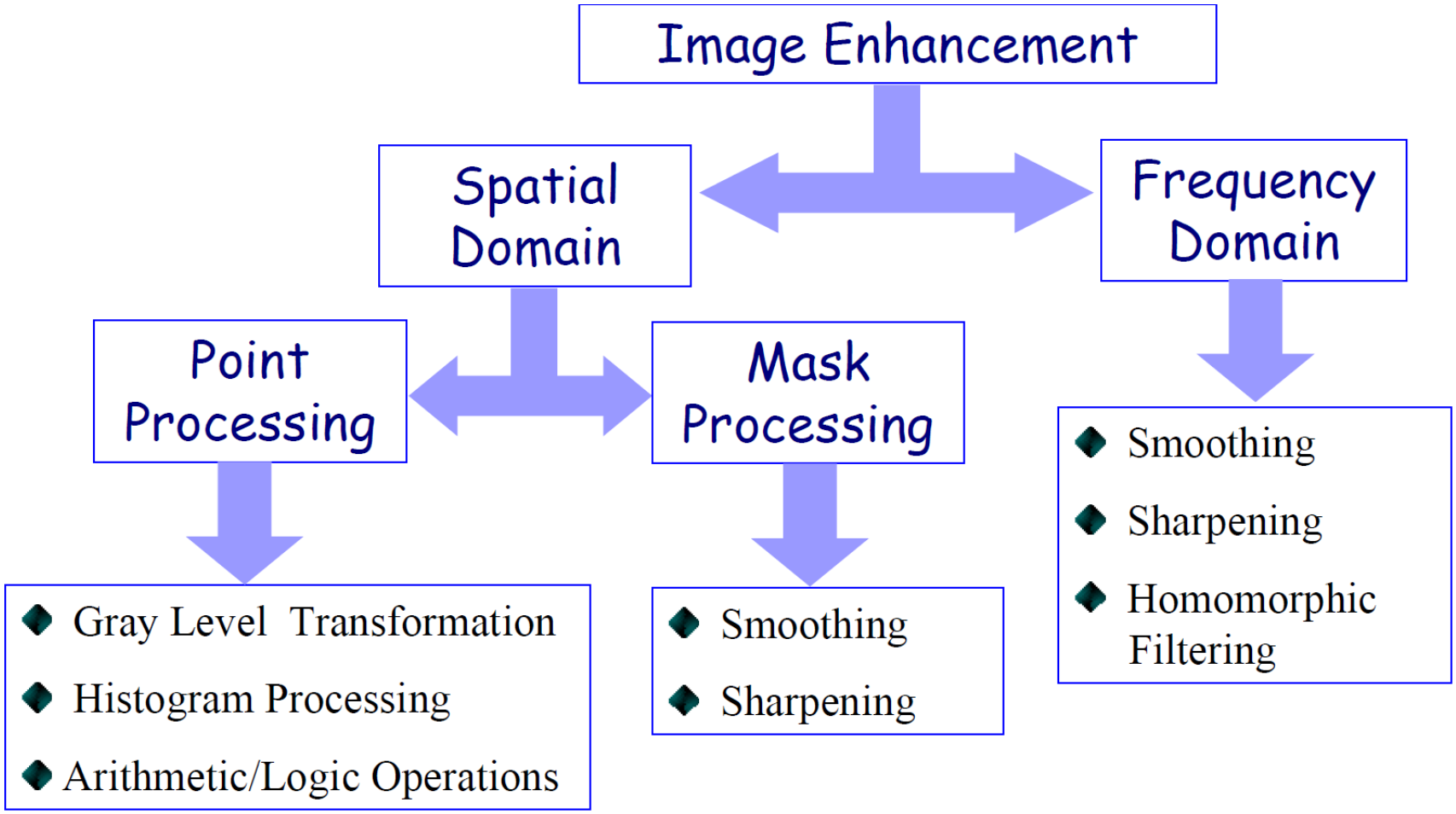
Image Enhancement in Spatial Domain (1)

Lecturer: Asst. Prof. Dr. Thittaporn Ganokratanaa

Image Enhancement Framework



What is image enhancement?

- To process an image so that the result will be "more suitable" than the original image for a specific application.
- The suitableness is up to each application.
- A method that is quite useful for enhancing an image may not necessarily be the best approach for enhancing another image.

What is a better image?

- For human visual
  - The visual evaluation of image quality is a highly subjective process.
  - It is hard to standardize the definition of a good image.
- For machine perception
  - The evaluation task is easier.
  - A good image is one which gives the best machine recognition results.
- A certain amount of trial and error usually is required before a particular image enhancement approach is selected.

Spatial Domain

Point Processing

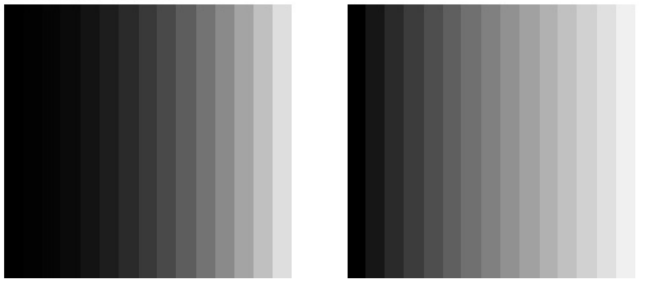
Gray Level Transformations

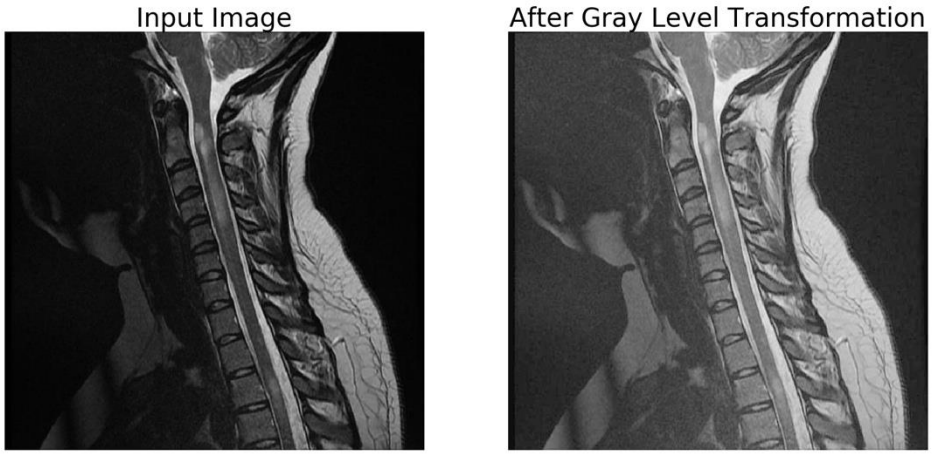
❖ Gray level transformation is used to generate new gray level in image

$$s(i, j) = T(r(i, j))$$

Where $s(i, j)$ is output image with new gray level

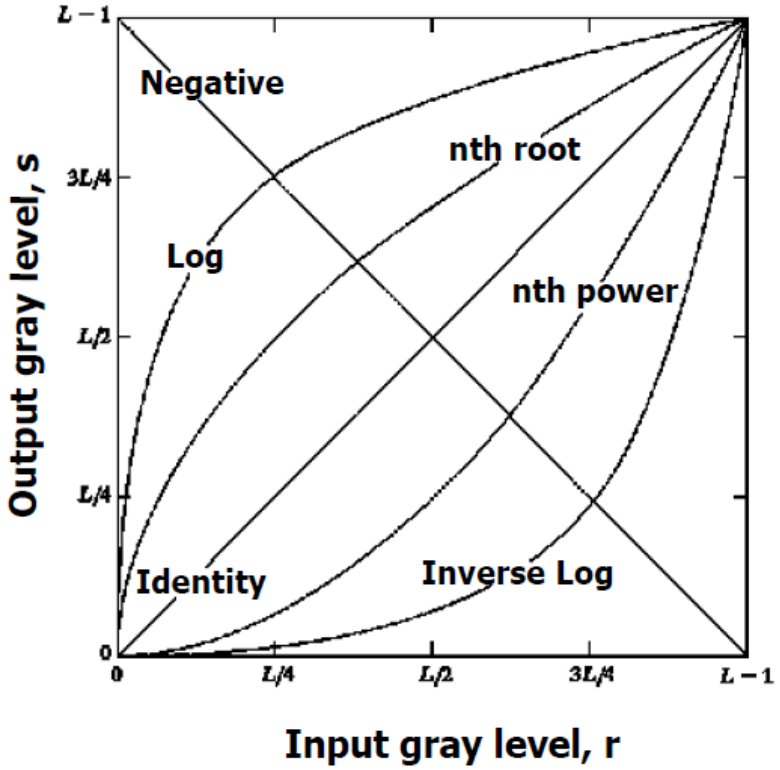
$T(r(i, j))$ is function to transform current gray level of input image $r(i, j)$





Gray Level Transformations

- Linear function
  - Negative: $s(i,j) = L - 1 - r(i,j)$
  - Identity : $s(i,j) = r(i,j)$
- Logarithm function
  - Log : $s(i,j) = c \times \log(1 + r(i,j))$
  - Inverse-Log
- Power-law function
  - γ Power: $s(i,j) = c \times r(i,j)^{\gamma}$
  - $\frac{1}{\gamma}$ Root $s(i,j) = c \times r(i,j)^{1/\gamma}$

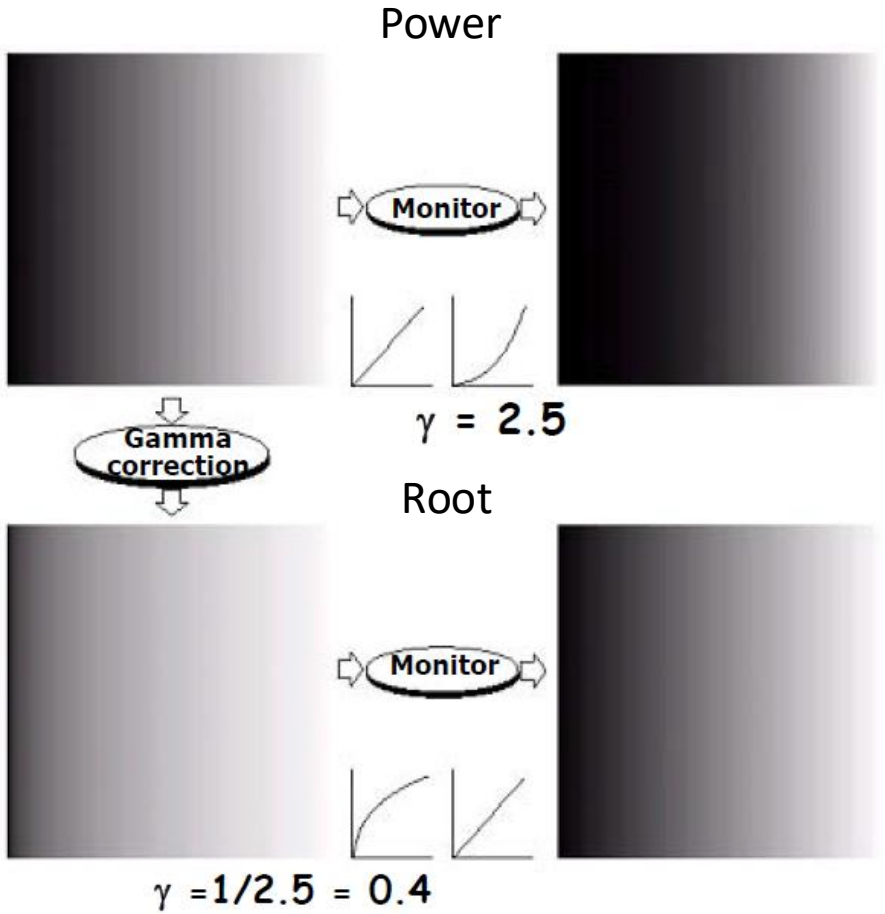


Gamma Correction

- Cathode ray tube (CRT), display device, has an intensity-to-voltage response that is a power function, with γ varying from 1.8 to 2.5
  - The picture will become darker.

- Gamma correction is done by pre-processing the image before inputting it to the monitor with

$$s(i, j) = c \times r(i, j)^{1/\gamma}$$



Power gives brightness to darkness
Root gives darkness to brightness

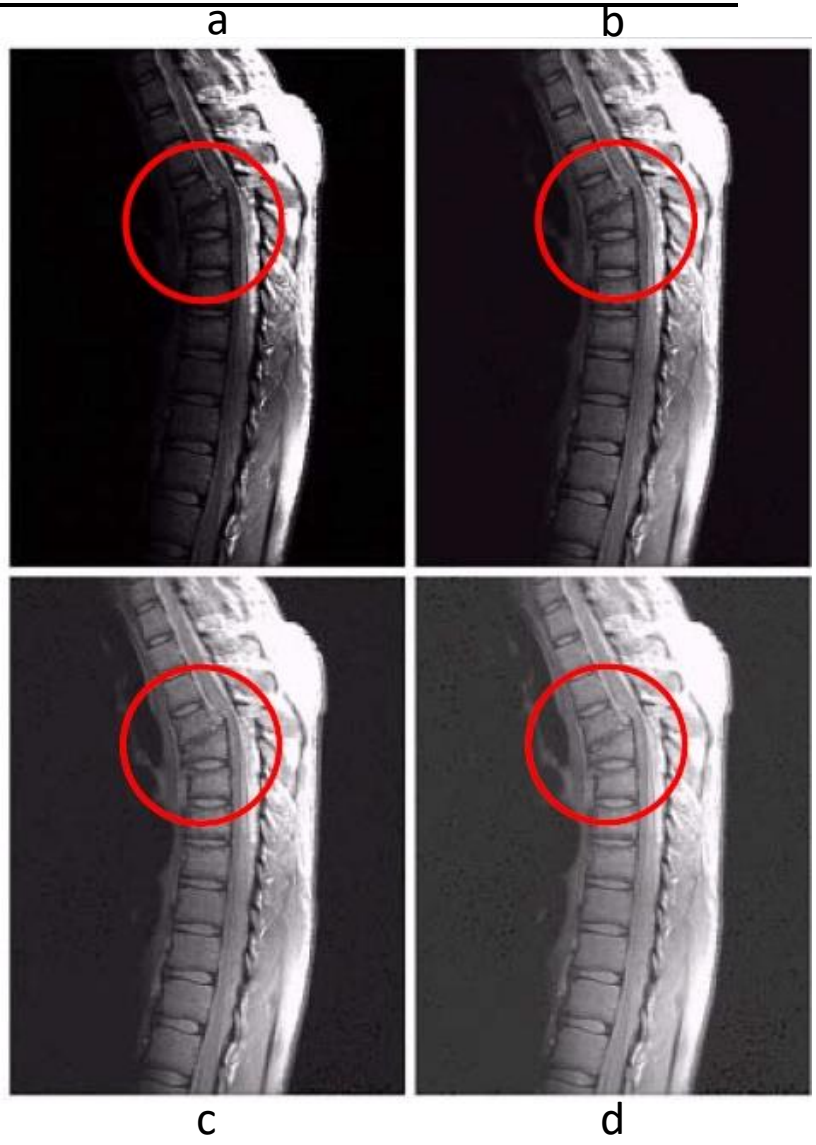
Another Example: MRI

a) A magnetic resonance image of an upper thoracic human spine with a fracture dislocation and spinal cord impingement
   - The picture is predominately dark
   - An expansion of gray levels are desirable -> **needs γ < 1**

b) Result after power-law transformation with γ= 0.6,c=1

c) Transformation with γ= 0.4 (best result)

d) Transformation with γ = 0.3 (under acceptable level)



Another Example

(a) Image has a washed-out appearance; it needs a compression of gray levels -> **needs $\gamma > 1$**

(b) Result after power-law transformation with $\gamma = 3.0$ (suitable)

(c) Transformation with $\gamma = 4.0$ (suitable)

(d) transformation with $\gamma = 5.0$ (high contrast, the image has areas that are too dark, some detail is lost)

Effect of Decreasing Gamma

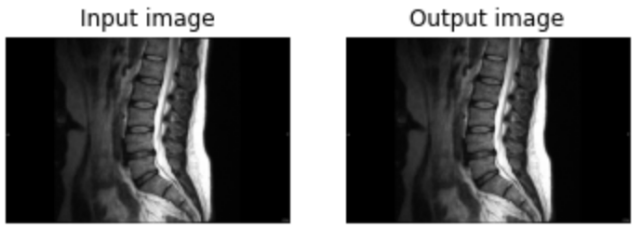
- When the γ is reduced too much, the image begins to reduce contrast to the point where the image started to have very slight "wash-out" look, especially in the background

Python code

```python
import numpy as np
import matplotlib.pyplot as plt
import cv2

def GLT(image, transform, coeff = 1.0, gamma = 1.0):
    #build  lookup table to map the pixel value [0,255] to their gray level transformation
    if transform == 'negative':
        table = np.array([256-1-i for i in np.arange(0,256)]).astype("uint8")

    elif transform == 'identity':
        table = np.array([i for i in np.arange(0,256)]).astype("uint8")

    elif transform == 'log':
        table = np.array([10*coeff*(np.log10(1+i)) for i in np.arange(0,256)]).astype("uint8")

    elif transform == 'invlog':
        table = np.array([10*coeff/(np.log10(1+i)+1) for i in np.arange(0,256)]).astype("uint8")

    elif transform == 'root':
        invGamma = 1.0/gamma
        table = np.array([coeff*((i/255.0)**invGamma)*255 for i in np.arange(0,256)]).astype("uint8")

    elif transform == 'power':
        table = np.array([coeff*((i/255.0)**gamma)*255 for i in np.arange(0,256)]).astype("uint8")

    return cv2.LUT(image, table)
```

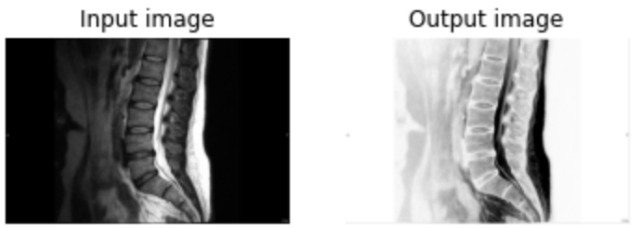
Python code (identity)

```
img = cv2.imread('lumbar.png')
transform = 'identity'
coeff = 1 # For negative c = -1; for identity c = 1; for other, c is integer
gamma = 2.5 # Gamma is used for root and power

out_img = GLT(img,transform,coeff = coeff, gamma = gamma)

plt.subplot(121)
plt.imshow(img,'gray')
plt.title('Input image', fontsize=12)
plt.axis("off")

plt.subplot(122)
plt.imshow(out_img,'gray')
plt.title('Output image', fontsize=12)
plt.axis("off")
plt.show()
```



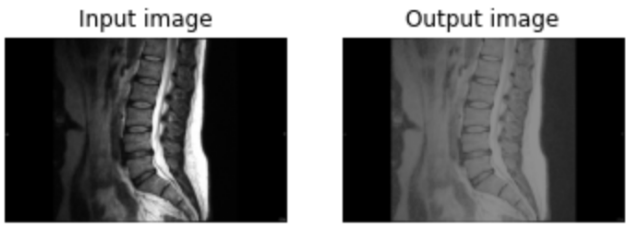
Python code (negative)

```
img = cv2.imread('lumbar.png')
transform = 'negative'
coeff = -1 # For negative c = -1; for identity c = 1; for other, c is integer
gamma = 2.5 # Gamma is used for root and power

out_img = GLT(img,transform,coeff = coeff, gamma = gamma)

plt.subplot(121)
plt.imshow(img,'gray')
plt.title('Input image', fontsize=12)
plt.axis("off")

plt.subplot(122)
plt.imshow(out_img,'gray')
plt.title('Output image', fontsize=12)
plt.axis("off")
plt.show()
```



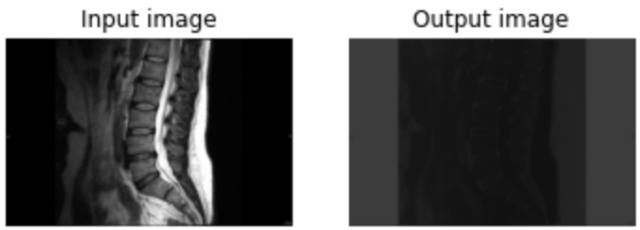
Python code (log)

```
img = cv2.imread('lumbar.png')
transform = 'log'
coeff = 6 # For negative c = -1; for identity c = 1; for other, c is integer
gamma = 2.5 # Gamma is used for root and power

out_img = GLT(img,transform,coeff = coeff, gamma = gamma)

plt.subplot(121)
plt.imshow(img,'gray')
plt.title('Input image', fontsize=12)
plt.axis("off")

plt.subplot(122)
plt.imshow(out_img,'gray')
plt.title('Output image', fontsize=12)
plt.axis("off")
plt.show()
```



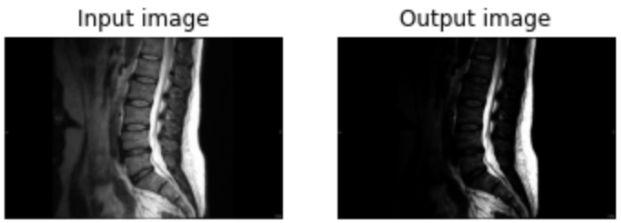
Python code (inverse log)

```
img = cv2.imread('lumbar.png')
transform = 'invlog'
coeff = 6 # For negative c = -1; for identity c = 1; for other, c is integer
gamma = 2.5 # Gamma is used for root and power

out_img = GLT(img,transform,coeff = coeff, gamma = gamma)

plt.subplot(121)
plt.imshow(img,'gray')
plt.title('Input image', fontsize=12)
plt.axis("off")

plt.subplot(122)
plt.imshow(out_img,'gray')
plt.title('Output image', fontsize=12)
plt.axis("off")
plt.show()
```



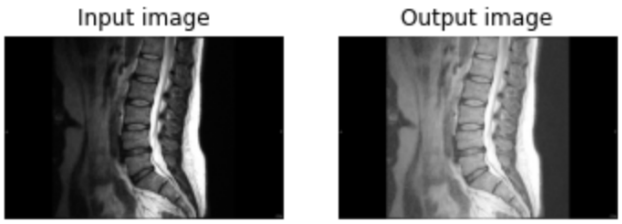
Python code (power)

```
img = cv2.imread('lumbar.png')
transform = 'power'
coeff = 1 # For negative c = -1; for identity c = 1; for other, c is integer
gamma = 2.5 # Gamma is used for root and power

out_img = GLT(img,transform,coeff = coeff, gamma = gamma)

plt.subplot(121)
plt.imshow(img,'gray')
plt.title('Input image', fontsize=12)
plt.axis("off")

plt.subplot(122)
plt.imshow(out_img,'gray')
plt.title('Output image', fontsize=12)
plt.axis("off")
plt.show()
```



Python code (root)

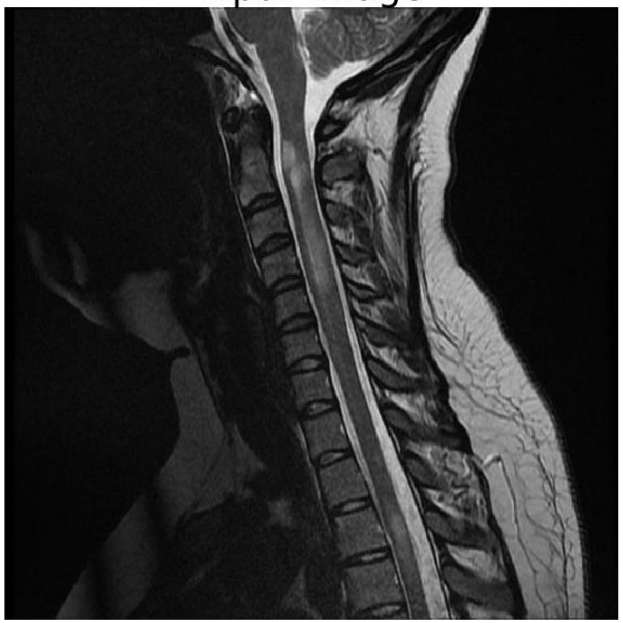
```
img = cv2.imread('lumbar.png')
transform = 'root'
coeff = 1 # For negative c = -1; for identity c = 1; for other, c is integer
gamma = 2.5 # Gamma is used for root and power

out_img = GLT(img,transform,coeff = coeff, gamma = gamma)

plt.subplot(121)
plt.imshow(img,'gray')
plt.title('Input image', fontsize=12)
plt.axis("off")

plt.subplot(122)
plt.imshow(out_img,'gray')
plt.title('Output image', fontsize=12)
plt.axis("off")
plt.show()
```



Practice

- Choose gray level transformation function to get result as picture below

Input Image

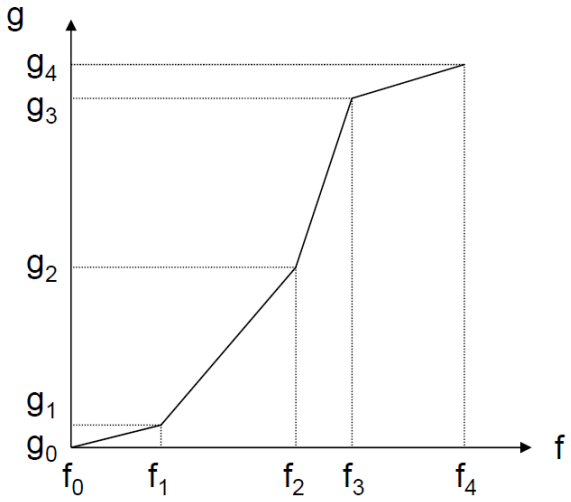


After Gray Level Transformation

Piecewise-Linear Transformations

- Linear stretching:
  - Enhance the dynamic range by linearly stretching the original gray levels to the range of

$$g(f) = af + b$$



- For piecewise-linear transformation

  K segments

  – Starting position of input $\{f_k, k = 0, \ldots K-1\}$

  – Starting position of output $\{g_k, k = 0, \ldots, K-1\}$

  – Transform function

- Advantage:
  - The form of piecewise functions can be arbitrarily complex
- Disadvantage:
  - Their specification requires considerably more user input
- Examples are:
  - Piecewise linear contrast stretching
  - Gray-level slicing
  - Bit-plane slicing

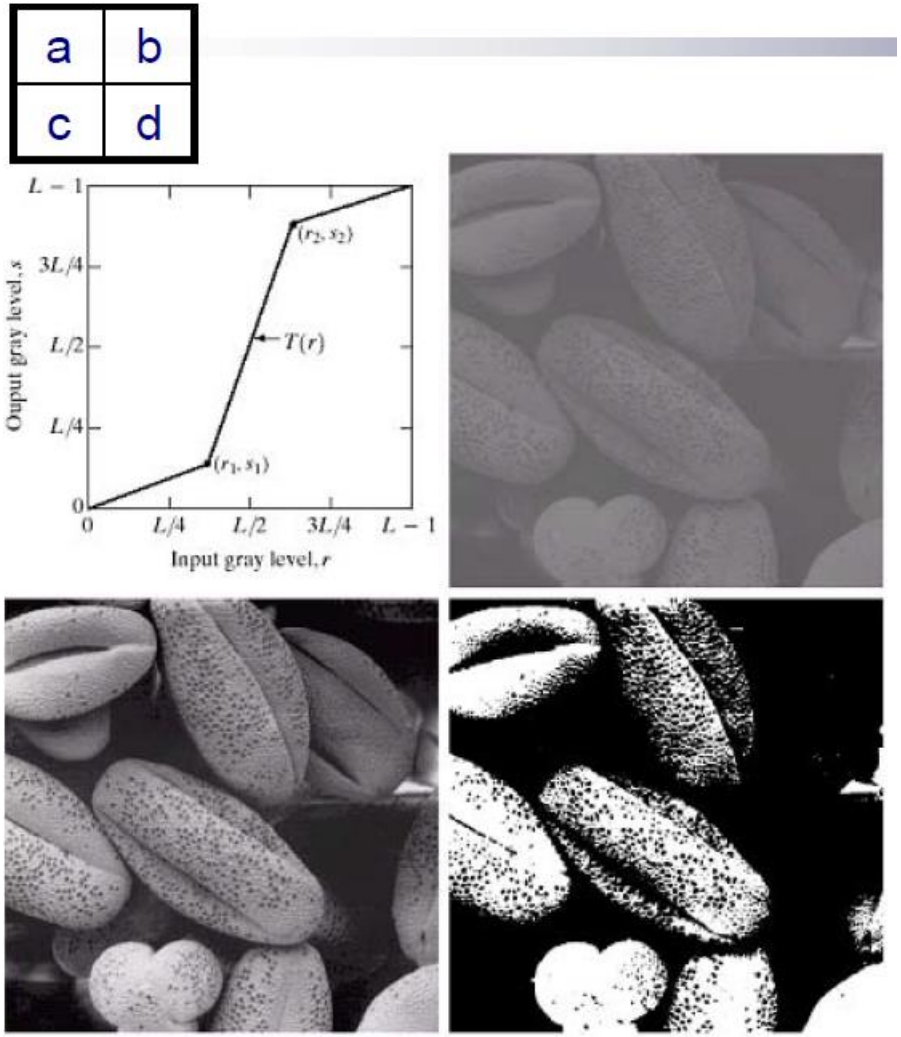
Piecewise Linear Contrast Stretching

(a) Increase the dynamic range of the gray levels in the image

(b) a low-contrast image : result from poor illumination, lack of dynamic range in the imaging sensor, or even wrong setting of a lens aperture of image acquisition

(c) result of contrast stretching: ($r1$,$s1$) = ($r$min,0) and ($r2$,$s2$) = ($r$max,L-1)

(d) result of thresholding



Gray Level Slicing

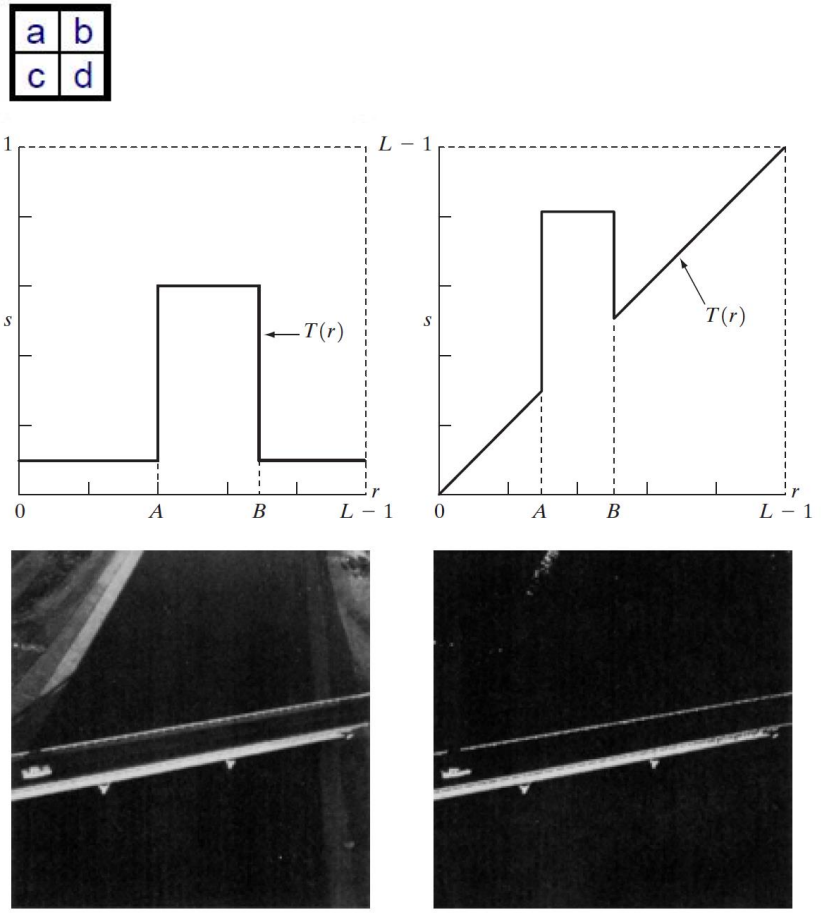
- Highlighting a specific range of gray levels in an image
  - Display a high value of all gray levels in the range of interest and a low value for all other gray levels

(a) transformation highlights range [*A*,*B*] of gray level and reduces all others to a constant level

(b) transformation highlights range [*A*,*B*] but preserves all other levels

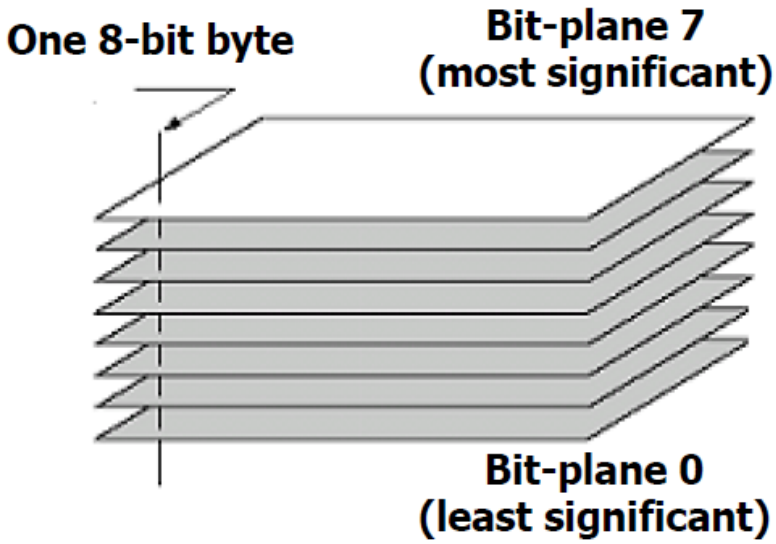
(c) original image

(d) image after applying the transformation in (a)



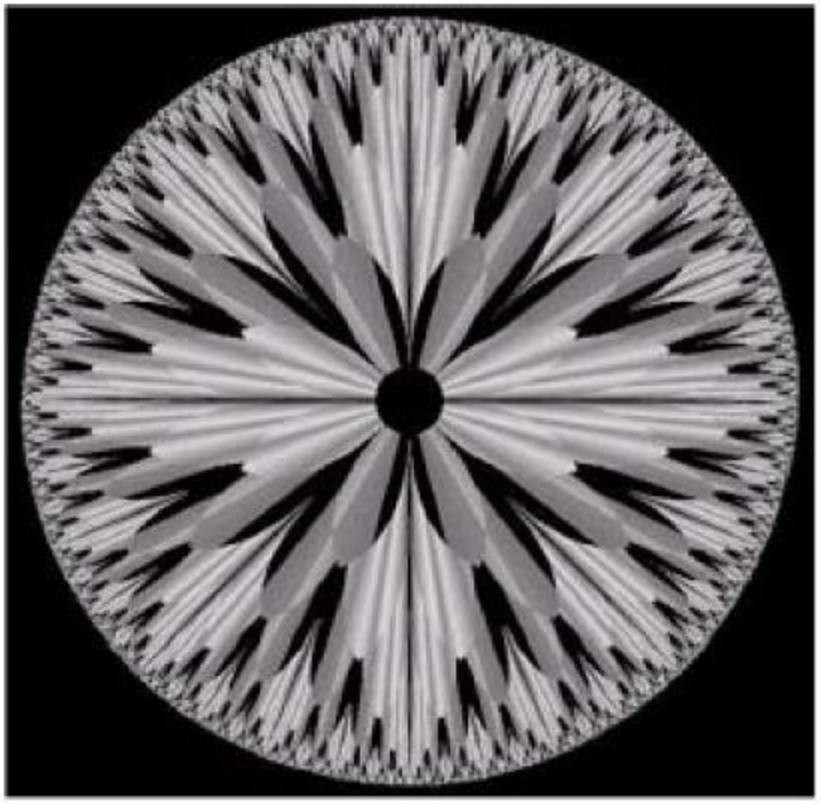
Bit-Place Slicing

- Highlighting the contribution made to total image appearance by specific bits
- Suppose each pixel is represented by 8 bits
- Higher-order bits contain the majority of the visually significant data
- Useful for analysing the relative importance played by each bit of the image
- Useful for image compression



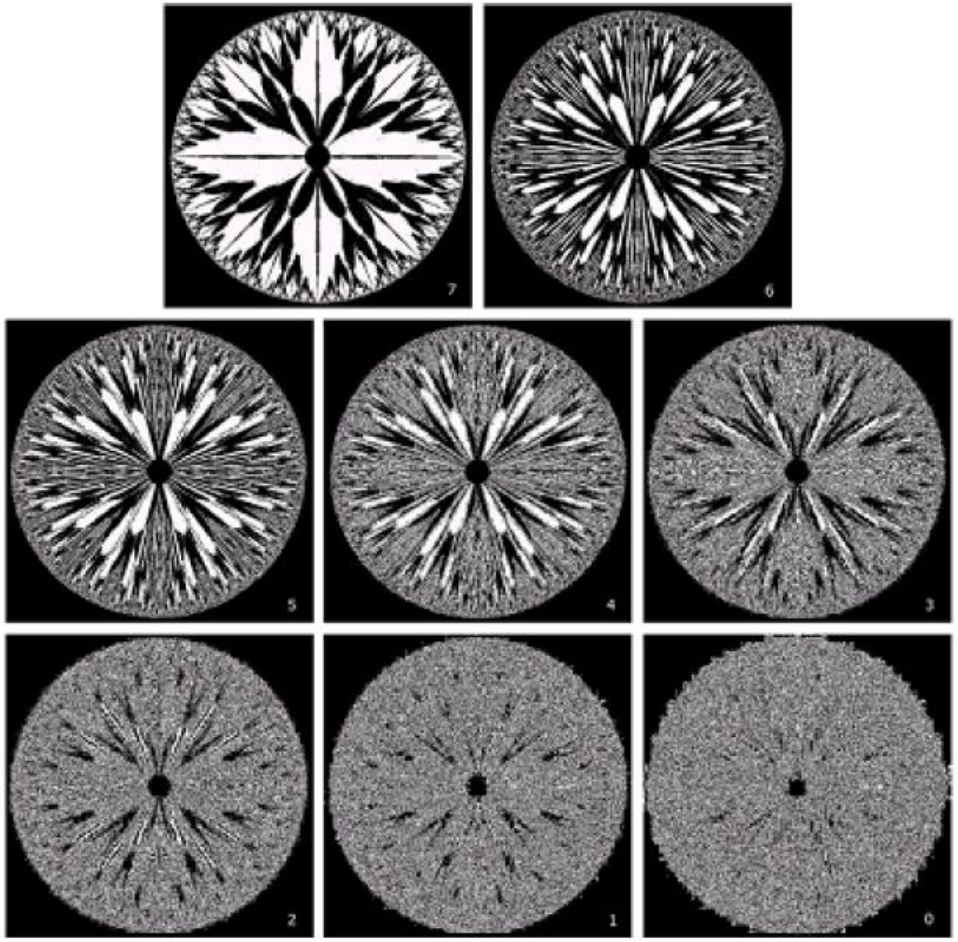
Example of Bit-Plane Slicing

- The (binary) image for bit-plane 7 can be obtained by processing the input image with a thresholding (gray-level transformation).
  - Map all levels between 0 and 127 to 0
  - Map all levels between 129 and 255 to 255



An 8-bit fractal image

All Eight Bit-Planes



Bit-plane 7	Bit-plane 6	
Bit-plane 5	Bit-plane 4	Bit-plane 3
Bit-plane 2	Bit-plane 1	Bit-plane 0

Histogram Equalization

❖ Histogram: a discrete function that maps the k^{th} gray level r_k to the number of pixels with gray level r_k in the image n_k.

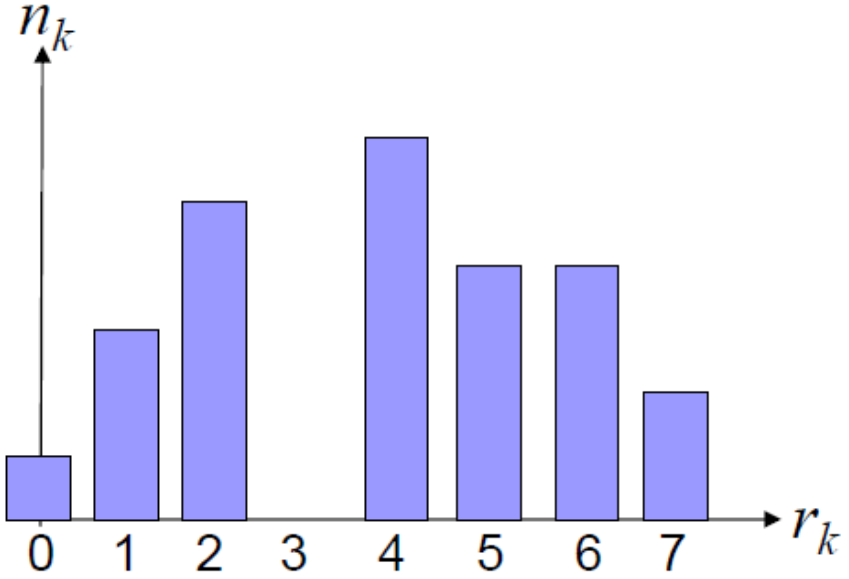
$$h(r_k) = n_k$$

where

- $k = [0, L\text{-}1]$;
- r_k = the k^{th} gray level
- n_k = the number of pixels in the image having gray level r_k
- $h(r_k)$ = histogram of a digital image with gray levels r_k

Example of Histogram

1	5	7	4	1
2	6	7	0	2
6	6	4	1	2
6	4	4	4	2
5	5	5	2	4



- Gray level in this example, the lowest is 0, highest is 7 -> it has L = 8 levels
- Counting $\boldsymbol{n_k}$, e.g., the number of gray level 6th ($\boldsymbol{r_6}$ = 6) is four then, $\boldsymbol{n_6}$ = 4

Normalized Histogram

- Normalized histogram is the division each histogram at gray level $\boldsymbol{r_k}$ by the total number of pixels in the image,

$$NormalizedHistogram = \frac{n_k}{n} = \frac{n_k}{\sum_{\forall k} n_k} = \frac{n_k}{n_0 + n_1 + \ldots + n_k}$$

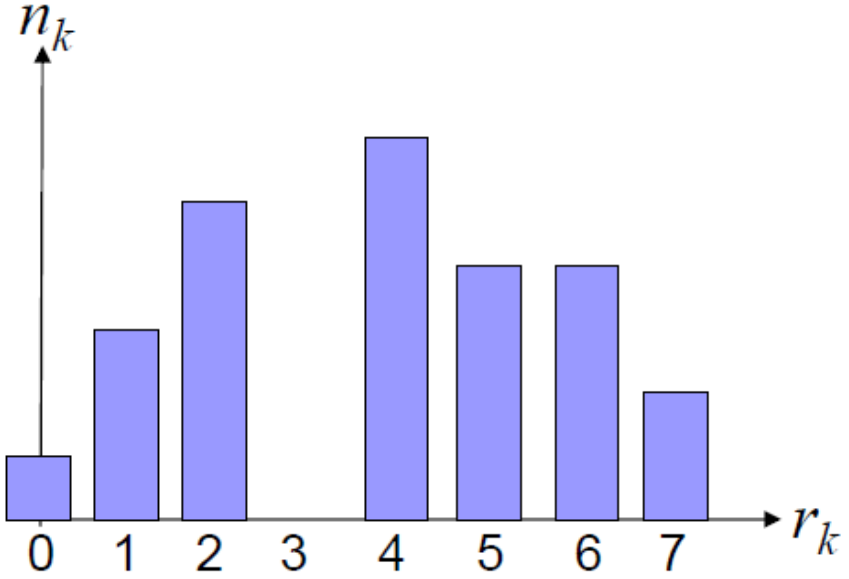
- Normalized histogram tells us how much of percentage of each histogram at gray level $\boldsymbol{r_k}$ has in whole image.
- We called as probability density function (pdf) $p(r_k)$.

$$p(r_k) = \frac{n_k}{n}$$

- $\sum_{\forall k} p(r_k) = \frac{n_0 + n_1 + \ldots + n_k}{n} = \frac{n}{n} = 1$

Example of Histogram

1	5	7	4	1
2	6	7	0	2
6	6	4	1	2
6	4	4	4	2
5	5	5	2	4



- In this example, we have 5x5 images
- Then total pixel, $\boldsymbol{n}$ = 5x5 = 25
- Define the pdf or Normalize Histogram of gray

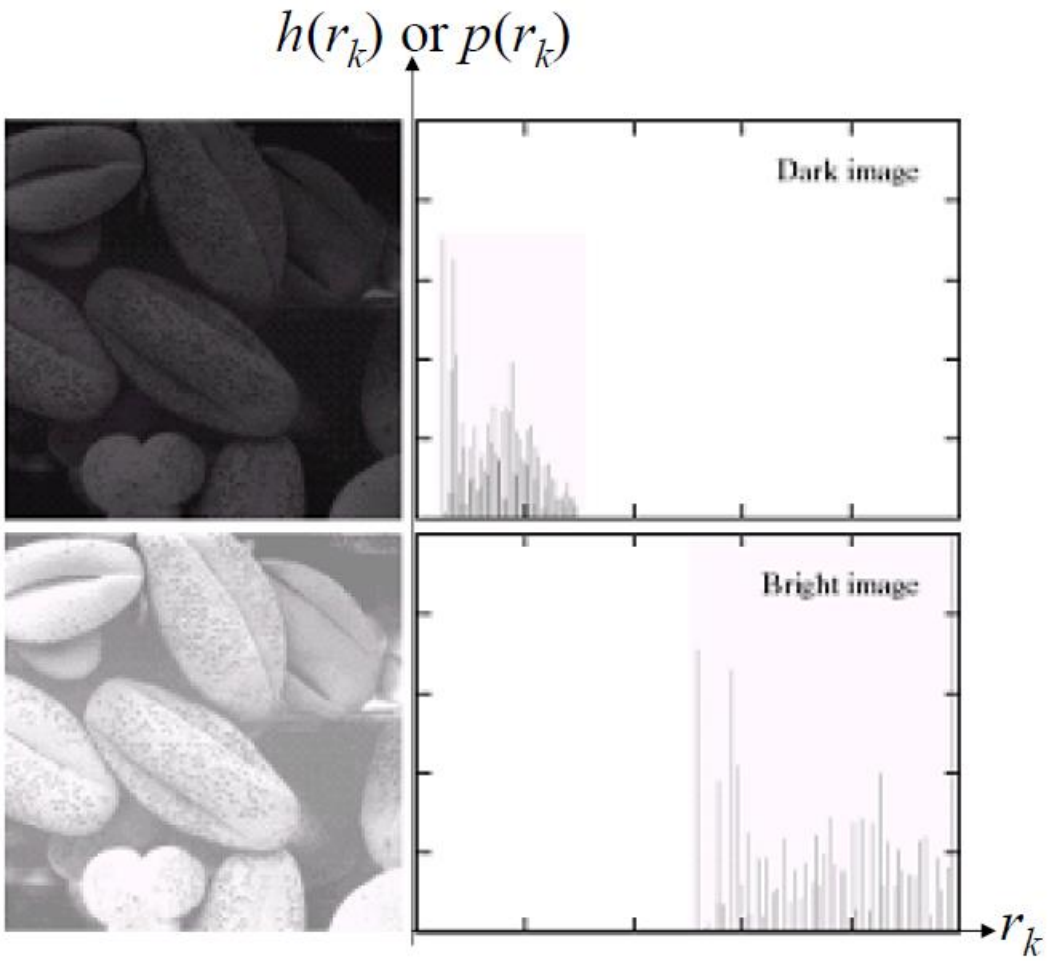
level 6th -> $\boldsymbol{n_6}$ = 4 -> $p(r_6) = \frac{n_6}{n} = \frac{4}{25} = 0.16$

Histogram Processing

- Basic for numerous spatial domain processing techniques
- Used effectively for image enhancement
- Information inherent in histograms is also useful in image compression and segmentation

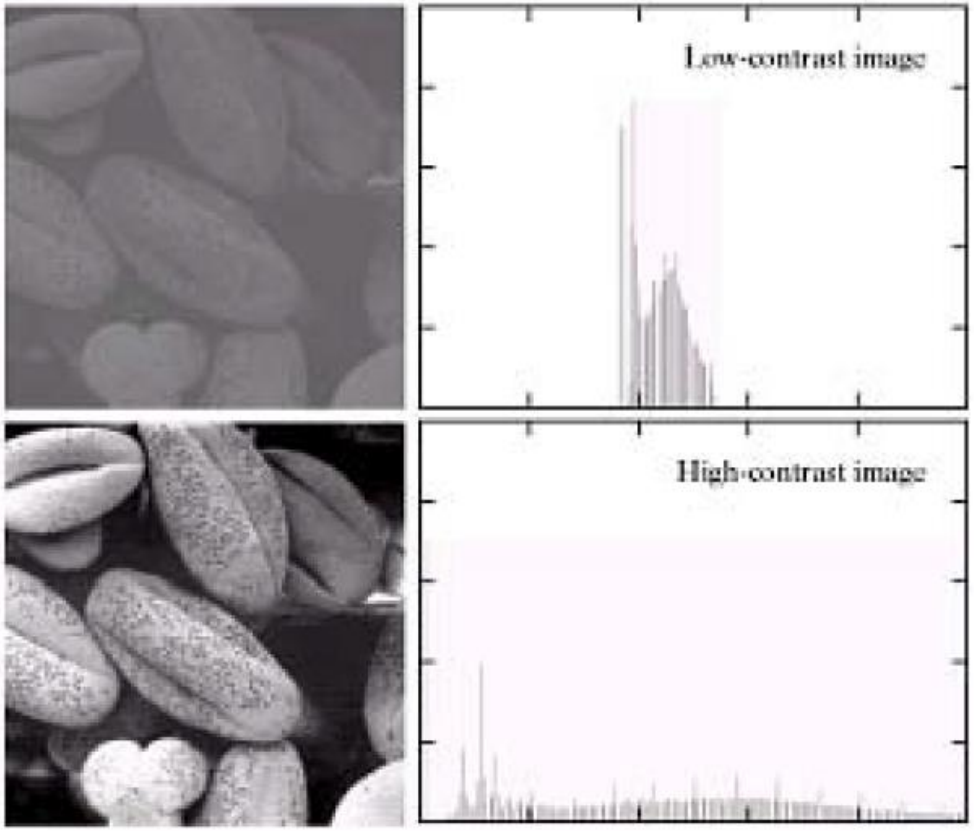
Examples

- Dark image
  - Components of the histogram are concentrated on the low side of the gray scale.
- Bright image
  - Components of histogram are concentrated on the high side of the gray scale.



Examples

- Low-contrast image
  - histogram is narrow and centered toward the middle of the gray scale
- High-contrast image
  - histogram covers broad range of the gray scale and the distribution of pixels is not too far from uniform, with very few vertical lines being much higher than the others



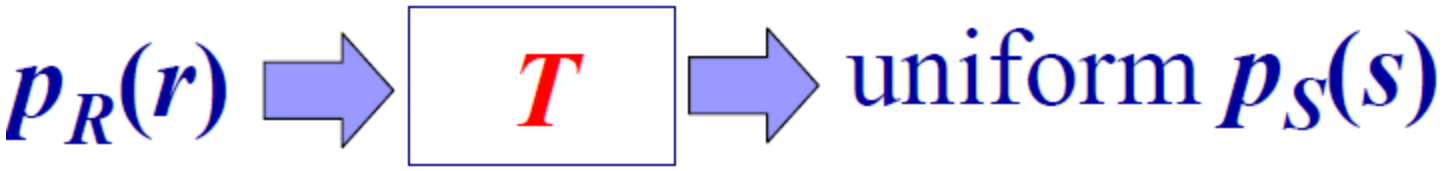
Histogram Transformation

- From previous examples, we notice that a high contrast image has a histogram that spreads out (almost) uniformly over the gray scale range.
- Therefore, if we distribute the histogram of a "poor" image to have a wider range, the quality of the image should improve.
- In other words, we want to adjust the (normalized) histogram (or the PDF) of the input image so that it spreads out more uniformly.

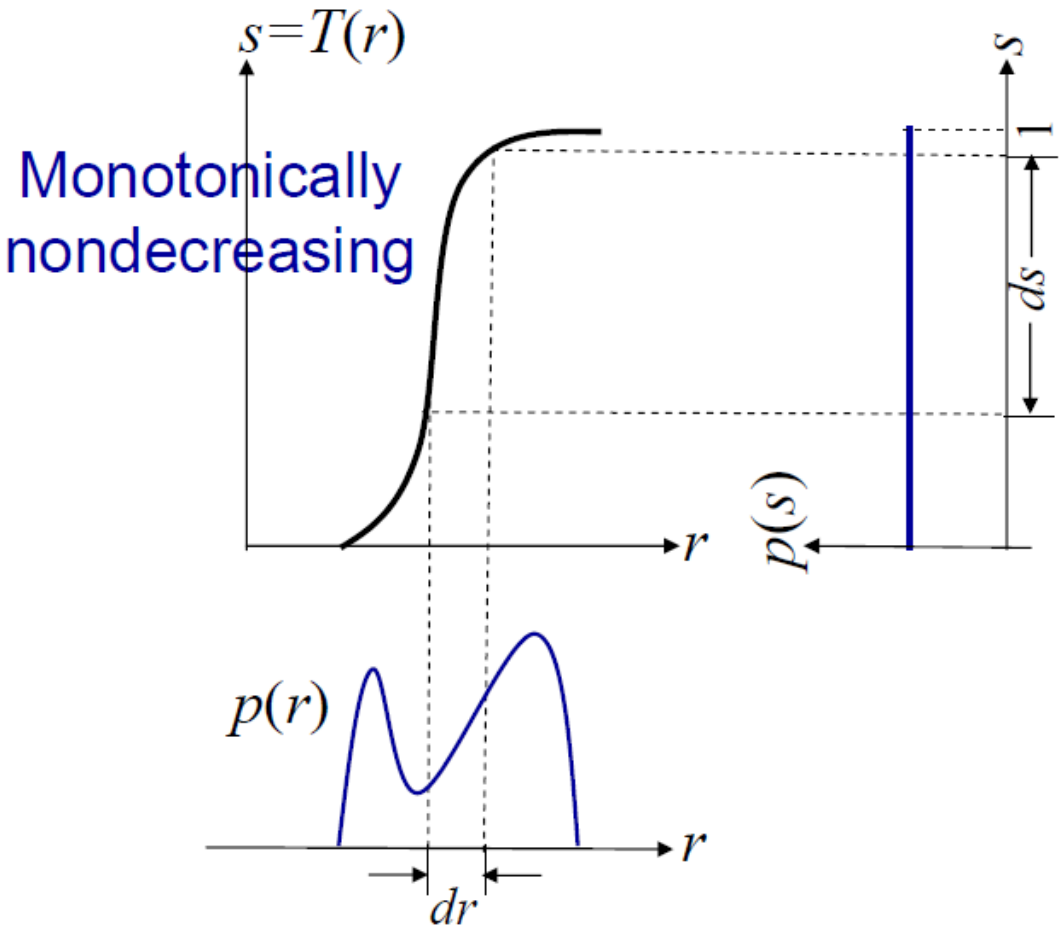
But how do we do it?

Histogram Equalization Design

- What do we have?
  - A (normalized) histogram or $p_R(r)$ of the input image that may not be uniformly distributed
- What do we want?
  - Another (normalized) histogram $p_S(s)$ for an output image that is (more) uniformly distributed
- What do we have to do?
  - Find a transformation T such that



Derivation: Histogram Equalization



$$\int p(s)ds = \int p(r)dr$$

we want $p(s) = 1$

$$s = \int p(r)dr$$

discrete

$$s_k = \sum_{j=0}^{k} p(r_j)$$

$$= \sum_{j=0}^{k} \frac{n_j}{n}$$

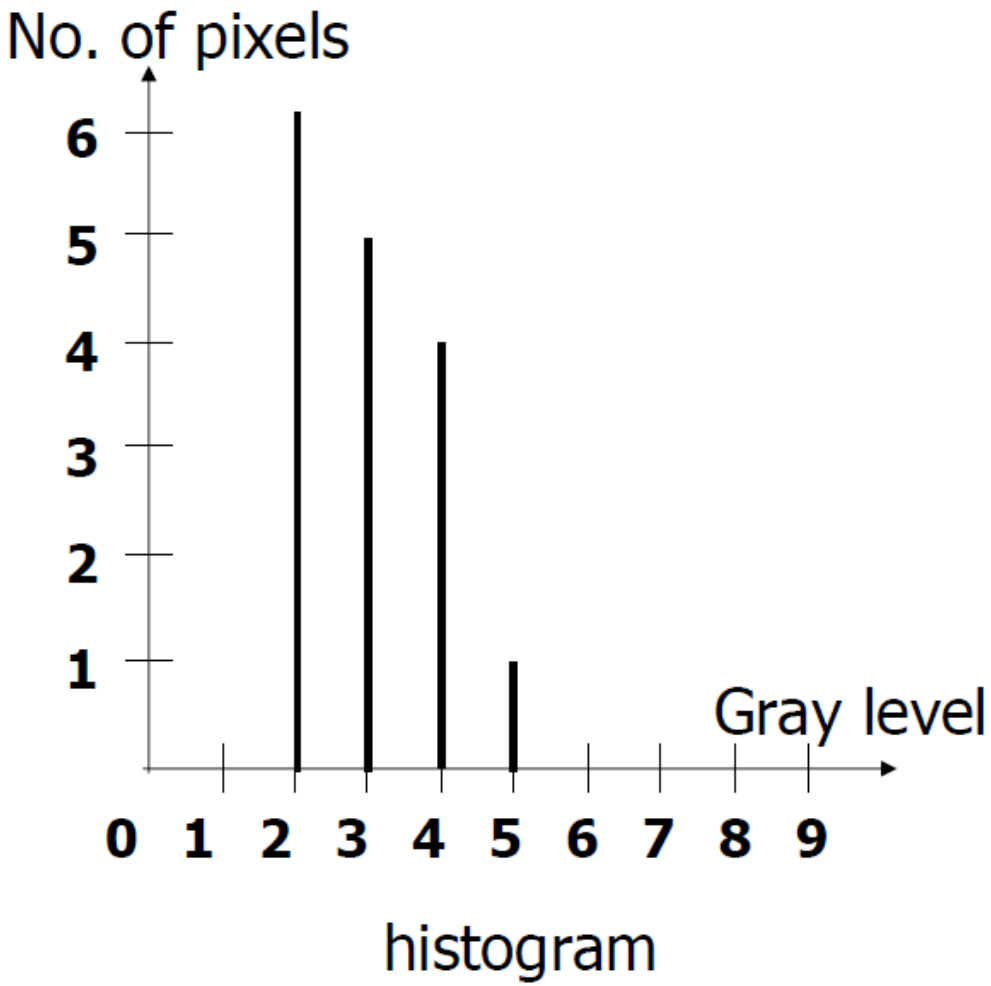
where $k = 0, 1, \ldots, L-1$

Example

2	3	3	2
4	2	4	3
3	2	3	5
2	4	2	4

4x4 image

Gray scale = [0,9]



histogram

Calculation: Histogram Equalization

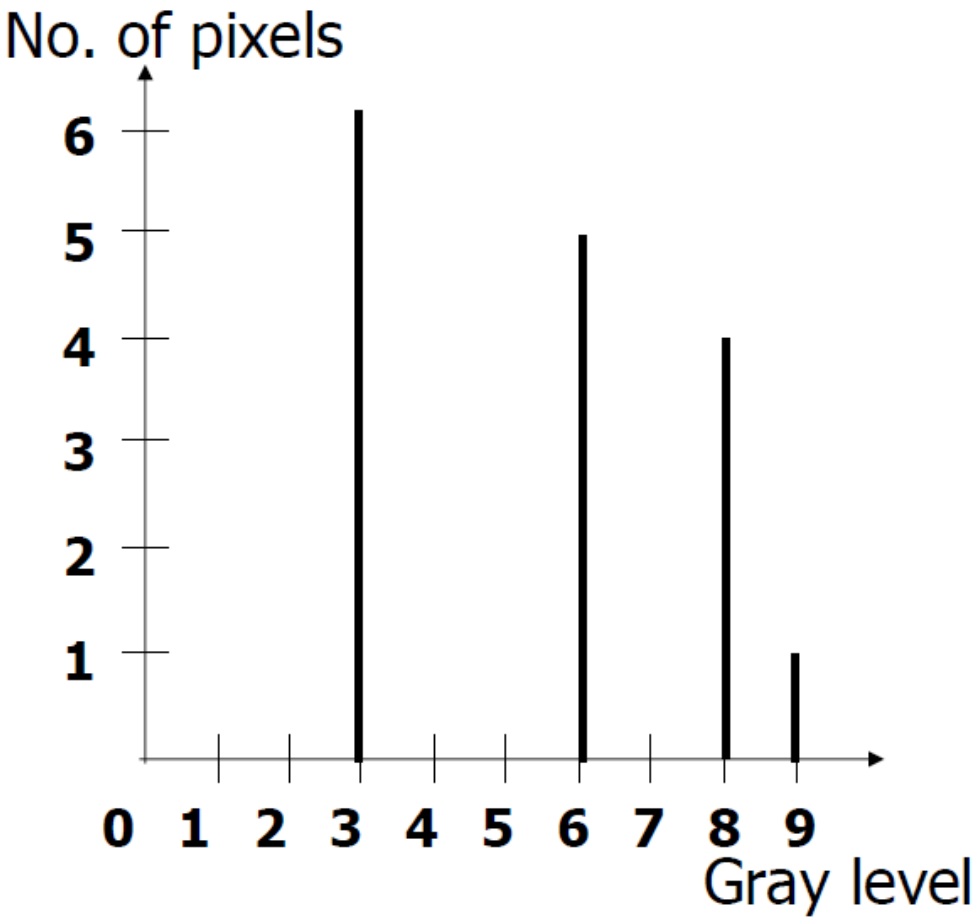
GrayLevel (r_k)	0	1	2	3	4	5	6	7	8	9
No. of pixels	0	0	6	5	4	1	0	0	0	0
$\sum_{j=0}^{k} n_j$	0	0	6	11	15	16	16	16	16	16
$s_k = \sum_{j=0}^{k} \frac{n_j}{n}$	0	0	$\frac{6}{16}$	$\frac{11}{16}$	$\frac{15}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{16}{16}$	$\frac{16}{16}$
$s_k \times 9$	0	0	3.3 ≈ 3	6.1 ≈ 6	8.4 ≈ 8	9	9	9	9	9

Example

3	6	6	3
8	3	8	6
6	3	6	9
3	8	3	8

Output image

Gray scale = [0,9]

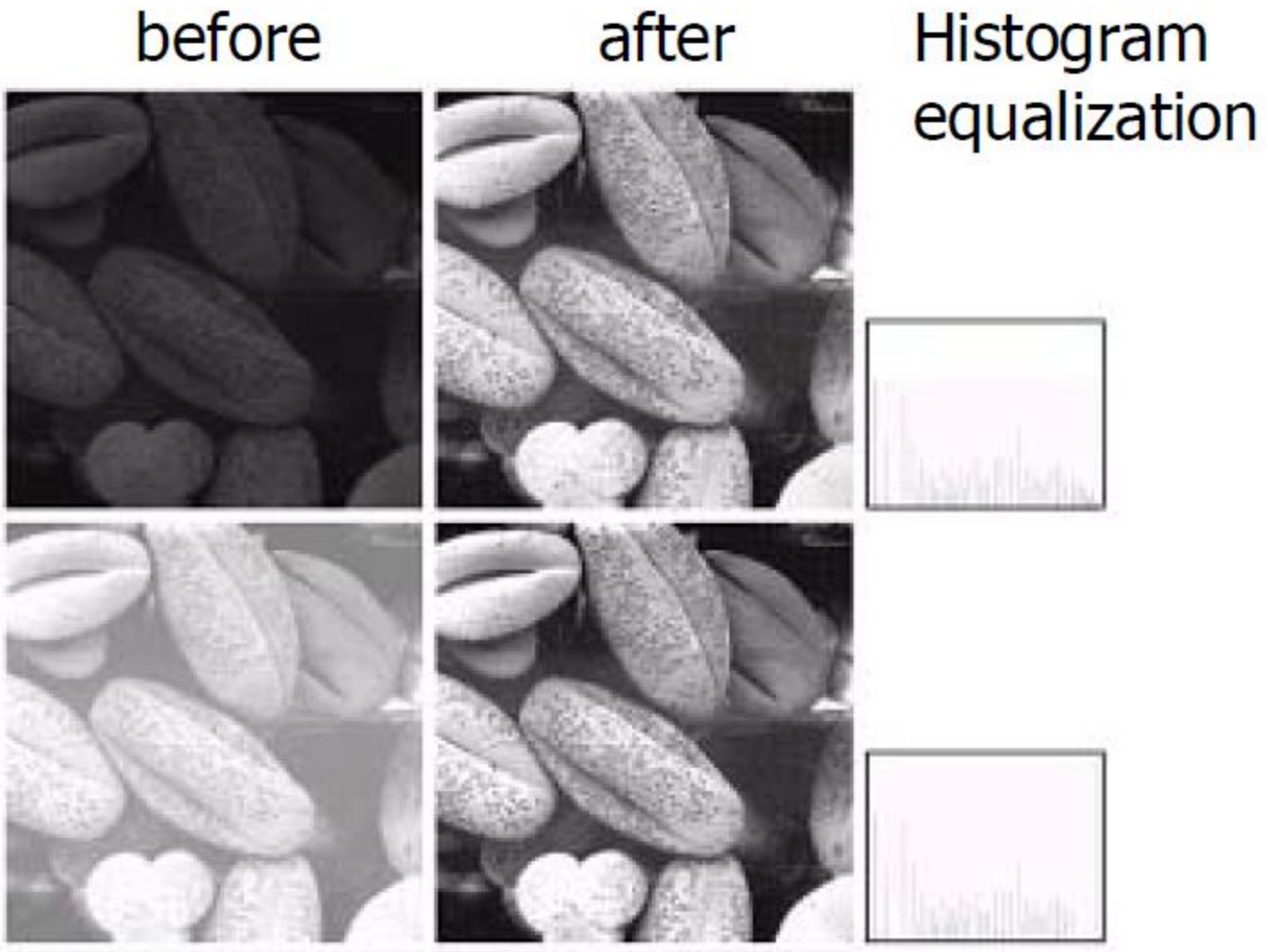


Histogram equalization

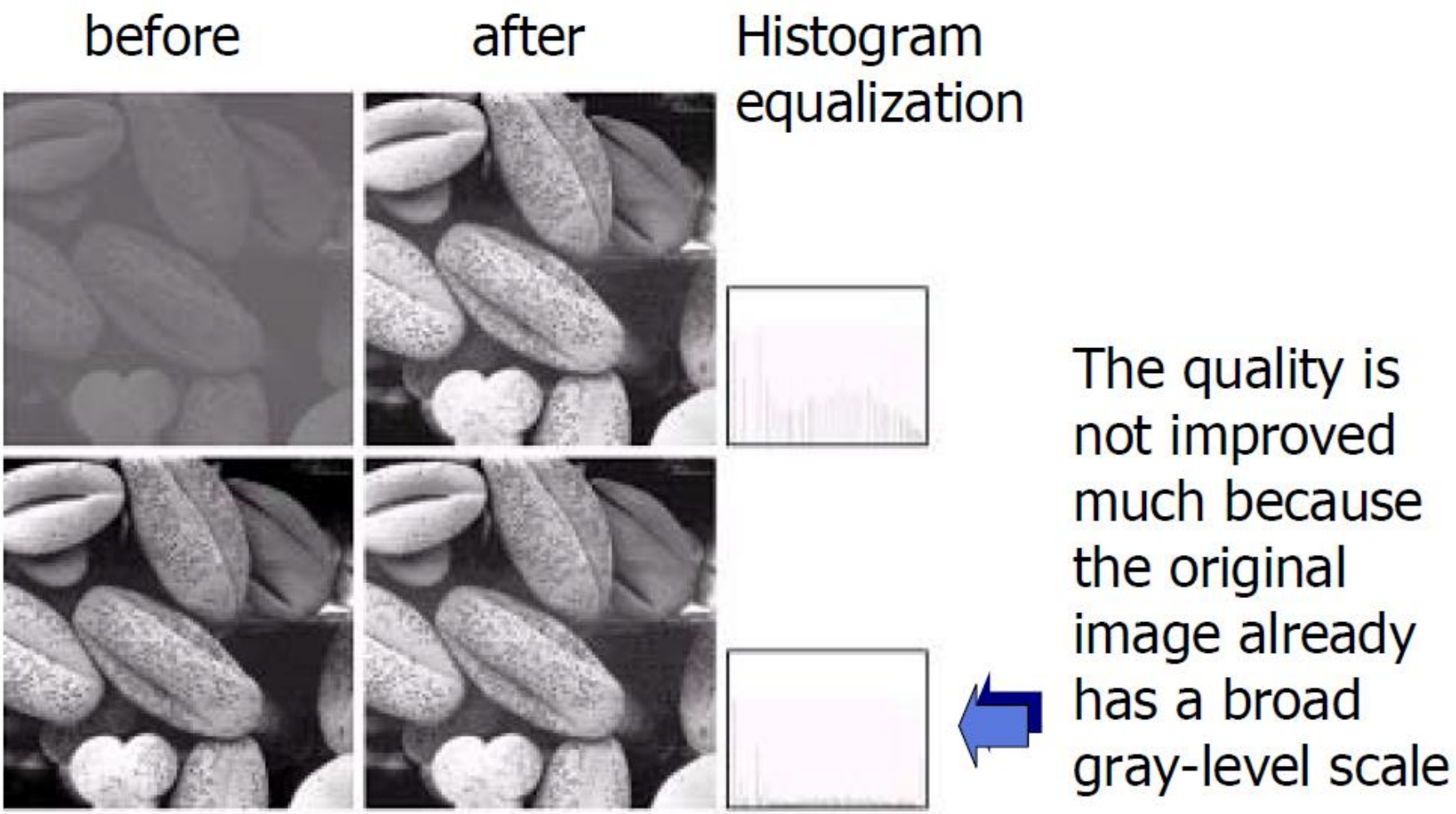
Histogram Equalization

- 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 in the output image
- In discrete space, *it cannot be proven*, in general, *that this discrete transformation will produce the discrete equivalence of a uniform probability density function*, which would be a uniform histogram.

Examples



Examples


before
after
Histogram equalization
The quality is not improved much because the original image already has a broad gray-level scale

Summary

- Histogram equalization distributes the gray level to reach the maximum gray level (white) because the cumulative distribution function equals 1 when $0 \le r \le L-1$
- If the cumulative numbers of gray levels are slightly different, they will be mapped to little different or same gray levels as we may have to approximate the processed gray level of the output image to integer number
- Thus the discrete transformation function can't guarantee the one to one mapping relationship

Histogram Matching (Specification)

- Histogram equalization generates only one type of output image (the one with uniform histogram).
- Sometimes, we may want a particular shape for the output histogram (which may not be uniform)
  -> Histogram Matching or Histogram Specification

(We don't cover Histogram Matching and Specification in this lesson)

Global vs. Local Enhancement

- Histogram processing methods are global processing, in the sense that pixels are modified by a transformation function based on the gray-level content of an entire image.
- Sometimes, we may need to enhance details over small areas in an image, which is called a local enhancement. (Need to use Histogram Statistic)

Arithmetic/Logic Operations

- Arithmetic/Logic operations perform on pixel by pixel basis between two or more images (except for NOT operation which performs on only a single image.)
- The AND and OR used for masking = processing only on the region of interest (ROI)

Operation	Definition
NOT	$c = \bar{a}$
OR	$c = a + b$
AND	$c = a.b$
XOR	$c = a \oplus b = a.\bar{b} + \bar{a}.b$

Logic Operations on Gray Levels

- Logical operators are often used to combine two (mostly binary) images.
- In the case of integer images, the logical operator is normally applied in a bitwise way.
- Light represents a binary 1, and dark represents a binary 0
- NOT operation = negative transformation

A	B	Q
0	0	0
0	1	0
1	0	0
1	1	1

AND

A	B	Q
0	0	1
0	1	1
1	0	1
1	1	0

NAND

A	B	Q
0	0	0
0	1	1
1	0	1
1	1	1

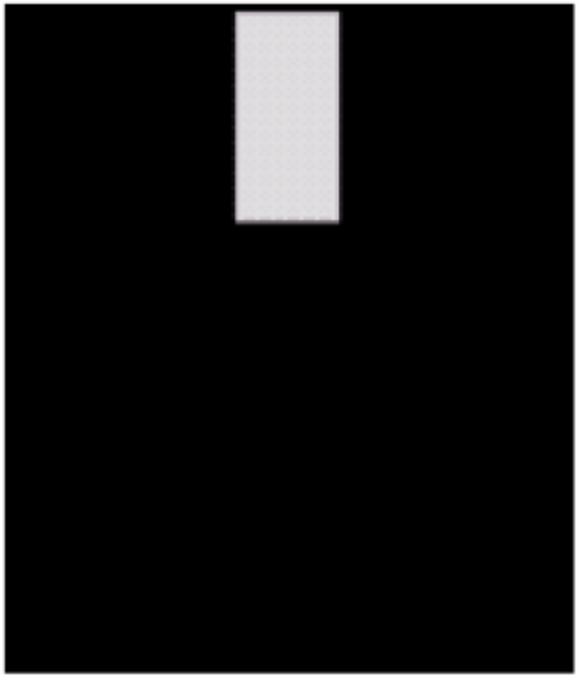
OR

A	B	Q
0	0	1
0	1	0
1	0	0
1	1	0

NOR

AND Operation

original image

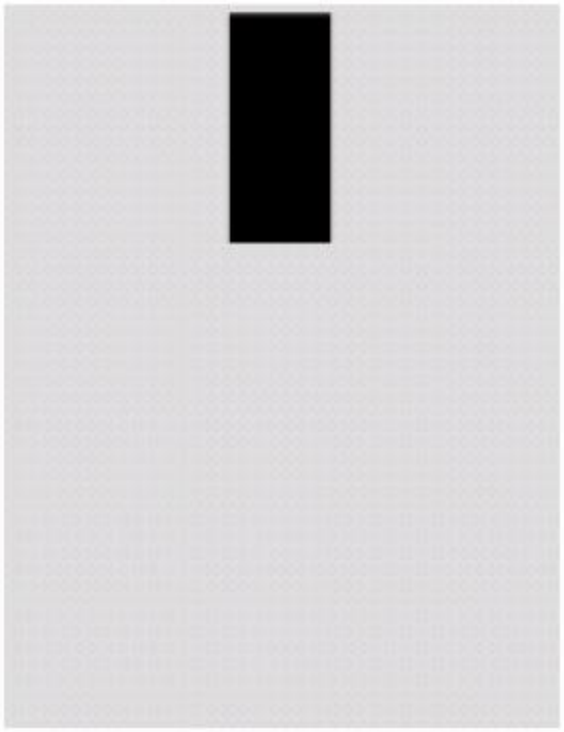


AND image mask

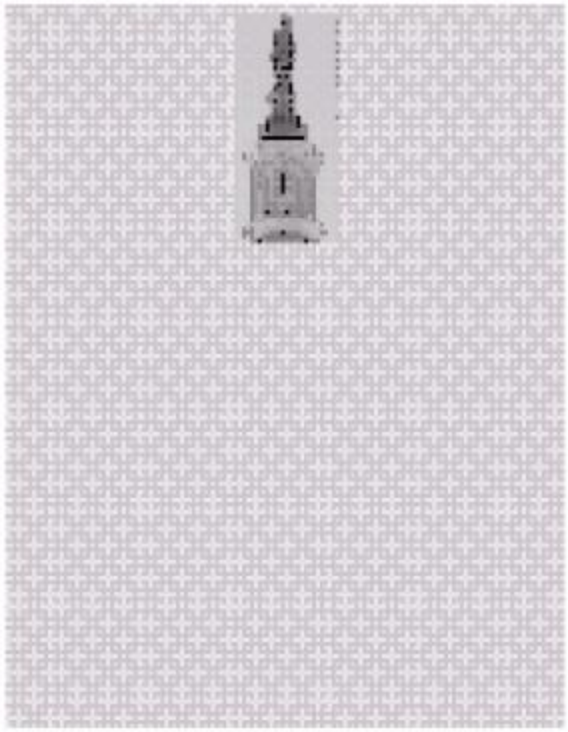
result of AND operation

OR Operation

original image



OR image mask



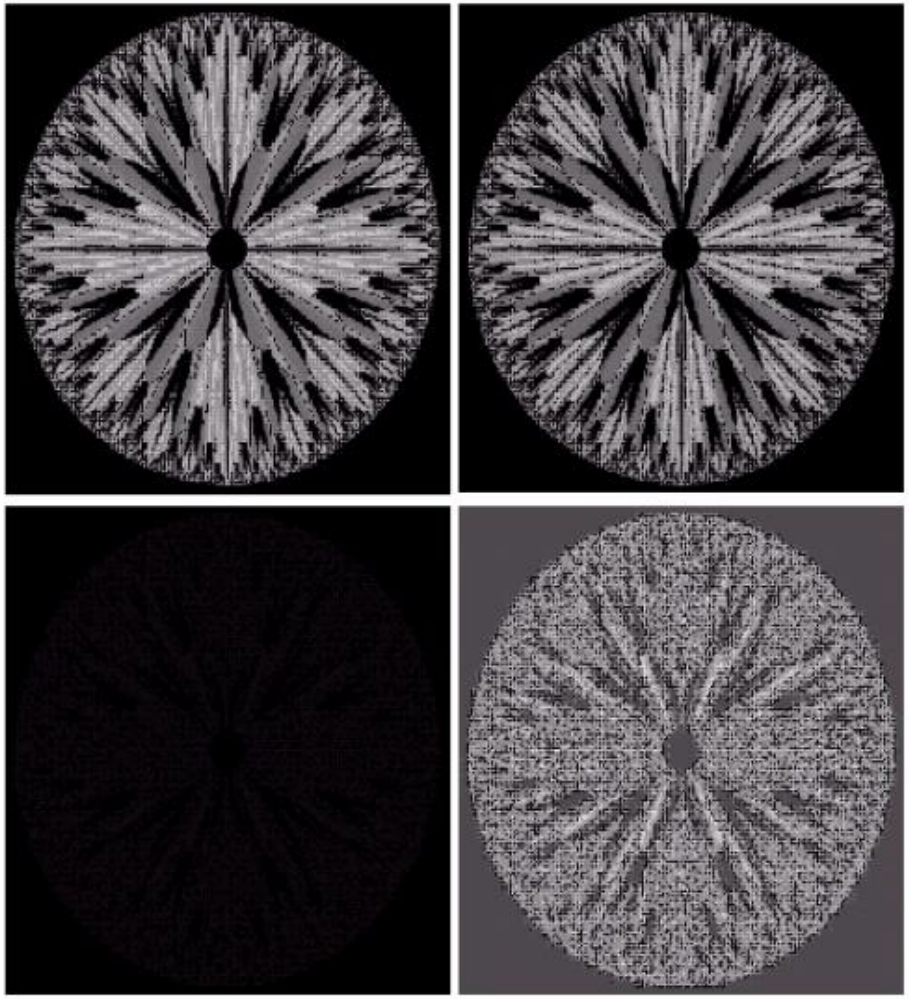
result of OR operation

Image Subtraction

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

- Enhancement of the differences between images

Image Subtraction



a	b
c	d

- a). original fractal image
- b). result of setting the four lower-order bit planes to zero
  - refer to the bit-plane slicing
  - the higher planes contribute significant detail
  - the lower planes contribute to finer detail
  - image b). almost visually identical to image a), with a very slightly drop in overall contrast due to less variability of the gray-level values in the image.
- c). difference between a). and b). (nearly black)
- d). histogram equalization of c). (perform contrast stretching transformation)

Q&A

thittaporn.gan@kmutt.ac.th