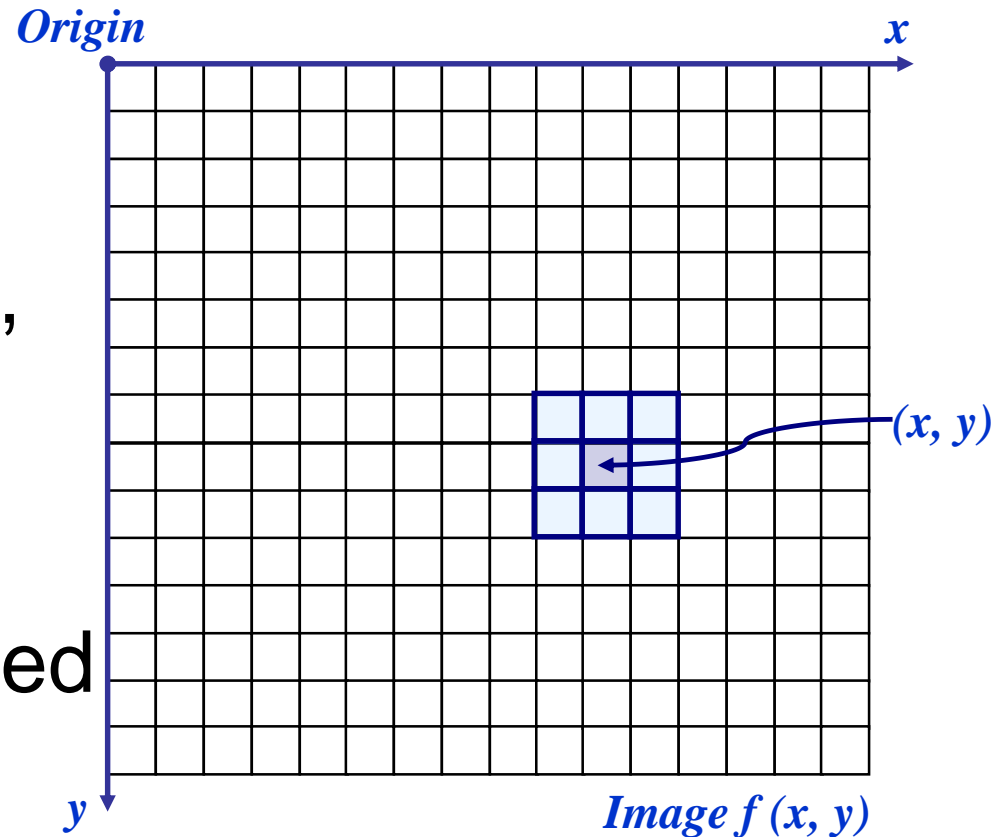
$$g(x, y) = T[f(x, y)]$$

where

$f(x, y)$: the input image,

$g(x, y)$: the processed image

T : some operator defined
some neighbourhood
of (x, y)



Basic Spatial Domain Image Enhancement

The simplest spatial domain operations occur when the neighbourhood is simply the pixel itself: **Point Processing**

In this case T is referred to as a *gray level transformation function* or a *point processing operation*

Point processing operations take the form

$$s = T (r)$$

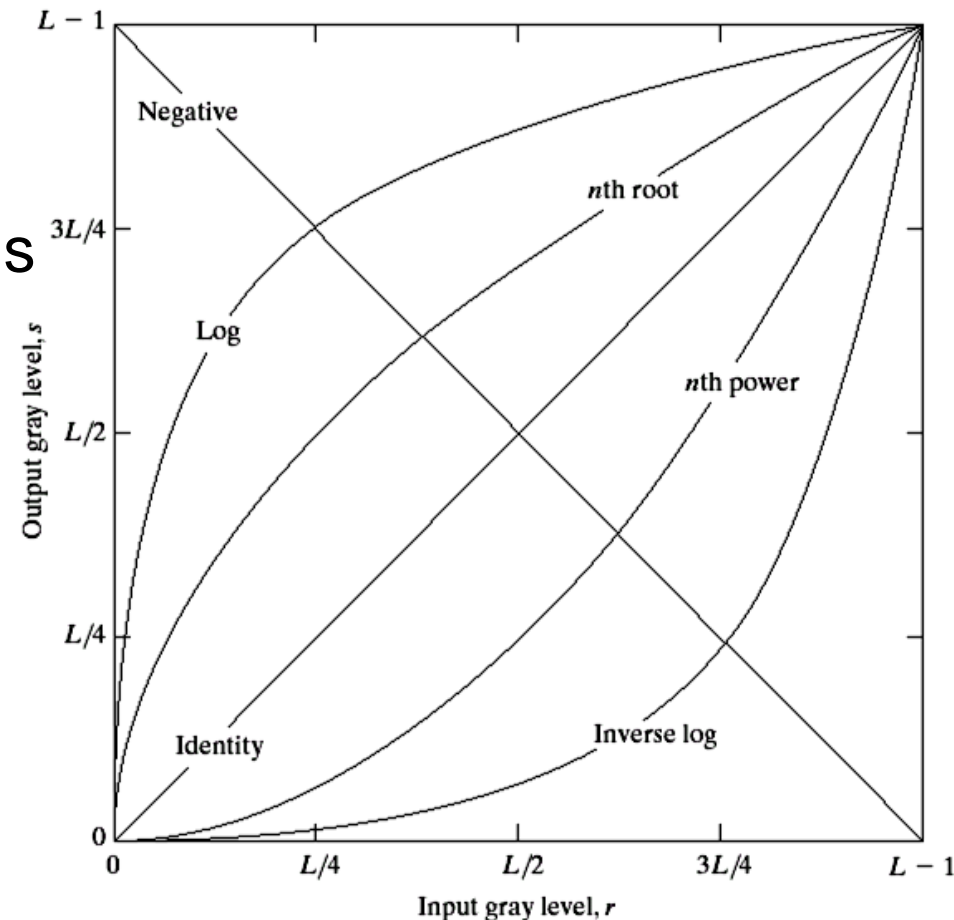
where s refers to the processed image pixel value and r refers to the original image pixel value

Basic Spatial Domain Image Enhancement

- Point processing
 - The neighborhood is of size 1×1
 - Gray-level transformation
- Mask processing (遮罩处理) or filtering
 - The neighborhood is defined as a mask, filter, or window.
 - Filtering

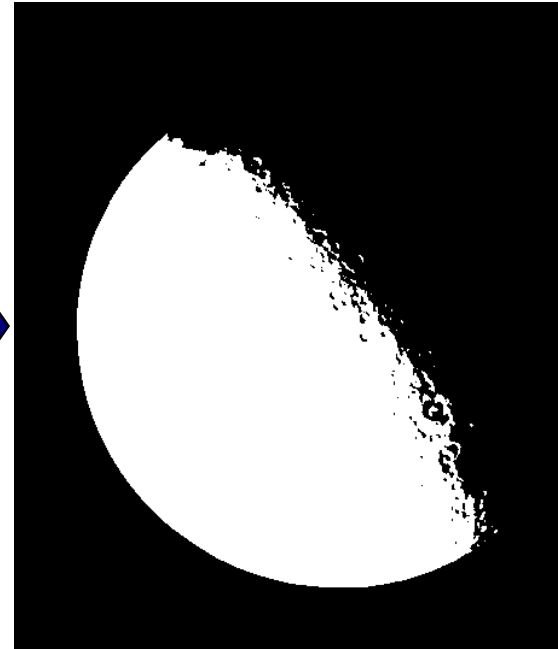
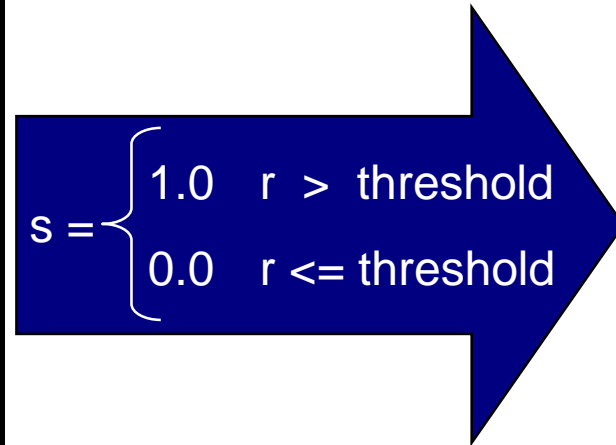
In this lecture we will look at image enhancement point processing techniques:

- Thresholding
- Logarithmic transformation
- Power law (幂律) transforms
- Gray level slicing (分切)
- Bit plane slicing
- Image subtraction
- Image averaging



Thresholding (閾値) transformations are particularly useful for segmentation (分割) in which we want to isolate an object of interest from a background

$$s = \begin{cases} 1.0 & r > threshold \\ 0.0 & r \leq threshold \end{cases}$$



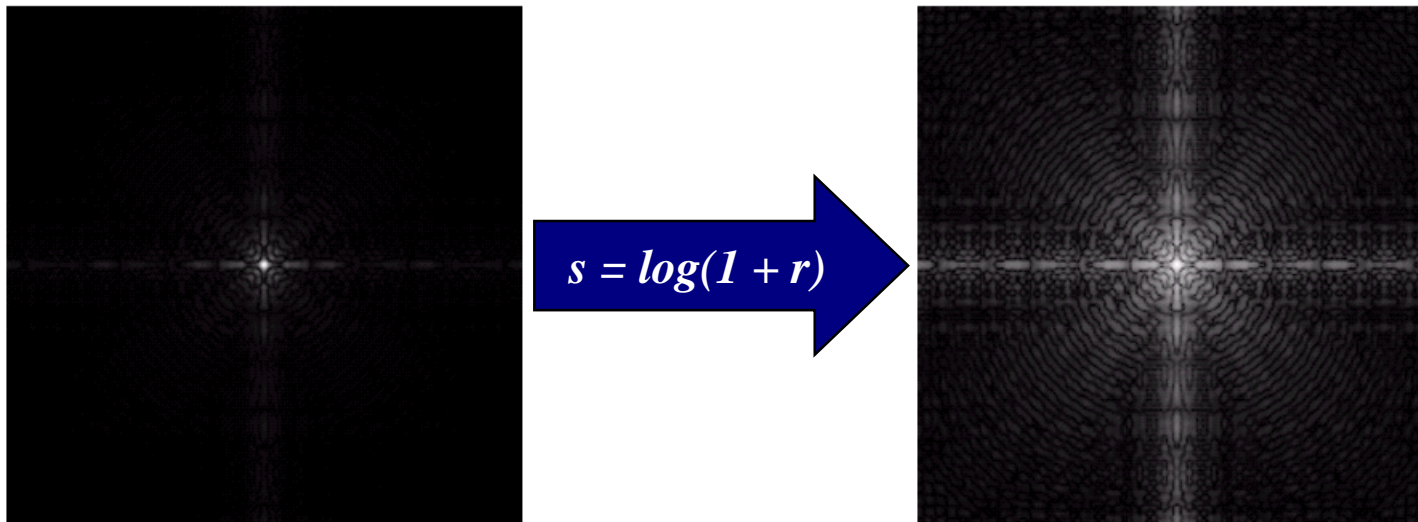
The general form of the log transformation is

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

Log functions are particularly useful when the input grey level values may have an extremely large range of values.

In Fourier transform, we usually encounter spectrum values that ranges from 0 to 10^6 or higher. But image display system cannot reproduce such a wide range of intensity values.

In the following example the Fourier transform of an image is put through a log transform to reveal more detail



Example: `logtransform.m`

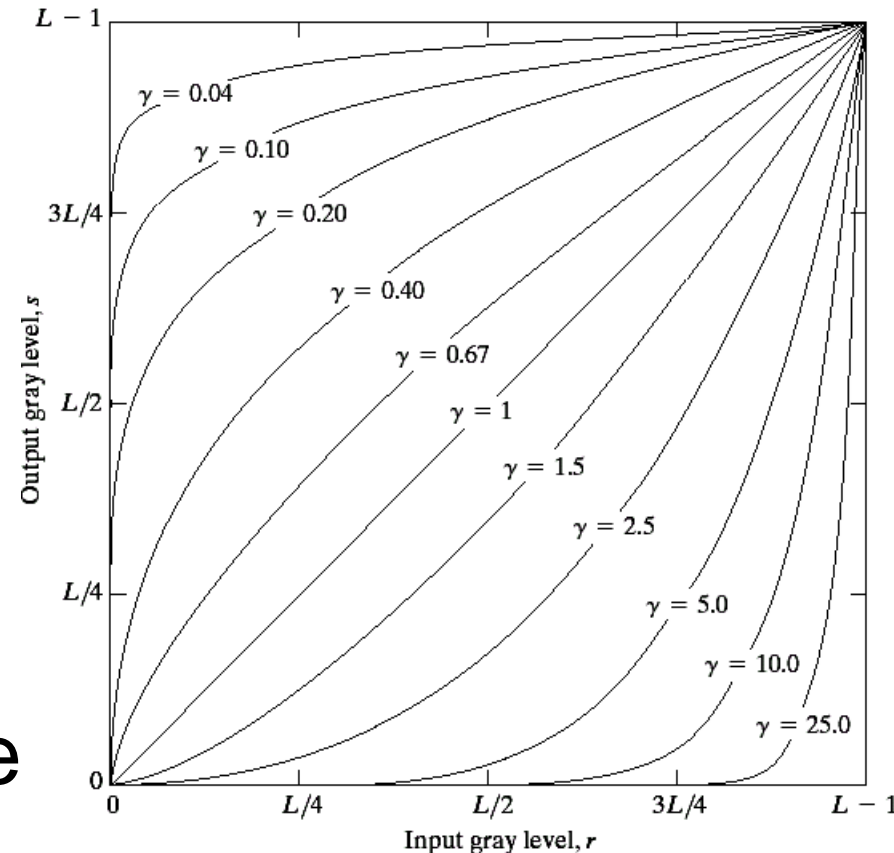
Power Law Transformations

Power law transformations have the following form

$$s = c * r^\gamma$$

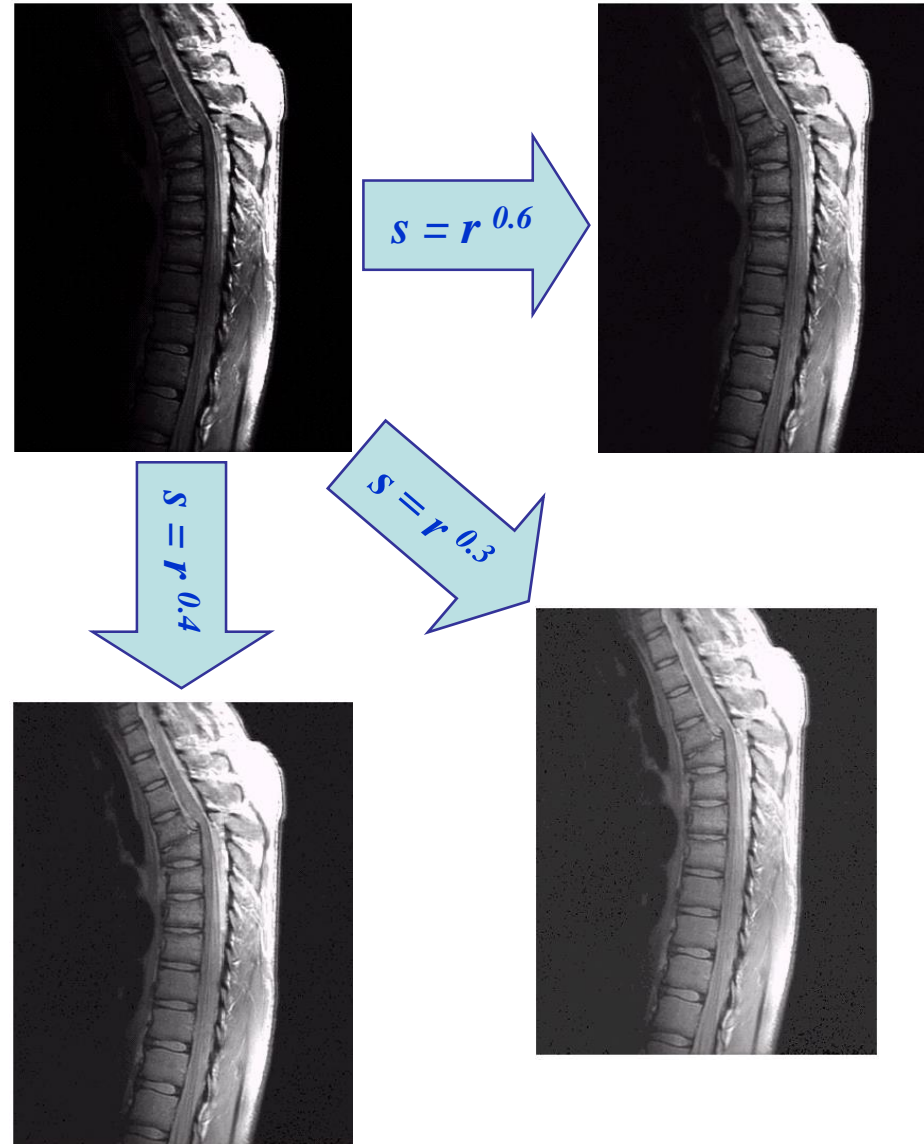
Map a narrow range of dark input values into a wider range of output values or vice versa

Varying γ gives a whole family of curves



Power Law Example (cont...)

- ◆ The images to the right show a magnetic resonance (MR) image of a fractured (断裂) human spine (脊柱).
- ◆ Different curves highlight different detail.

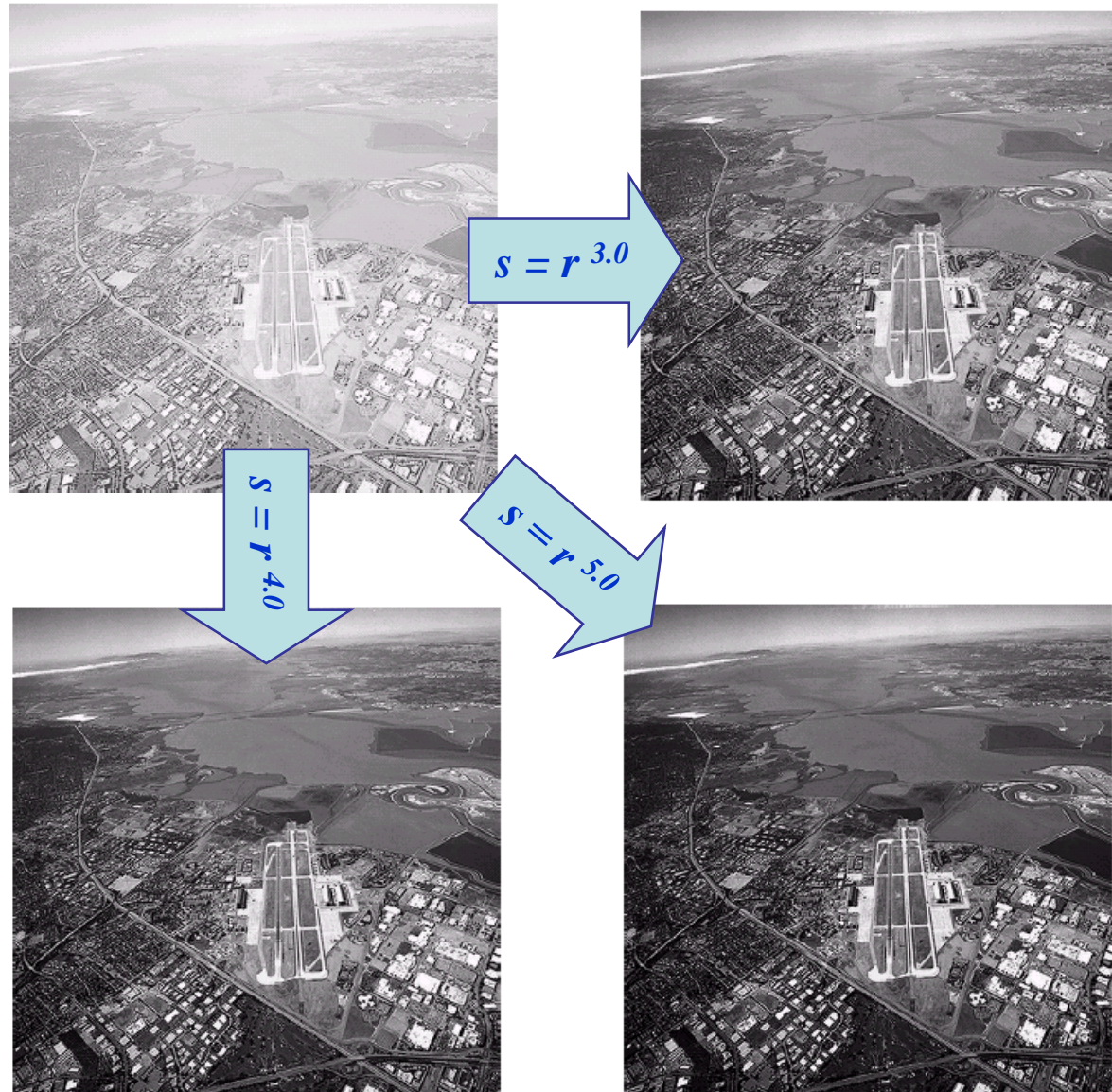


Exam: [powertransform.m](#)

Power Law Transformations (cont...)

- ◆ An aerial photo of a runway is shown.
- ◆ This time power law transforms are used to darken the image.

Exam:
powertransform2.m



Many of you might be familiar with gamma correction of computer monitors.

Problem is that display devices, print devices do not respond linearly to different intensities, and they respond according to a power law:

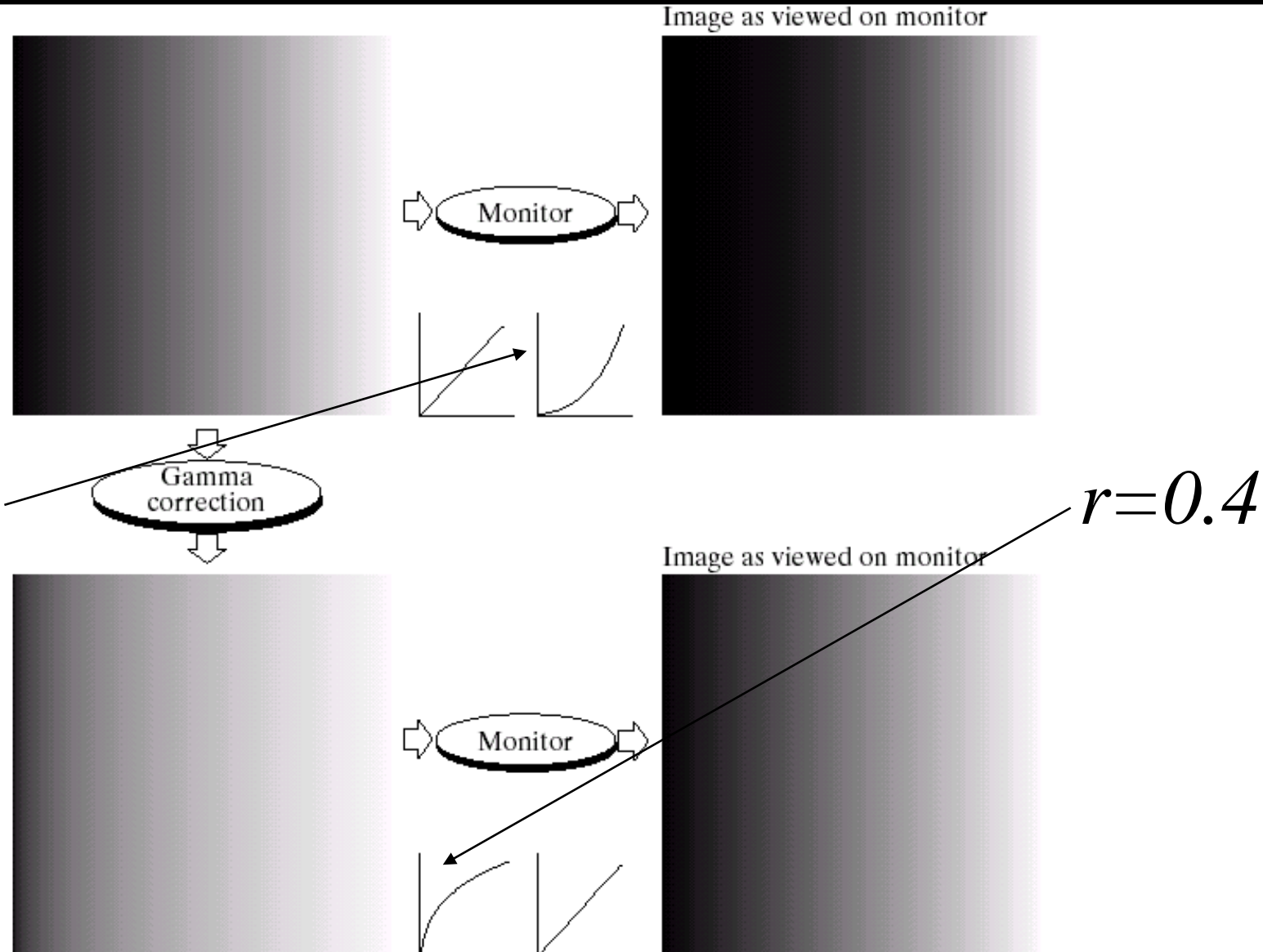
$$s = r^\gamma$$

For CRT display, $\gamma=1.8\sim 2.5$

Can be corrected using a *nth* root transform:

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

Gamma Correction



❖ Prior Knowledge about gamma correction

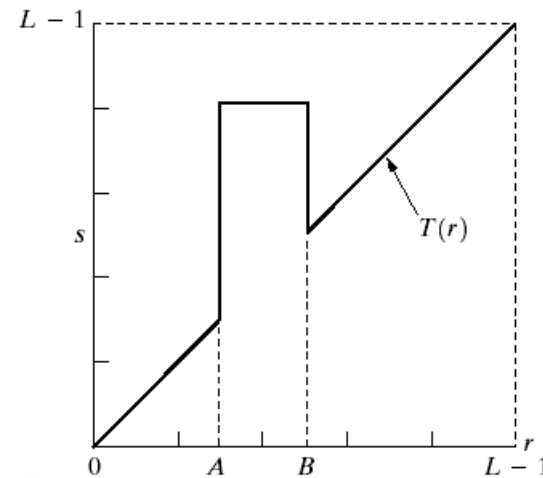
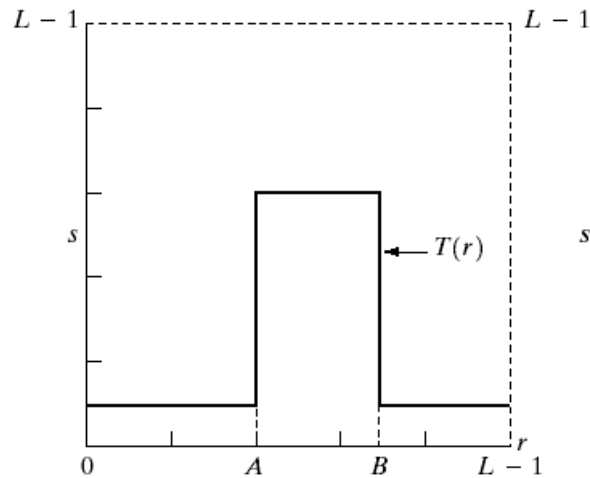
- Varying the value of gamma correction changes not only the brightness, but also the ratio of red to green to blue.

❖ Applications

- Internet
- Millions of people and millions of monitors
- Gamma represents an “average ” of the types of monitors and computer systems
- Scanners, Printers have different values of gamma

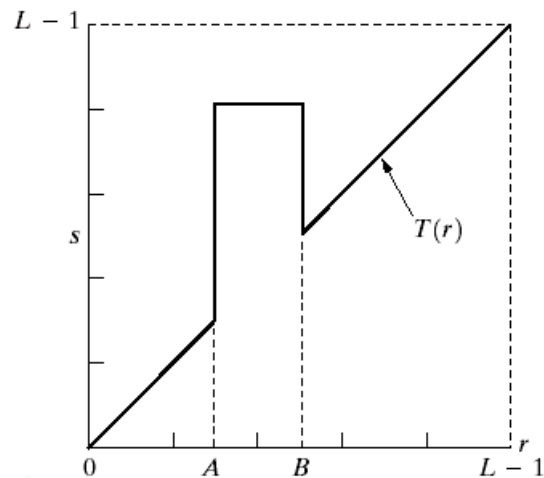
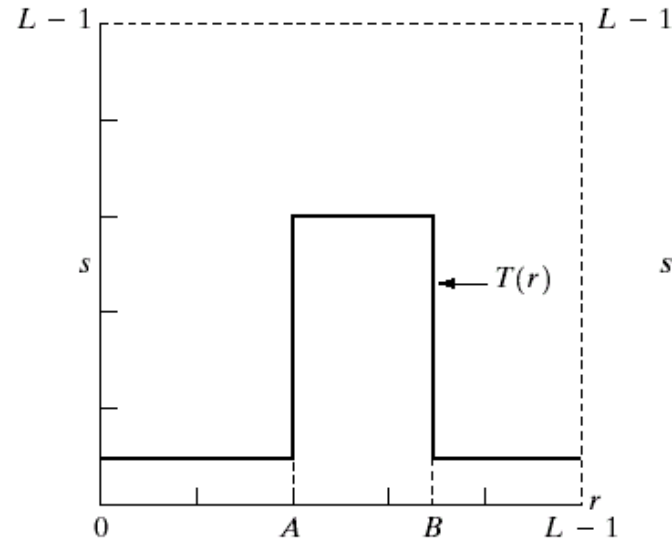
Highlights a specific range of grey levels

- Similar to thresholding
- Other levels can be suppressed or maintained
- Useful for highlighting features in an image

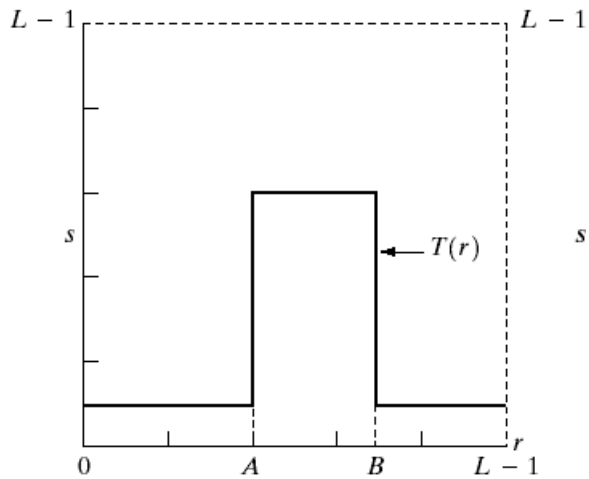


$$y = \begin{cases} s1 & x < a, x > b \\ s2 & a \leq x \leq b \end{cases}$$

$$s2 > s1$$



$$y = \begin{cases} x & x < a, x > b \\ c & a \leq x \leq b \end{cases}$$

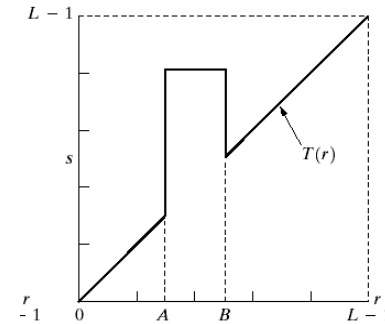
Exam: `grayslicing.m`

```
a=210;  
b=240;
```

```
l=find(x<a);  
y(l)=0;
```

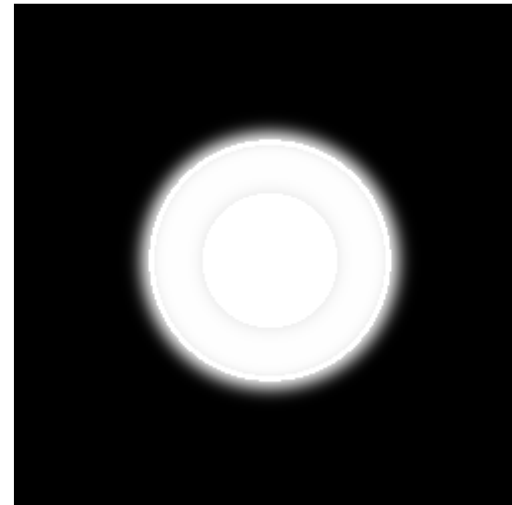
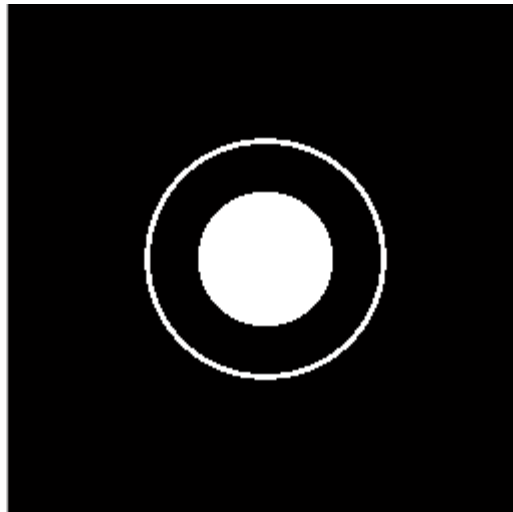
```
l2=find(x>b);  
y(l2)=0;
```

```
l3=find(x>=a&x<=b);  
y(l3)=255;
```



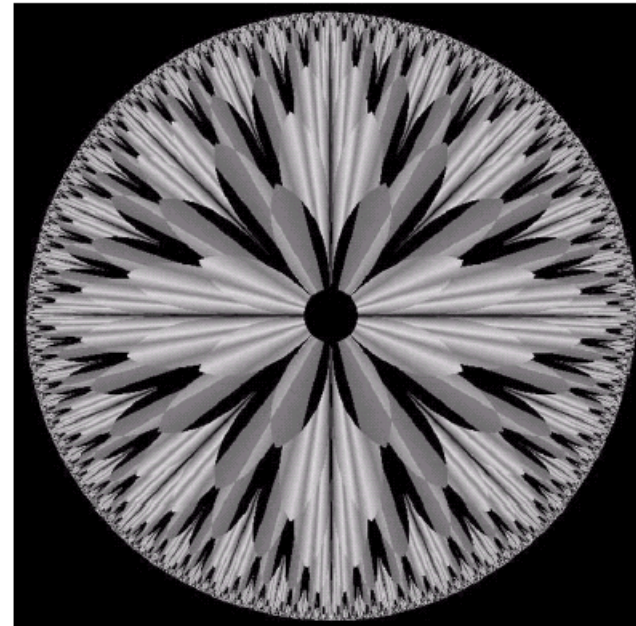
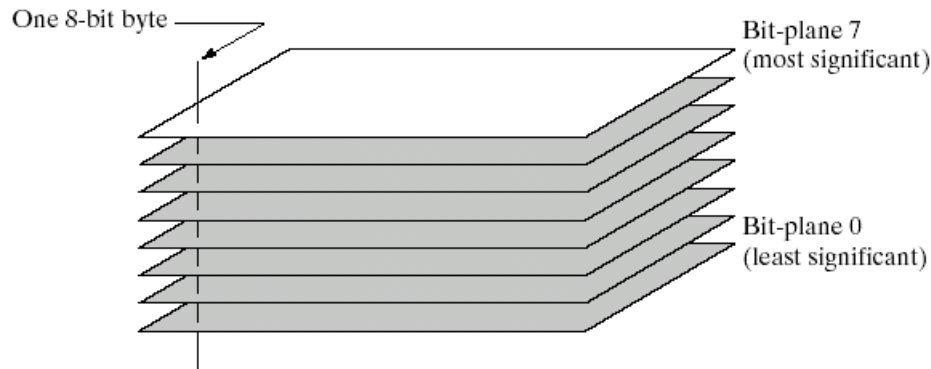
```
a=210;  
b=240;
```

```
y=x;  
l3=find(x>=a&x<=b);  
y(l3)=255;
```



Often by isolating particular bits of the pixel values in an image we can highlight interesting aspects of that image

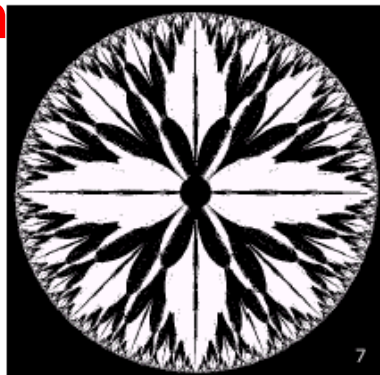
- Higher-order bits usually contain most of the significant visual information
- Lower-order bits contain subtle details



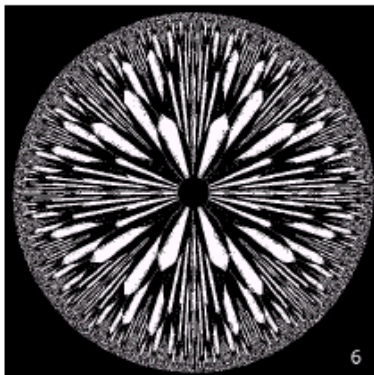
Bit Plane Slicing (cont...)

Exam: Bitplaneslicing.m

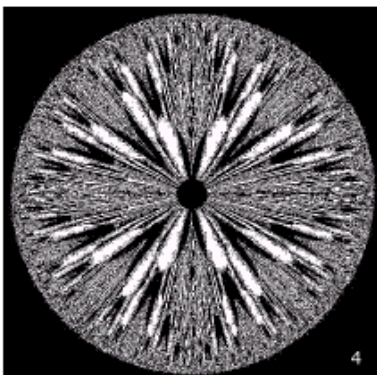
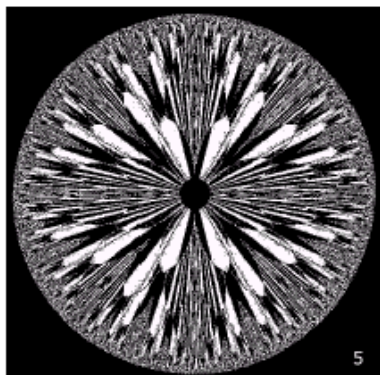
[10000000]



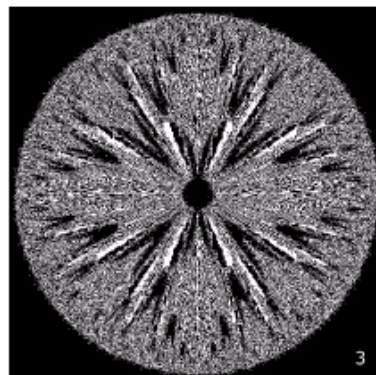
[01000000]



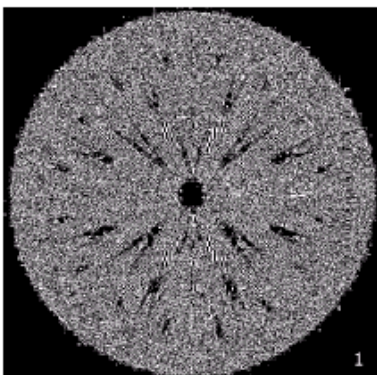
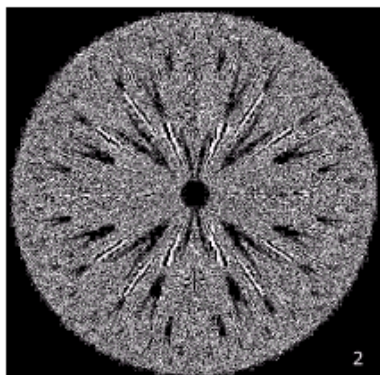
[00100000]



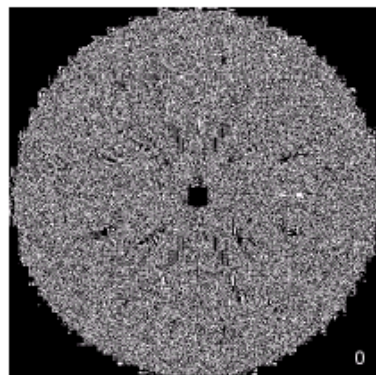
[00001000]



[00000100]



[00000001]



$$g(x, y) = f(x, y) - h(x, y)$$

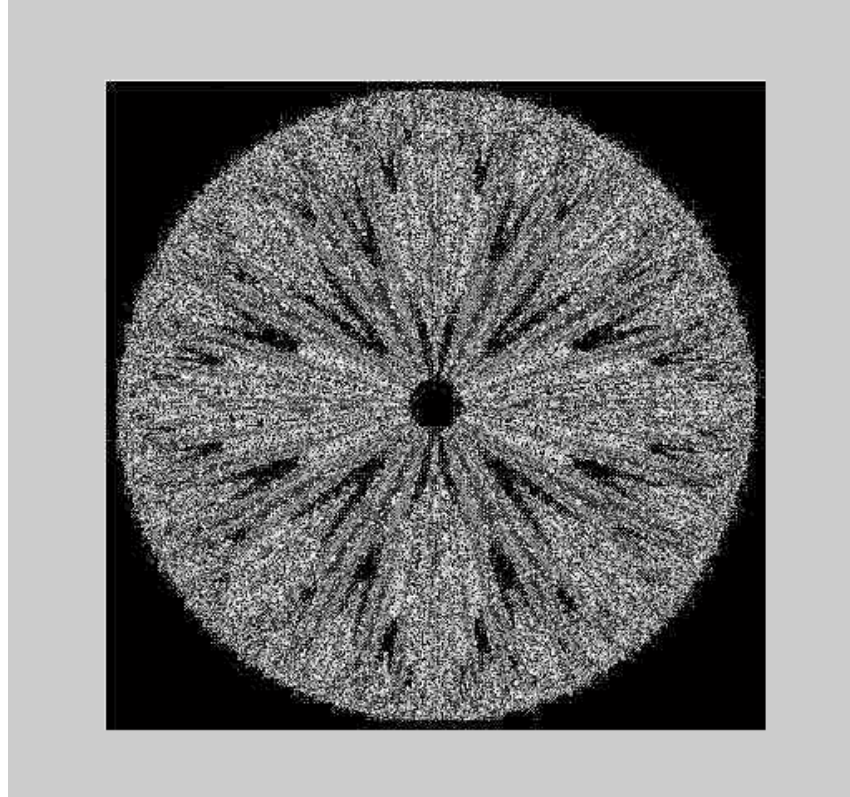
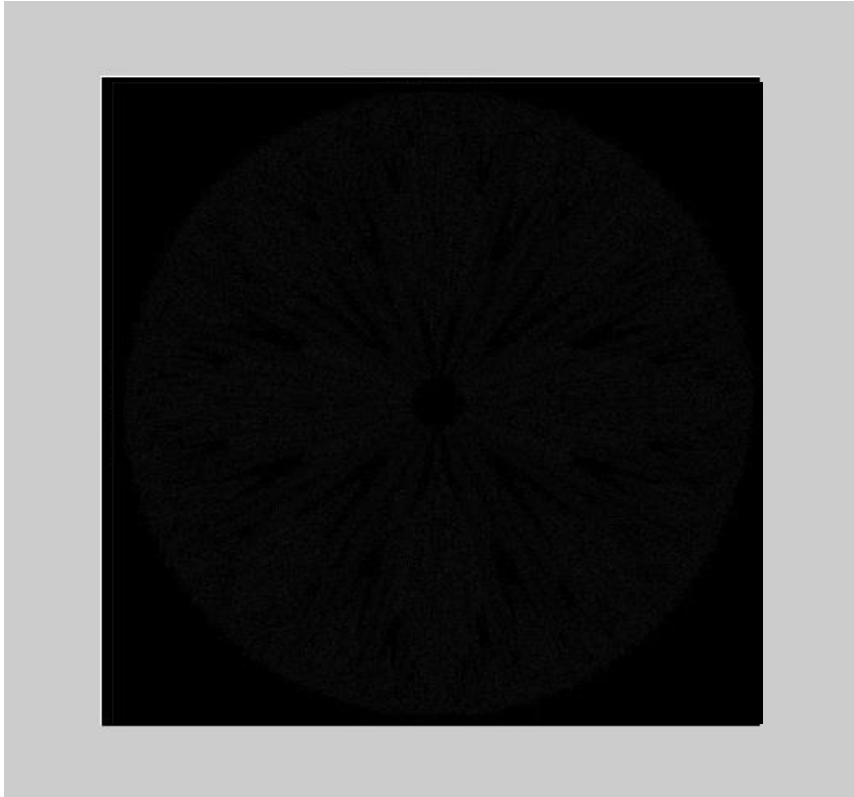
f and h are two images.

Note: most images are displayed using 8 bits (even 24-bit color images consists of three 8 bit channels). Thus we expect image values not to be outside the range from 0 to 255.

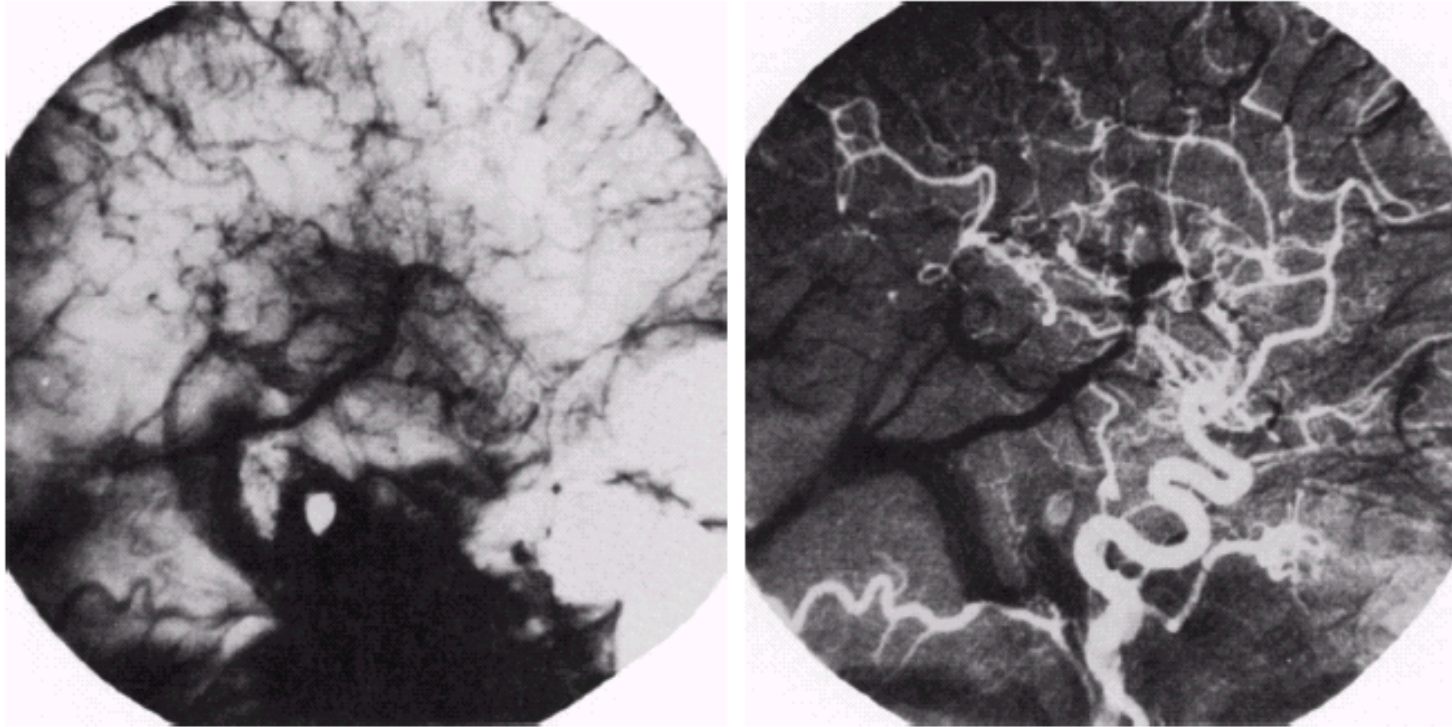
In subtraction, the results should be in the range -255 to 255. so some sort of scaling is required to display the results.

a) Add 255 and then divide by 2

b) $y = x - \min(x)$; $z = y * 255 / \max(y)$.



Exam: subtraction.m



a b

FIGURE 3.29

Enhancement by image subtraction.
(a) Mask image.
(b) An image (taken after injection of a contrast medium into the bloodstream) with mask subtracted out.

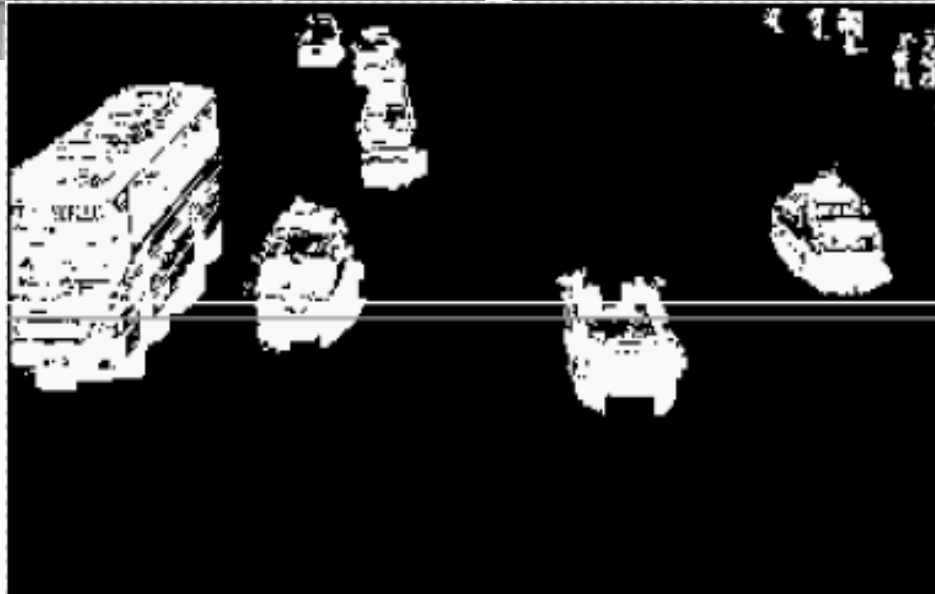
Mask mode radiography

– $h(x,y)$ is the mask

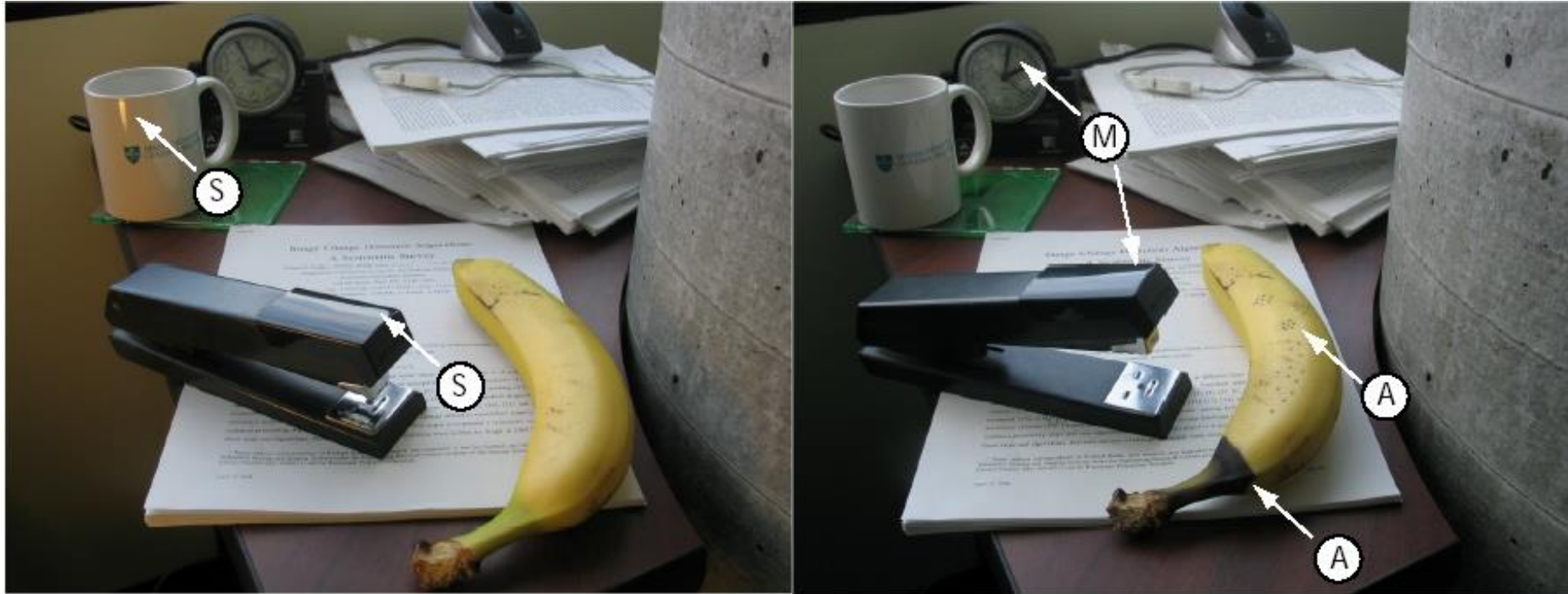
Change detection is another major application by using image subtraction. Such as:

- Tracking moving vehicles
- Tracking walking persons
- Change detections

Image Subtraction



Examples: Change Detection



A noisy image:

$$g(x, y) = f(x, y) + n(x, y)$$

where $n(x)$ is the noise, and it has zero average value

Then Averaging M different noisy images can reduce the noise:

$$\bar{g}(x, y) = \frac{1}{M} \sum_{i=1}^M g_i(x, y)$$

$$= f(x, y) + \frac{1}{M} \sum_{i=1}^M n_i(x, y)$$

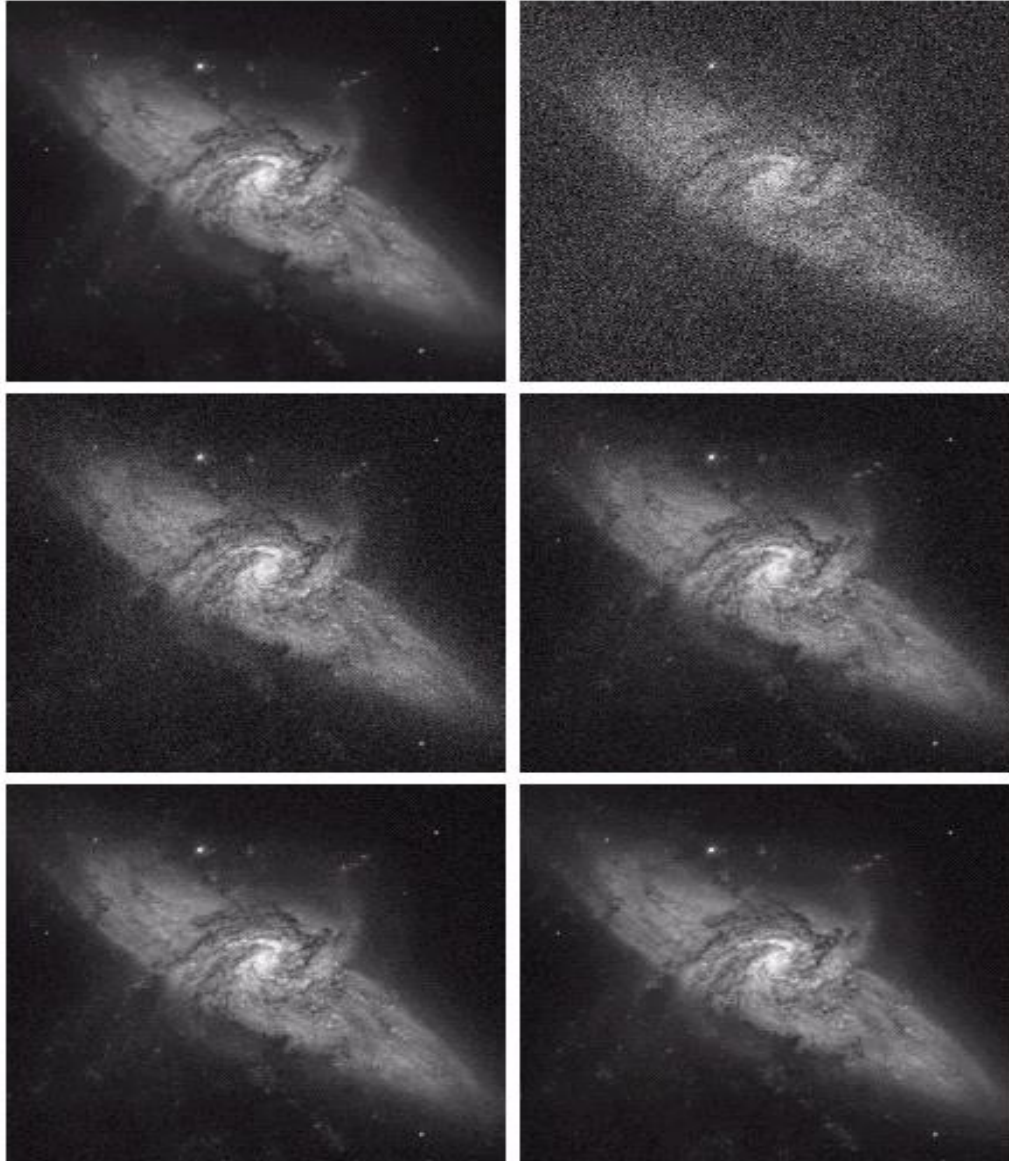
$$\sigma_{\bar{g}(x,y)}^2 = \frac{1}{M} \sigma_{n(x,y)}^2$$

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{M}} \sigma_{n(x,y)}$$

As M increases, the uncertainty of the pixel values at each location decreases.

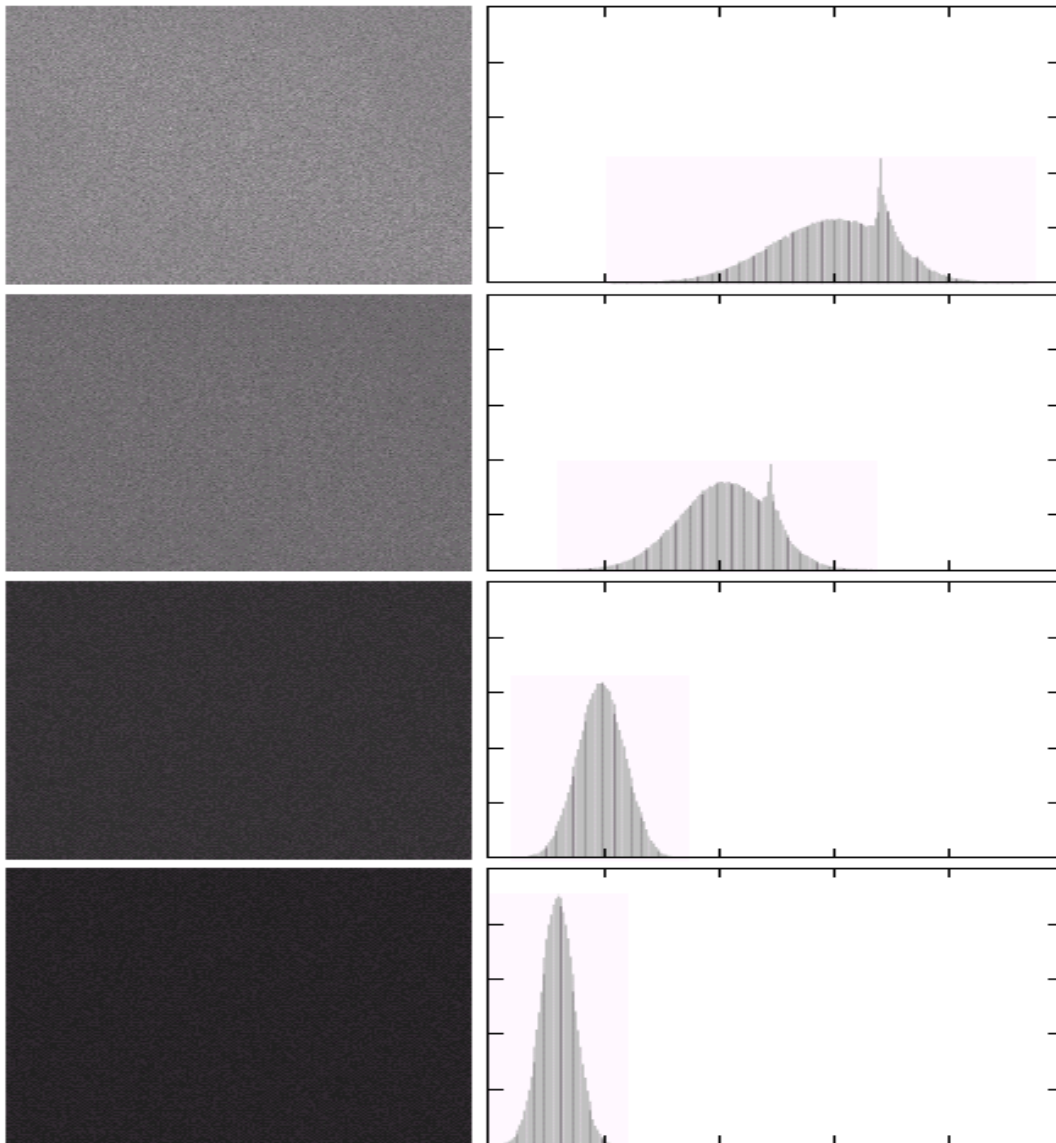
- This means that $\bar{g}(x,y)$ approaches $f(x,y)$ as the number of noisy images used in the averaging process increases.

$$\sigma_{\bar{g}(x,y)} = \frac{1}{\sqrt{M}} \sigma_{n(x,y)}$$

 $M=8,16,64,128$

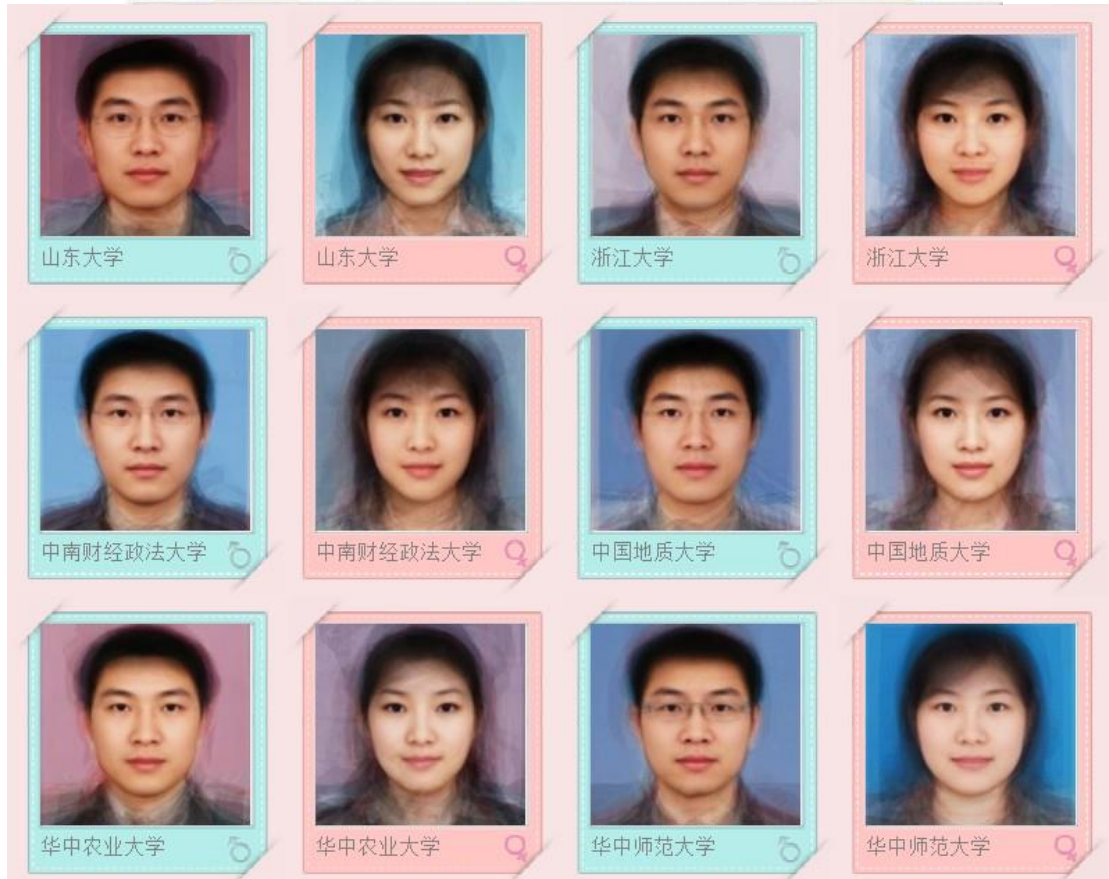
a	b
c	d
e	f

FIGURE 3.30 (a) Image of Galaxy Pair NGC 3314. (b) Image corrupted by additive Gaussian noise with zero mean and a standard deviation of 64 gray levels. (c)–(f) Results of averaging $K = 8, 16, 64$, and 128 noisy images. (Original image courtesy of NASA.)

**a b****FIGURE 3.31**

(a) From top to bottom: Difference images between Fig. 3.30(a) and the four images in Figs. 3.30(c) through (f), respectively. (b) Corresponding histograms.

Image Averaging



In average, the results of gray level should be in the range 0 to $255 \cdot k$. so some sort of scaling is required to display the results.

a) Divide by k .

b) $y = x - \min(x)$; $z = y \cdot 255 / \max(y)$.

The histogram (直方图) of a digital image with gray levels from 0 to $L-1$ is a discrete function:

$$p_r(r_k) = n_k \quad 0 \leq r_k \leq L-1 \quad k = 0, 1, 2, \dots, L-1$$

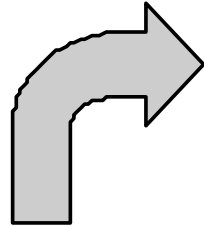
- r_k is the k th gray level
- n_k is the pixels in the image with that gray level

Normalized histogram: $p(r_k) = n_k/n$

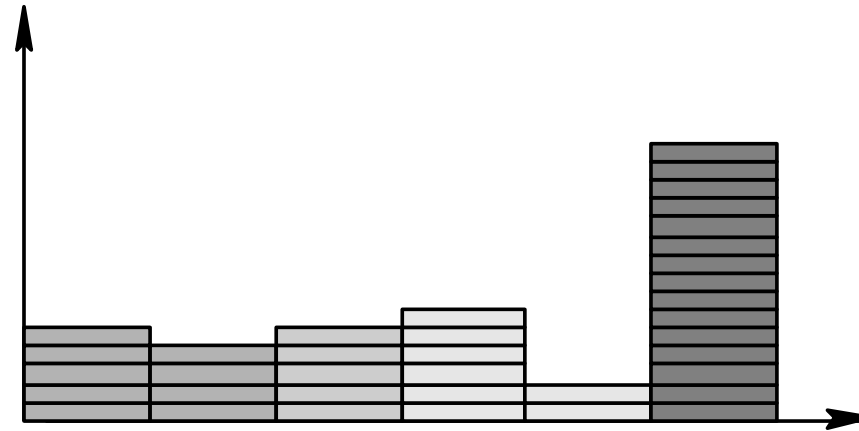
- n is the total number of pixels in the image
- sum of all components = 1

The histogram of an image shows us the distribution of grey levels in the image. It is useful in image processing, especially in enhancement and segmentation.

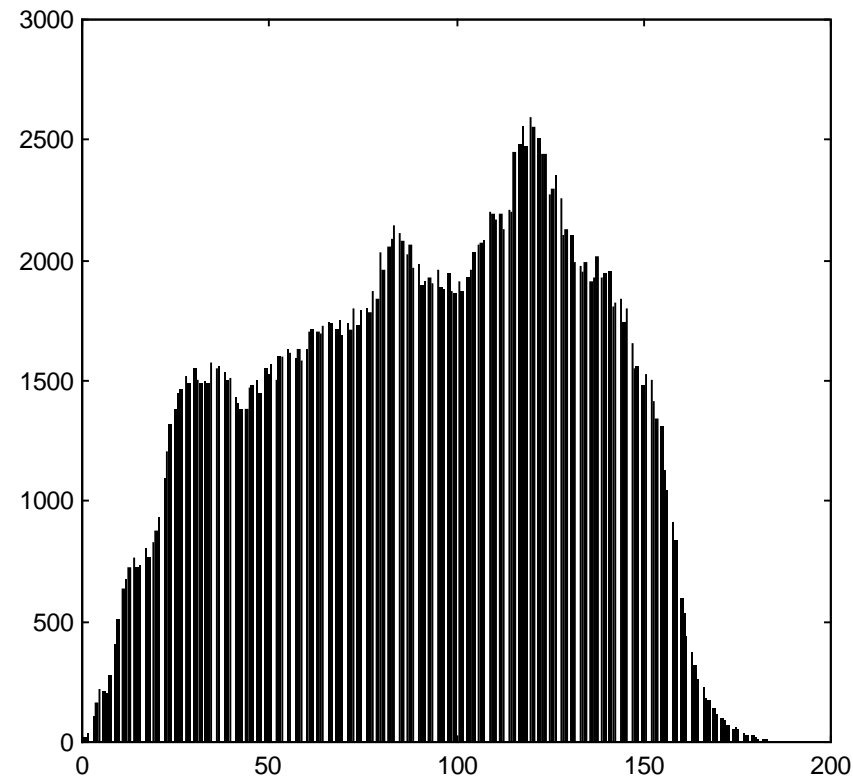
1	2	3	4	5	6
6	4	3	2	2	1
1	6	6	4	6	6
3	4	5	6	6	6
1	4	6	6	2	3
1	3	6	4	6	6



1	2	3	4	5	6
5	4	5	6	2	14
$5/36$	$4/36$	$5/36$	$6/36$	$2/36$	$14/36$



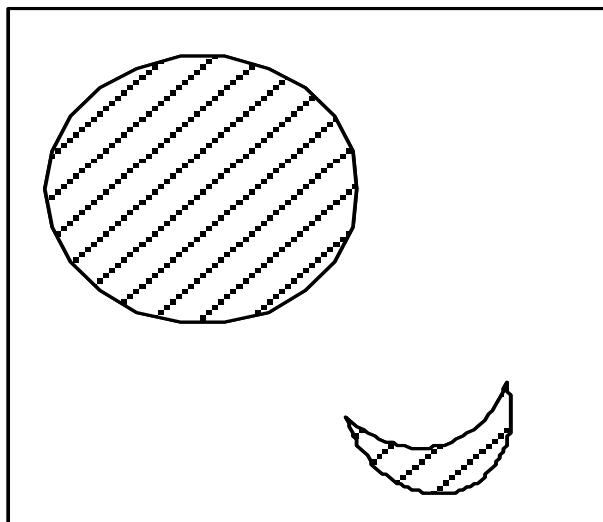
Histogram based Enhancement



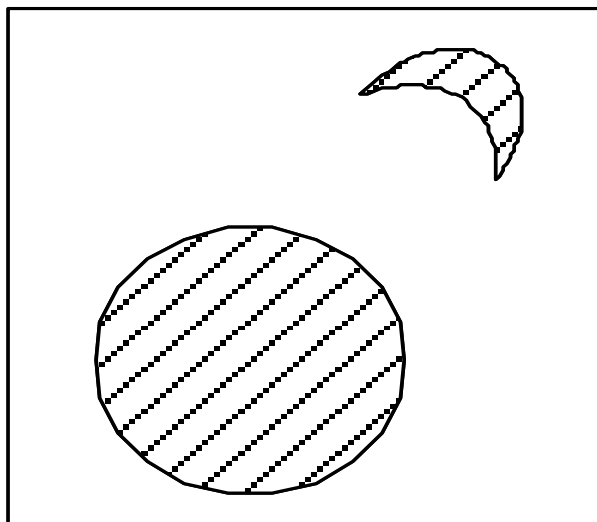
MATLAB function `>imhist(x)`

Properties of Image Histograms

- ◆ The histogram only shows the distribution of grey levels in the image, and **it doesn't include the location information of pixels.**
- ◆ One image has its corresponding histogram, but different images may have the same histograms. (不同的图像可能有相同的直方图)



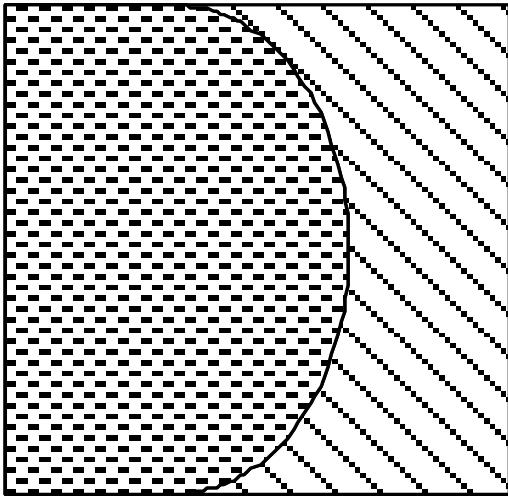
(a)



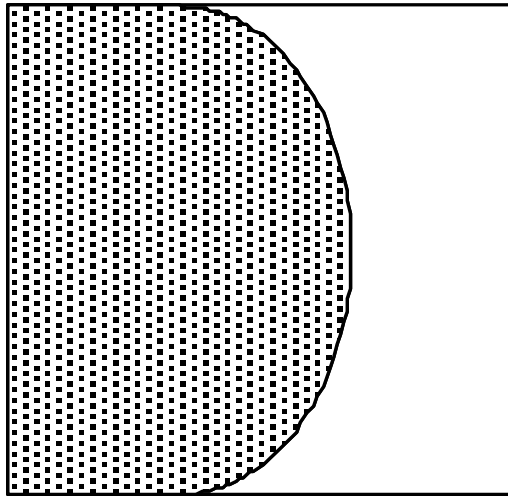
(b)

Properties of Image Histograms

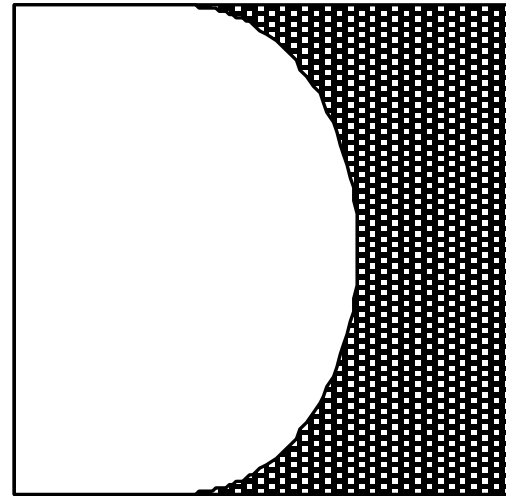
- ◆ Since the histogram is based on the statistics of the grey levels of pixels, the histogram of an image is equal to the sum of all its subimages.



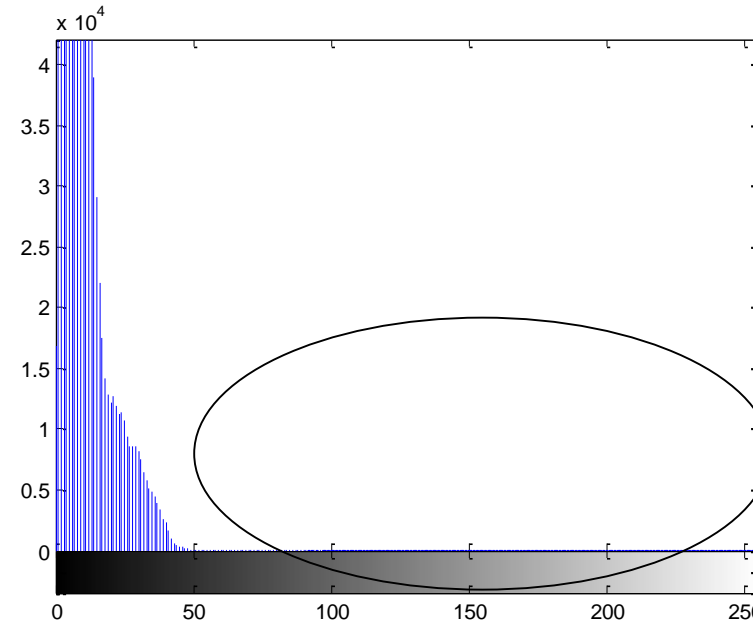
(a)



(b)

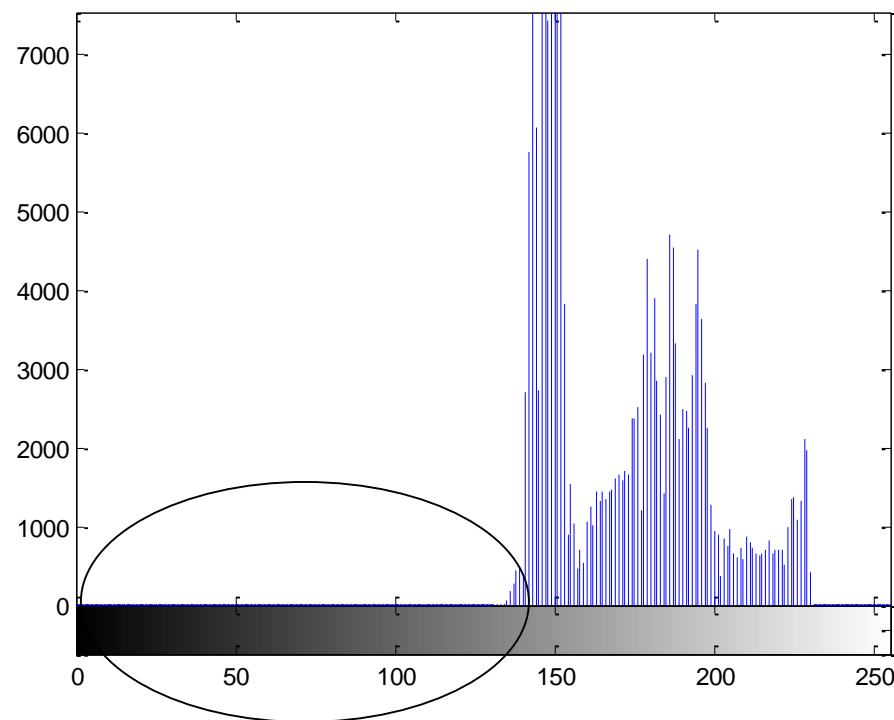


(c)



It is a baby in the cradle!

Histogram information reveals that image is under-exposed
(曝光不足)

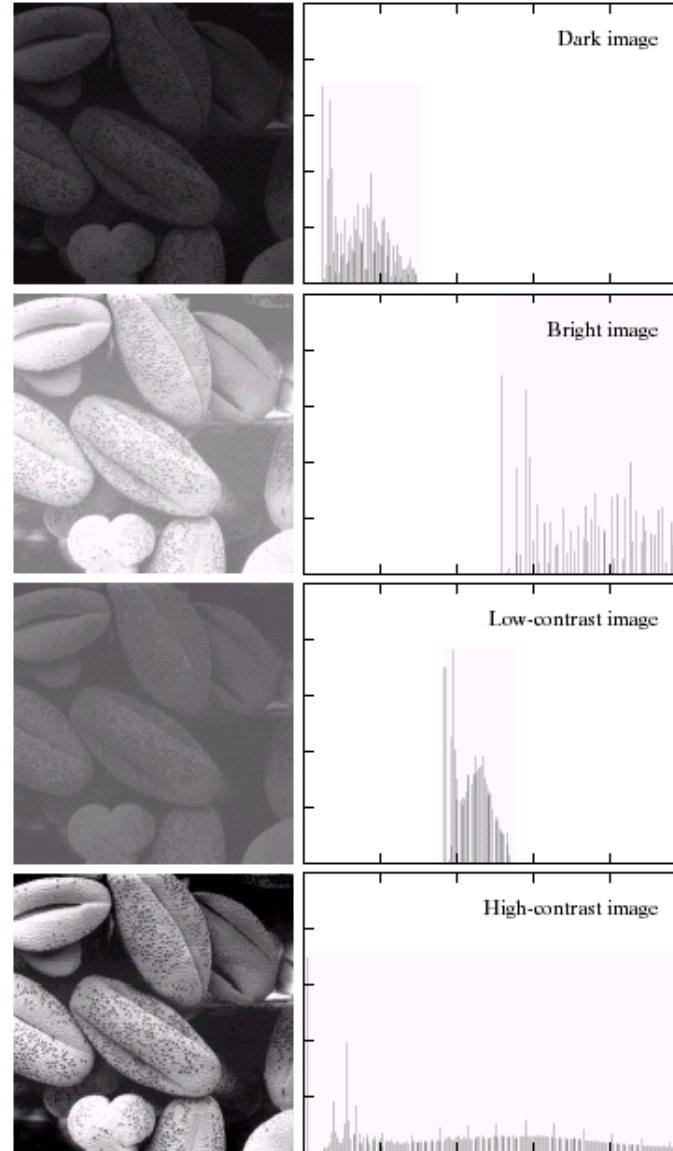


Over-exposed (过曝光) image

A selection of images and their histograms

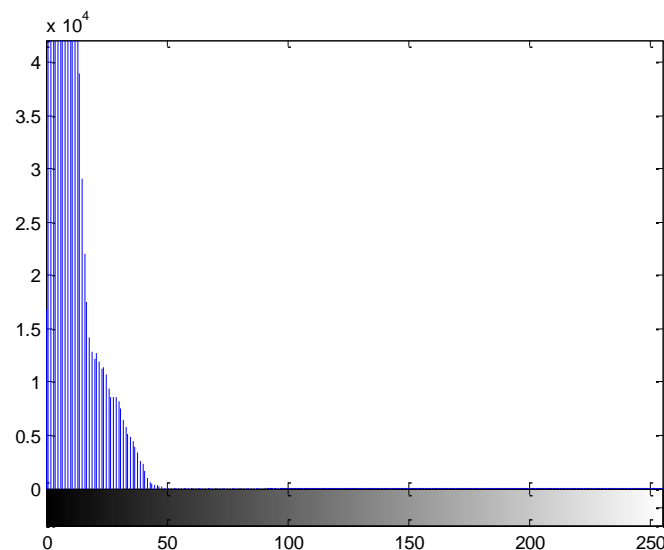
Notice the relationships between the images and their histograms

Note that the high contrast image has the most evenly spaced histogram



Histogram equalization (直方图均衡化)

- Basic idea: find a map $T(r)$ such that the histogram of the modified (equalized) image is flat (uniform).



Let r represent the gray levels, which is in the interval $[0, L-1]$, with $r=0$ representing black and $r=L-1$ representing white. For any r , define a transformation:

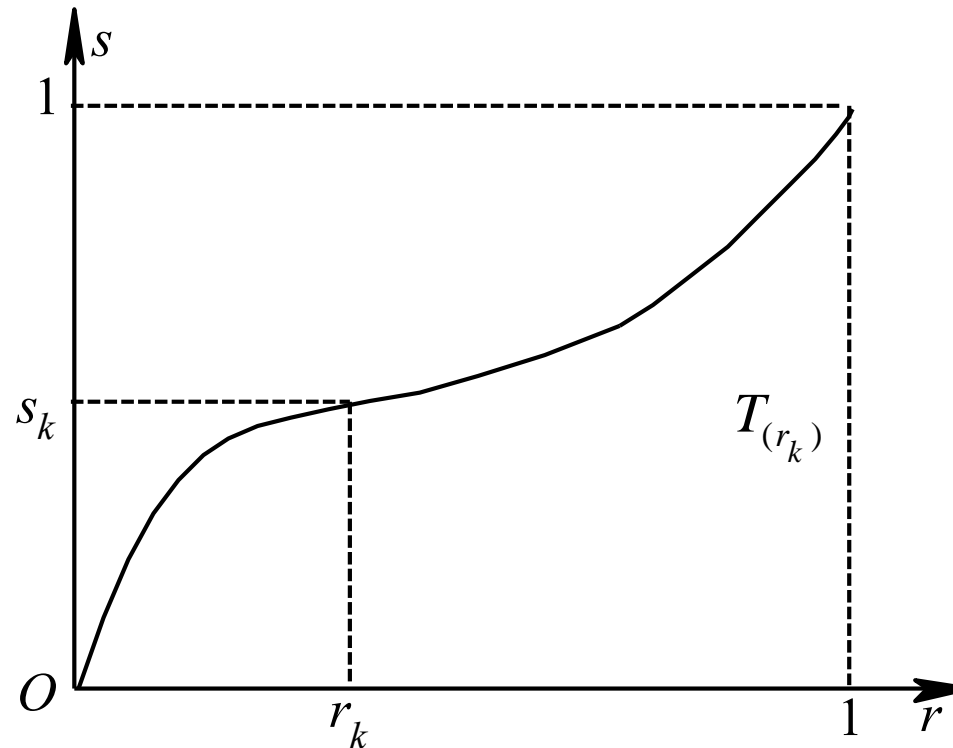
$$s = T(r)$$

That outputs a level s for every pixel value r in the input image. We assume that the transformation function $T(r)$ satisfies the following conditions:

(1) $T(r)$ is **single-value** and **monotonically increasing** in the interval $0 \leq r \leq L-1$;

(2) $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$.

- The condition (1)
preserves the increasing
order from black to white
in the output image.
- The condition (2)
guarantees that the
output gray levels will be
in the same range as the
input levels.



Let $p_r(r)$ and $p_s(s)$ denote the **probability density function** of random variables r and s , respectively. A basic result from **probability theory** is that, if $p_r(r)$ and $T(r)$ are known and $T^{-1}(s)$ satisfies condition (1). Then $p_s(s)$ can be obtained using a rather simple formula:

$$s=T(r), \quad p_s(s) = p_r(r) \left| \frac{dr}{ds} \right| \quad (3.3-3)$$

- ❑ Thus the probability density function (PDF) of s is determined by the gray level PDF of input image and by the chosen transformation function.
- ❑ So for a given input image, we can change its histogram by some transformation, it is the idea of histogram equalization

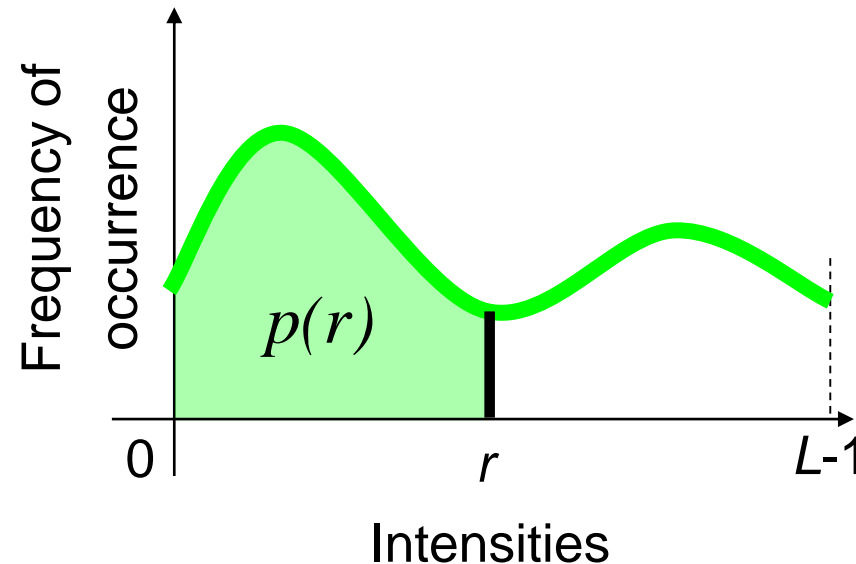
Take the following equation as the transform function:

$$s = T(r) = (L - 1) \int_0^r p_r(\omega) d\omega \quad (3.3-4)$$

This transform satisfy the two conditions:

- (1) $T(r)$ is single-value and monotonically increasing in the interval $0 \leq r \leq L-1$;
- (2) $0 \leq T(r) \leq L-1$ for $0 \leq r \leq L-1$.

- Probability density functions are always positive. The integral of a function is the area under the function. So it should be single values and monotonically increasing to satisfy (1).
- Similarly, the integral of the probability density function for variables in the range $[0, L-1]$ also in the range $[0, L-1]$



$$\frac{ds}{dr} = \frac{dT(r)}{dr} = (L - 1) \frac{d}{dr} \left[\int_0^r p_r(\omega) d\omega \right] = (L - 1)p_r(r) \quad (3.3-5)$$

By Leibniz's rule, we know that the derivative of a definite integral with respect to its upper limit is simply the integrand evaluated at the limit.

From (3.3-3) $p_s(s) = p_r(r) \left| \frac{dr}{ds} \right|$

$$p_s(s) = p_r(r) \left| \frac{dr}{ds} \right| = p_r(r) \left| \frac{1}{(L - 1)p_r(r)} \right| = \frac{1}{L - 1}, 0 \leq s \leq L - 1 \quad (3.3-6)$$

From (3.3-4), it can get a random variable s . It is important to note that $T(r)$ depends on $p_r(r)$. Eq.(3.3-6) indicates that the $p_s(s)$ always is uniform, independent of the form of $p_r(r)$.

Histogram Equalization

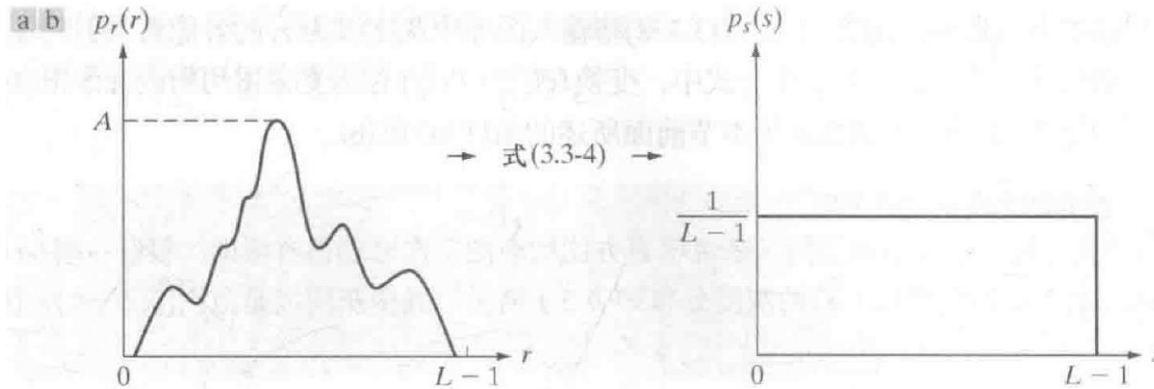


图 3.18 (a) 一个任意的 PDF; (b) 对所有灰度级 r 应用式 (3.3-4) 中的变换的结果。具有均匀 PDF 的结果灰度 s 与 r 的 PDF 的形式无关

So we can use $p_r(r)$, as the transformation function to produce an output image which has an uniform distribution in histogram, that is called histogram equalization.

$$s = T(r) = (L - 1) \int_0^r p_r(\omega) d\omega$$


Histogram Equalization

For discrete value we deal with probabilities and summations instead of probability density functions and integrals.

$$p_r(r_k) = \frac{n_k}{n} \quad 0 \leq r_k \leq 1 \quad k = 0, 1, \dots, L - 1$$

where n is the total number of pixels, n_k is the number of pixels that have gray level r_k , L is the total number of possible gray levels. $p_r(r_k)$ is the probability of occurrence of gray level r_k .

Histogram


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

Steps:

- ◆ Find probability of the input image. $p_r(r_k) = \frac{n_k}{n}$
- ◆ Find the cumulative distribution based on the probability.

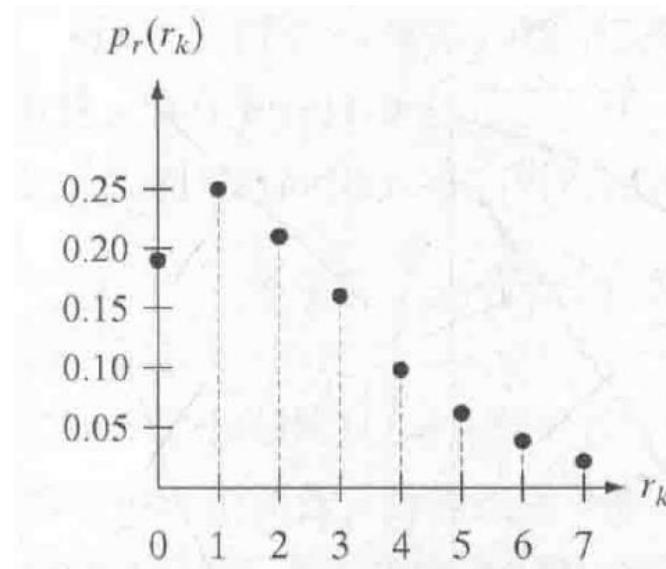
$$s_k = T(r_k) = (L - 1) \sum_{j=0}^k \frac{n_j}{n}$$

- ◆ Map the original gray-level value to the resulting value obtained in Step 3.

Histogram Equalization

An example, where the image is 64×64 pixels in size, with 8 grey levels. The distribution is as following table.

r_k	n_k	$p_r(k)=n_k/n$
$r_0=0$	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



Histogram Equalization

Processing steps:

$$s_0 = T(r_0) = 7 \sum_{j=0}^0 P_r(r_j) = 7 \times 0.19 = 1.33$$

$$s_1 = T(r_1) = 7 \sum_{j=0}^1 P_r(r_j) = 7P_r(r_0) + 7P_r(r_1) = 7 \times 0.44 = 3.08$$

$$s_2 = T(r_2) = 7 \sum_{j=0}^2 P_r(r_j) = 7P_r(r_0) + 7P_r(r_1) + 7P_r(r_2) = 7 \times 0.65 = 4.55$$

$$s_3 = T(r_3) = 7 \sum_{j=0}^3 P_r(r_j) = 7P_r(r_0) + 7P_r(r_1) + 7P_r(r_2) + 7P_r(r_3) = 7 \times 0.81 = 5.67$$

Then, we get

$$s_4 = 6.23$$

$$s_5 = 6.65$$

$$s_6 = 6.86$$

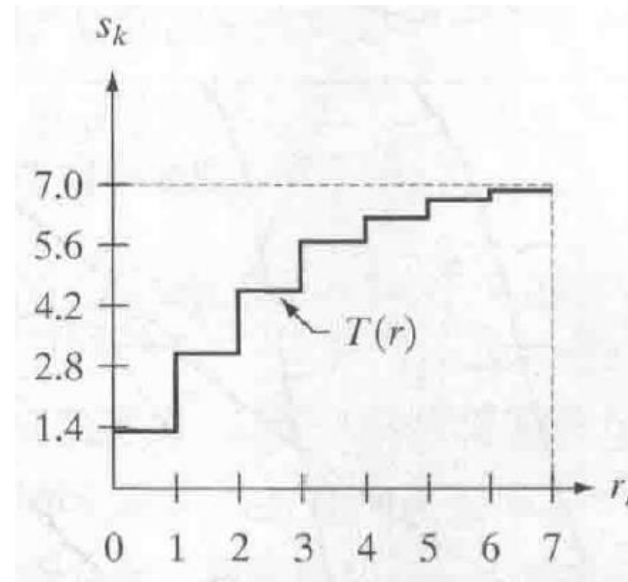
$$s_7 = 7.00$$

r_k	n_k	$p_r(k)=n_k/n$
$r_0=0$	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

Histogram Equalization

$$\begin{array}{ll} s_0 = 1.33 \rightarrow 1 & s_4 = 6.23 \rightarrow 6 \\ s_1 = 3.08 \rightarrow 3 & s_5 = 6.65 \rightarrow 7 \\ s_2 = 4.55 \rightarrow 5 & s_6 = 6.86 \rightarrow 7 \\ s_3 = 5.67 \rightarrow 6 & s_7 = 7.00 \rightarrow 7 \end{array}$$

The transform function is shown in b).
Since the grey level is 8, and we should
adjust the output values to the nearest
integer numbers.



(b)

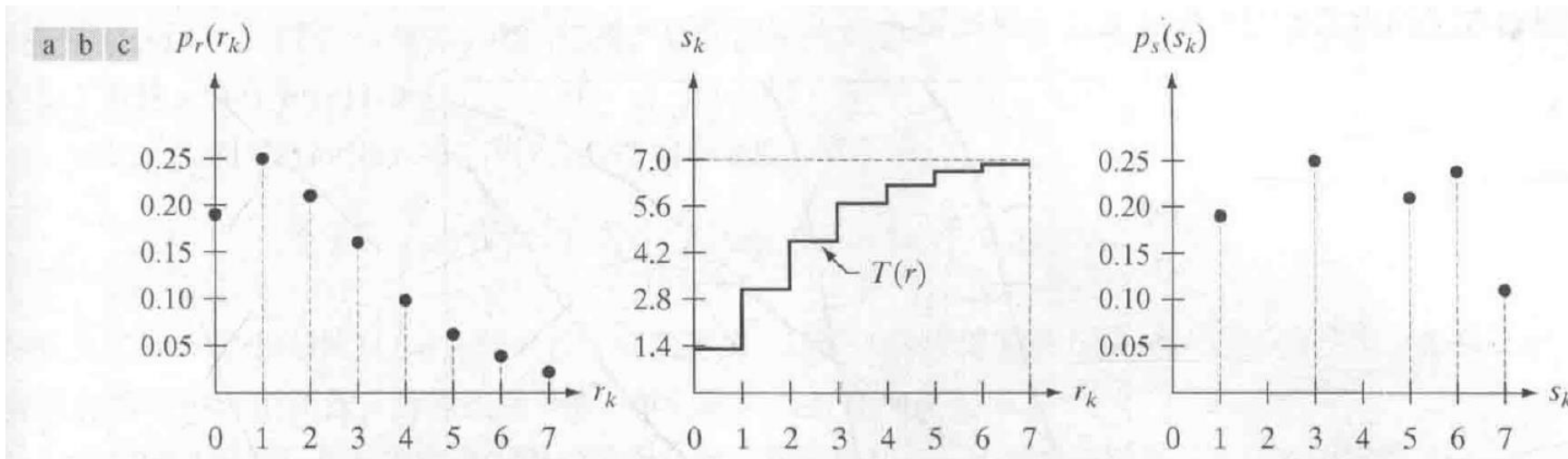
Histogram Equalization

We can see that there are only 5 effective grey levels in the output image:

$$s_0 \approx 1, \quad s_1 \approx 3, \quad s_2 \approx 5, \quad s_3 \approx 6, \quad s_4 \approx 6, \quad s_5 \approx 7, \quad s_6 \approx 7, \quad s_7 \approx 7$$

r_k	n_k	P_r	s_k	n'_k	P_s
$r_0=0$	790	0.19	$s_0=1$	790	0.19
$r_1=1$	1023	0.25	$s_1=3$	1023	0.25
$r_2=2$	850	0.21	$s_2=5$	850	0.21
$r_3=3$	656	0.16	$s_3=6$	985	0.24
$r_4=4$	329	0.08	$s_4=6$		
$r_5=5$	245	0.06	$s_5=7$	448	0.11
$r_6=6$	122	0.03	$s_6=7$		
$r_7=7$	81	0.02	$s_7=7$		

Histogram Equalization



(a) Histogram of last table

(b) Transform function

(c) After equalization

Image Example: histeq_demo.m

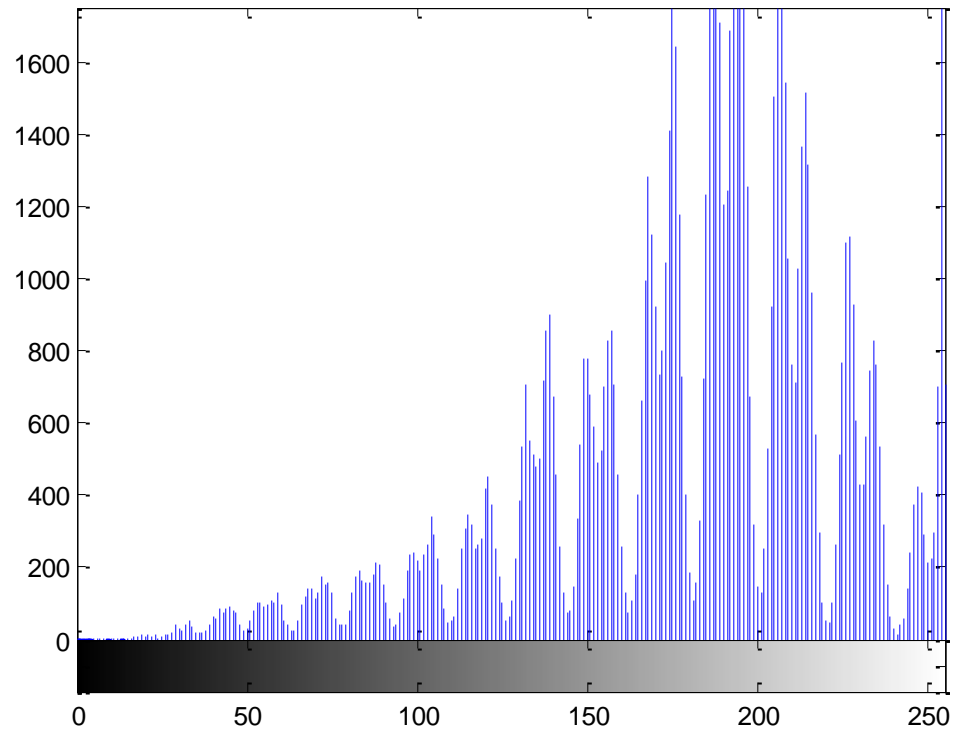


Before

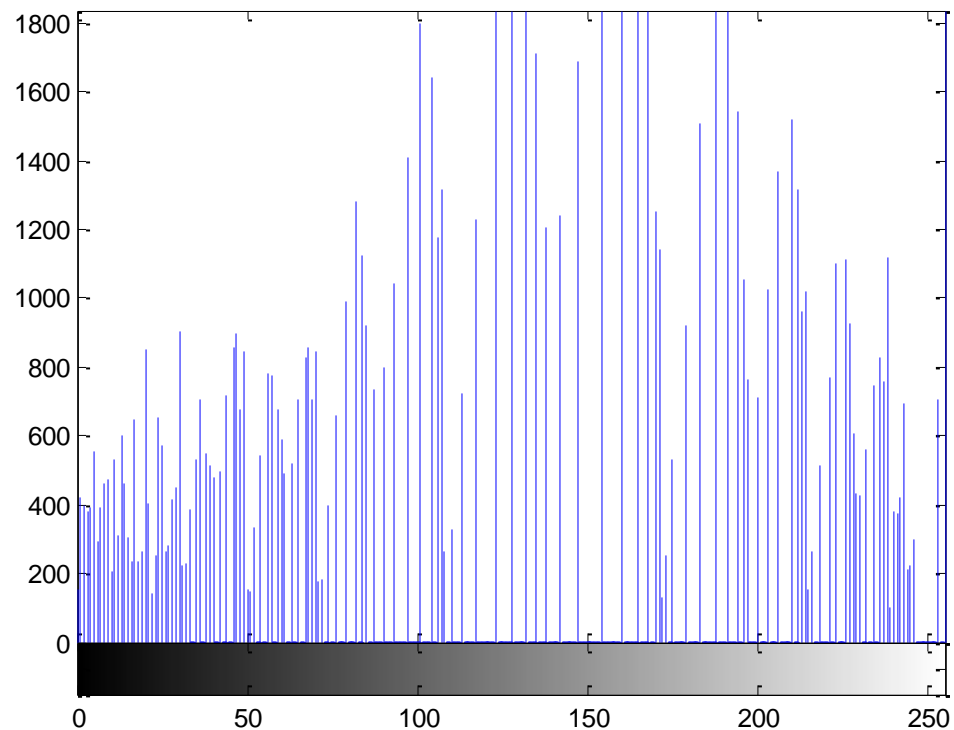


After

Histogram Equalization

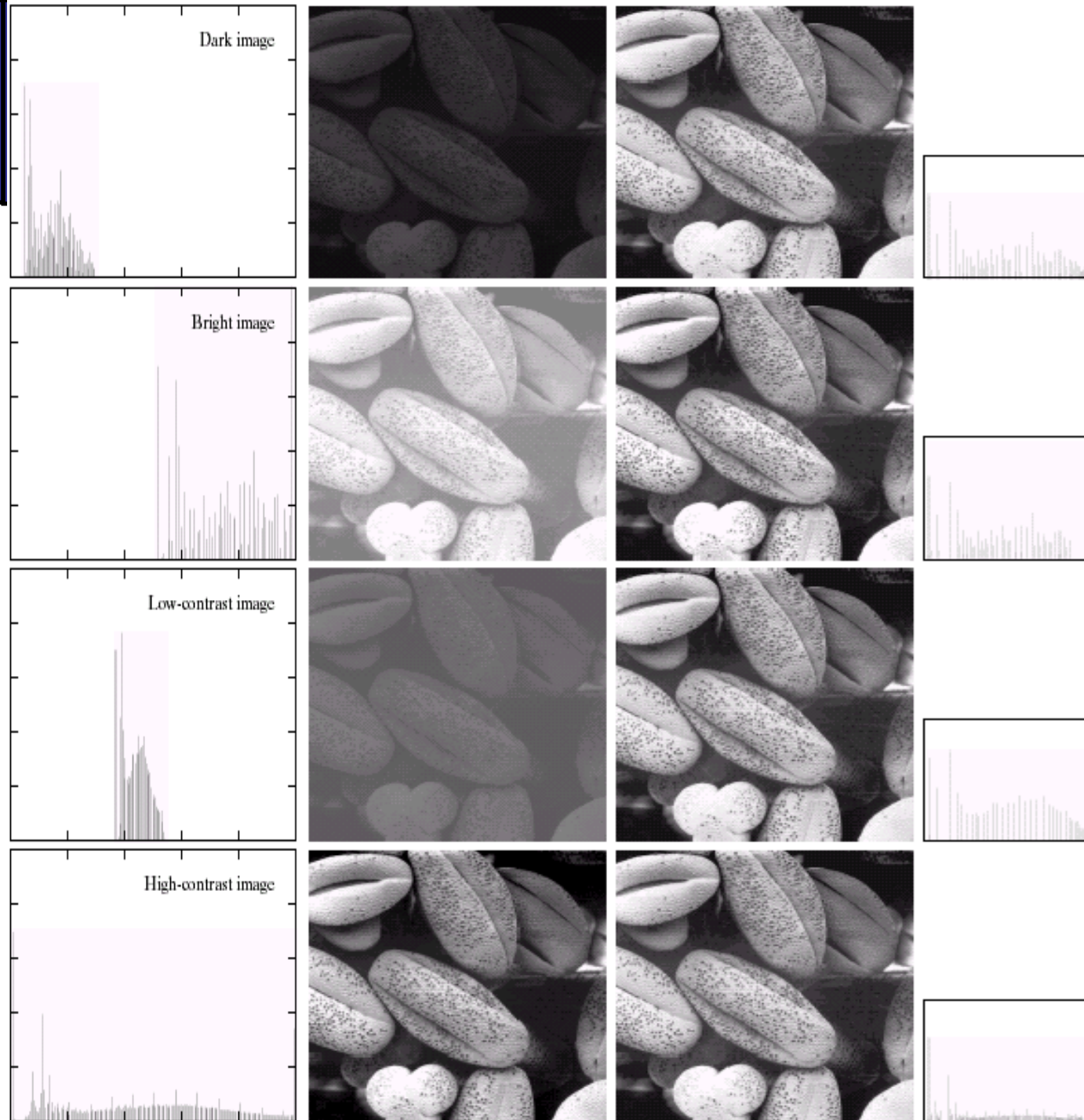


before equalization

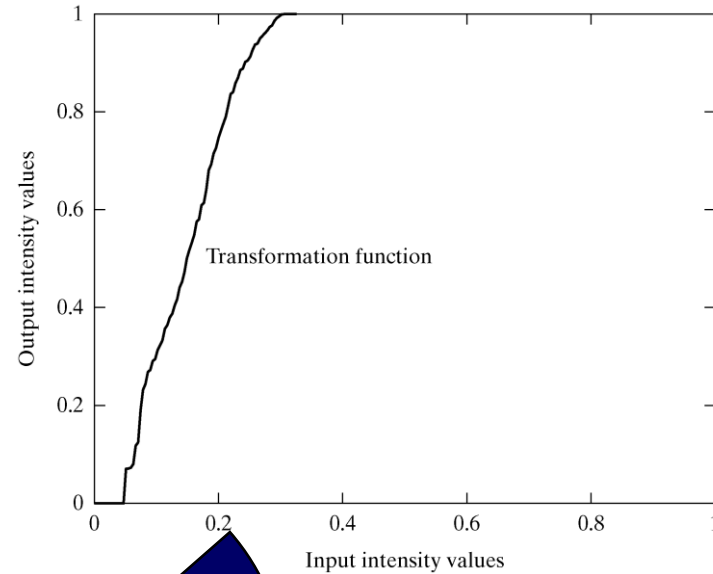
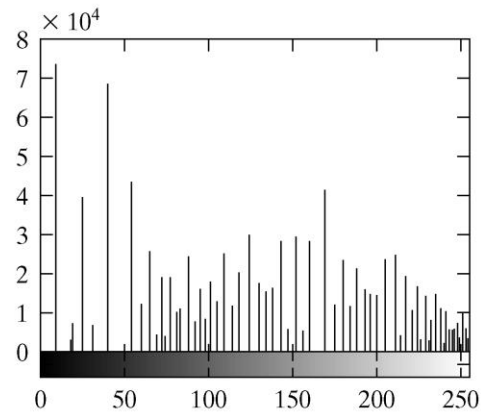
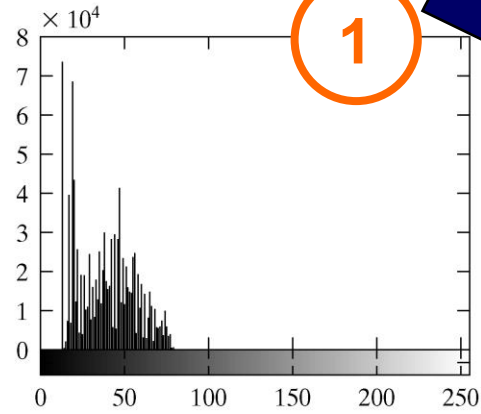
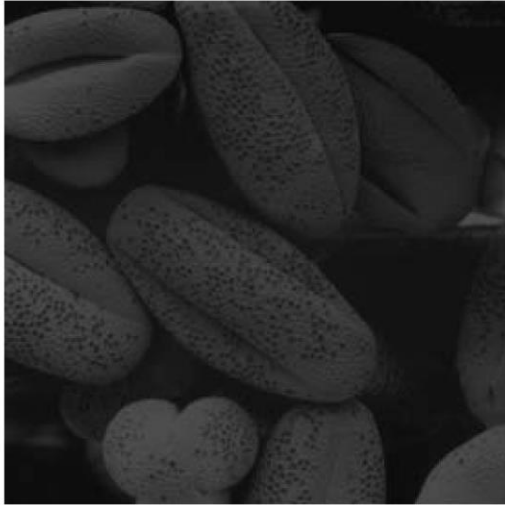


after equalization

Histogram Equalization

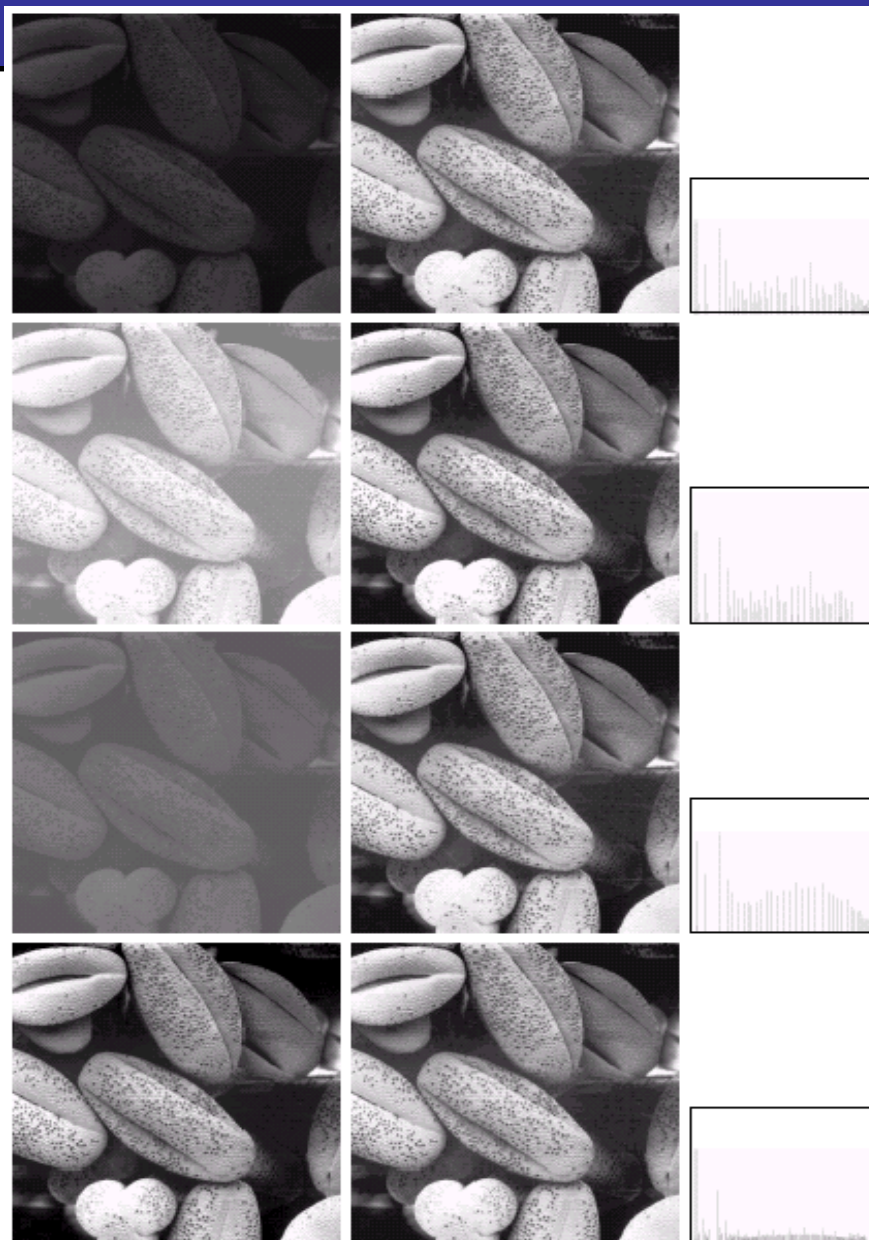
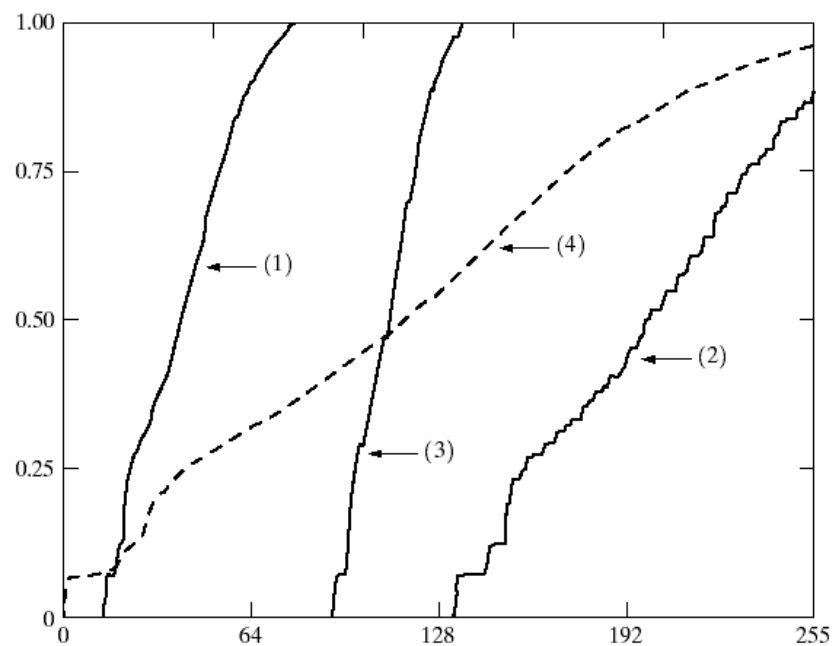


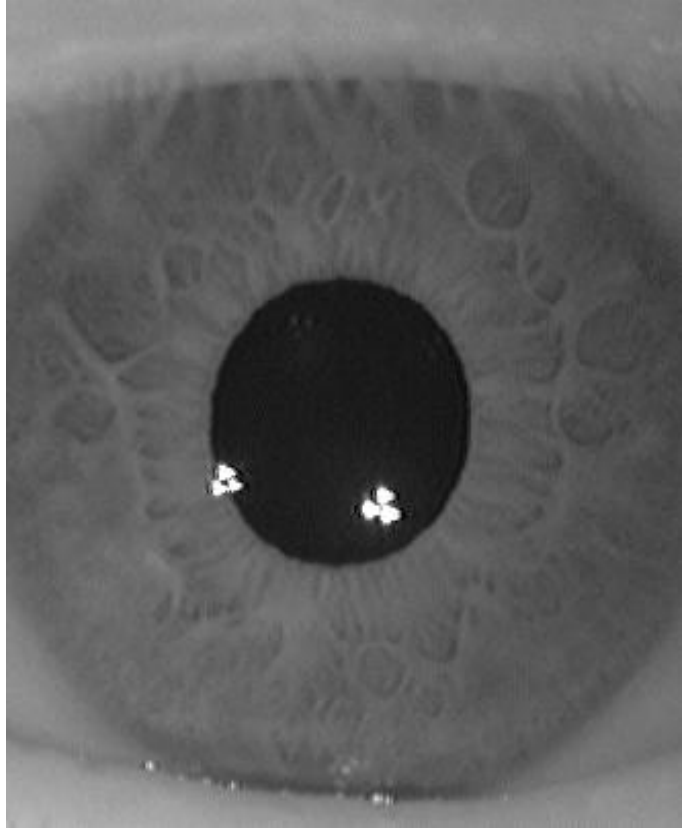
Equalisation Transformation Function



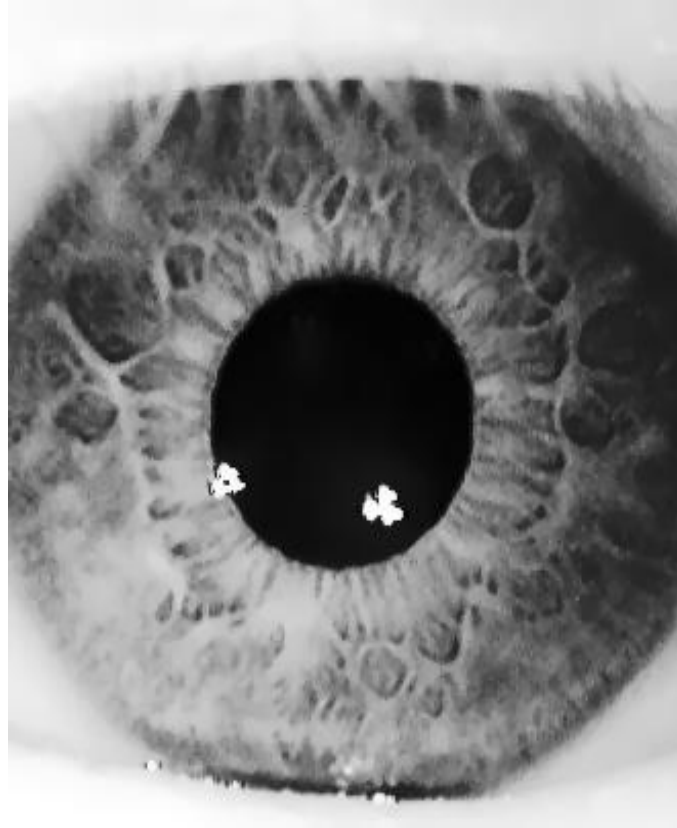
Equalization Transformation Functions

图20对应的直方图均衡化函数





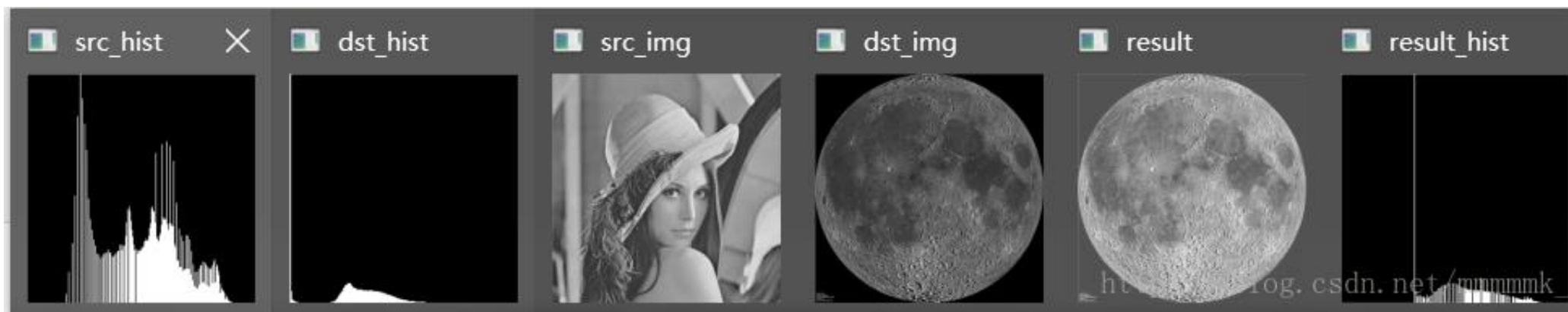
Before



After

Histogram Matching(Specification)

1. Compute the histogram $p_r(r)$ of the given image, and use it to find the histogram equalization transformation in Eq. (3.3-13). Round the resulting values, s_k , to the integer range $[0, L - 1]$.
2. Compute all values of the transformation function G using the Eq. (3.3-14) for $q = 0, 1, 2, \dots, L - 1$, where $p_z(z_i)$ are the values of the specified histogram. Round the values of G to integers in the range $[0, L - 1]$. Store the values of G in a table.
3. For every value of s_k , $k = 0, 1, 2, \dots, L - 1$, use the stored values of G from step 2 to find the corresponding value of z_q so that $G(z_q)$ is closest to s_k and store these mappings from s to z . When more than one value of z_q satisfies the given s_k (i.e., the mapping is not unique), choose the smallest value by convention.
4. Form the histogram-specified image by first histogram-equalizing the input image and then mapping every equalized pixel value, s_k , of this image to the corresponding value z_q in the histogram-specified image using the mappings found in step 3. As in the continuous case, the intermediate step of equalizing the input image is conceptual. It can be skipped by combining the two transformation functions, T and G^{-1} , as Example 3.8 shows.



Histogram of reference image

Histogram of input image

Reference image

Input image

Image after matching

Histogram of result image

We have looked at:

- ◆ Different kinds of point processing image enhancement
- ◆ Histogram processing