Imperial College
London

# 4 – Intensity Transformation

Prof Peter YK Cheung

Dyson School of Design Engineering

URL: www.ee.ic.ac.uk/pcheung/teaching/DE4_DVS/
E-mail: p.cheung@imperial.ac.uk



In the previous two lectures, we have been considering how human capture visual information and send them to the brain. This lecture is the first of several that consider how visual information can be processed by computers for the purpose of enhancement, restoration, reconstruction and interpretation.

This lecture is based on the materials found in the first half of Chapter 3 of the textbook, “Digital Image Processing”, 4th Edition, by RC Gonzalez and RE Woods.

## Spatial Domain Processing

- Visual data is captured and stored in computer
- Only using intensity (or luminous) information
- Scene is sampled at regular spatial grid in X and Y direction
- General equation of processing is:

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

  - $f(x,y)$ is the captured visual information at (x,y)
  - $T[.]$ is the operator defined over a neighborhood of point ( x, y)
  - $g(x,y)$ is the output image at (x,y)



Visual data are captured using image sensor device such as conventional camera, infra-red camera, x-ray imager, Scanning Electron Microscope sensor or MRI reconstructed system. They are two-dimensional data in a regular grid. These data can be represented as a M x N x B matrix, where B is the number of components (or channels) that form the image. Each element within the matrix, data are often stored as unsigned 8-bit integers (uint8), unsigned 16-bit intergers (uint16) or double precision 64-bit floating points (double).

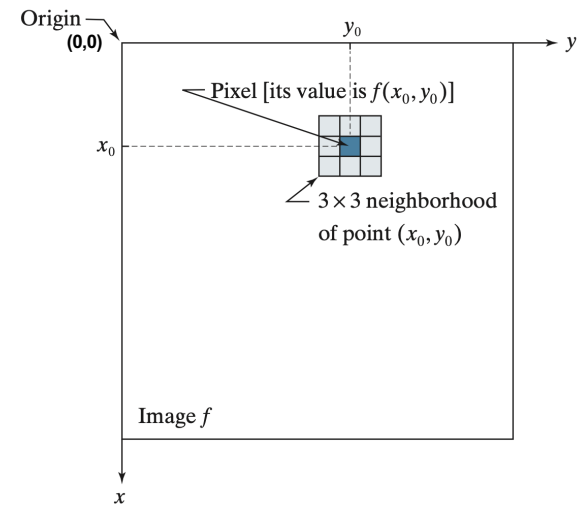
For grayscale image, B = 1; for RGB, XYZ or HSV colour images, B = 3. For this Lecture, we will consider intensity or grayscale images only.

In this lecture, we only deal with each pixel intensity individually, independent of its neighbour. f(x,y) is the input image and g(x,y) is the output image. T is the operation applied pixel-by-pixel to the input image to produce the output image.

In the next lecture, T operates over the neighbourhood of the pixel f(x,y) to derive the outpout g(x,y).

## Neighbourhood operation

Matlab origin – (1,1)




- $(x_0, y_0)$ – arbitrary location in image
- Neighbourhood is a rectangle centred on $(x_0, y_0)$
- This shows a 3x3 neighbourhood
- Operator T is applied to the pixel values of neighbouring pixel surrounding $f(x_0, y_0)$
- The result of T is output image value $g(x_0, y_0)$
- Processing the entire image requires such an operation performed over the entire image, pixel-by-pixel, starting from the origin

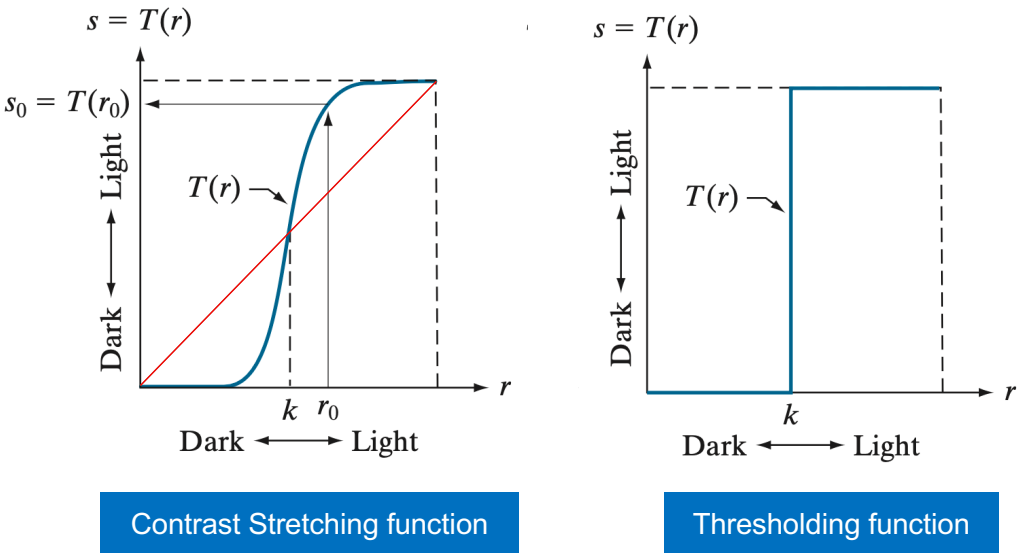


In an image with M x N pixels, the origina is in the top left corner normally referred to as the coordinate (0, 0). However, for Matlab, this is location (1, 1) because Matlab matrix indices start from 1 and not 0.

Consider an aribitrary pixel at location ($x_0$, $y_0$). Its immediate neighbours are the 8 pixels surranding ($x_0$, $y_0$). The operator T could be derived purely from f ($x_0$, $y_0$), as for this Lecture, or from all its immediate neighbours, or even go beyond to the neighbours of the immediate neighbours. This is called **neighbourhood processing**. The operator T will produce the output value for g($x_0$, $y_0$).

Repeating the T operator on EVERY PIXEL in the image produces the entire output image.

## Examples of Operator




Contrast Stretching function

Thresholding function

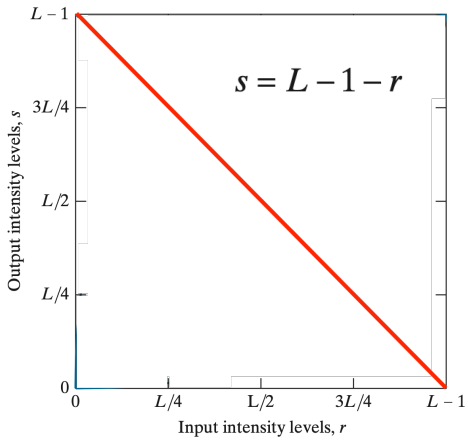


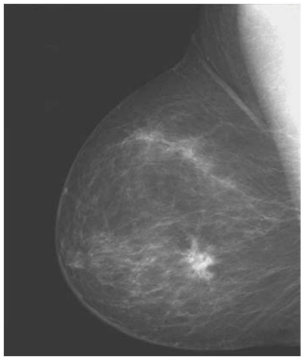
In this example, the input image intensity at f ($x_0$, $y_0$) is $r_0$. The operator T is depicted in the "transfer" function (mapping input value to output value) as shown in the two graphs above.

The red line shown on the left plot is the unity transfer function where $s = T(r) = r$ – i.e. output is the same as input and no change is made.

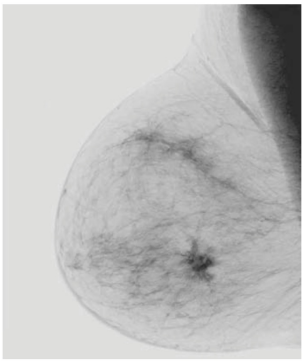
The left function stretches the contrast in the mid-region around the input intensity value *k*. The right function is a thresholding function. Any pixel intensitive above *k* is made maximum intensity (white) while below is made minimum (black).

## Negative Transformation



Original Image



Negative Image

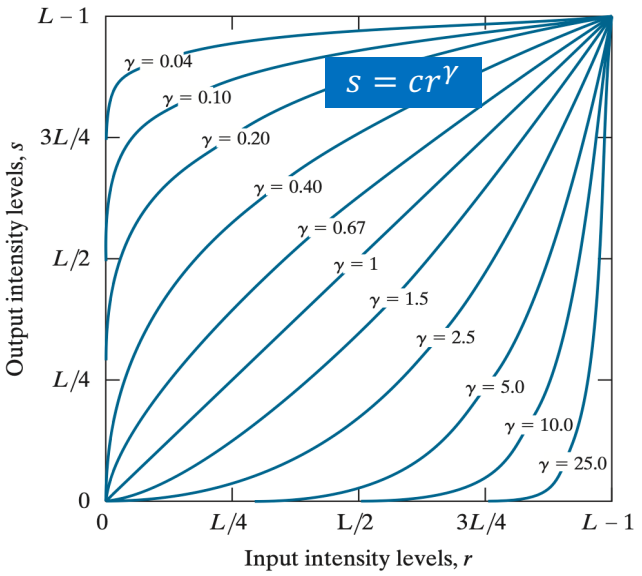
- r is the original pixel value in range [0, L-1]
- s is the resulting pixel value after negative operator



Sometimes, it is helpful to produce a negative image. Assuming the intensity values is between 0 to L-1 (i.e. there are L intenstive levels), the the transformation is simply $L-1-r$ where r is the intensity at f ($x_0$, $y_0$).

Here is an example of the orginal image of a mammory photo. The negative image shows the lesion much clearer.

## Power-law Transformation (Gamma Correction)

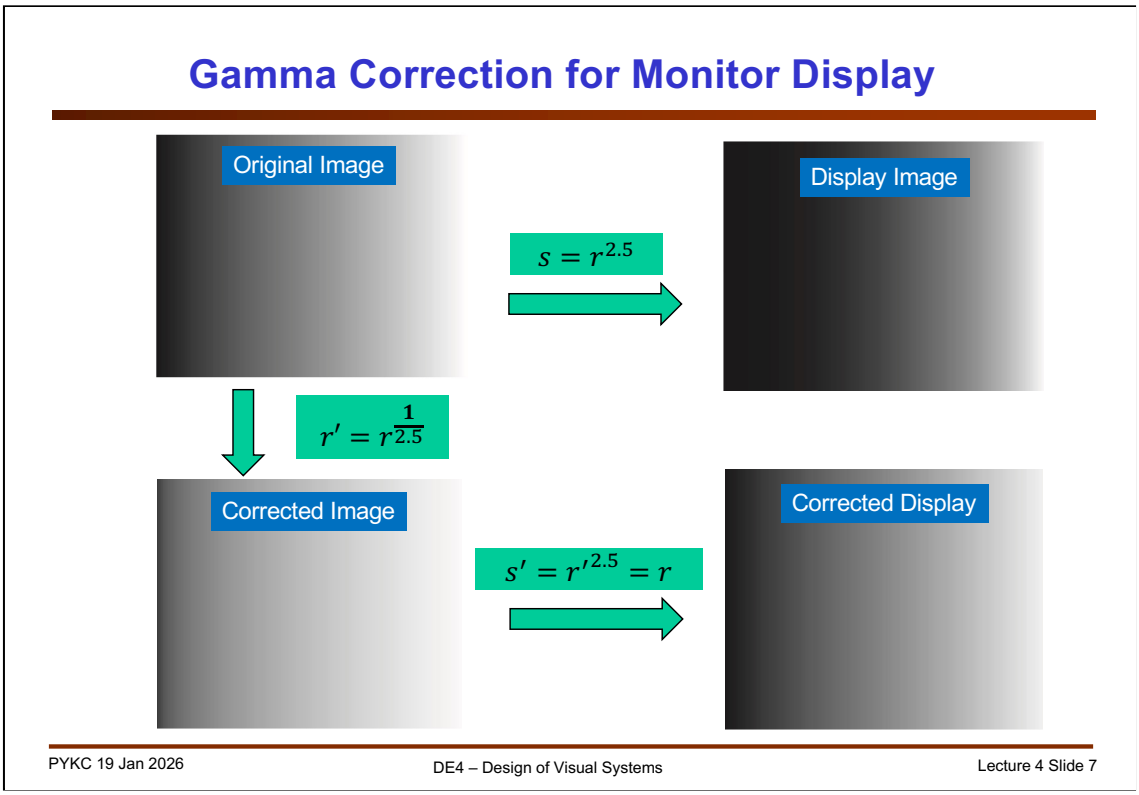



- $c$ and $\gamma$ are positive constants
- Better known as Gamma Correction
- Assume c = 1
- Used for processing an image for display on monitors
- Different values of $\gamma$ produce a family of functions
- $\gamma = 1$ means image is not changed
- $\gamma > 1$ produces in darker image
- $\gamma < 1$ produces a ligher image
- $s = r^{\gamma}$ and $s = r^{\frac{1}{\gamma}}$ are inverse of each other



How our eyes see light intensity is very different from how a display reproducing that intensity value. Therefore, when reproducing an image on a display so that we see the same level of contrast as the original image requires correction. This is called "**Gamma Correction**".

The Gamma function is given by: $s = cr^{\gamma}$, where c and $\gamma$ are both constants. c is the gain value and will adust the overall brightness of the display. $\gamma$ is chosen to perform the correction. If $\gamma$ is 1, then nothing changes. If $\gamma > 1$, a darker image is produced. If $\gamma < 1$, a lighter image is produced.




Here is an example illustrate the effect of Gamma correction.

If an image is sent to a monitor which changes what is displayed according to the function $s = r^{2.5}$, one would see a much darker image.

However, if we make the correction first by apply the INVERSE of the monitor gamma function, pre-distorting the pixel values according to the function $s = r^{1/2.5}$, combining with the monitor's own distortion, what is produced on the display is now the original image with the original contrast level.

## Power-law Contrast Enhancement (1)

$s = r^{\gamma}$

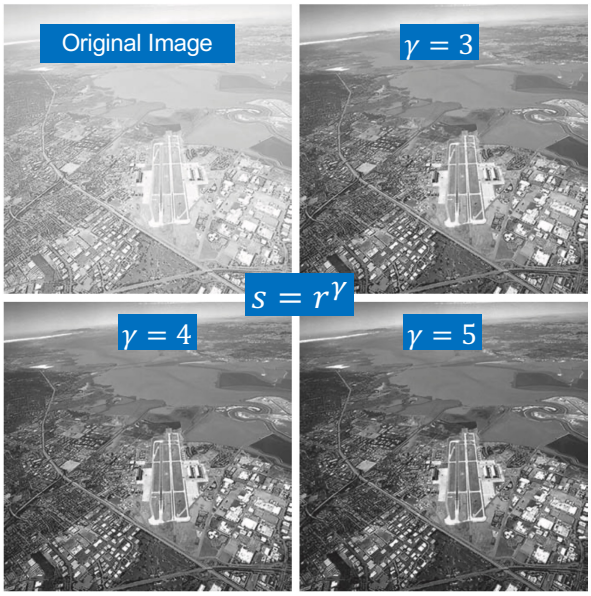
- $\gamma < 1$ lightens the image and shows break in spine

Original Image | $\gamma = 0.6$ | $\gamma = 0.3$



Here is an example to show the effect of varing the value of $\gamma$. This is an X-ray image of a patient's spine with a fracture. The original image has such a contrast that the break is hardly visible. However, if it is Gamma corrected with $\gamma = 0.3$, the break is clearly visible.

## Power-law Contrast Enhancement (2)

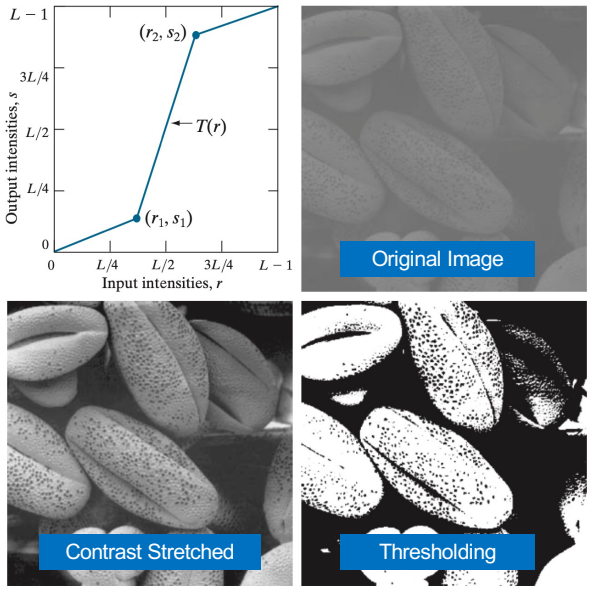



- Original image over exposed
- $\gamma > 1$ darkens the image



Here is another example of using power-law enhancement to improve an over-expose arial photograph of an airport scene.

## Piecewise Linear Transformation (Contrast Stretching)



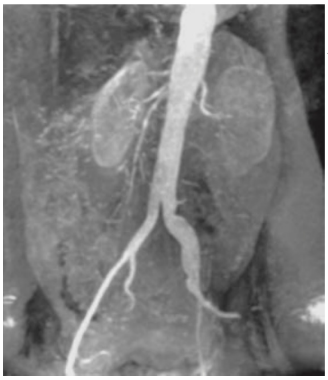

- Transformation function made up of segments of lines
- Gradient of line segments always positive
- Not a mathematical function
- Lighten and darken different grayscale regions
- E.g. low contrast electron microscope image of pollens enlarged by x700
- After contrast stretching using the piecewise linear transformation
- After thresholding only

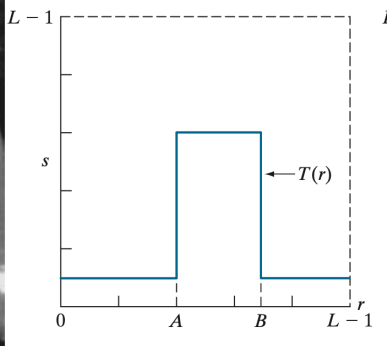


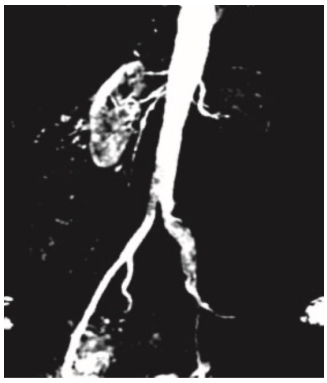
In power-law correction, the relationship between output intensity and input intensity (the transfer function T) is expressed as an exponential mathematical function. This can be limiting. An alternative approach is to construct the function as several straight-line segments – a **piecewise linear function** as shown above.

This approach allows ANY function to be approximated. Here is an example of using such an approach to stretch the constrast in the intensity in the middle third range from $r_1$ to $r_2$ while de-emphasizing the first and last third range. As you can see, this produces a much better image of the microscope image of pollens.

## Piecewise Linear Transformation (Intensity-level slicing) (1)





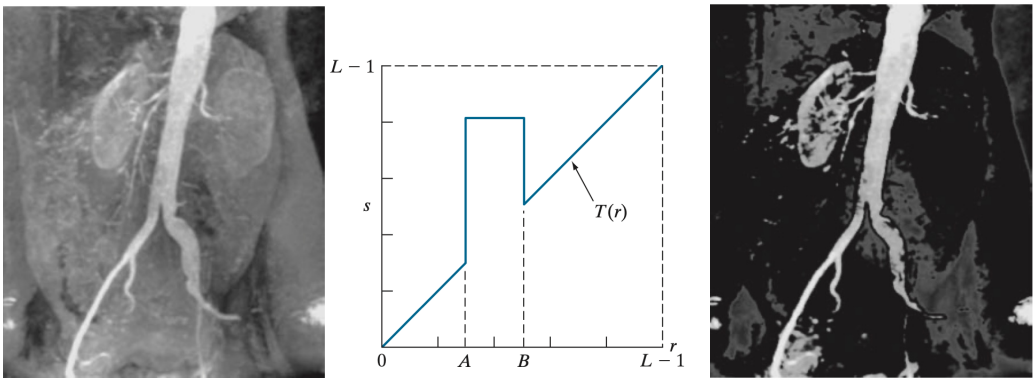



- Similar to thresholding, but use two thresholds instead of one
- Result: highlight region of image with intensity between A and B
- E.g. Aortic angiogram image with interesting feature having intensity with defined region
- Only the arteries and part of heart highlighted



Here is another piecewise linear approach to enhancement. The image is that of a aortic angiogram. The feature of interest produces intensity in the range between A and B. This is a selective thresholding method known as intensity-level slicing. It extract the feature of interest and set all other pixels to black. This of course destroys the remaining image except the extracted feature.

## Piecewise Linear Transformation (Intensity-level slicing) (2)

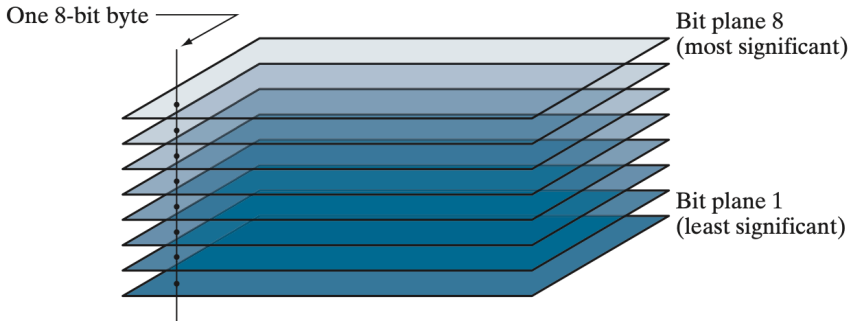



- Similar to the previous, but preserve original image intensity except for the region with intensity between A and B
- Resulting image shows features of other part of body, but still highlighting the region of interest.
- Effectively superimposing the highlighted part onto the original image



An alternative to the slicing approach is to keep the 45-degree line in the function T, but emphasize the feature of interest by mapping the pixels with intensities between A and B white. This extracts the features of interest while in the context of the rest of the image.

## Bit-plane slicing



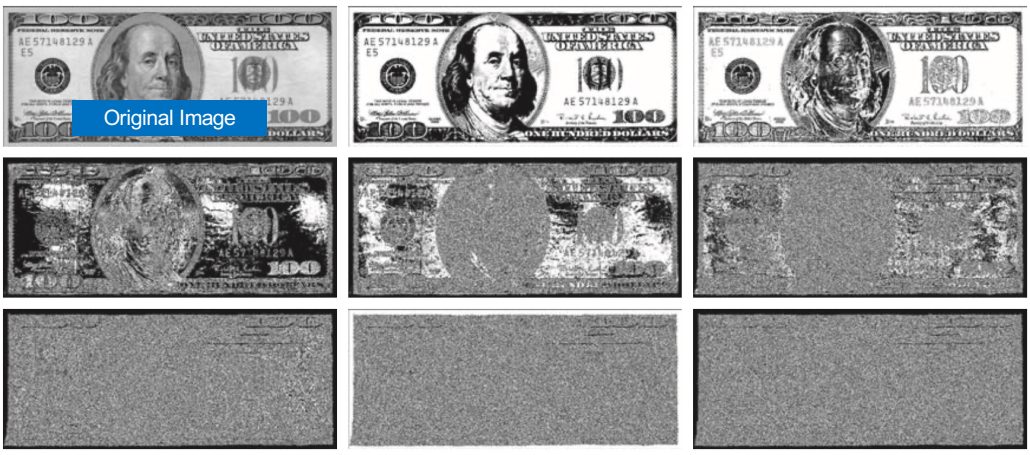

- Intensity is 8-bit number
- Slice each pixel intensity values into images of EACH BIT
- Produces 8 separate images – what for?



Pixels are stored in binary numbers as bits. For an 8-bit intensity value, there are 8 separate bits from bit 1 to bit 8.

Bit-plane slicing is to produce from the original image, 8 separate images, one for each bit of the intensity value.

## Bit-plane slicing of a US bank note (1)




- Slicing original image of bank note shown in top left
- Clearly the most-significant bit image is most important (not surprising)
- Also shows that image of the bottom 4-bits contain little information



The purpose of this is to figure out which bits you may drop without degrading the image too much. Here is an example of a 8-bit intensity image of a US $100 bank note. It is clear that the bit-plane 8 is the most important. Also, the last four bit-plane images show that they add little to the overall image.

## Bit-plane slicing of a US bank note (2)

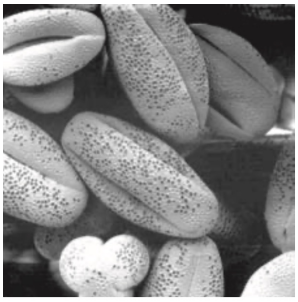
Original with 8-bit intensity

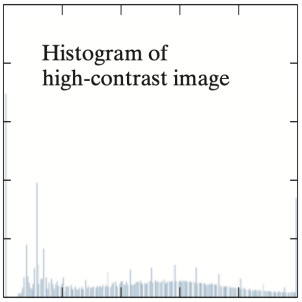
With only most-significant 4 bits



Consequently, we may choose to store each pixel as 4-bit instead of 8-bit numbers. Here is the results of retaining only the four most-significant bits of the intensity value. You can hardly tell any difference between this and the original.

## Formal Definition of Histogram






Let $r_k$, for $k = 0,1,2,..., L-1$, denote the intensities of an $L$-level digital image, $f(x, y)$.

The **unnormalized histogram** of $f$ is defined as:

$$h(r_k) = n_k \quad \text{for } k = 0,1,2,\ldots,L-1$$

$n_k$ is the number of pixels in $f$ with intensity $r_k$.

The subdivisions of the intensity scale are called **histogram bins**.

The **normalized histogram** of image $f$ of dimension (MxN)is defined as:

$$p(r_k) = \frac{h(r_k)}{MN} = \frac{n_k}{MN}$$



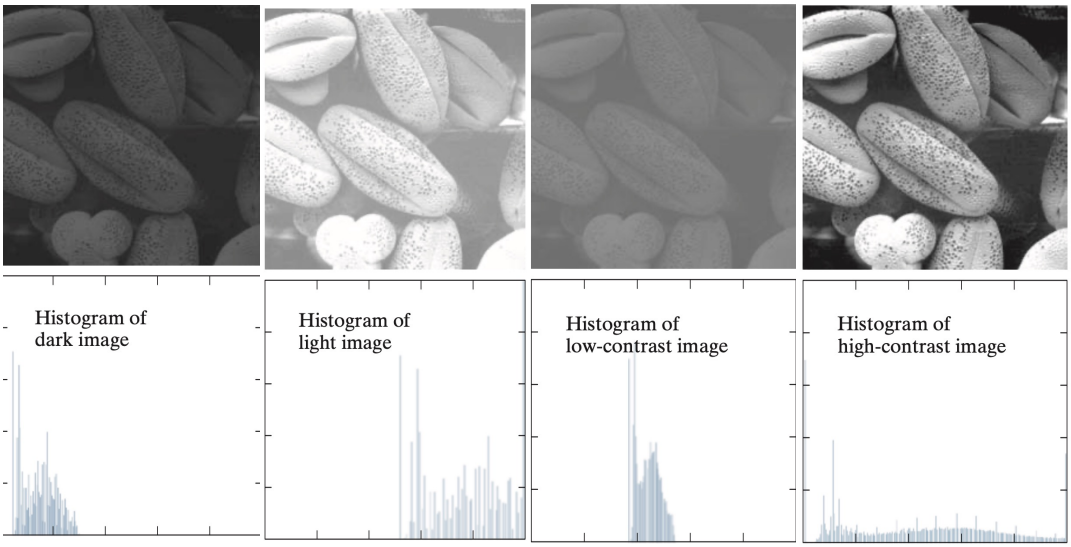
We will now move to another approach where we modify the pixel intensity DEPENDING on the image that we are processing. This allows us to adapt the transformation function T to the image at hand.

A common technical is called **histogram equalization**.

Here is the definition of the histogram of an image, which is simply the count of each intensity value and put them into bins.

We also often normalize the histogram by dividing the count by the total number of pixel. This provides the probability of a given intensity being found in the image. Here the image is of dimension M x N.

## Histogram affects contrast

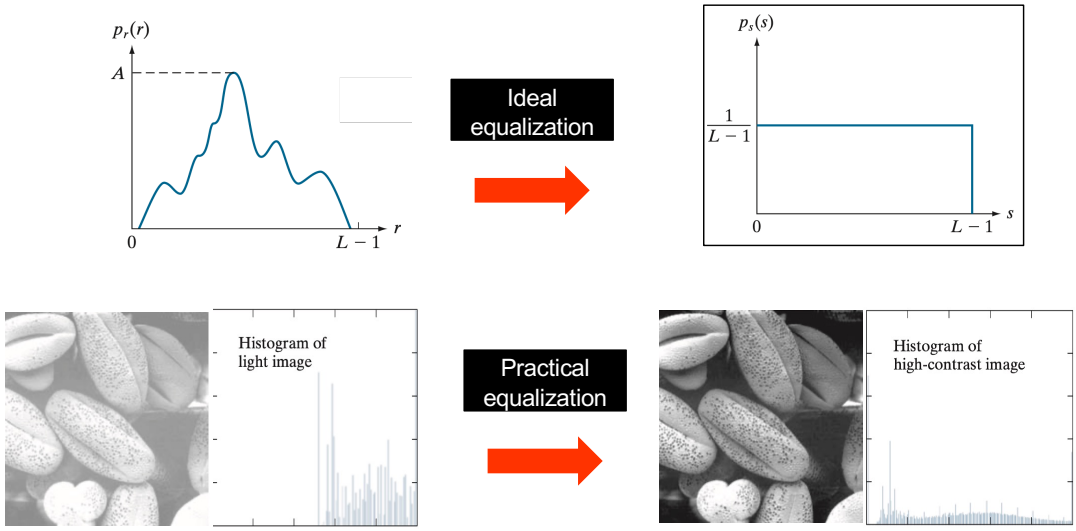





This slides shows four microscopic images of pollens with different contrast levels. The right-most image is the best because the image make use of the entire intensity range while the other images intensity values are clustered toward light or dark.

Therefore, it is desirable in many cases to find T such that the histogram is nearly flat across the entire intensity range, as in the right-most image. This is known as **Histogram Equalization**.

## Histogram Equalization

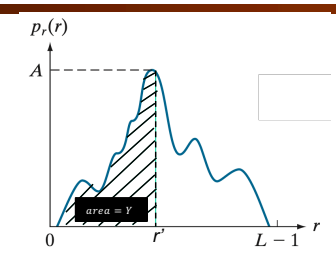





Ideally, the equalized histogram should be completely flat. This is generally not possible for a finite image in the discrete domain. However, one could achieve something close to this ideal.

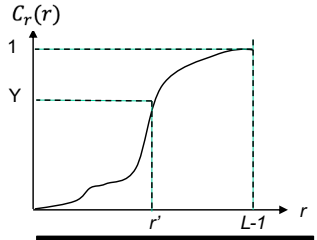
In the example shown here. The histogram equalized image has a histogram that is more or less flat.

## PDF and CDF




Probability Distribution Function PDF

Integrate = area under curve




Type equation here.Cumulative Distribution Function

$$C_r(r) = (L-1)\int_0^r p_r(w)dw$$

- Histogram equalization (i.e. flattening of the PDF) can be achieved by using the CDF as the intensity transformation function

$$s = T(r) = (L-1)\int_0^r p_r(w)dw$$



How can this be achieved? To do so, one has to understand the concept of **cumulative distribution function** (CDF).

A normalized histogram is a discrete form of the **probability distribution function** (PDF). It is so-called because if one wants to find out the probability of any intensity falling between A and B of an image with a PDF function $p_r(r)$ , one can simply integrate the function between A and B, or obtain its area under the curve:

$$p_r(r)_{AtoB} = \int_A^B p_r(r)dr$$

Cumulative distribution function is simply the plot of the area under the curve moving from 0 to *r'* with *r'* increasing from 0 to L-1 (full intensity). That is, we simply accumulate the probability as r increases from 0 to L-1.

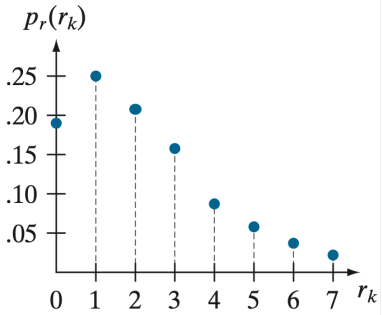
For the example show above, the value of $C_r(r')\ at\ r = r'$ is Y, which is the total area of $p_r(r)$ from 0 to r', shown in shaded region.

Without proofing it here (although it is in the textbook chapter 3), histogram equalization is achieved by **using the CDF as the transformation function T**.

This is intuitively reasonable because the CDF is increasing fastest where the PDF has large values. That means that we stretch the contrast more in intensity range where most pixels belong.

## An example – 3-bit intensity distribution of a 64 x 64 image

| $r_k$ | $n_k$ | $p_r(r_k) = n_k/MN$ |
|---|---|---|
| $r_0 = 0$ | 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 |

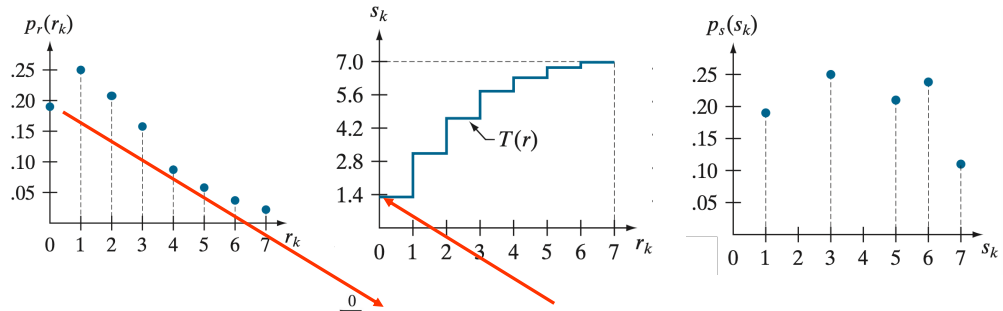





Here is a small example to show how histogram equalization is done, step-by-step.

The image is 64 x 64, and there are only 8 intensity level. The histogram bin values and the normalized discrete PDF is as shown. The intensity level is more common in low values than high values.

## Compute the CDF and use as intensity transform function




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

| | | | |
|---|---|---|---|
| $s_0 = 1.33 \rightarrow 1$ | $s_2 = 4.55 \rightarrow 5$ | $s_4 = 6.23 \rightarrow 6$ | $s_6 = 6.86 \rightarrow 7$ |
| $s_1 = 3.08 \rightarrow 3$ | $s_3 = 5.67 \rightarrow 6$ | $s_5 = 6.65 \rightarrow 7$ | $s_7 = 7.00 \rightarrow 7$ |

- Note that after equalization, the final histogram have no values at 0 and 2!
- Equalized histogram is approximately flat.

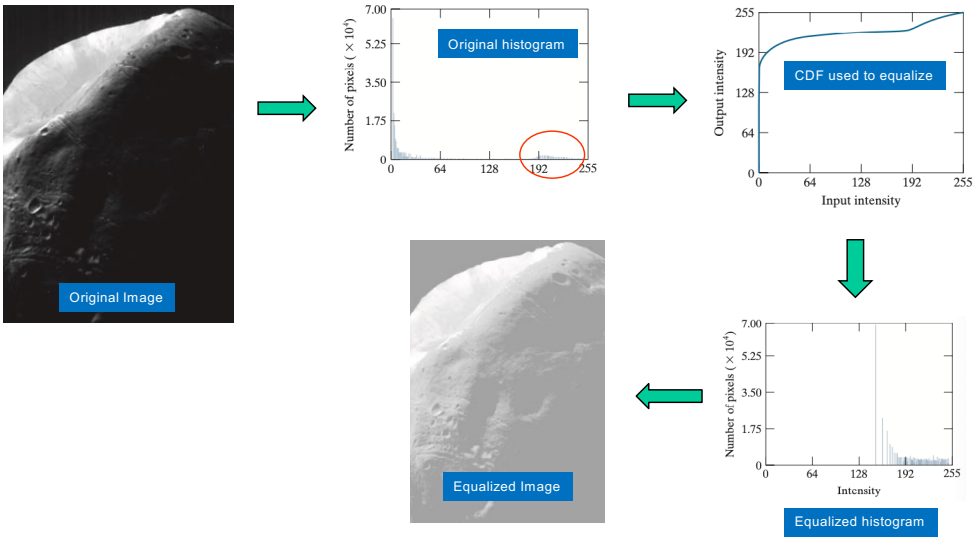


We construct the middle graph by accumulating the PDF scaled to the maximum intensity level (in this case, 7). The T mapping for intensity value of 0 is calculated as 7 x 0.19 = 1.33, which is rounded to 1.

So input 0 is mapped to 1; level 1 is mapped to 3; 2 to 5; etc..

After applying the transformation to all pixels and obtain the output histogram, we obtain the PDF on the right. It is now more flat than before. Note that we only have pixels of 5 distinct values. There are no longer any pixels with values 0, 2 and 4.

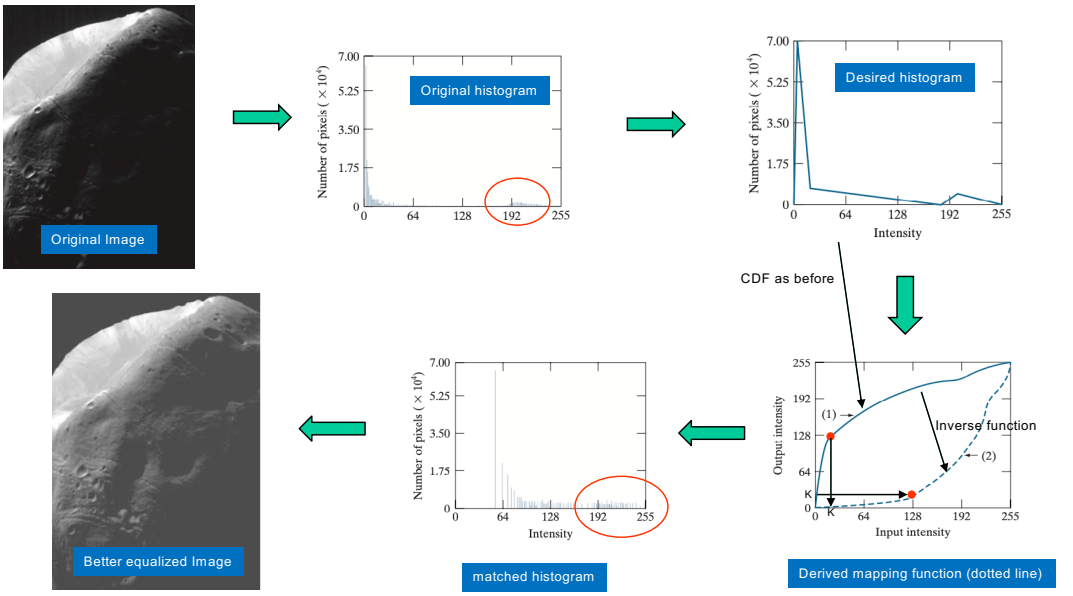
## Histogram equalization may not work






Histogram equalization may not always provide a good enhanced image. Here is an example of a photo of the moon. The histogram equalized image is washed out because most of the image is dark, but the features of interest produced histogram near to the light area. This is a bimodal distribution. Making the histogram flat just focus in making dark regions light, and the entire image is now too light.

## Histogram Matching






Instead of equalizing the histogram, it is much better in this case to transform the original such that processed image has a distribution of your choice. In this case, a desire histogram (not flat) is constructed from the original histogram, retaining the bimodal distribution. We then compute the CDF as before. How, from the CDF we produce the inverse function and use that to do the transformation.

The inverse function is obtained as follow. Consider the pixel output intensity of 128, the CDF tells us that its input intensity should be k. We now reverse these two values: we use 128 as the x input, and k as the inverse function output.

The resulting histogram and image is as shown. Now the enhanced image shows all the details!

## Matlab Functions related to this Lecture

### imadjust

Adjust image intensity values or colormap

**Syntax**

```
J = imadjust(I)
J = imadjust(I,[low_in high_in])
J = imadjust(I,[low_in high_in],[low_out high_out])
J = imadjust(I,[low_in high_in],[low_out high_out],gamma)
```

### imhist

Histogram of image data

**Syntax**

```
[counts,binLocations] = imhist(I)
[counts,binLocations] = imhist(I,n)
```

### histeq

Enhance contrast using histogram equalization

**Syntax**

```
J = histeq(I)
J = histeq(I,n)
J = histeq(I,hgram)
```

### imhistmatch

Adjust histogram of 2-D image to match histogram of reference image

**Syntax**

```
J = imhistmatch(I,ref)
J = imhistmatch(I,ref,nbins)
```

### stretchlim

Find limits to contrast stretch image

**Syntax**

```
lowhigh = stretchlim(I)
lowhigh = stretchlim(I,Tol)
```



Matlab provides many built-in functions in the Image Processing Toolbox. You will be experimenting with them in the Laboratory session next Thursday.