Digital Image Processing

Image Enhancement

(Point + Histogram Processing)

By: Dr. Hafeez

Contents

Over the next few lectures we will look at image enhancement techniques working in the spatial domain:

- What is image enhancement?
- Different kinds of image enhancement
- What is point processing?
  - Negative images
  - Thresholding
  - Logarithmic transformation
  - Power law transforms
  - Grey level slicing
  - Bit plane slicing
- Histogram processing

A Note About Grey Levels

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

- Where 0 is black and 255 is white

There is no reason why we have to use this range

- The range [0,255] stems from display technologes

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

What Is Image Enhancement?

Image enhancement is the process of making images more useful and better.

The reasons for doing this include:

1. **Highlighting interesting detail in images**
2. **Removing noise from images**
3. **Making images more visually appealing**

Image Enhancement Examples

Original Image

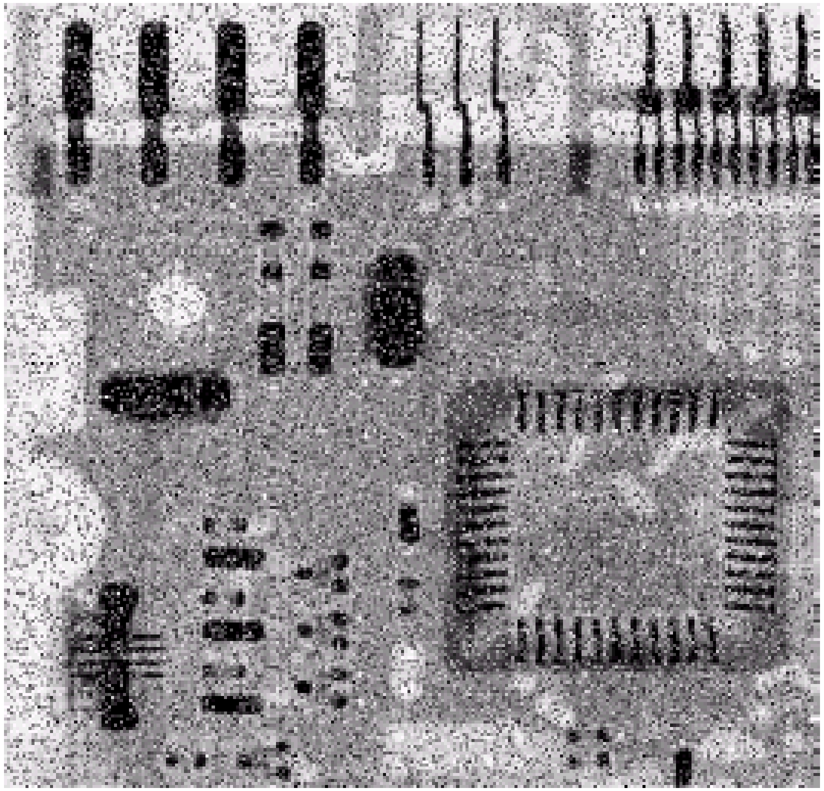
Gamma Enhanced Image

MRI of a fractured spine.

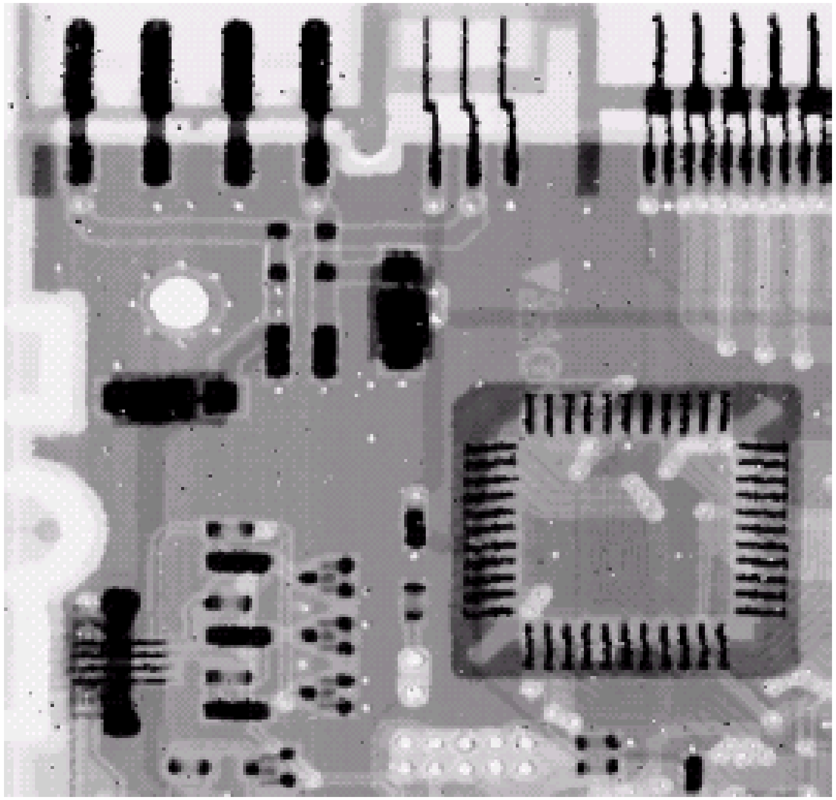
Image Enhancement Examples (cont…)

Image Enhancement Examples (cont…)

Original Image



Median Filtered Image

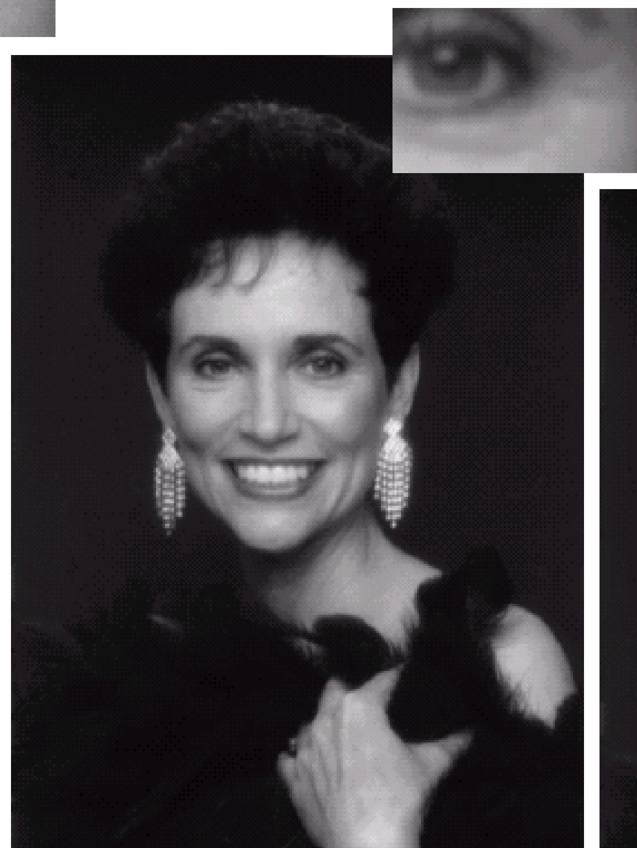


Removing Noise from the image

Image Enhancement Examples (cont…)

Smooth Images

Original Image



Spatial & Frequency Domains

There are **two broad categories** of image enhancement techniques:

1. Spatial domain techniques
   - **Direct manipulation of image pixels**
2. Frequency domain techniques
   - **Manipulation of Fourier transform or wavelet transform of an image**

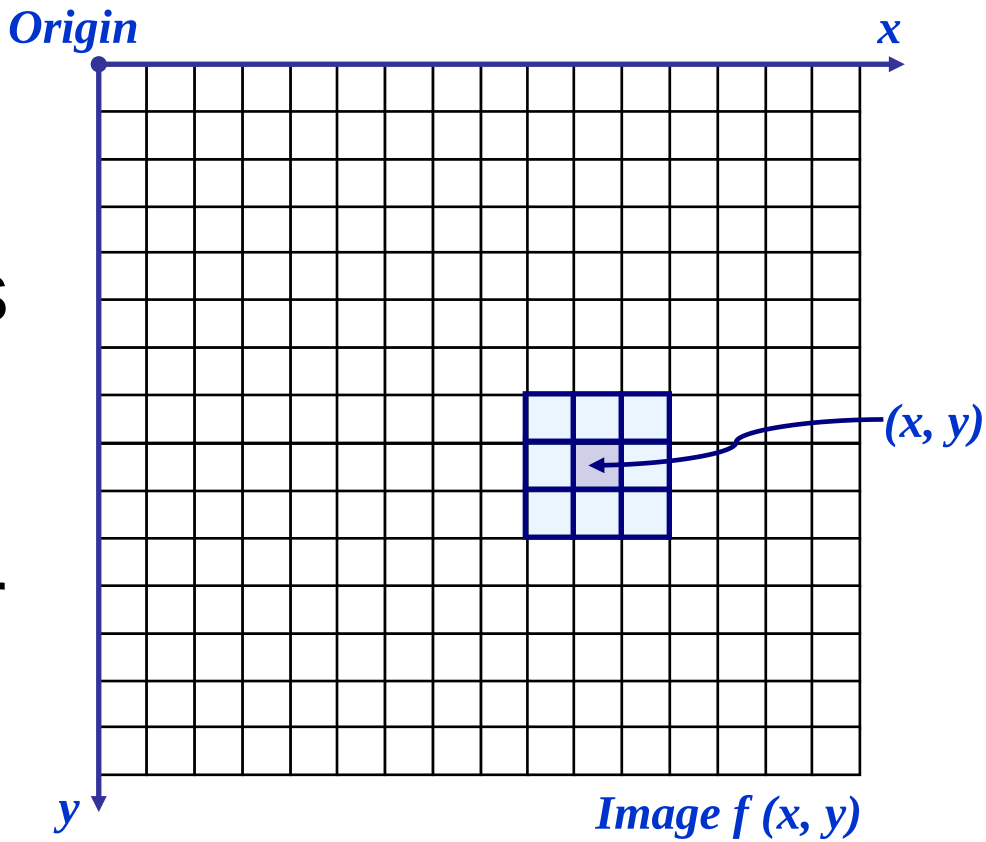
For the moment we will concentrate on techniques that operate in the spatial domain

Basic Spatial Domain Image Enhancement

Most spatial domain enhancement operations can be reduced to the form

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

where $f\ (x, y)$ is the input image, $g\ (x, y)$ is the processed image and T is some operator defined over some neighbourhood of (x, y)



Point Processing

The simplest spatial domain operations occur when the neighbourhood is simply the pixel itself

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

Point processing operations take the form

$$s = T(r)$$

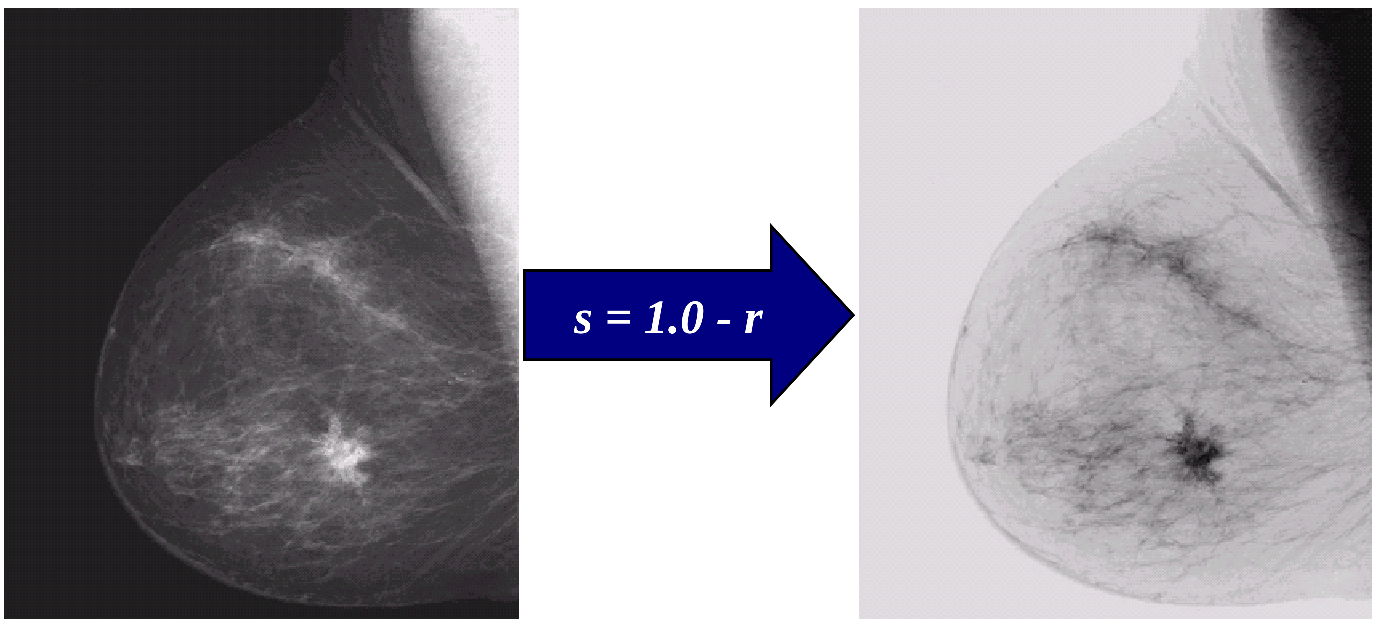
where s refers to the processed image pixel value and r refers to the original image pixel value

Point Processing Example: Negative Images

Negative images are useful for enhancing white or grey detail embedded in dark regions of an image

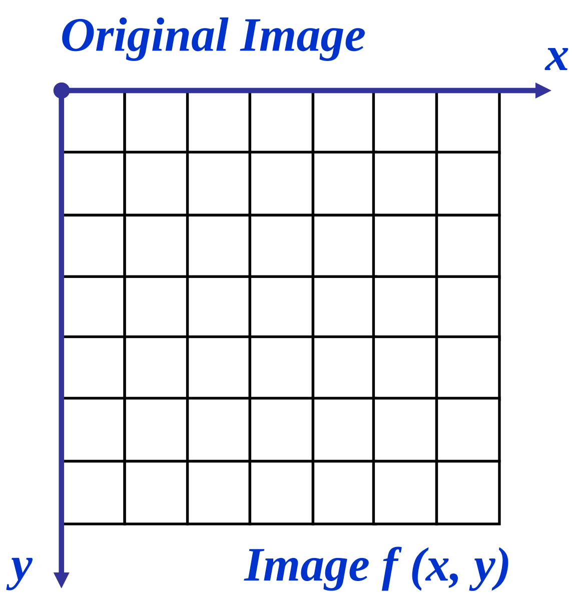
- Note how much clearer the tissue is in the negative image of the mammogram below

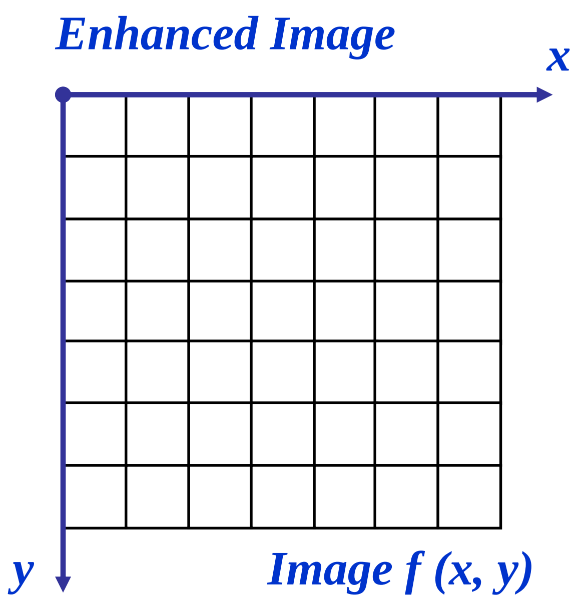
Original Image



Negative Image

Point Processing Example: Negative Images (cont…)



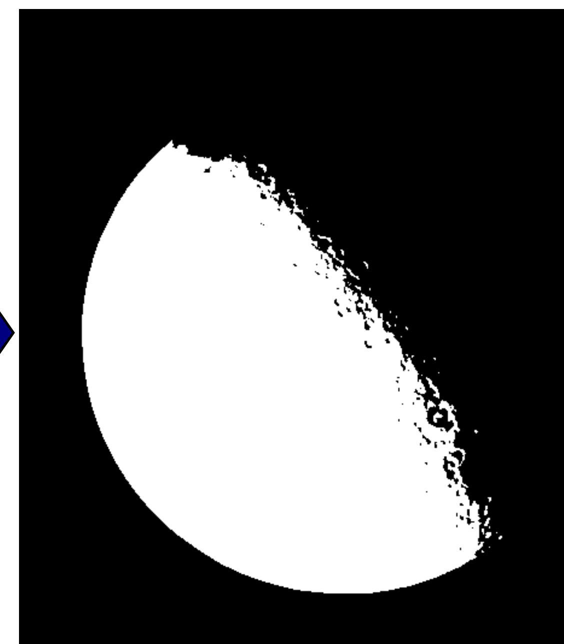


$$s = intensity_{max} - r$$

Point Processing Example: Thresholding

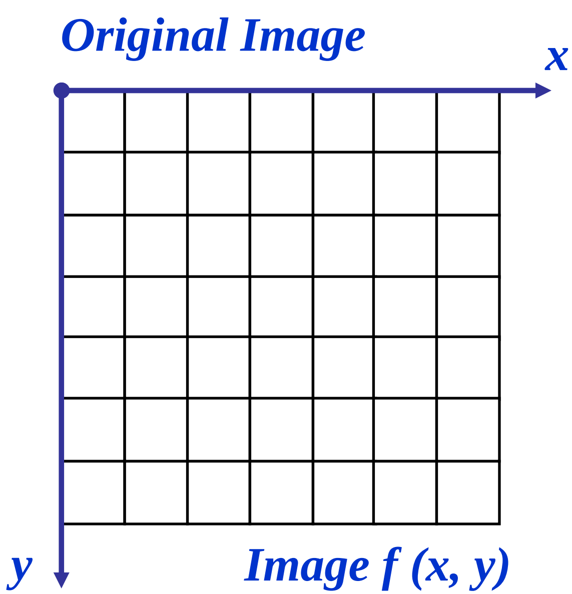
Thresholding transformations are particularly useful for segmentation in which we want to isolate an object of interest from a background

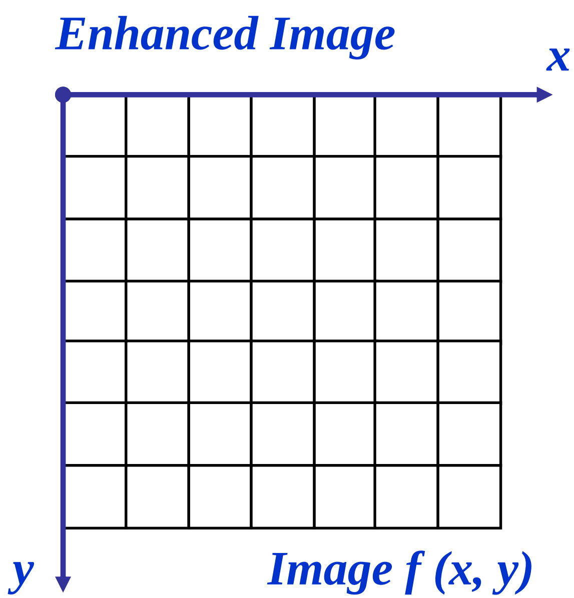
$$s = \begin{cases} 1.0 & r > \text{threshold} \\ 0.0 & r <= \text{threshold} \end{cases}$$



Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Point Processing Example: Thresholding (cont…)





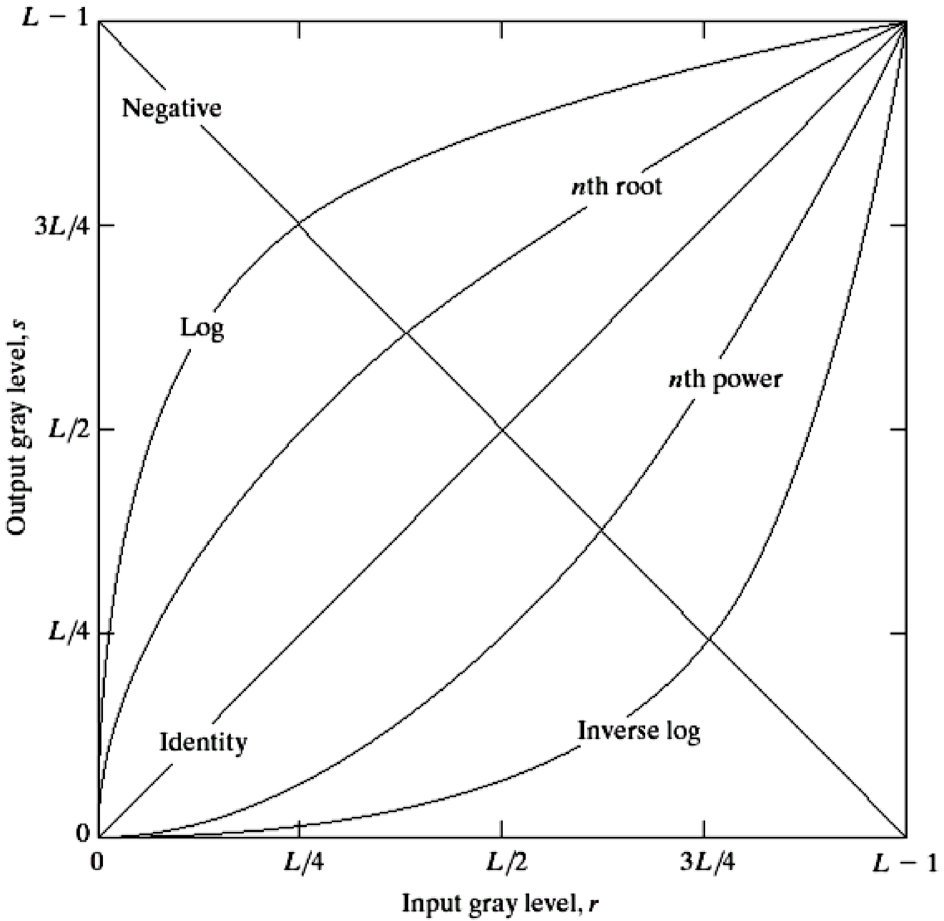
$$s = \begin{cases} 1.0 & r > threshold \\ 0.0 & r <= threshold \end{cases}$$

Basic Grey Level Transformations

There are many different kinds of grey level transformations

Three of the most common are shown here

- Linear
  - Negative/Identity
- Logarithmic
  - Log/Inverse log
- Power law
  - n^{th} power/n^{th} root



Logarithmic Transformations

The general form of the log transformation is

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

The log transformation maps **a narrow range** of **low input grey** level values into a **wider range of output values (stretching)**

The **inverse log** transformation performs the opposite transformation

Log transformation increases brightness and contrast of dark regions while reduces contrast of light regions.

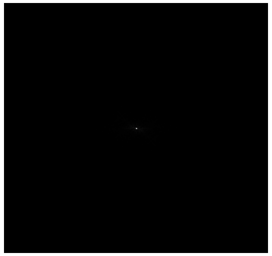
Logarithmic Transformations (cont…)

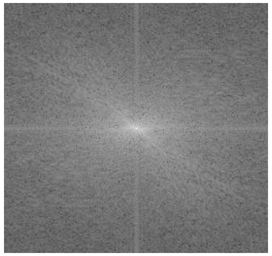
- Log functions are particularly useful when the input grey level values may have an extremely large range of values
- In the following example the Fourier transform of an image is put through a log transform to reveal more detail

lena

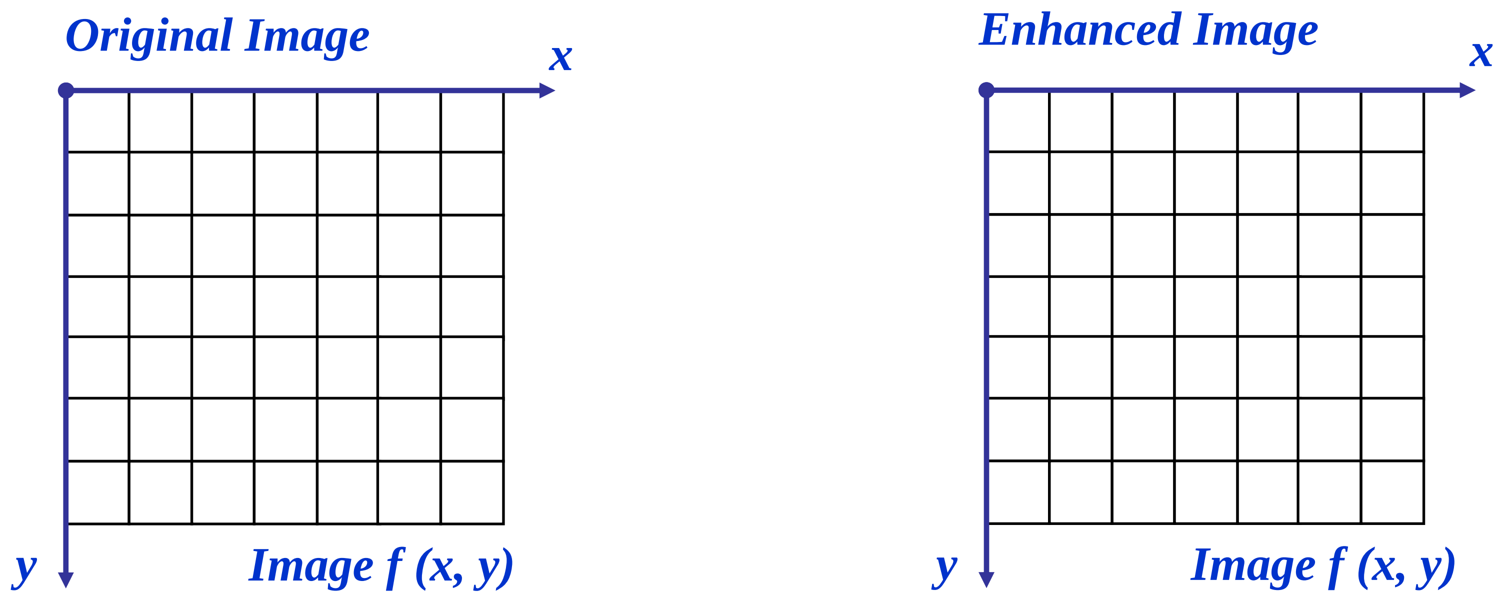
$s = \log_e(1 + r)$

FFT(lena)





Logarithmic Transformations (cont…)



$$s = log_e(1 + r)$$

We usually set c to 1

Grey levels must be in the range [0.0, 1.0]

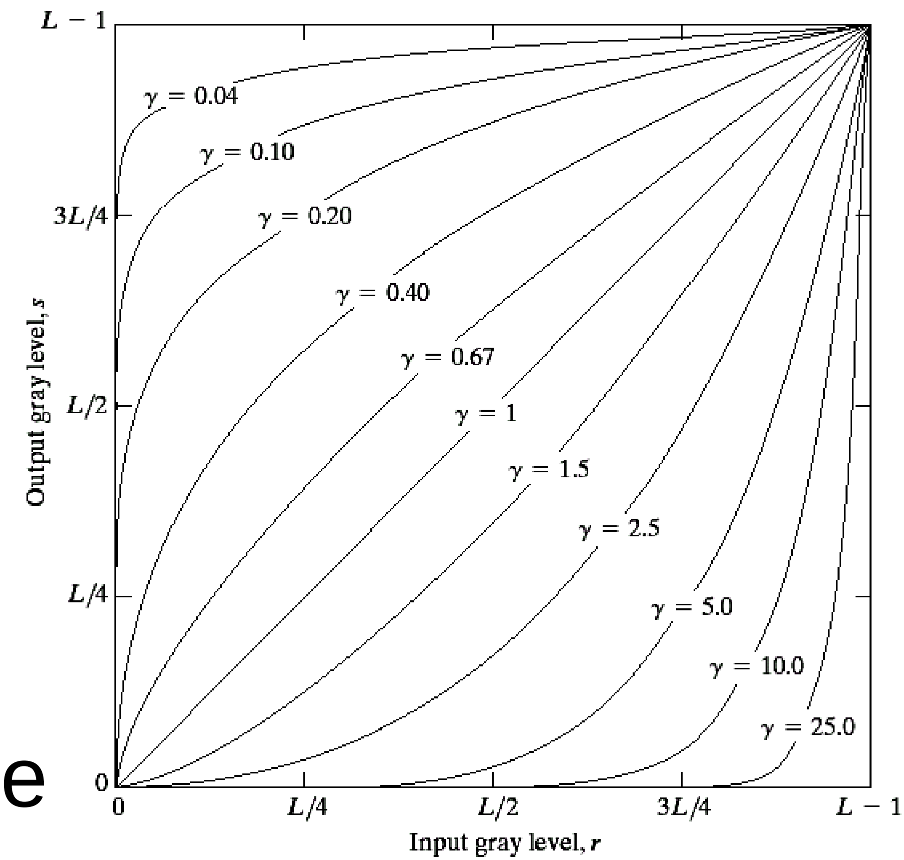
Power Law Transformations

Power law transformations have the following form

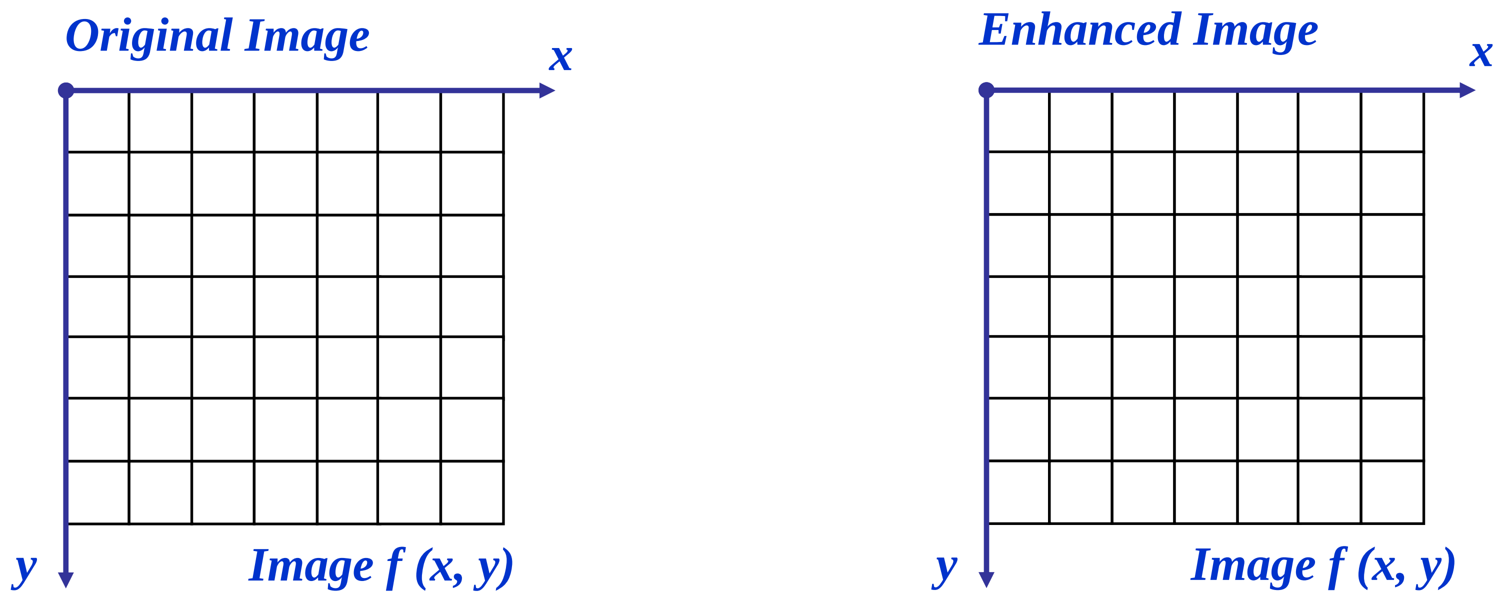
$$s = c * r^{\gamma}$$

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 Transformations (cont…)



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

We usually set c to 1

Grey levels must be in the range [0.0, 1.0]

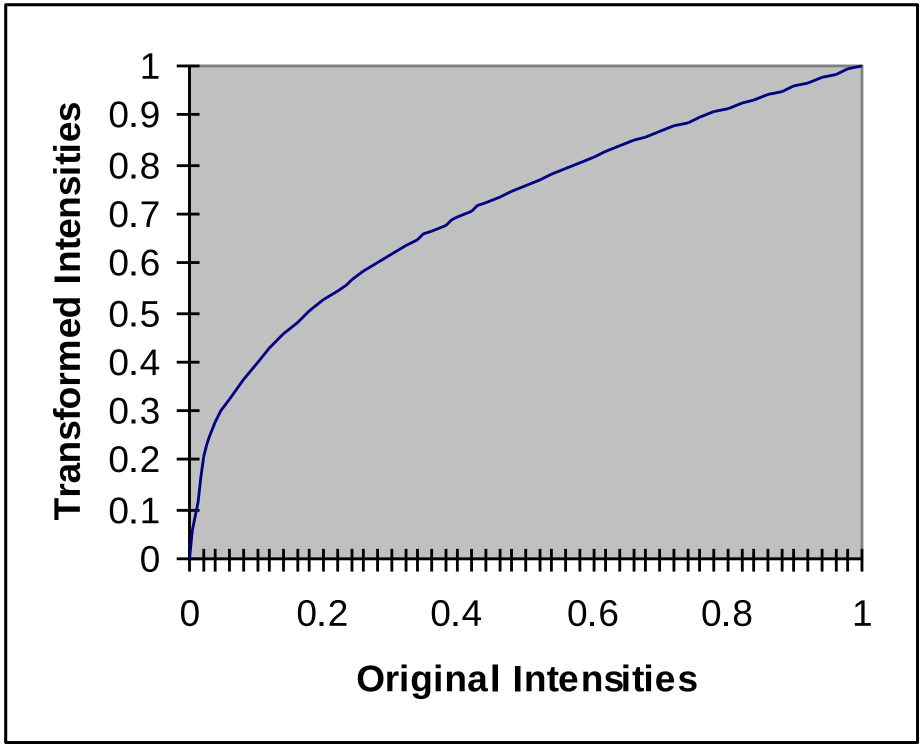
Power Law Example

Power Law Example (cont…)

$$\gamma = 0.6$$

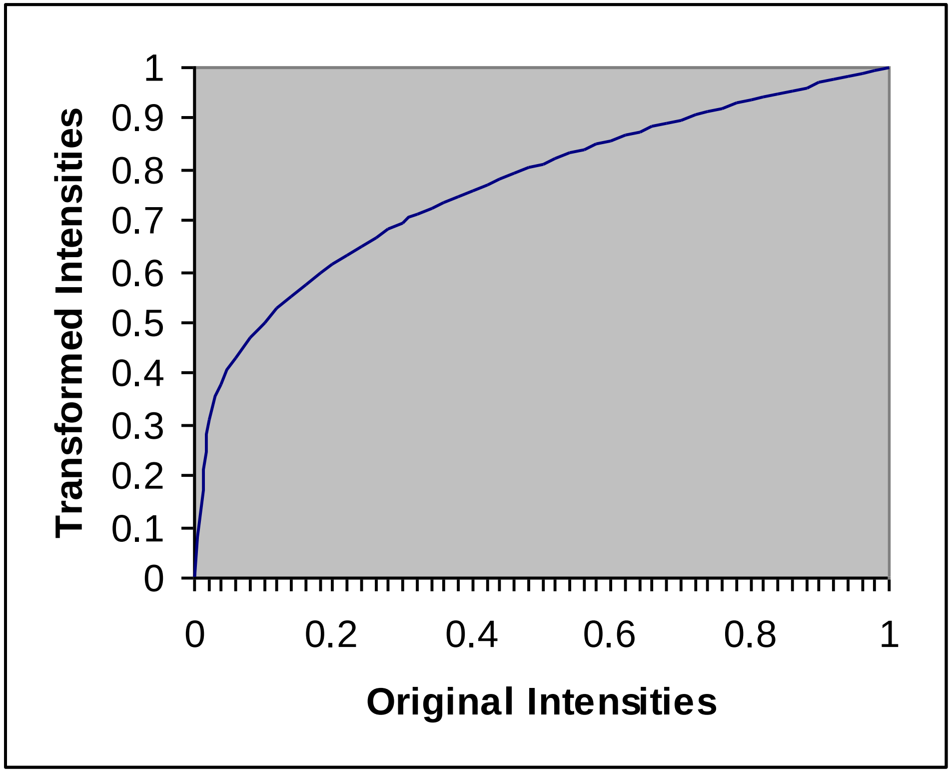
Power Law Example (cont…)

$$\gamma = 0.4$$



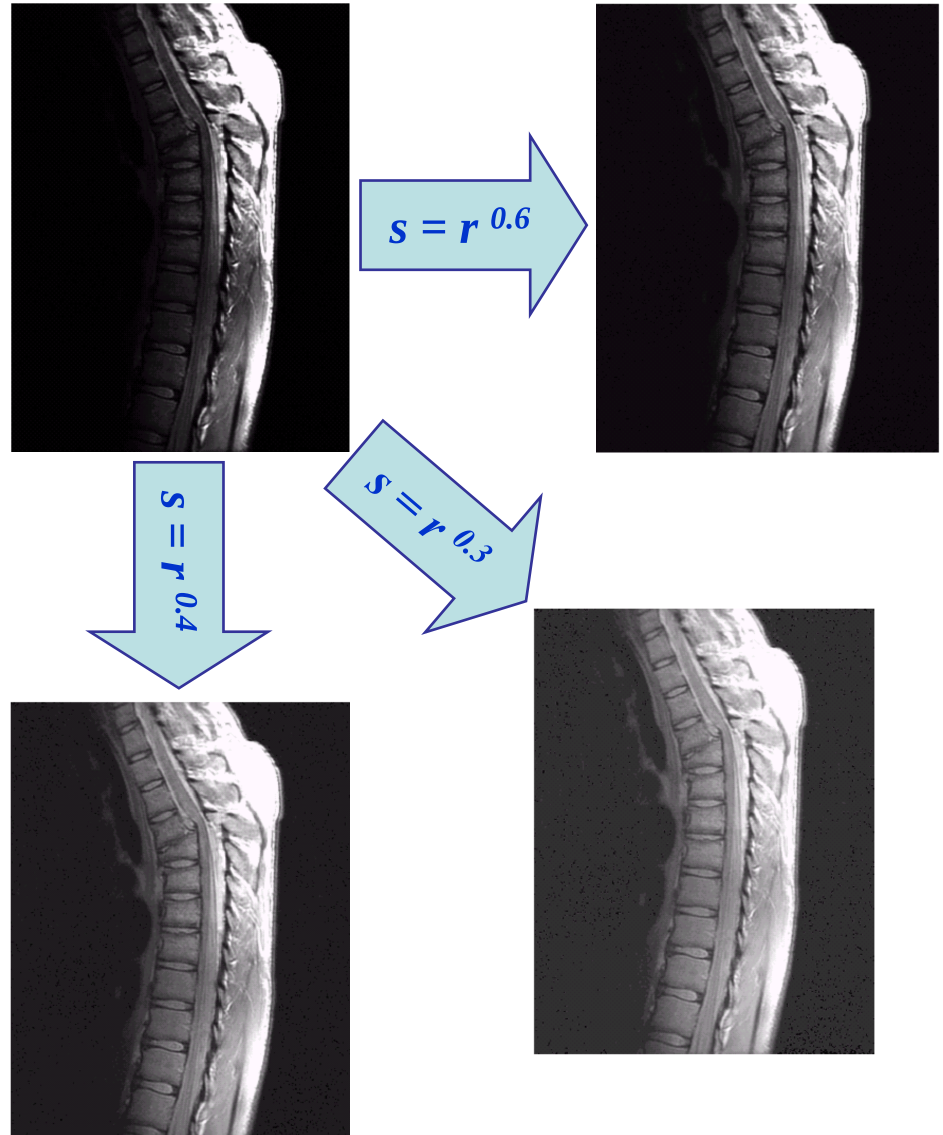
Power Law Example (cont…)

$$\gamma = 0.3$$



Power Law Example (cont…)

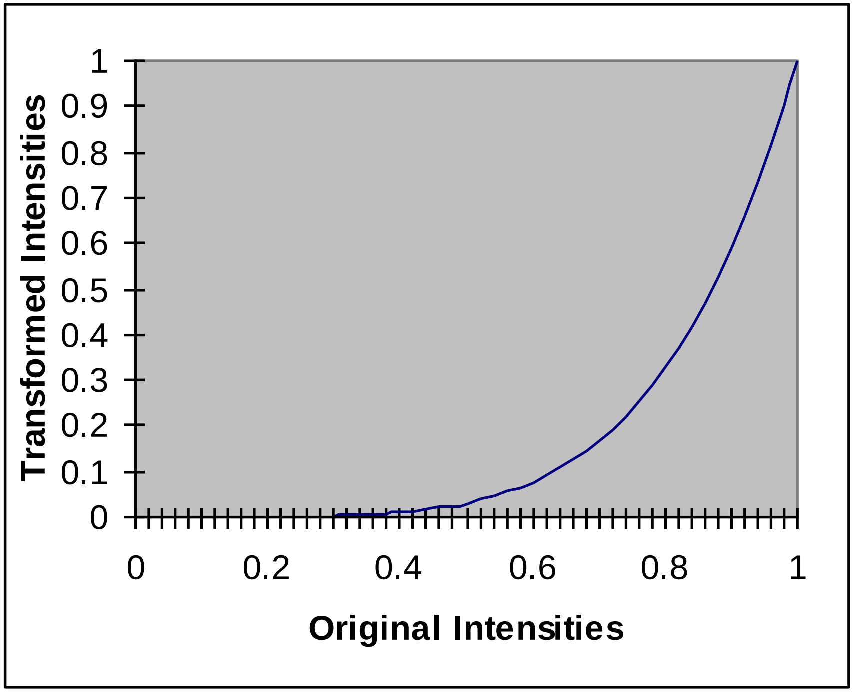
- The images to the right show a magnetic resonance (MR) image of a fractured human spine
- Different curves highlight different detail



Power Law Example

Power Law Example (cont…)

$$\gamma = 5.0$$

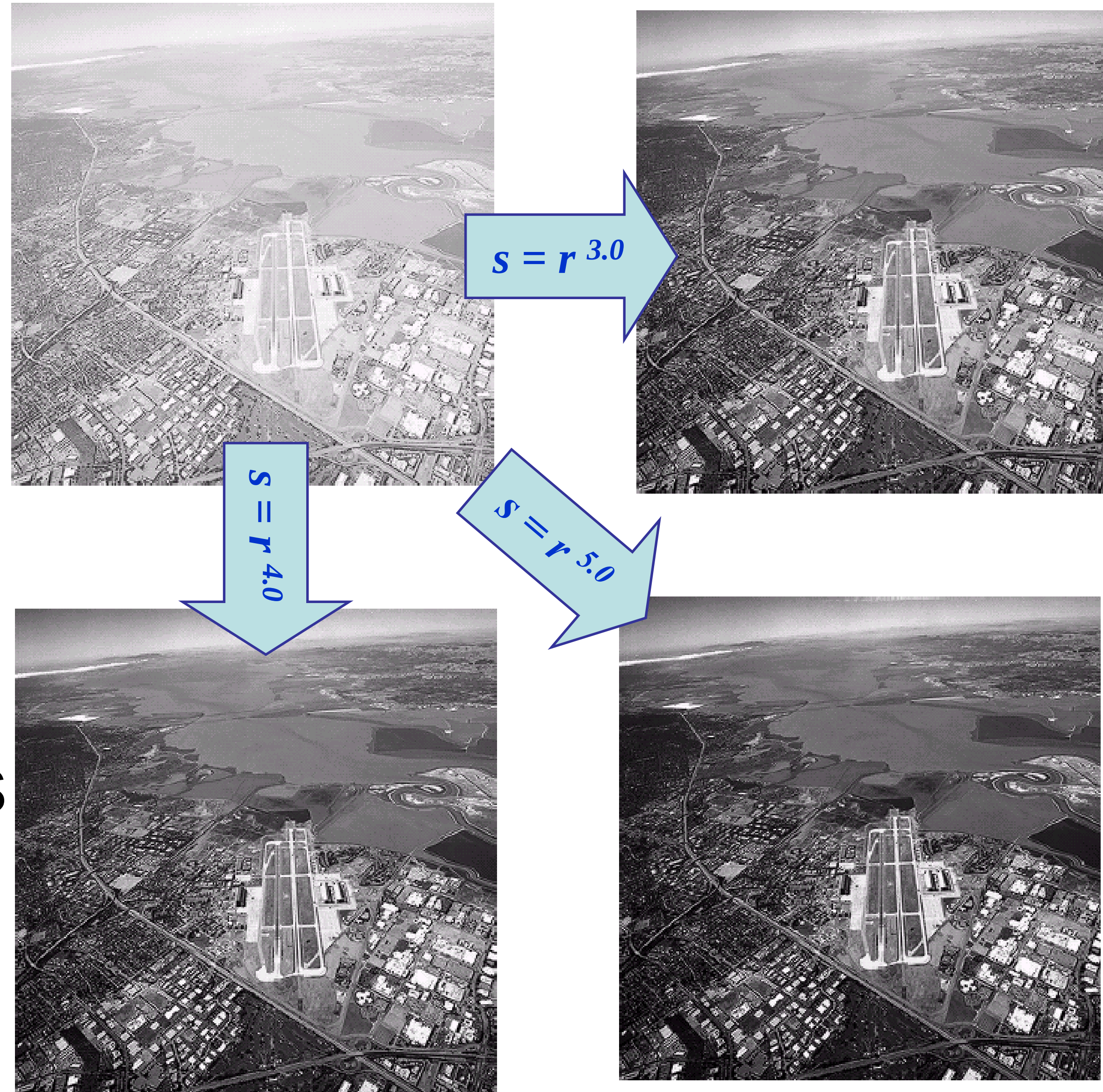


Power Law Transformations (cont…)

An aerial photo of a runway is shown

This time power law transforms are used to darken the image

Different curves highlight different detail

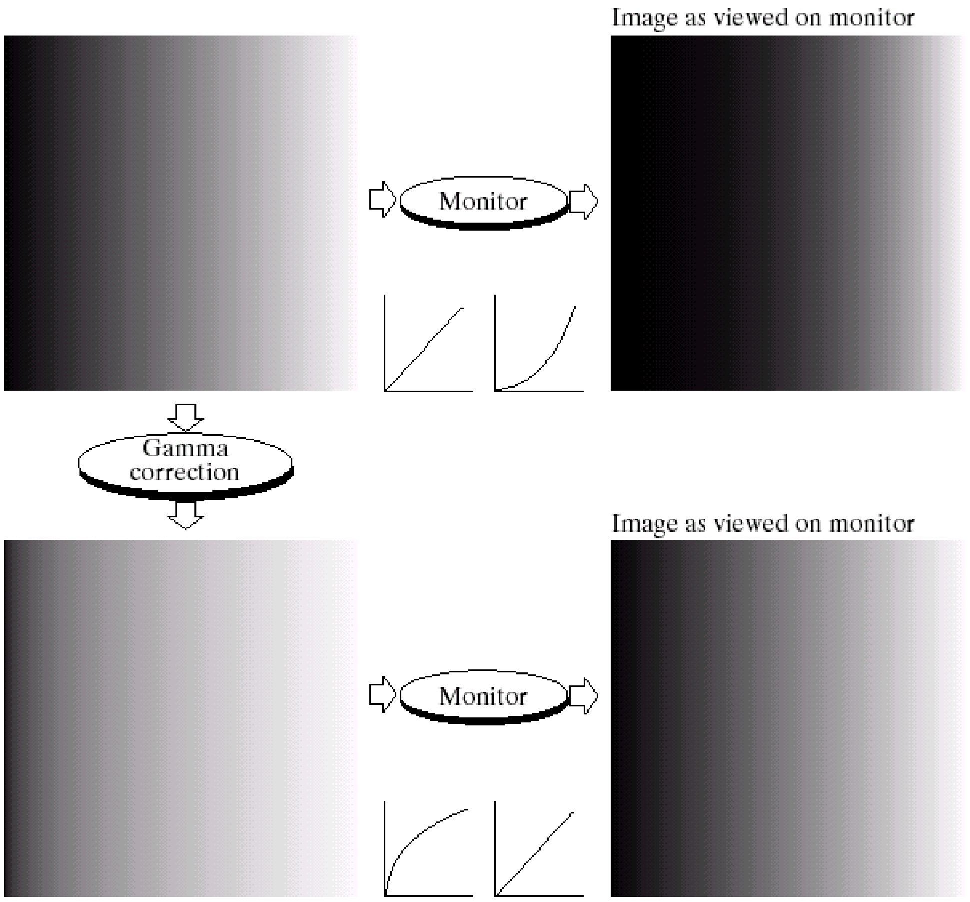


Gamma Correction

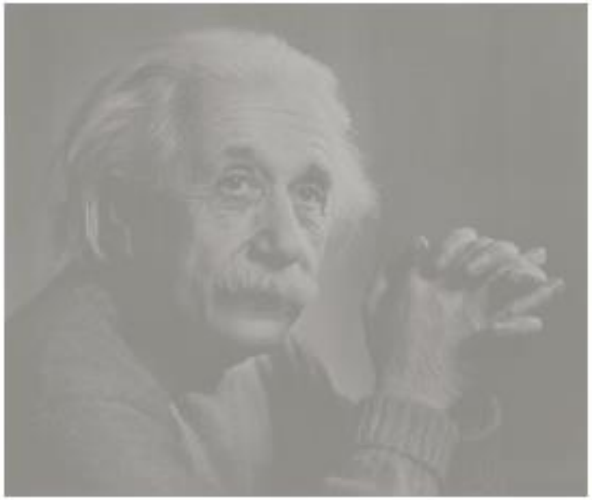
Many of you might be familiar with gamma correction of computer monitors

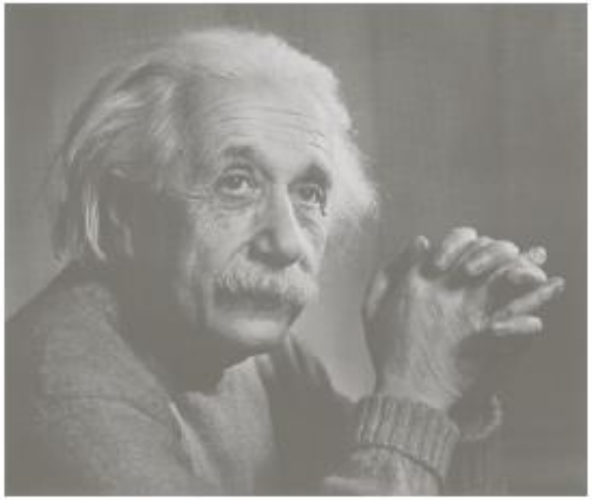
Problem is that display devices do not respond linearly to different intensities

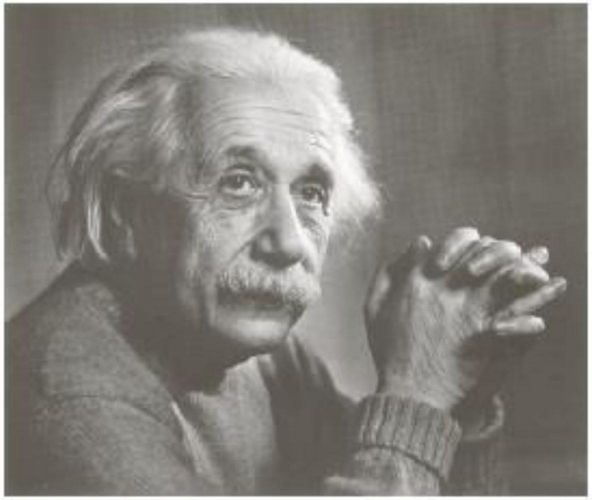
Can be corrected using a log transform



More Contrast Issues



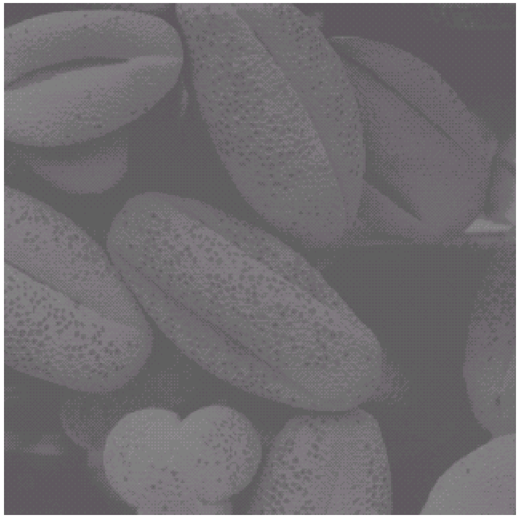


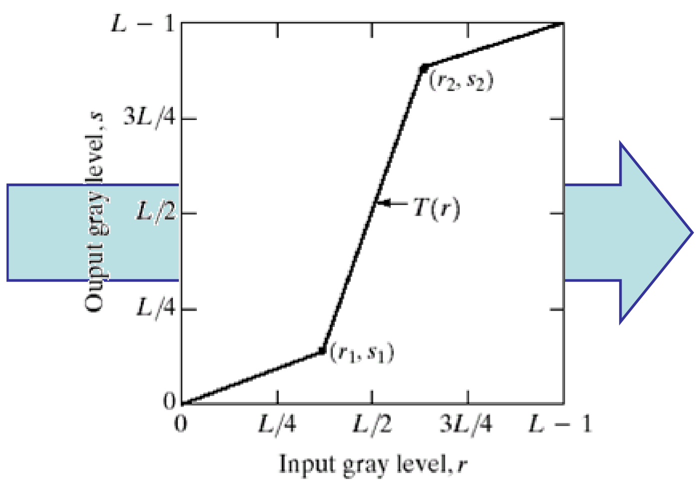


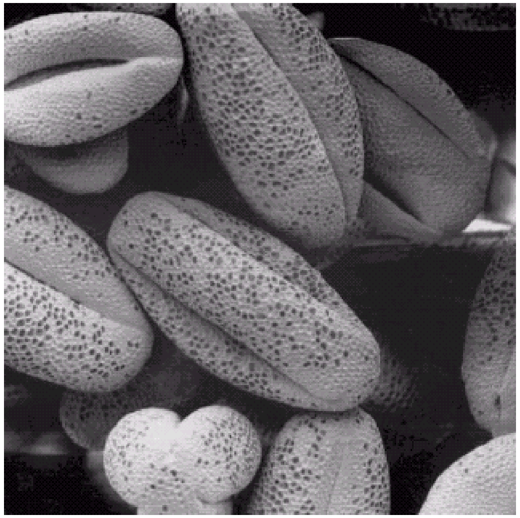
Piecewise Linear Transformation Functions

Rather than using a well defined mathematical function we can use arbitrary user-defined transforms

The images below show a contrast stretching linear transform to add contrast to a poor quality image







Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Contrast Stretching

We can **fix images** that have **poor contrast** by applying a pretty simple contrast specification

The interesting part is how do we decide on this transformation function?

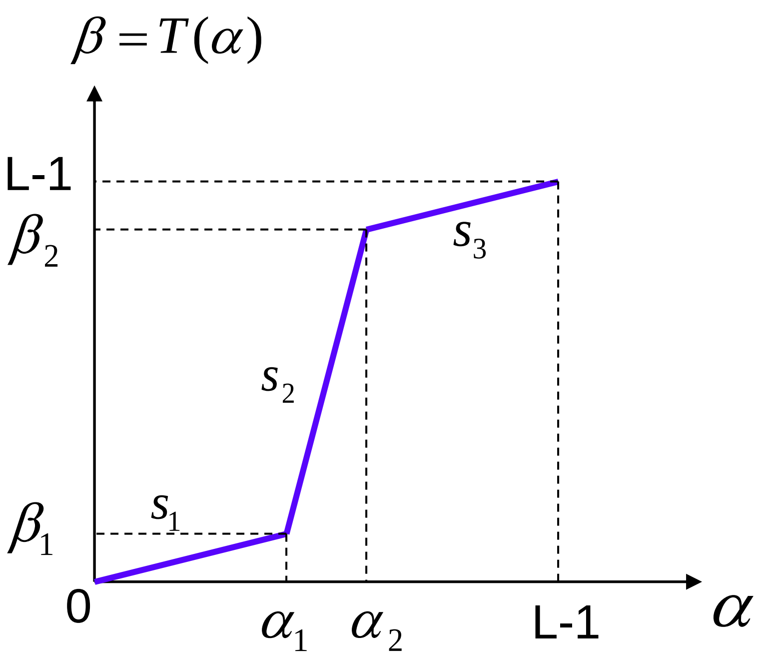
$$s = (r - c)\left(\frac{b - a}{d - c}\right) + a$$

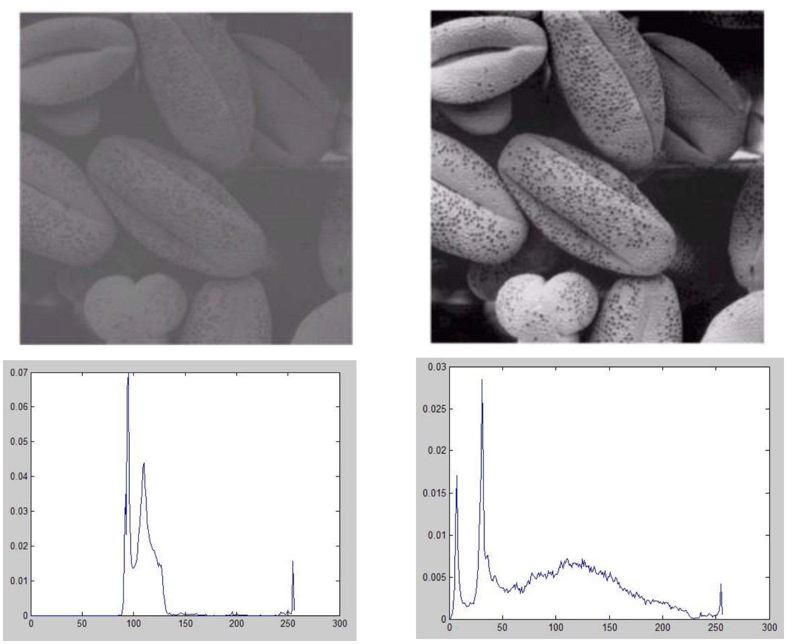
Where s is the output intensity, **r** is the input intensity, **a** and **b** are the lower and upper limit of intensities in gray level image (usually 0 and 255 for 8 bit gray images), **c** and **d** are the stretching points in the input image.

Contrast Stretching

Stretch the over-concentrated gray-levels

Piece-wise linear function, where the slope in the stretching region is greater than 1.

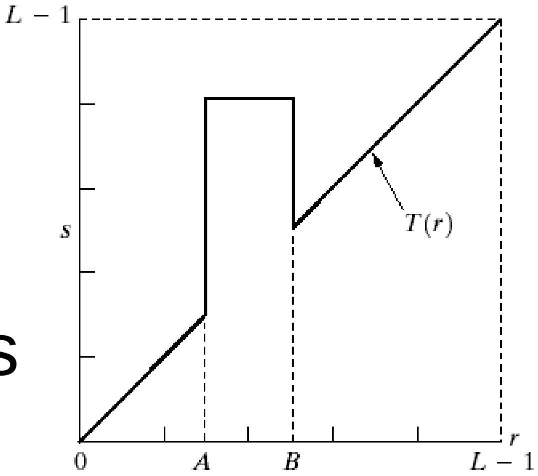


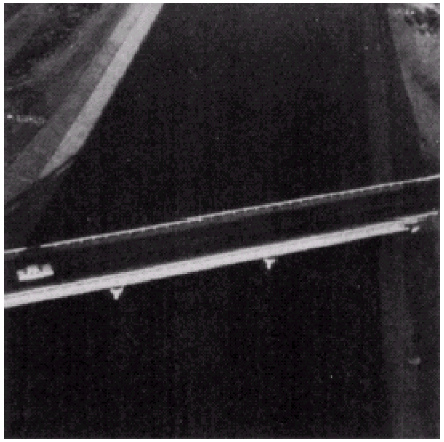


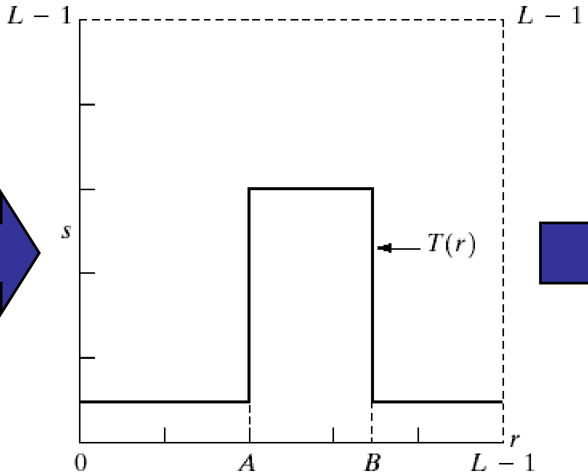
Gray Level Slicing

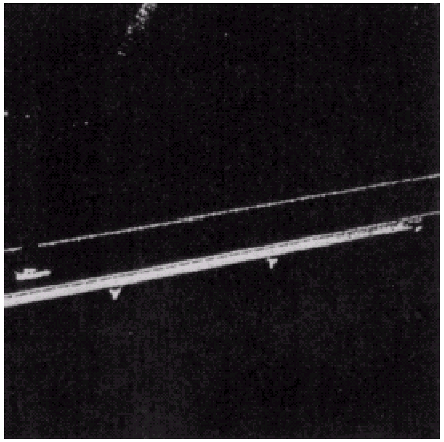
Highlights a specific range of grey levels

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







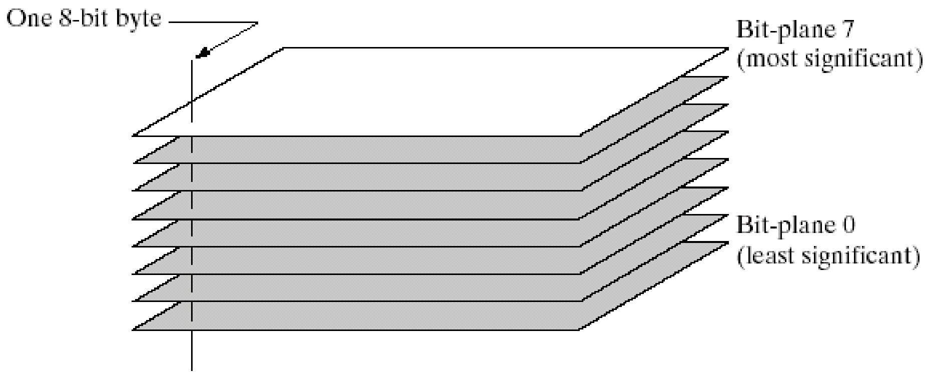


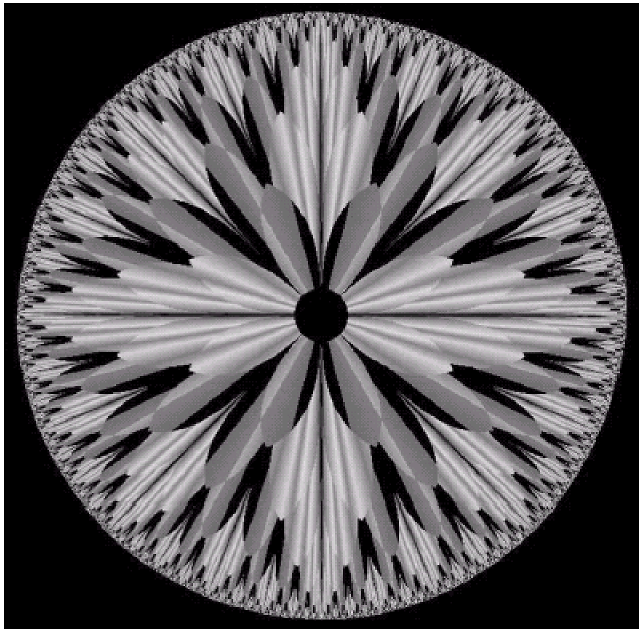
Bit Plane Slicing

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

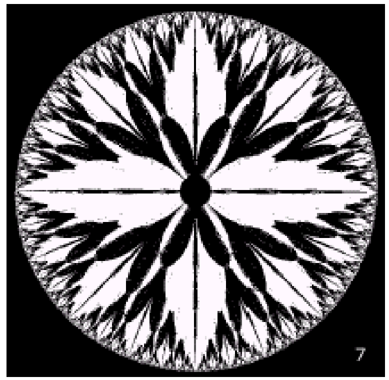
– Lower-order bits contain subtle details

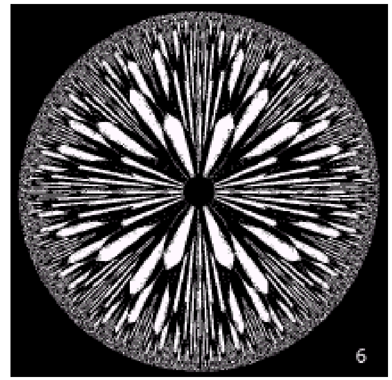




Bit Plane Slicing (cont…)

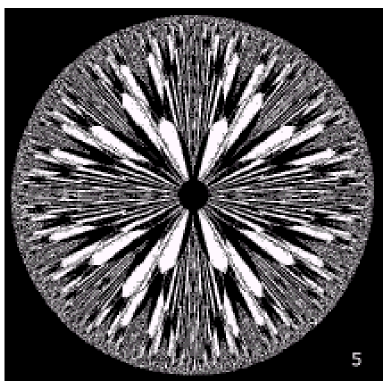
[10000000]

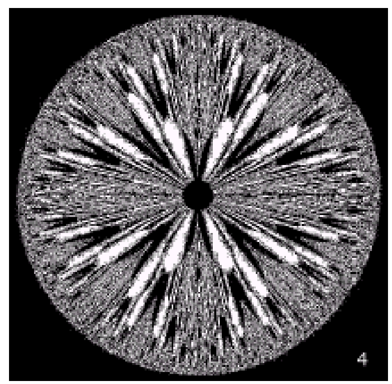


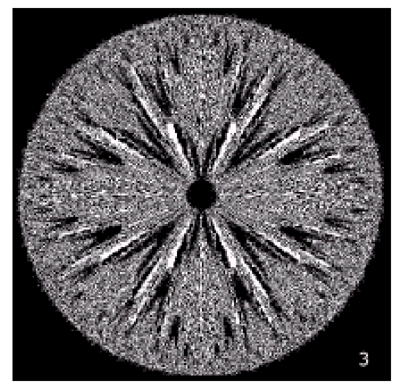


[01000000]

[00100000]

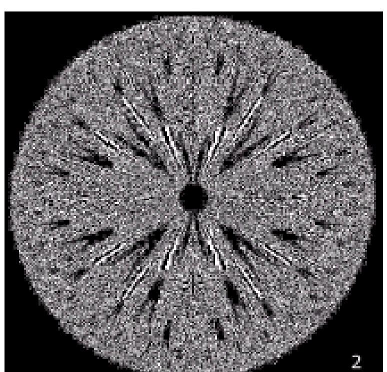


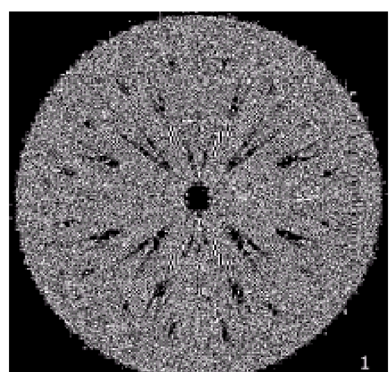


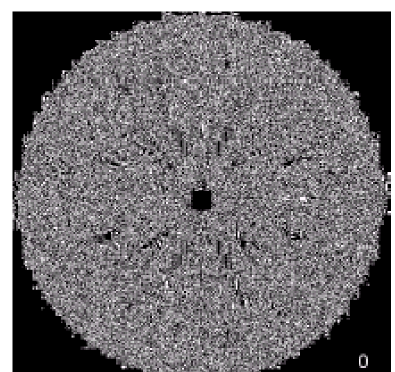


[00001000]

[00000100]



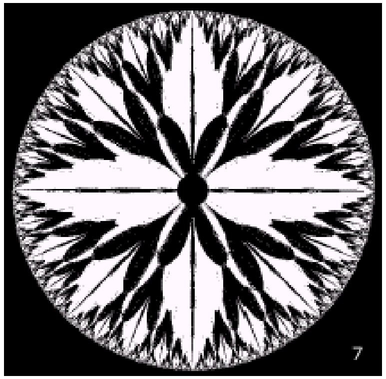


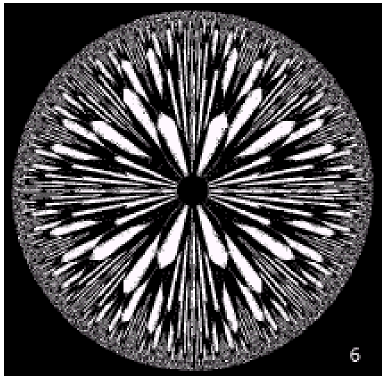


[00000001]

Bit Plane Slicing (cont…)

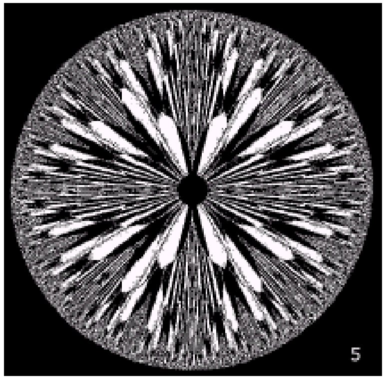
[10000000]

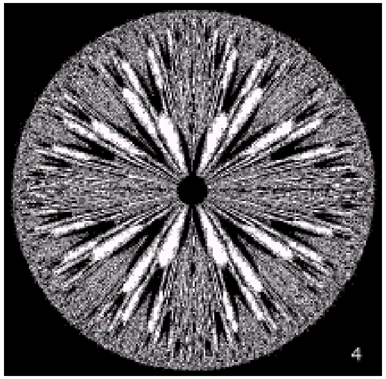


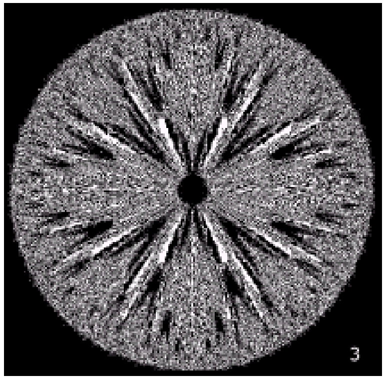


[01000000]

[00100000]

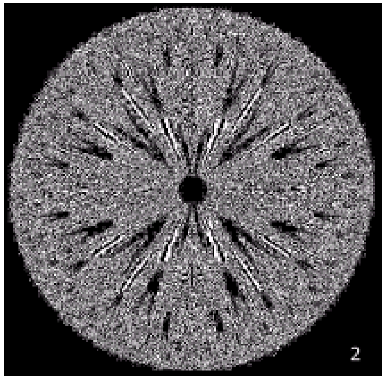


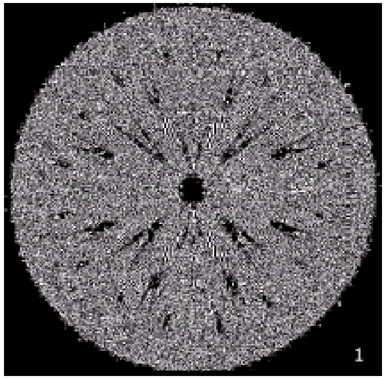


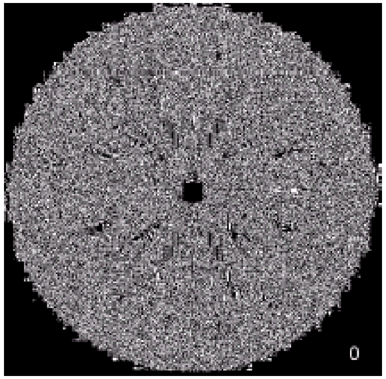


[00001000]

[00000100]







[00000001]

Bit Plane Slicing (cont…)

a b c
d e f
g h i

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

Bit Plane Slicing (cont…)

Original Image

Bit Plane Slicing (cont…)

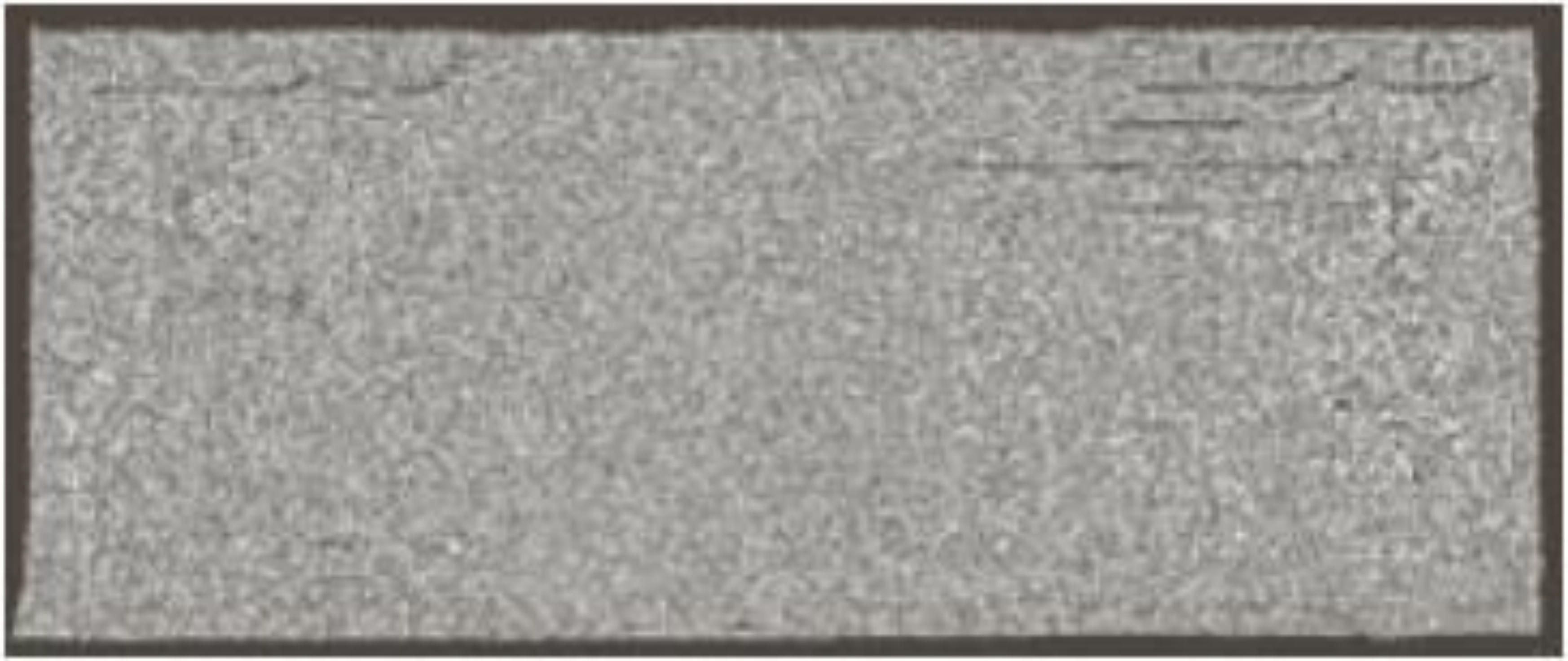
Bit Plane 1

Bit Plane Slicing (cont…)

Bit Plane 2

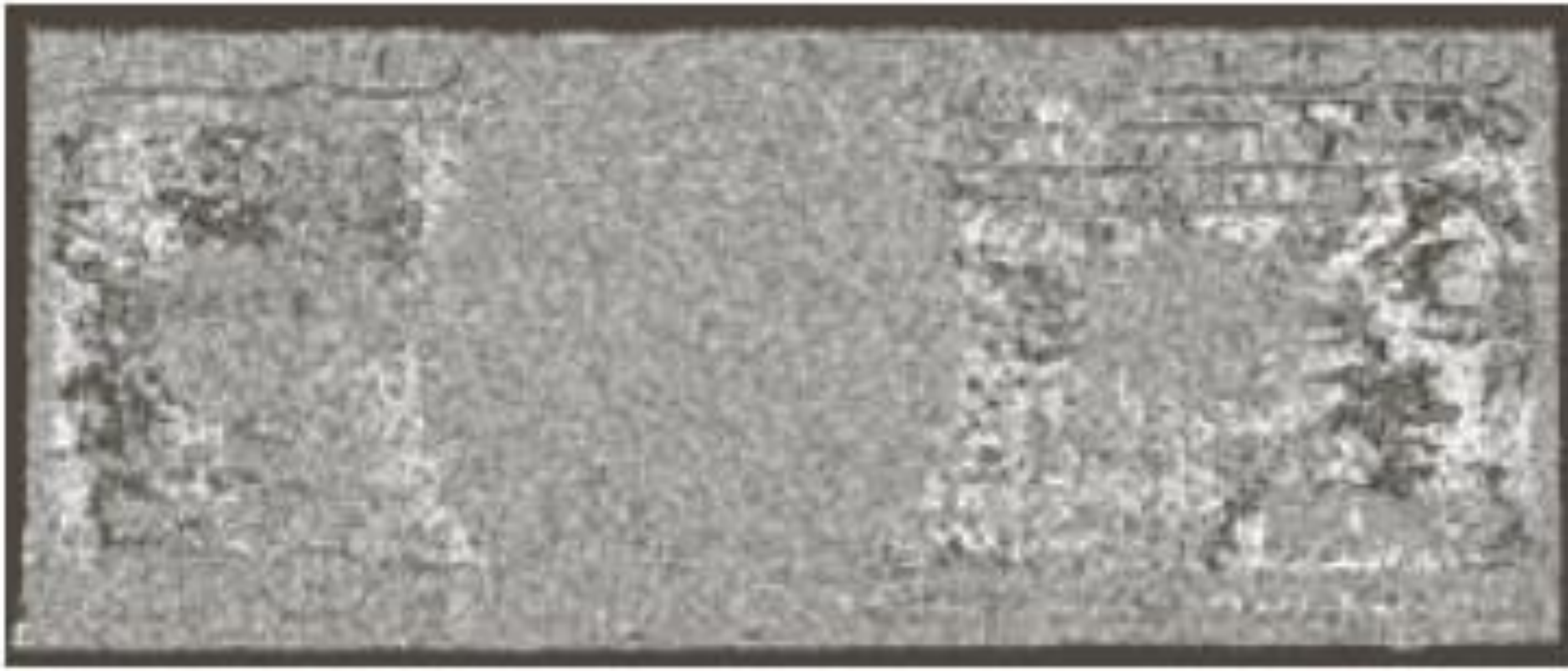
Bit Plane Slicing (cont…)

Bit Plane 3



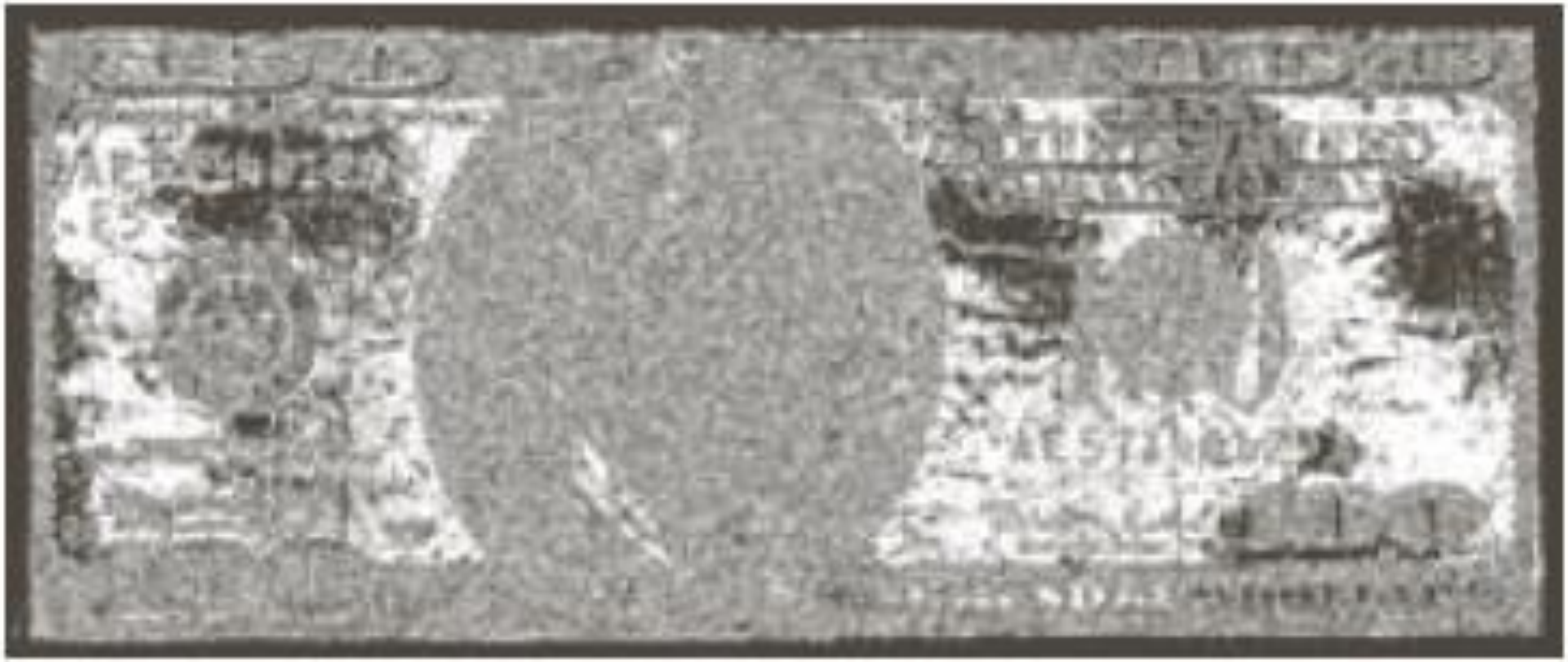
Bit Plane Slicing (cont…)

Bit Plane 4



Bit Plane Slicing (cont…)

Bit Plane 5



Bit Plane Slicing (cont…)

Bit Plane 6

Bit Plane Slicing (cont…)

Bit Plane 7

Bit Plane Slicing (cont…)

Bit Plane 8

Bit Plane Slicing (cont…)

Reconstructed image using only bit planes 8 and 7

Reconstructed image using only bit planes 8, 7 and 6

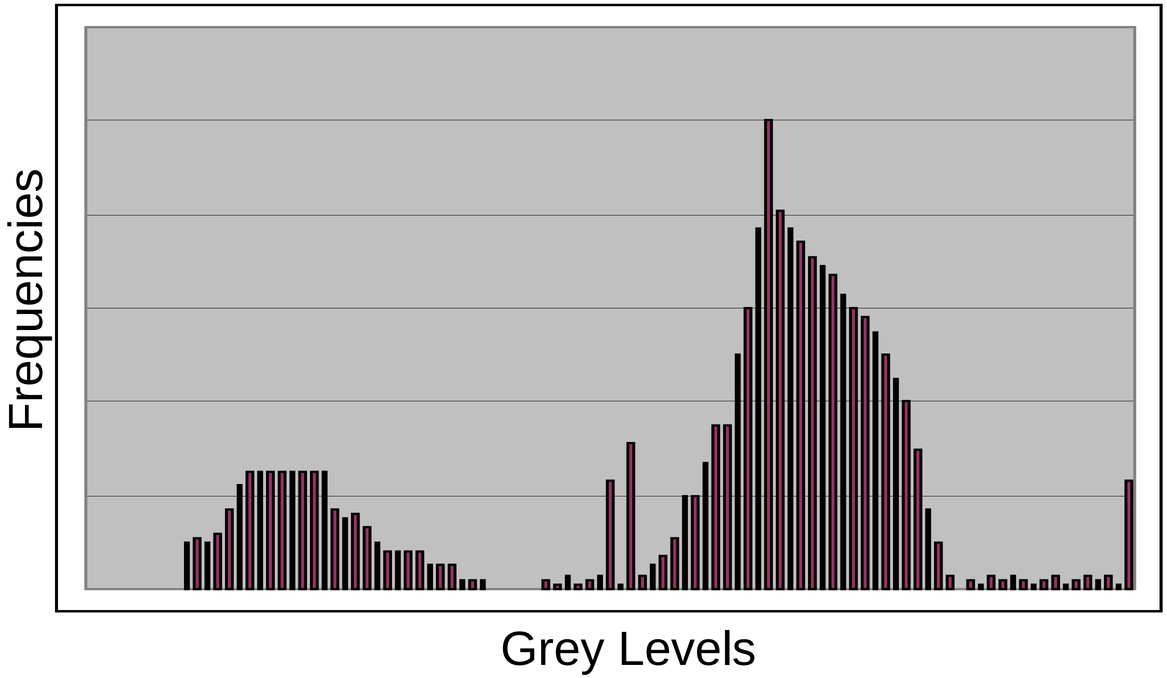
Reconstructed image using only bit planes 7, 6 and 5

Image Histograms

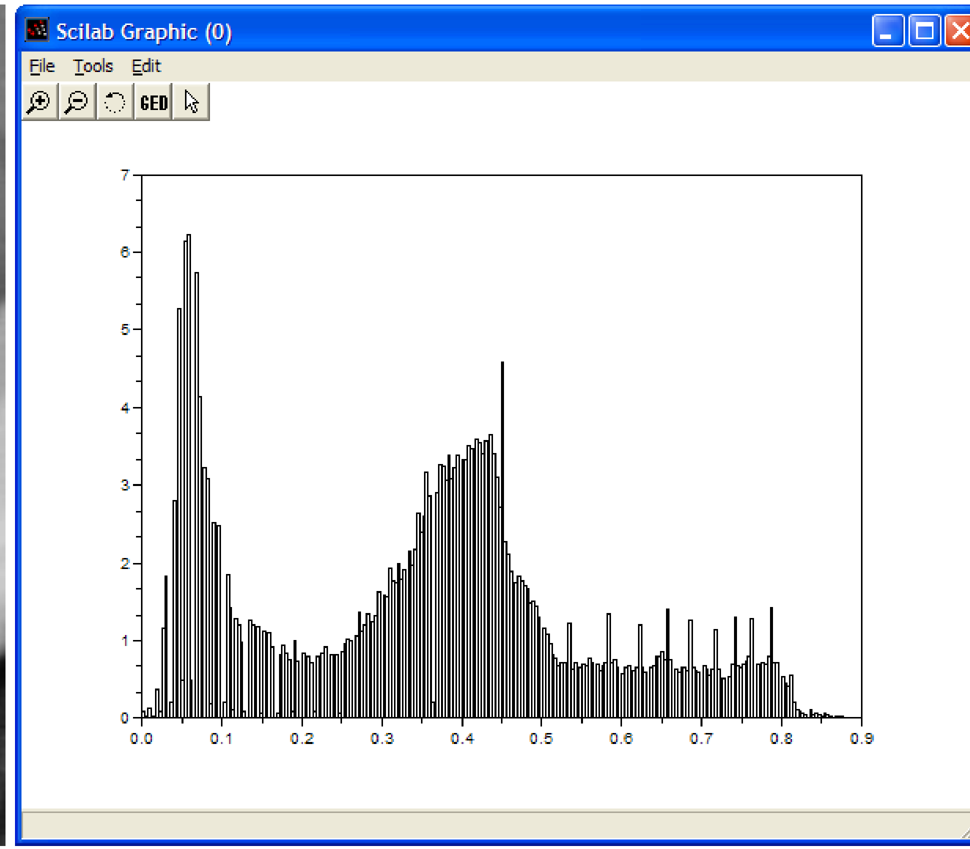
Recall that the histogram of an image shows us the distribution of grey levels in the image

Massively useful in image processing.

We will use it for image enhancement.



Histogram Examples (cont…)



Histogram Equalisation

Spreading out the frequencies in an image (or equalising the image) is a simple way to improve dark or washed out images

The formula for histogram equalisation is given where

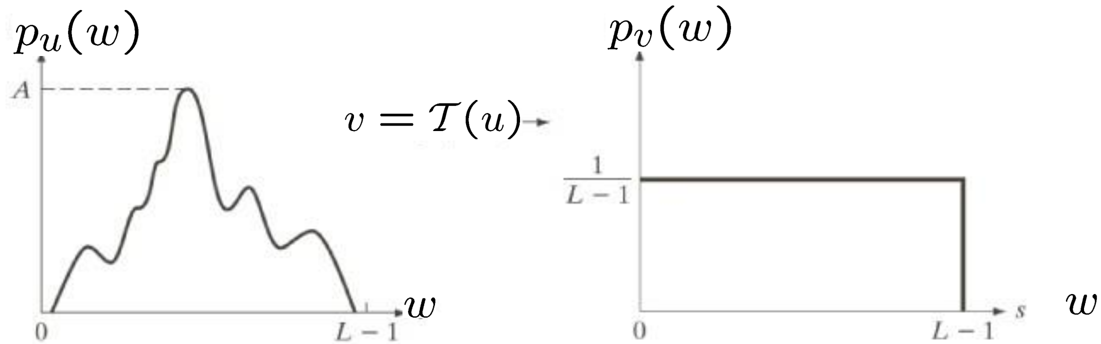
- r_k: input intensity
- s_k: processed intensity
- k: the intensity range (e.g 0.0 – 1.0)
- n_j: the frequency of intensity j
- n: the sum of all frequencies

$$
\begin{aligned}
s_k &= T(r_k) \\
&= \sum_{j=1}^{k} p_r(r_j) \\
&= \sum_{j=1}^{k} \frac{n_j}{n}
\end{aligned}
$$

Histogram Equalisation

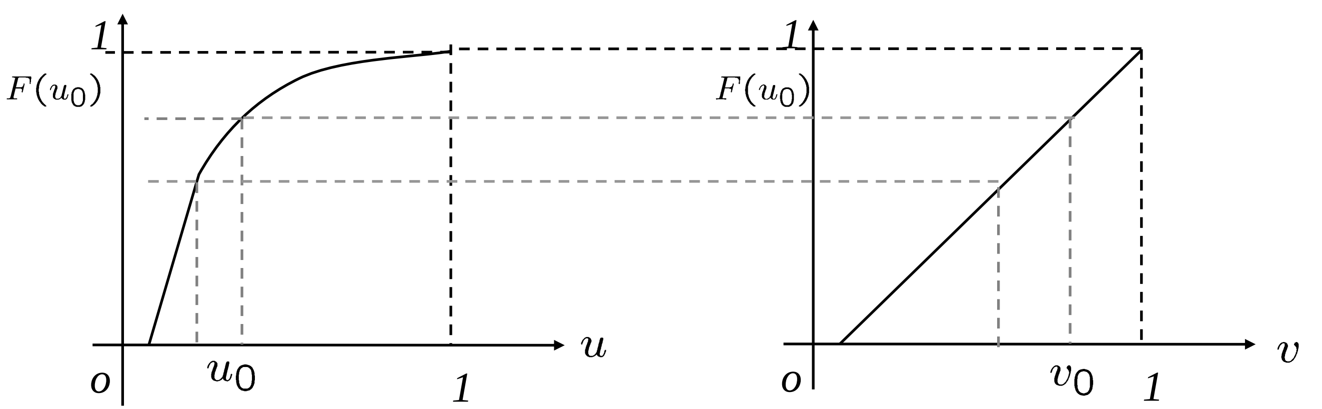
Goal: map the each luminance level to a new value such that the output image has approximately uniform distribution of gray levels

pdf



$$F_u(U) = P(U < u) = \int_0^u p_u(w)dw \qquad F_v(V) = P(V < v) = \int_0^v p_v(w)dw$$

cdf



Histogram Equalisation

PROOF(1-2):

**

We need ~~to~~ to prove that

if $s = T(r) = (L-1)\int_0^r p(r)\,dr$

transformation is applied on the input intensities then the resulting image will have an equalized histogram (normalized).

i.e.

$$p(s) = \frac{1}{L-1}$$

Histogram Equalisation

PROOF(2-2):

$$s = T(r) = (L-1)\int_0^r p(r)\,dr \longrightarrow \text{Transformation function.}$$

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

$$p(s) = p(r)\,\frac{dr}{ds} \longrightarrow (A)$$

$\frac{ds}{dr}$ = we can compute it.

$$\frac{d}{dr}T(r) = \frac{d}{dr}(L-1)\int_0^r p(r)\,dr.$$

$$= (L-1)\,p(r) \qquad \left(\frac{d}{dx}\int_0^x f(x)\,dx = f(x)\right)$$

$$\frac{ds}{dr} = p(r)(L-1) \longrightarrow (B)$$

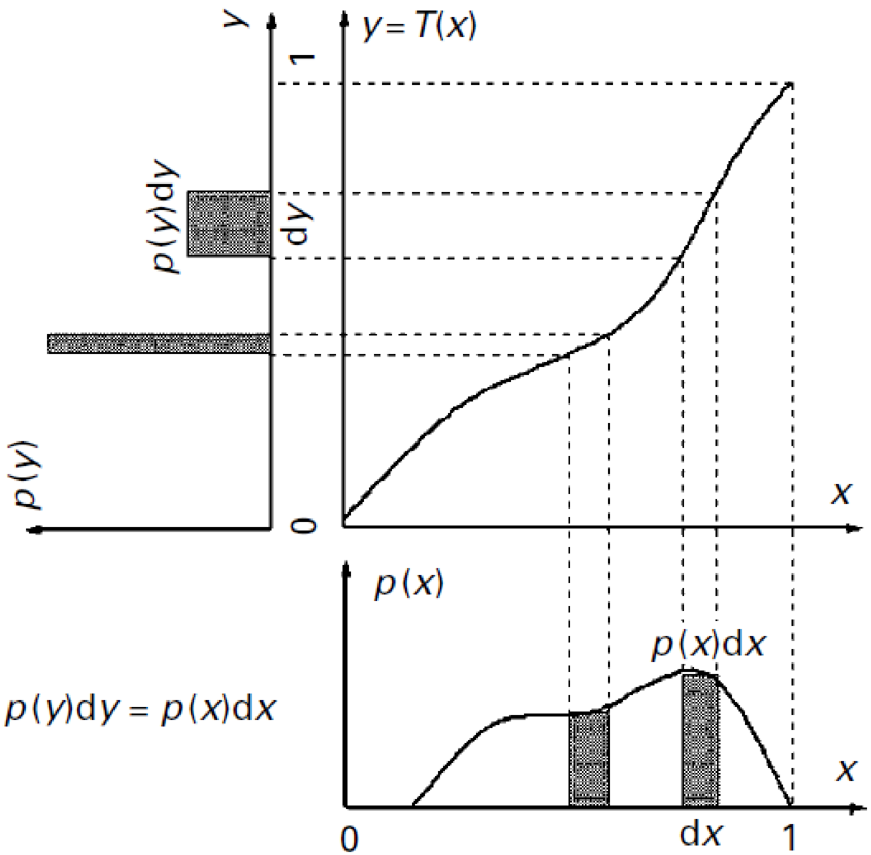
put in eq. (A)

$$p(s) = p(r)\cdot\frac{1}{(L-1)\,p_r}$$

$$p(s) = \frac{1}{(L-1)} \longrightarrow (C)$$

Equation (C), states that pdf of an equalized

Histogram Equalisation



PDF of the input image p(x) and output image p(y) will have the same mass.
i.e.,
p(y)dy=p(x)dx

Figure 5.12 The transfer function, $y = T(x)$, determines how the probability density function $p(x)$ is transformed into the probability density function $p(y)$.

Histogram Equalisation

r_k	n_k	$p_r(r_k) = n_k/MN$	$7 * \sum_u p(u)$	v
$r_0 = 0$	790	0.19	1.33	1
$r_1 = 1$	1023	0.25	3.08	3
$r_2 = 2$	850	0.21	4.55	5
$r_3 = 3$	656	0.16	5.67	6
$r_4 = 4$	329	0.08	6.23	6
$r_5 = 5$	245	0.06	6.65	7
$r_6 = 6$	122	0.03	6.86	7
$r_7 = 7$	81	0.02	7.00	7

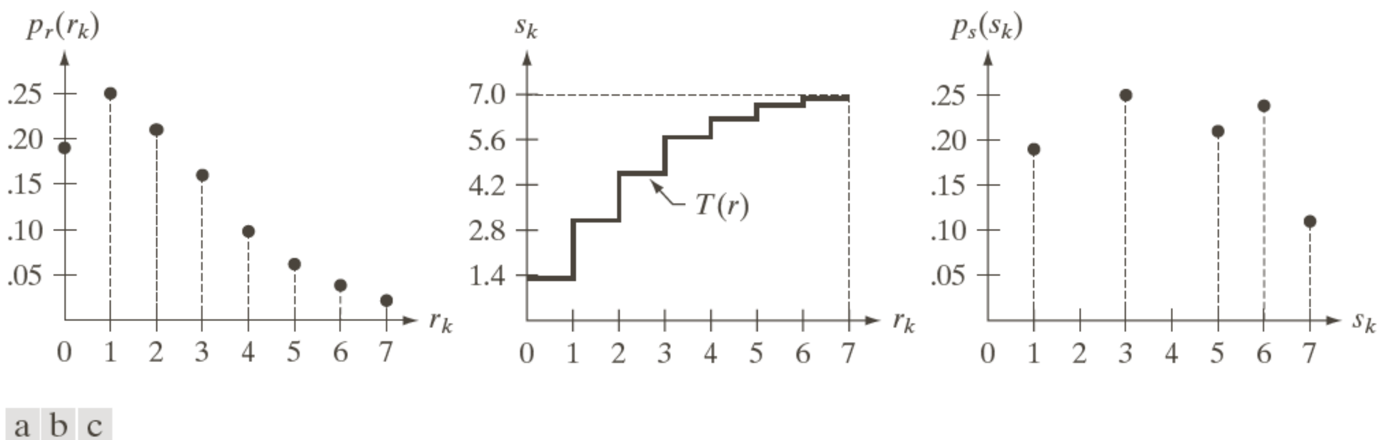
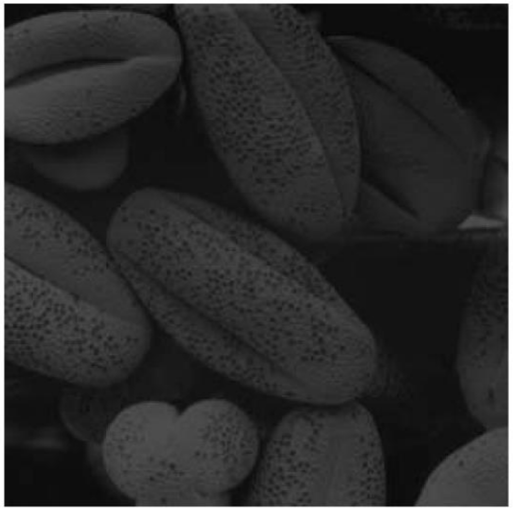
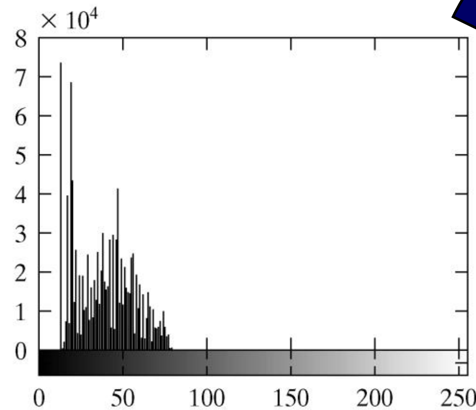
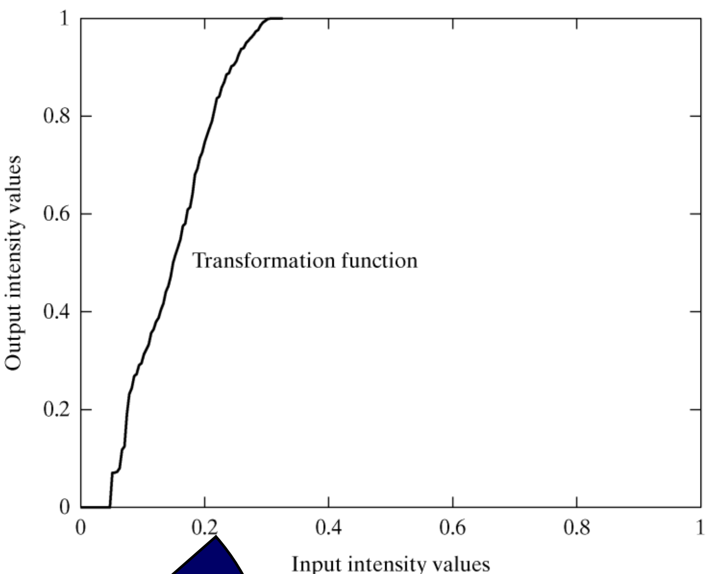


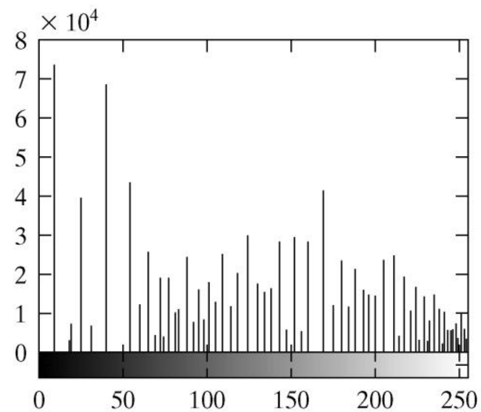
FIGURE 3.19 Illustration of histogram equalization of a 3-bit (8 intensity levels) image. (a) Original histogram. (b) Transformation function. (c) Equalized histogram.

Equalisation Transformation Function



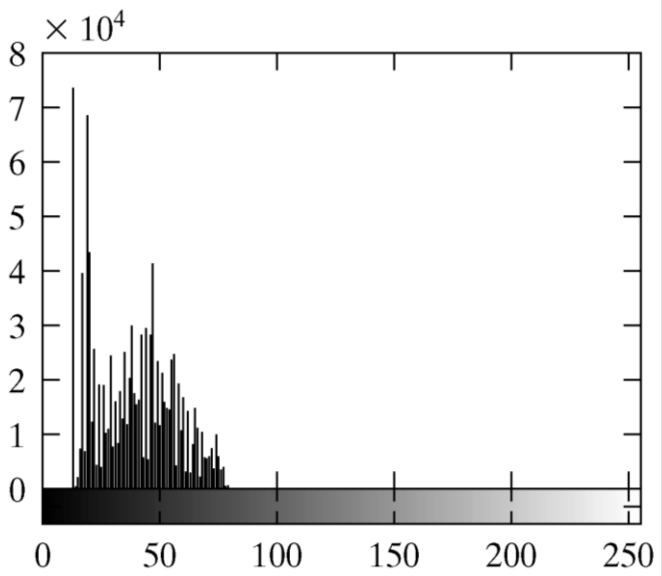


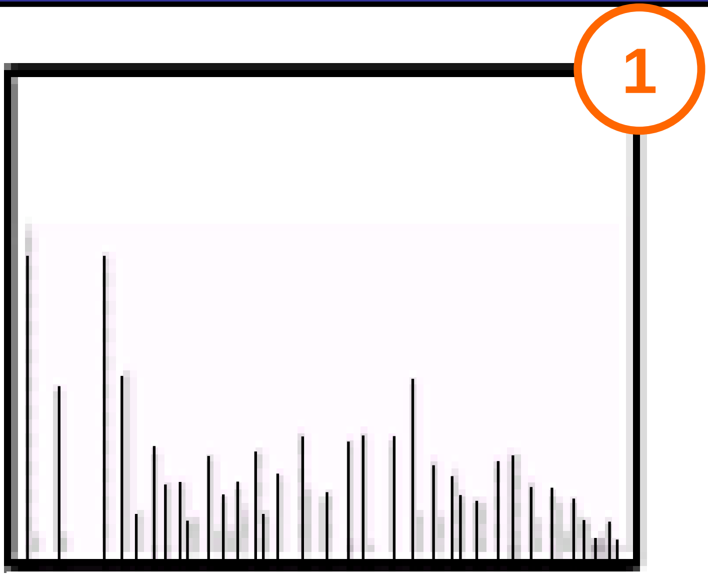


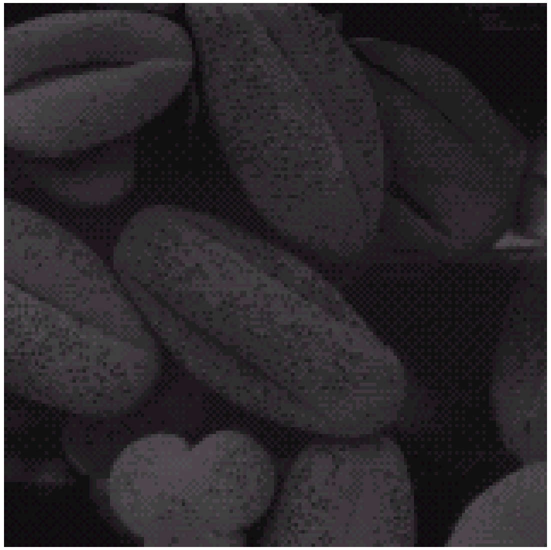


Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Equalisation Examples

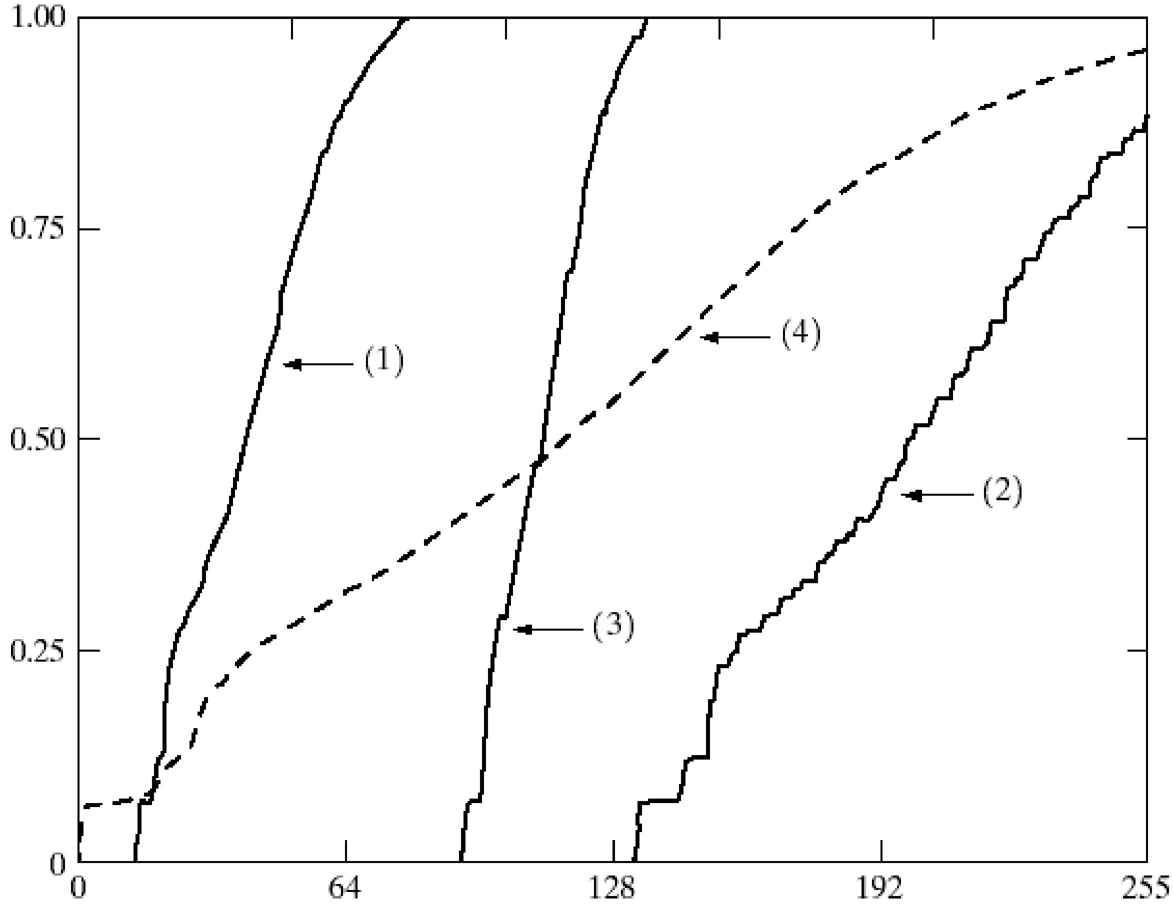




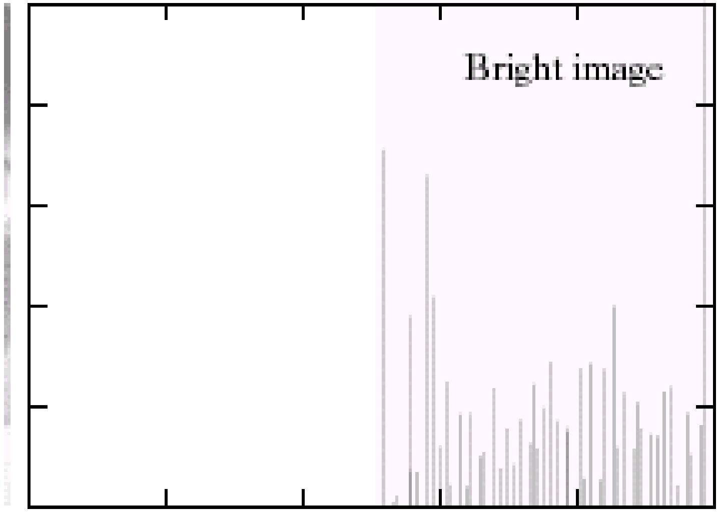


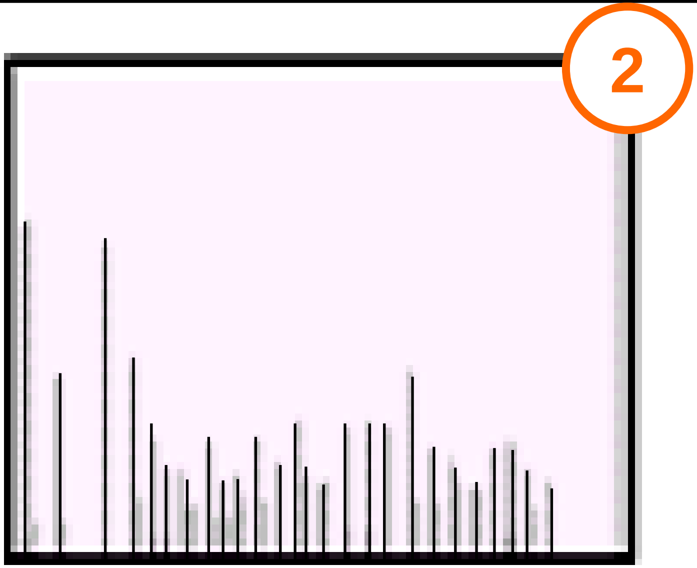
Equalisation Transformation Functions

The functions used to equalise the images in the previous example



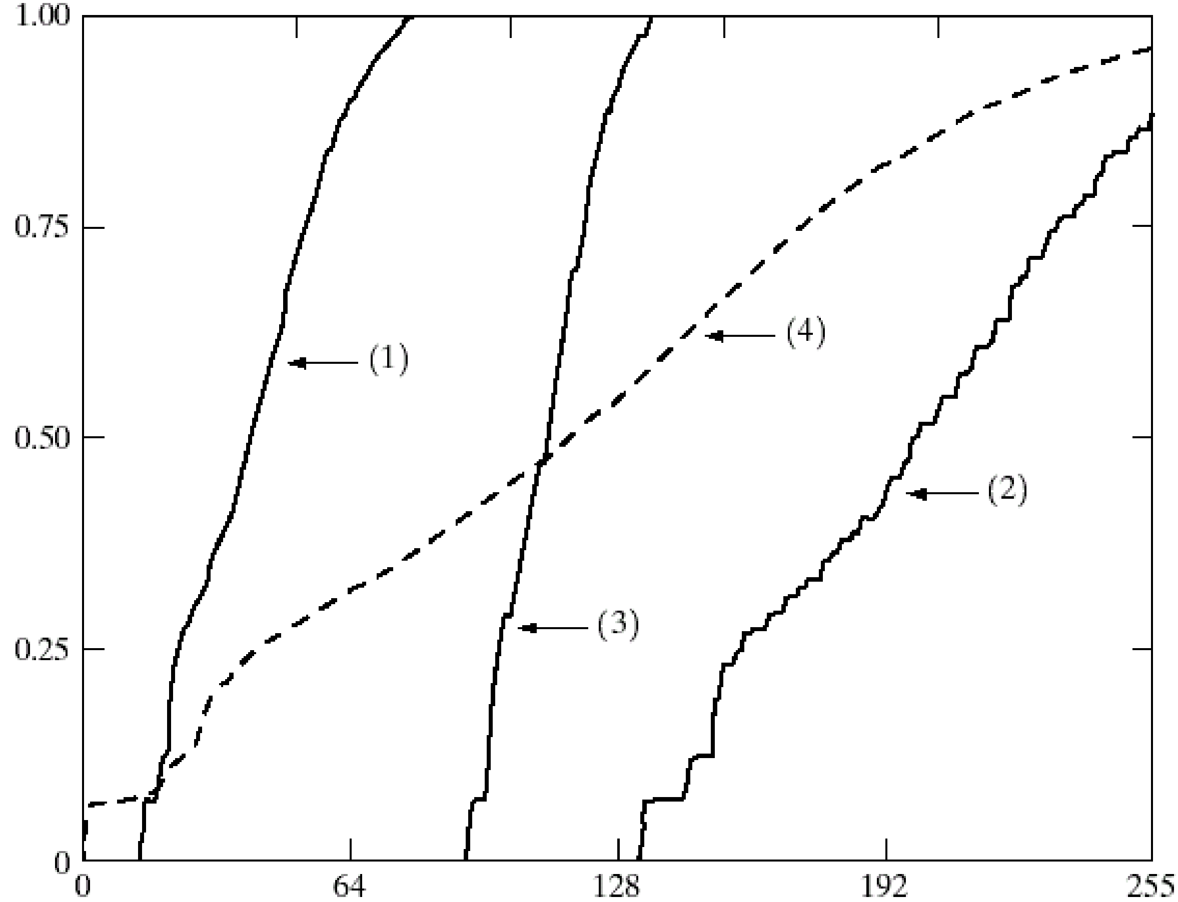
Equalisation Examples



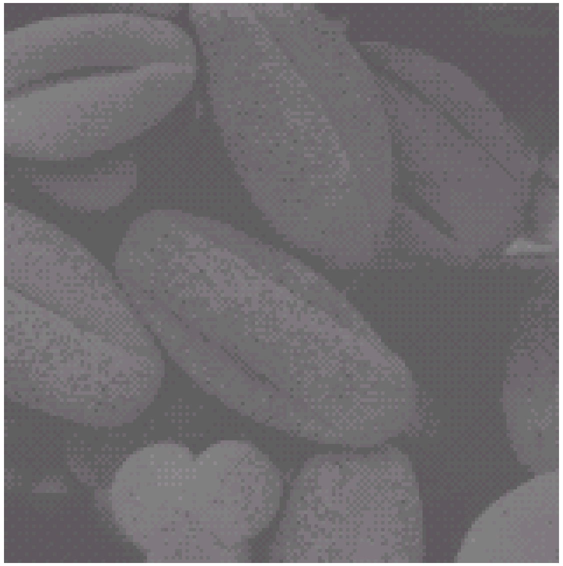


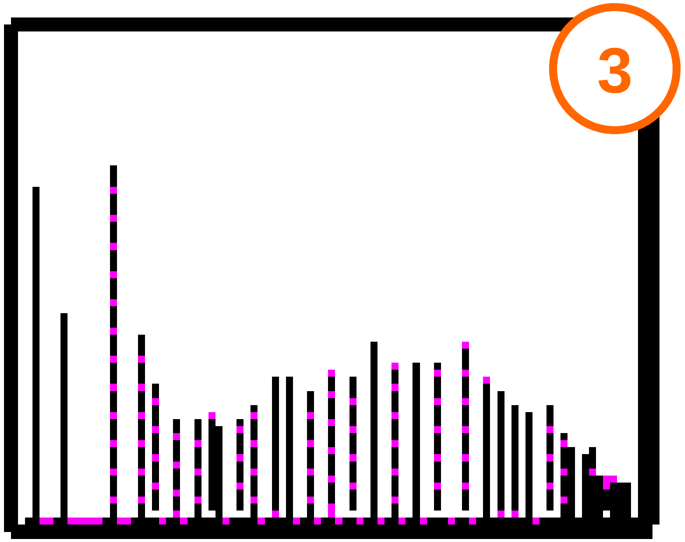
Equalisation Transformation Functions

The functions used to equalise the images in the previous example

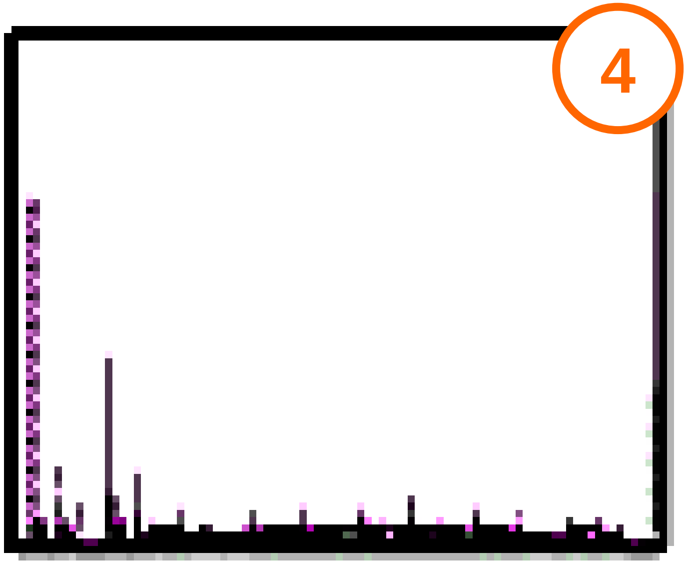


Equalisation Examples (cont…)





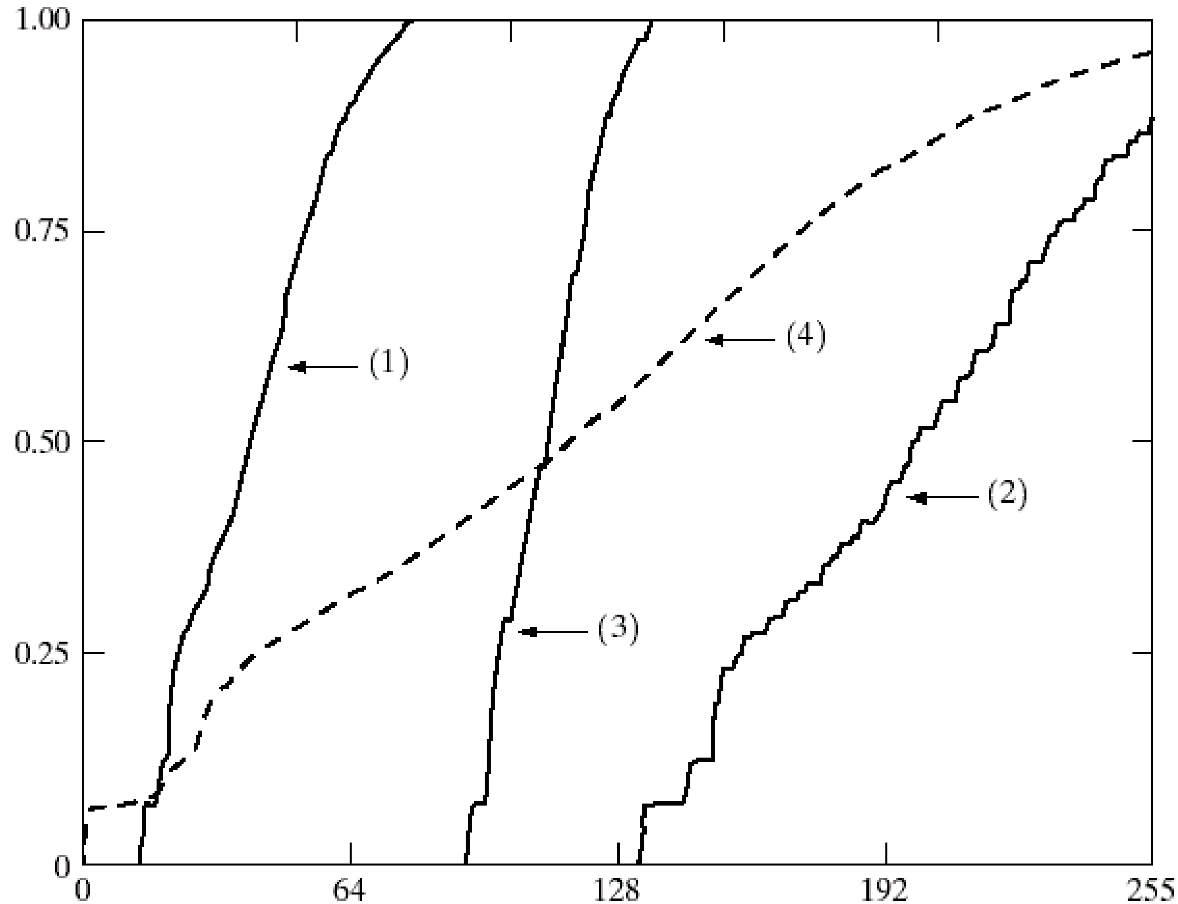




Images taken from Gonzalez & Woods, Digital Image Processing (2002)

Equalisation Transformation Functions

The functions used to equalise the images in the previous examples



Difference B/W CS and HE

Contrast stretching is all about **increasing the contrast** i.e., increasing the difference between the maximum intensity value in an image and the minimum one. All the rest of the intensity values are spread out between this range.

Histogram equalization is about **modifying the intensity values** of all the pixels in the image such that the **histogram is "flattened**" (in reality, the histogram can't be exactly flattened, there would be some peaks and some valleys, but that's a practical problem).

Difference B/W CS and HE

In **contrast stretching** there exists a one-to-one relationship of the intensity values between the source image and the target image i.e., the original image can be restored from the contrast-stretched image. Hence, the **process is reversible**.

Once **histogram equalization** is performed, there is no way of getting back the original image i.e., histogram equalization is an **irreversible** operation.

Exercise Histogram Equalization

Equalize the histogram of the following image:

5	**1**	**3**	**0**	**1**
7	**3**	**2**	**7**	**0**
1	**2**	**1**	**6**	**1**
7	**0**	**5**	**0**	**3**
1	**4**	**3**	**7**	**2**

Summary

We have looked at:

- Different kinds of image enhancement
- Spatial Enhancement Methods
  - Point Processing
  - Histogram equalisation

Next time we will start to look at some **neighbourhood based** image enhancement.