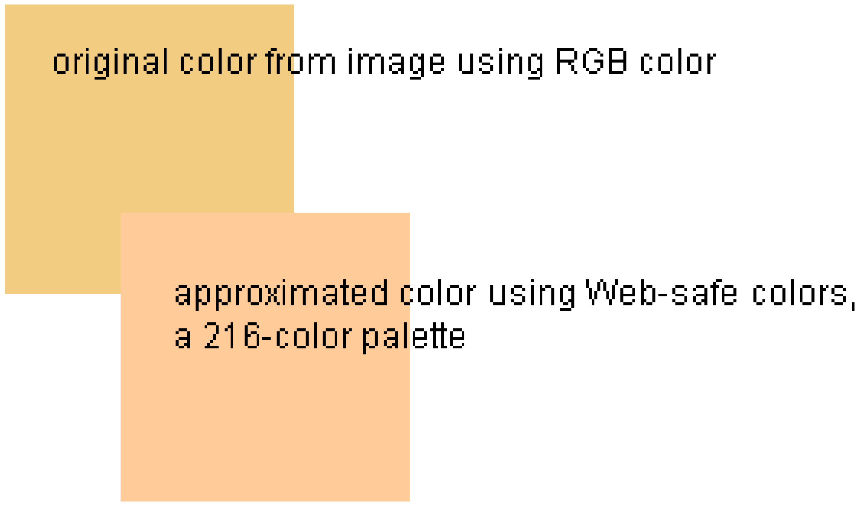
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

BENC 4173 Multimedia Technology & Application

Chapter 3: Image Technology (Part C)

- Dithering
- Histogram for Image Processing
- Image Compression Standard

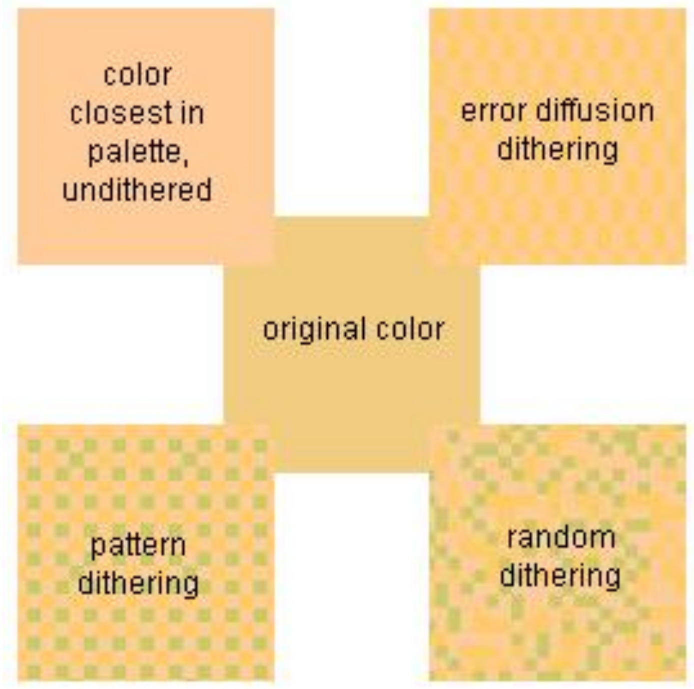
DITHERING


original color from image using RGB color
approximated color using Web-safe colors, a 216-color palette

- RGB color (24 bits) – image files can be quite large
- Using 8 bits can reduce the file size.
- Some of the colors will have to be approximated using colors in the chosen 256-color palette
- May lose some of the colors in the original image

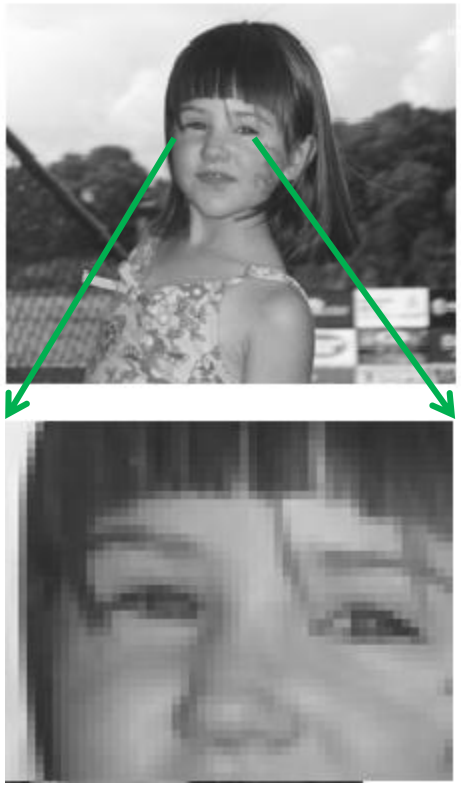
2^{24} = 16,777,216 possible colors

Dithering


color closest in palette, undithered
error diffusion dithering
original color
pattern dithering
random dithering

- Dithering is a technique for approximating a color that is unavailable in colors palette.
- Because the pixels are so small and close together, the eye blends the colors
- to reduce the bit depth of the image, and thus the file size, without greatly changing the appearance of the original image
- Three dithering algorithms: random, pattern and error diffusion dithering.

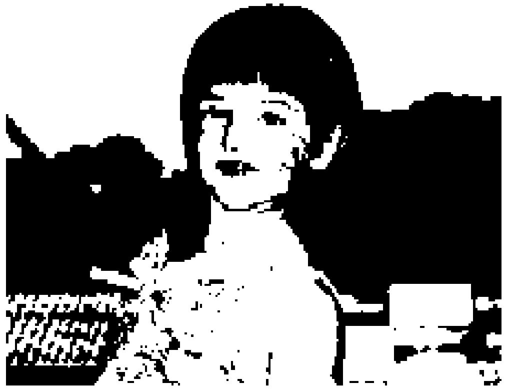
Example



- Example: grayscale photo with resolution of 225x180
- Close-up, the pixels are in varying shades of gray
- What if you wanted to use only one bit for each pixel in a grayscale image?
- This would mean, each pixel would have only one of two values (0, black or 1, white)

Threshold Dithering

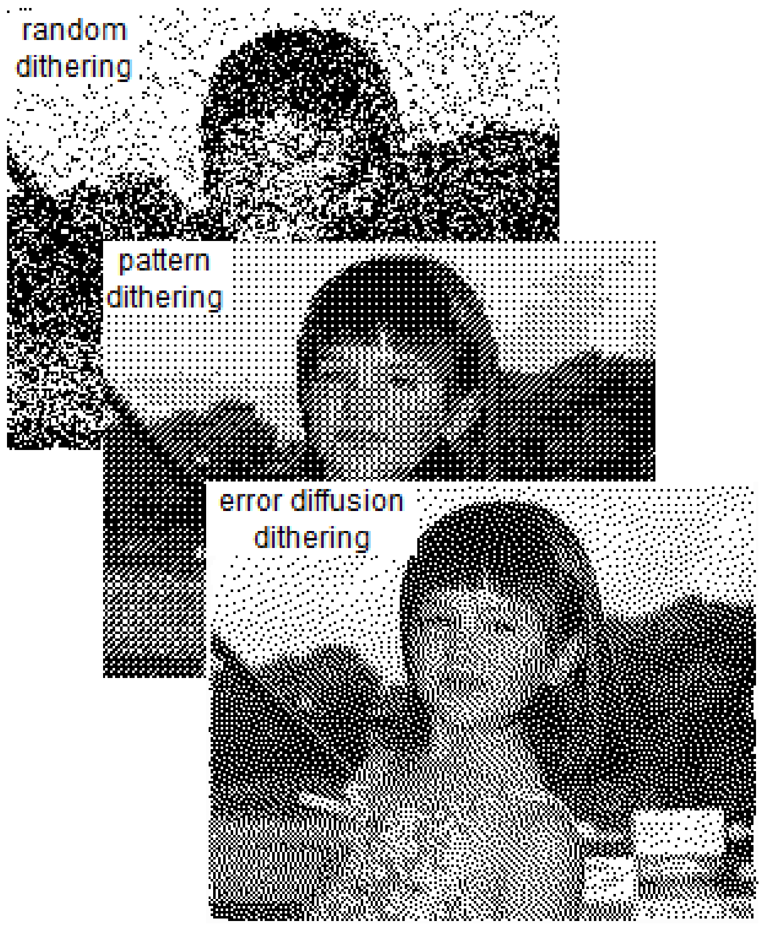
- The simplest way to display a grayscale image using only black and white pixels by the following:
- *If a pixel value is greater than 127, make it white. Otherwise, make it black.* (Why 127?)
- This is called threshold dithering.



Issue: large patches of black & white

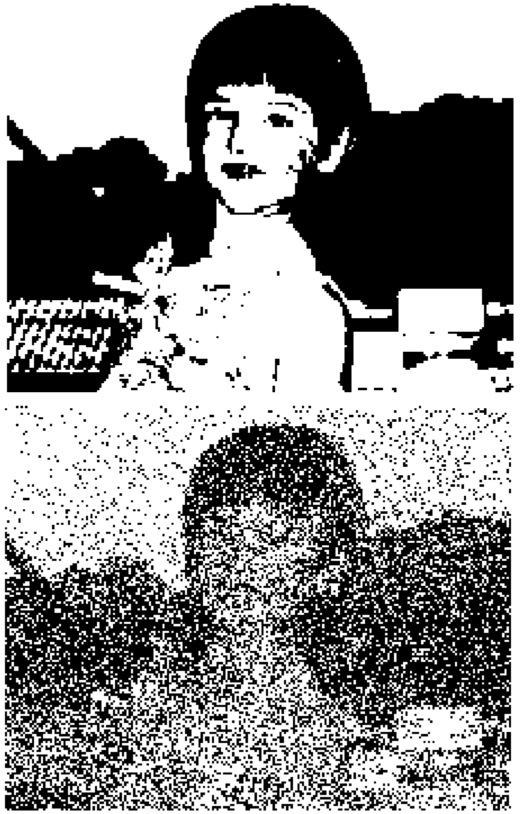
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Dithering Algorithms



- Random (Noise), Pattern and Error Diffusion Dithering
- These dithered versions of the picture are a little better than thresholding in the previous example.
- They create the effect of shades of gray (when combined by the eye)
- How the dithering algorithms are implemented?

Random Dithering



- For each pixel, a random number (between 0 and 2^b -1, b is the bit depth) is generated.
- If the pixel value is less than the random number, a bit of **0** is stored in the dithered image, making the pixel black. Otherwise, 1 is stored – pixel white.
- In contrast to thresholding, random dithering inserts random ‘noise’ into the picture, which breaks up the solid areas of black and white.

Random Dithering

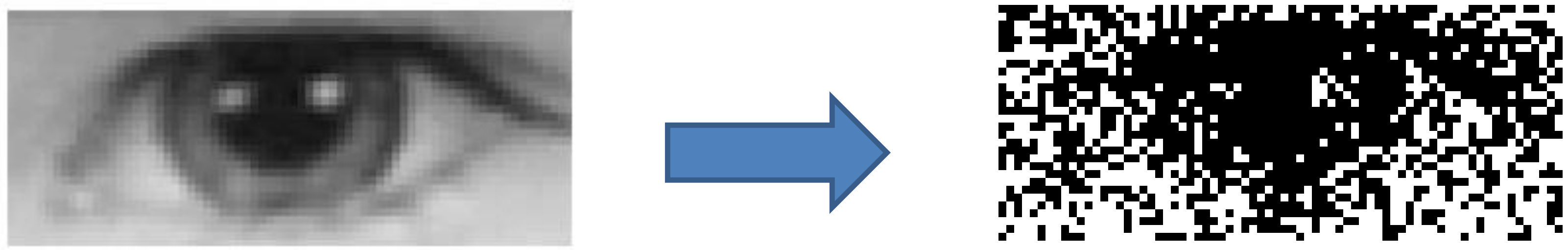
- The value of the selected pixel is 146.

- Picture of an eye (zoomed from the original resolution 288x120).
- The tiny red square indicates which pixel is being processed.
- Supposedly, random number is 77.
- Is **146<77** ?

Random Dithering

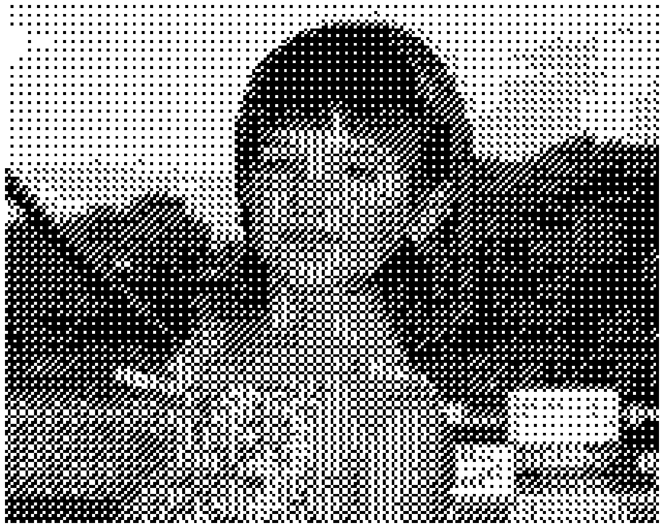
- The pixel value >= random number.
- Therefore, the square is white.
- Replaced the pixel with WHITE pixel

Random Dithering



- For the rest of the pixels.
- It is actually too noisy, but the effect can be softened by various means – like not inserting noise with every pixel

Pattern Dithering



- ***ordered dithering*** or the ***Bayer method***
- Creates a more patterned effect in the dithered image, as opposed to the noisiness of random dithering.
- The pattern 'stamped' onto the image, reflects the colors or grayscale values in the image.
- The pattern represented by a mask (half-tone screen) of numbers → which pixels are more inclined to black or white

The Mask

- Supposedly, we use a pattern that represented by the mask below:

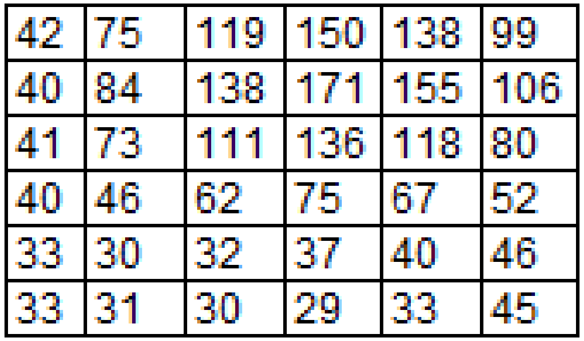
$$\begin{matrix} 8 & 3 & 4 \\ 6 & 1 & 2 \\ 7 & 5 & 9 \end{matrix}$$

***m* x *m* array of values between 1 and m^2**

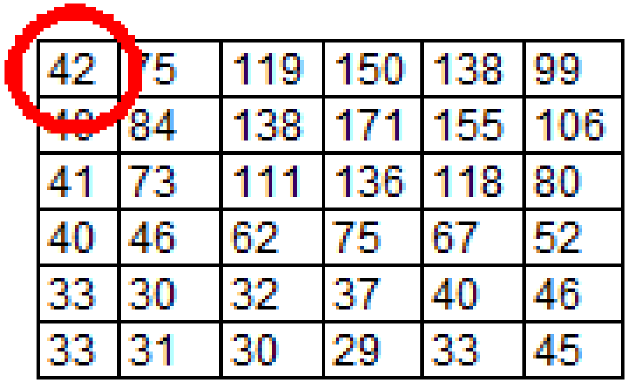
- The mask is successively placed over 3x3 blocks of pixels in the image.
- A small portion of the pixel data from the eye we just randomly dithered is given to the right.

42	75	119	150	138	99
40	84	138	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

Normalizing



42	75	119	150	138	99
40	84	138	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45



42	75	119	150	138	99
40	84	138	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

- Normalize the pixel values so that they are between 0 and 9, like mask value.i.e.256/9=28.
- Divide by 28
- Drop the remainder

8	3	4
6	1	2
7	5	9

Normalizing

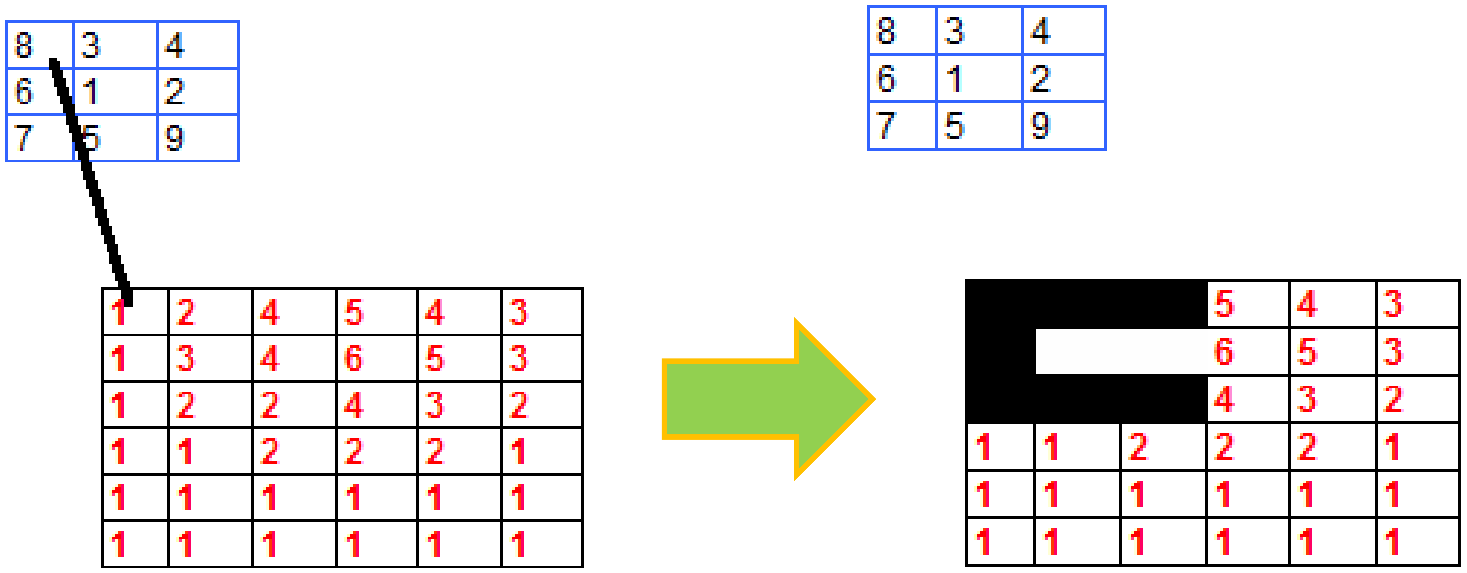
1	75	119	150	138	99
40	84	138	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

$\Rightarrow$

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

- After divide by 28

Comparing Pixels with Mask



8	3	4
6	1	2
7	5	9

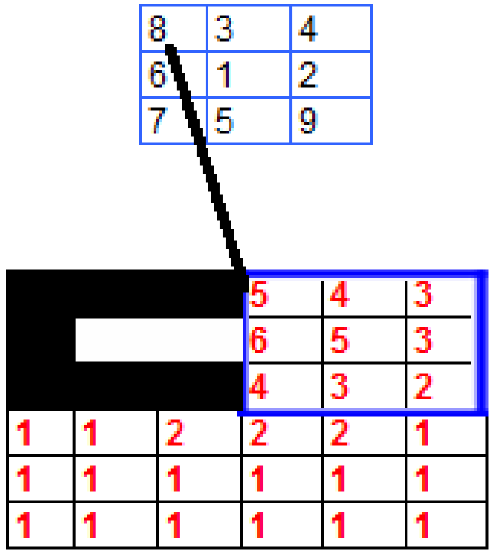
1	2	4	5	4	3
1	3	4	6	5	3
1	2	2	4	3	2
1	1	2	2	2	1
1	1	1	1	1	1
1	1	1	1	1	1

8	3	4
6	1	2
7	5	9

⬛	⬛	⬛	5	4	3
⬛	⬜	⬜	6	5	3
⬛	⬛	⬛	4	3	2
1	1	2	2	2	1
1	1	1	1	1	1
1	1	1	1	1	1

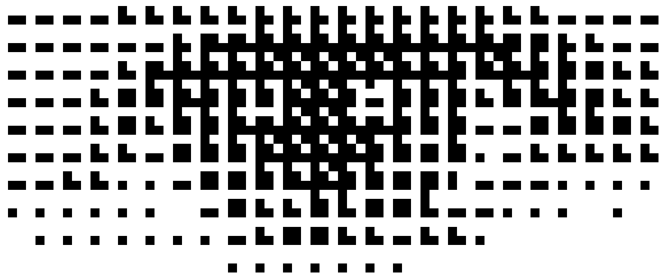
- Is **pixel value** > **mask value**?
- YES – white
- NO - black

Comparing Pixels with Mask



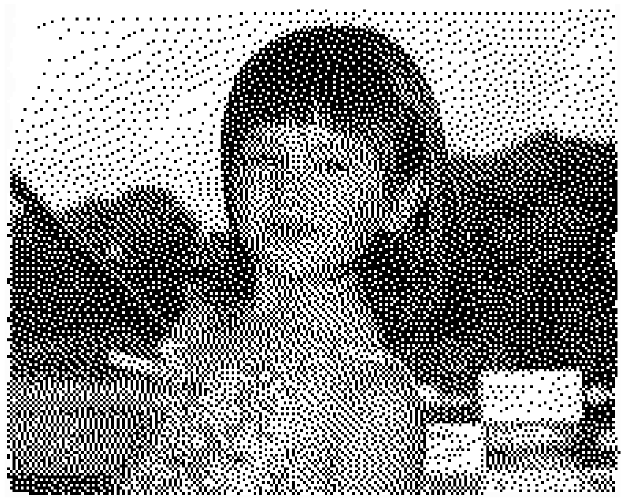
- Now, the mask moved to the three pixels to the right.
- Comparison repeated.

Pattern Dithering



- The method is easy and fast, but it can result in a crosshatched effect.
- There are many variations of this basic algorithm, applying mask of different sizes and types.

Error Diffusion Dithering



- a method that disperses the error, or difference between a pixel’s original value and the color (or grayscale) value available
- Creates a good simulation of grayscale values on many types of images.

Also known as *Floyd-Steinberg Algorithm*

How it works

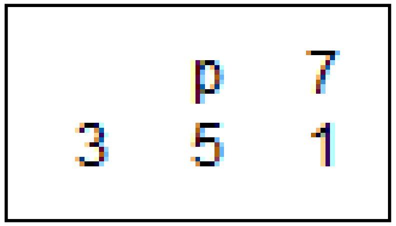
- for each pixel, the difference between the original color and the color available is calculated
- this error is divided up and distributed to neighboring pixels that have not yet been visited
- after all pixels have been visited, the color used for each pixel is the color in the reduced palette closest to the pixel's new value

Details of the algorithm include a determination of the **relative fraction of the error distributed** to neighboring pixels, the method for **choosing the closest available color**, and **the order** in which pixels are processed.
Adjustments can be made to reduce color bleeding, i.e., the bleeding of one color into another in the direction in which the error is dispersed.

Error Diffusion Dithering

The algorithm:

- For each pixel *p*, the error is distributed in a manner reflected in the pattern below:



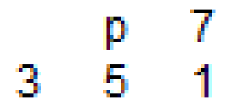
	p	7
3	5	1

- Each number in the mask pertains to the pixel that it "covers“.
- Notice that the number 7, 3, 5 and 1 add up to 16

- The pattern symbolizes:
  - *P* is the ‘central’ pixel, whose error is being distributed
  - The pixel to the right of *p* gets 7/16 of the error
  - The pixel below *p* gets 5/16 of the error
  - The pixel below and to the left gets 3/16 of the error
  - The pixel below and to the right gets 1/16 of the error

Step 1: Dispersing Errors

- The way to implement this is to move from left to right across the pixel rows.
- For each pixel *p*, if the pixel's value ≥ 128, then it is closer to 255. A simple threshold algorithm would have make such a pixel white (or 1).
- The difference between the pixel's original value and what we would like to make it (255) is effectively the error we are introducing.
- Since thresholding would make *p* LARGER, we want SUBTRACT some from *p*'s neighbors.



- If the pixel is closer to 0 (i.e., if the pixel's value < 128), then thresholding would make it black (or 0).
- Since thresholding would make such a pixel SMALLER, we should ADD a portion of the error to each of the neighboring pixels.
- The error is the difference between the pixel's original value and 0 – or simply the pixel's original value.

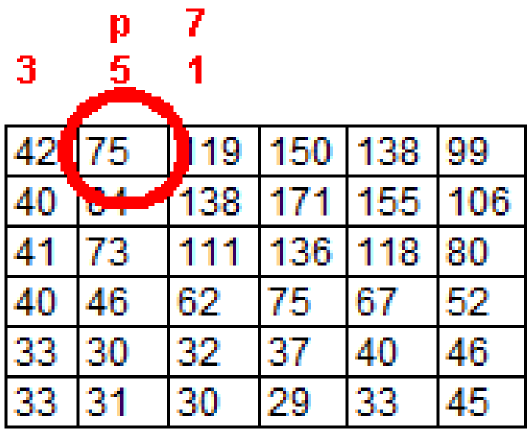
Example

	p	7
3	5	1

42	75	119	150	138	99
40	84	138	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

42	75 p	7 119	150	138	99
3 40	84 5	1 138	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

- First, position the mask over a portion of the pixels in the original image.



42	75	119	150	138	99
40	84	138	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

- Look at the pixel value corresponding to the space marked *p*.

- The pixel has a value of 75.
- 75 < 128
- In threshold dithering, we would make this pixel black – 0.
- Since, we decreasing this pixel's value from 75 to 0, we'll distribute 75 among the neighboring pixels.

3	p 5	7 1			
42	75	119	150	138	99
40	84	138	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

3	p 5	7 1			
42	75	152	150	138	99
40	84	138	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

- The pixel to the right of pixel p, (value 119), gets 7/16 of the error.
- **119** + (75*7/16) = 119+33 = **152**

- The rest of the neighboring pixels adjusted similarly.

119 + (75 * 7/16) = 119 + 33 = 152

40 + (75 * 3/16) = 40 + 14 = 54

84 + (75 * 5/16) = 84 + 23 = 107

138 + (75 * 1/16) = 138 + 5 = 143

	p	7			
3	5	1			
42	75	152	150	138	99
40	84	138	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

	p	7			
3	5	1			
42	75	152	150	138	99
54	107	143	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

p 7

3 5 1

42	75	152	150	138	99
54	107	143	171	155	106
41	73	111	136	118	80
40	46	62	75	67	52
33	30	32	37	40	46
33	31	30	29	33	45

- Then the mask is moved to the right one pixel and the process is repeated.

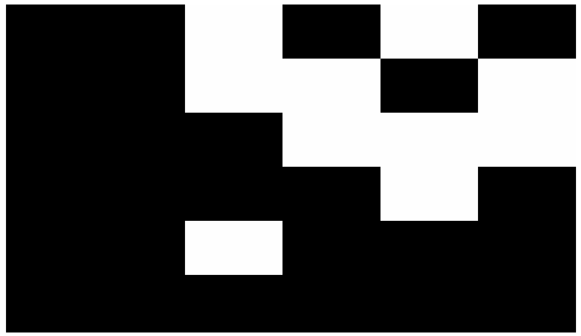
Step 2: Thresholding

42	75	152	105	184	68
54	88	169	146	92	142
58	84	106	160	198	129
56	92	123	124	146	11
50	82	135	11	18	47
48	34	0	28	39	46

- The pixel data after the error has been spread across this portion of the image.

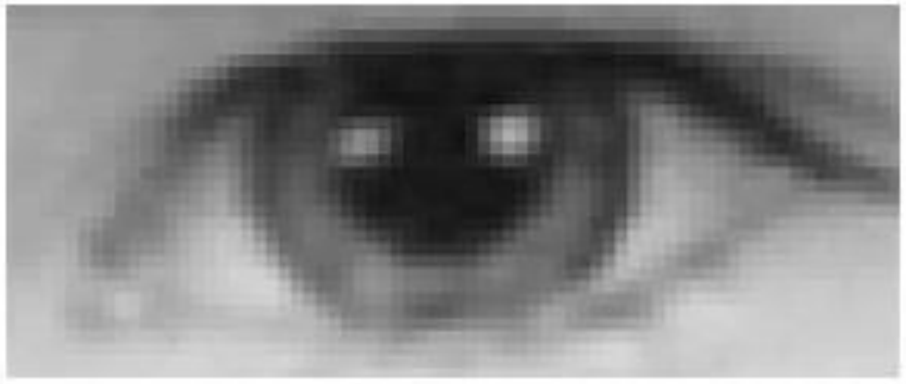
- The second step, use these new values, comparing them to the threshold value of 127.
- If a pixel's value > 127, changed to 1 (white)
- If less than or equal to 127, changed to 0 (black)

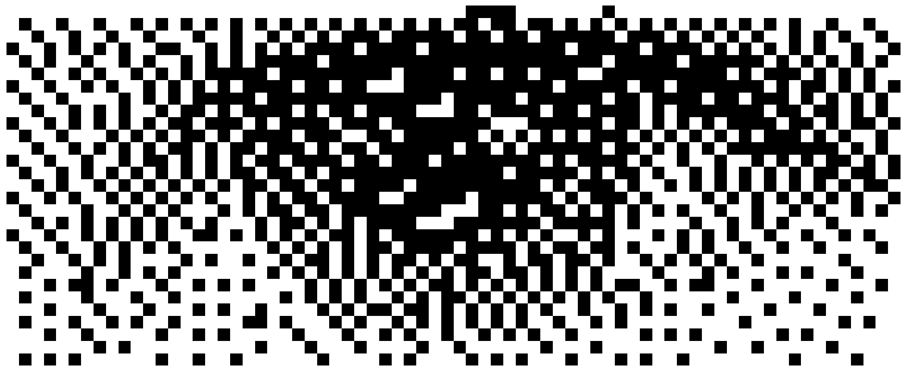
42	75	152	105	184	68
54	88	169	146	92	142
58	84	106	160	198	129
56	92	123	124	146	11
50	82	135	11	18	47
48	34	0	28	39	46



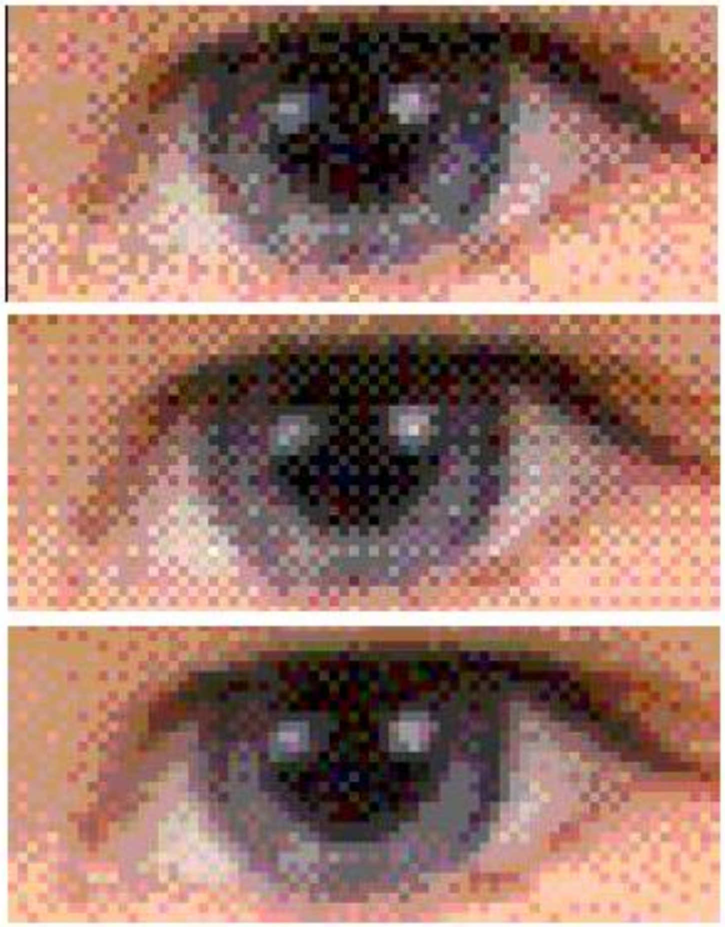
- Pixel's value > 127, changed to 1 (white)
- Pixel's value $\leq$127, changed to 0 (black)

Error Diffusion Dithering

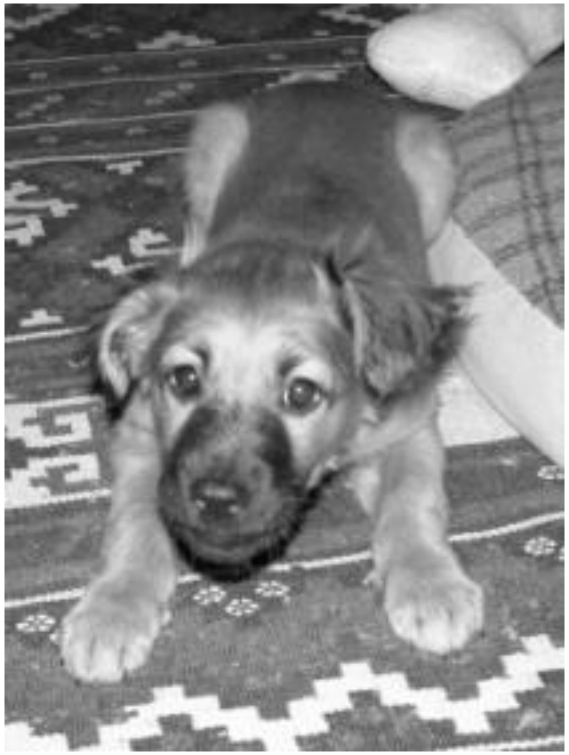


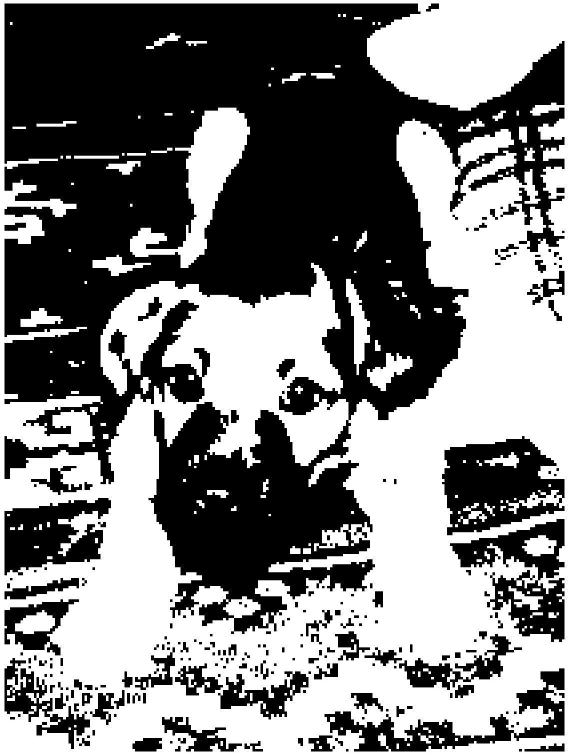


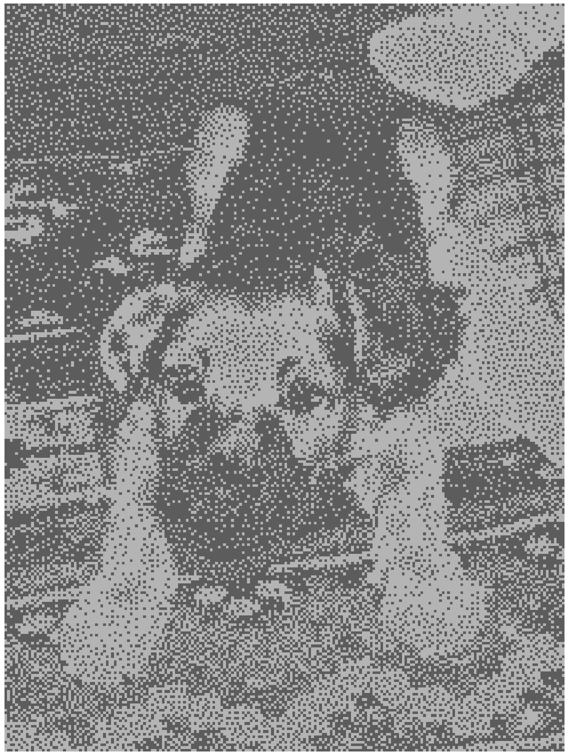
Color Dithering

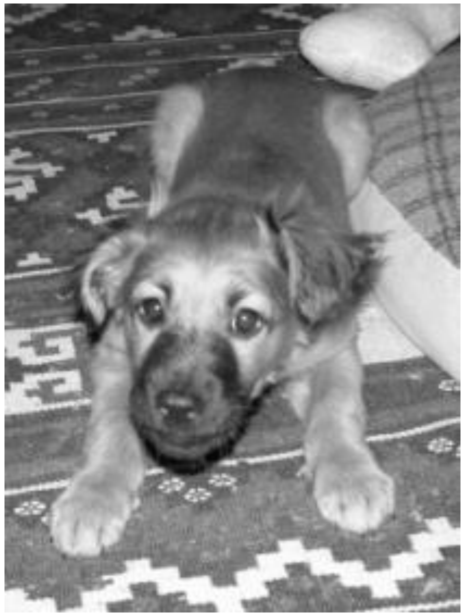


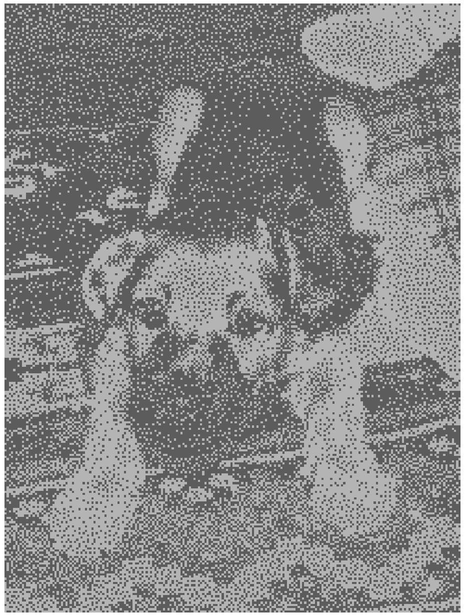
- The techniques are essentially the same with color images.
- Option: when convert and RGB to indexed color mode.

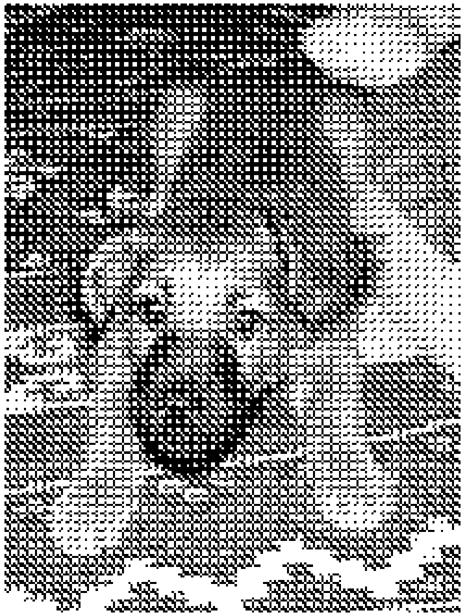

Original
(8 bits)

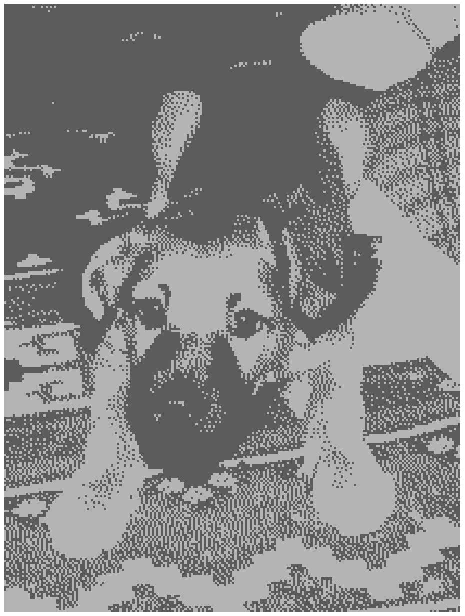

Uniform
Quantization
(1 bit)

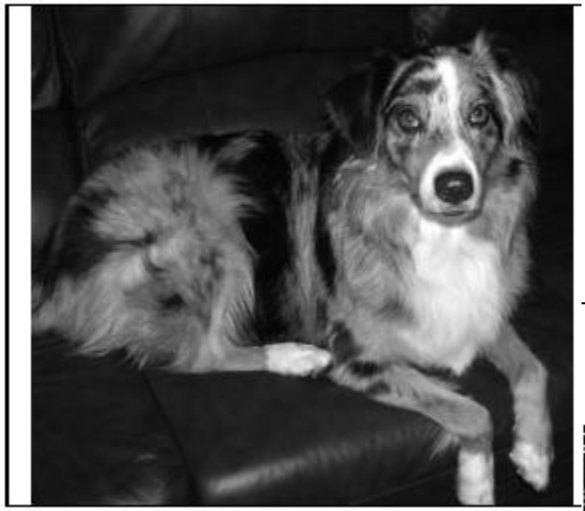

Random
Dither
(1 bit)

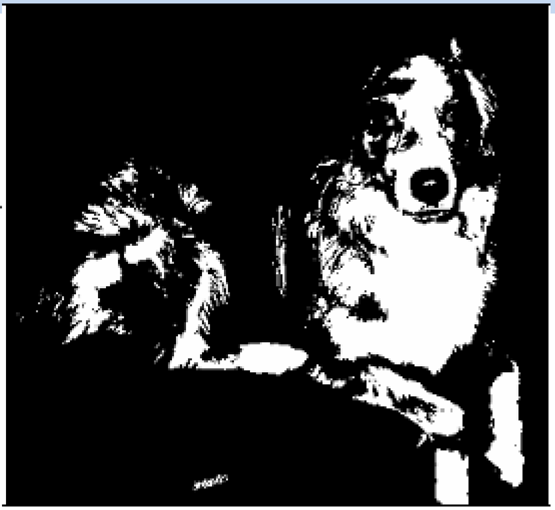

Original
(8 bits)


Random
Dither
(1 bit)

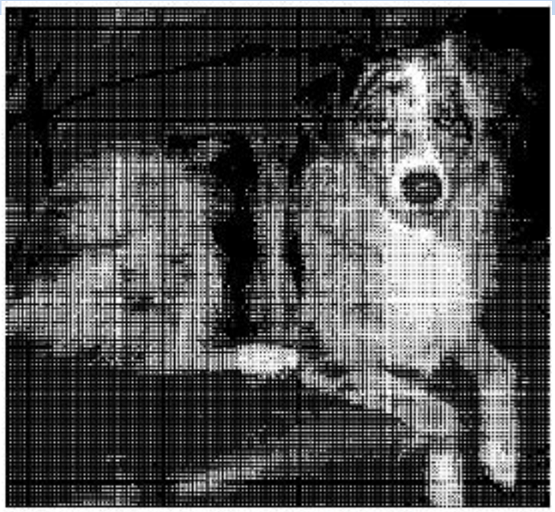

Ordered
Dither
(1 bit)


Floyd-Steinberg
Dither
(1 bit)

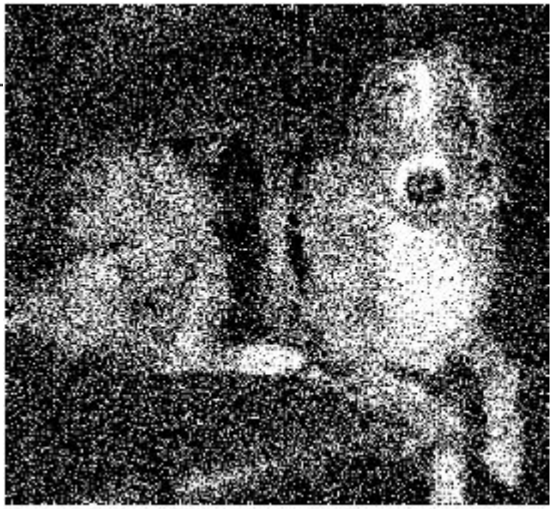




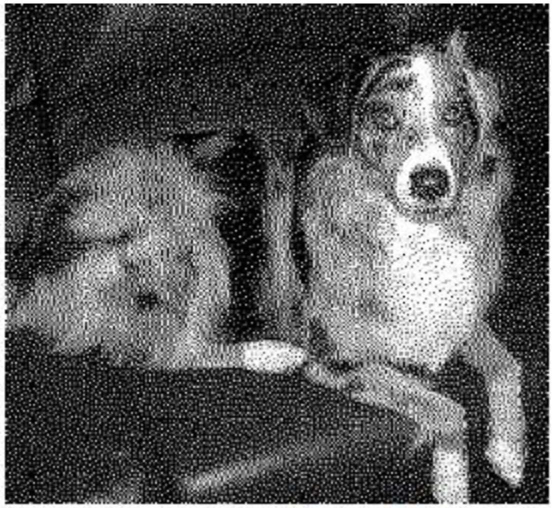
Thresholding



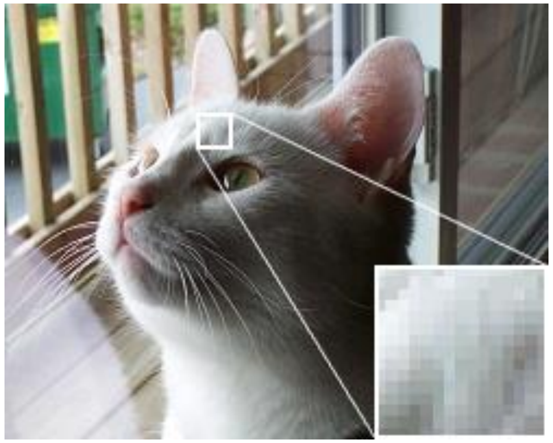
Pattern dithering



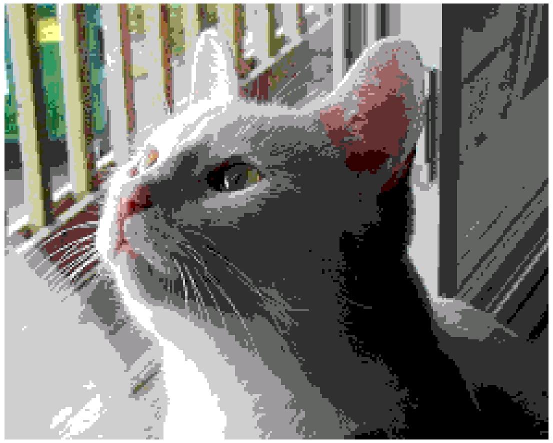
Noise (i.e., random) dithering



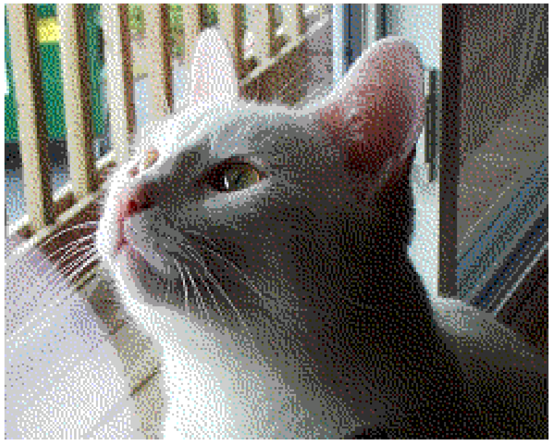
Error diffusion dithering



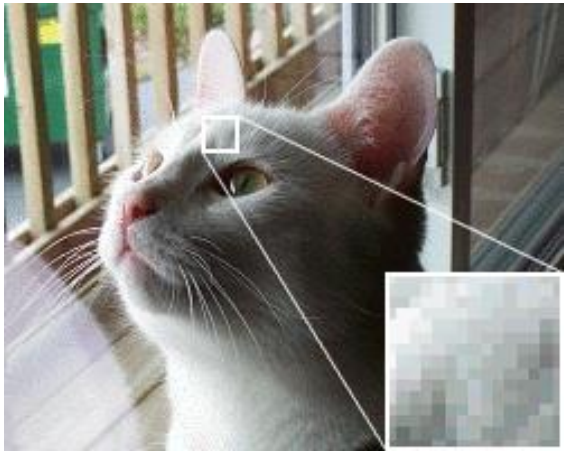
Original image



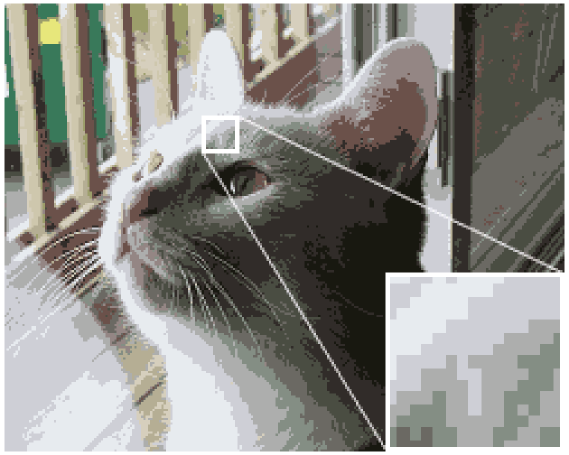
Web-safe palette, no dithering



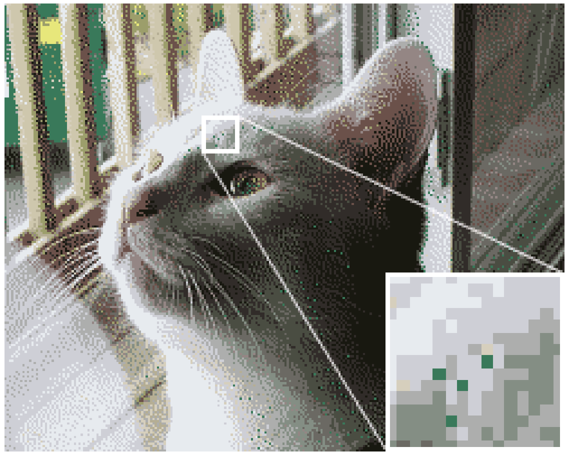
Web-safe palette, FS dithering



Optimized 256 color palette
FS dithering



Optimized 16 color palette
No dithering



Optimized 16 color palette
FS dithering

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

JPEG Compression

- Joint Photographic Experts Group
- suitable for reducing the size of photographic images or continuous-tone artwork – pictures where the scene transitions move smoothly from one color to another
- *.jpg* or *.jpeg* suffix; but it is also possible to apply JPEG compression to images saved in TIFF, PICT, and EPS files
- the algorithm removes closely-spaced changes in color that are not easily perceived by the human eye
- transforming the image data from the spatial domain to the frequency domain

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

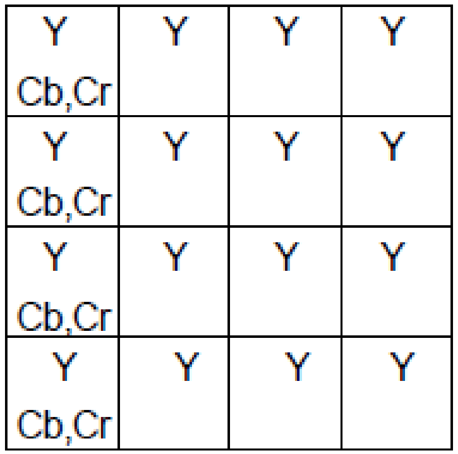
Chrominance subsampling

Step 1. Divide the image into 8 × 8 pixel blocks and convert RGB to a luminance/chrominance color model.

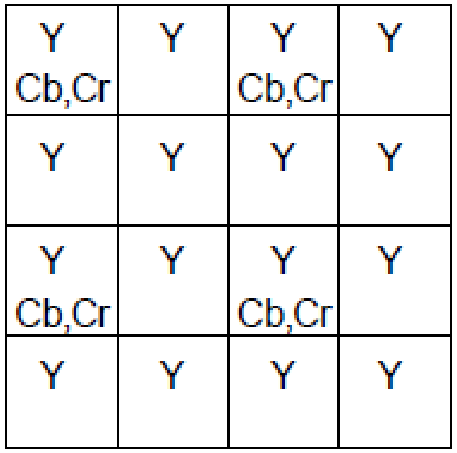
Detail: ***a*** is the number of Y samples in both rows

b is the number of Cb samples in the first row (and Cr samples)

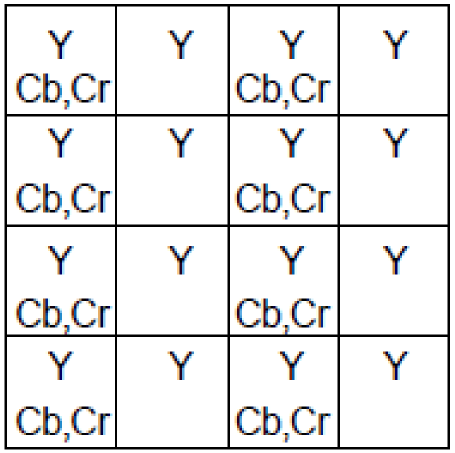
c is the number of Cb (and Cr samples) in the second row



4:1:1

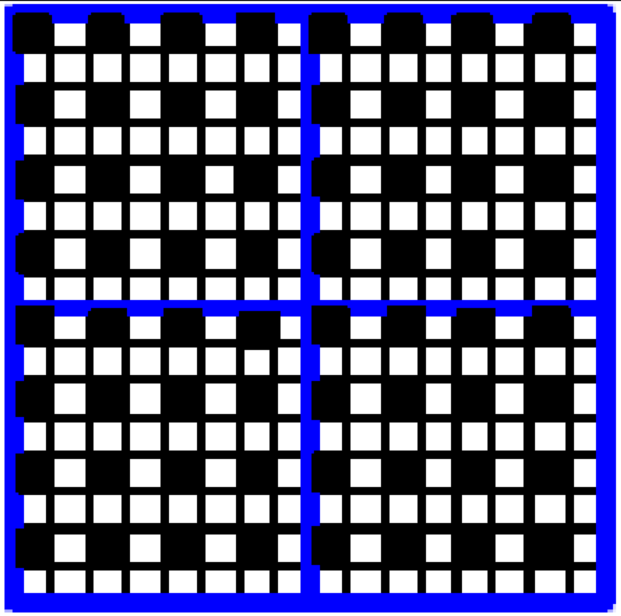


4:2:0



4:2:2

Chrominance subsampling



With 4:2:0 YCbCr chrominance subsampling, a 16 x 16 macroblock yields:
--four 8 x 8 blocks of Y values
--one 8 x 8 block of Cb values
--one 8 x 8 block of Cr values
Each black location represents one Cb and one Cr sample taken for a block of four pixels. Y samples are taken at all locations.

Chrominance subsampling

Consider the **reduction** in file size if **4:2:0** downsampling is used by counting how many bytes area of pixels with a width of **four pixels and a height of two pixels** – eight pixels total.

Assuming that each component requires one byte, an unsubsampled image requires 8 * 3 = 24 bytes. Subsampling at a rate of 4:2:0 requires 8 + (2 * 2) = 12 bytes (eight for the Y component but only two for each of the Cb and Cr). This is a **2:1 savings**.

Keep in mind that if 4:2:0 chroma subsampled **YCbCr color** is used, for every 16 × 16 pixel area, there are four 8 × 8 blocks of Y data and one each of Cb and Cr data.

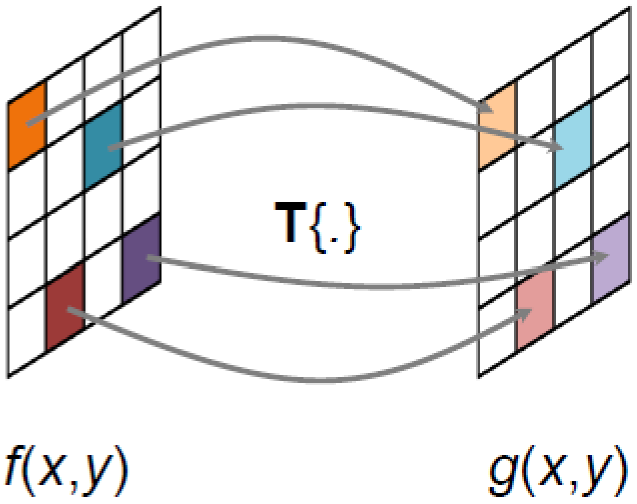
If **RGB** color is used, there are three 8 × 8 blocks – one each for R, G, and B – for every 8 × 8 pixel area.

Point Processing

- Types of neighbourhood
  - –**Point processing –1x1 (single pixel)**
  - –Neighbourhood processing –2x2 or greater.
- ***T** is known as a gray-level **transformation** function **s= T(r)***
- *s* and *r* denote the gray level of *f(x,y)* and *g(x,y)* respectively.
- *s* is output & *r* is input.

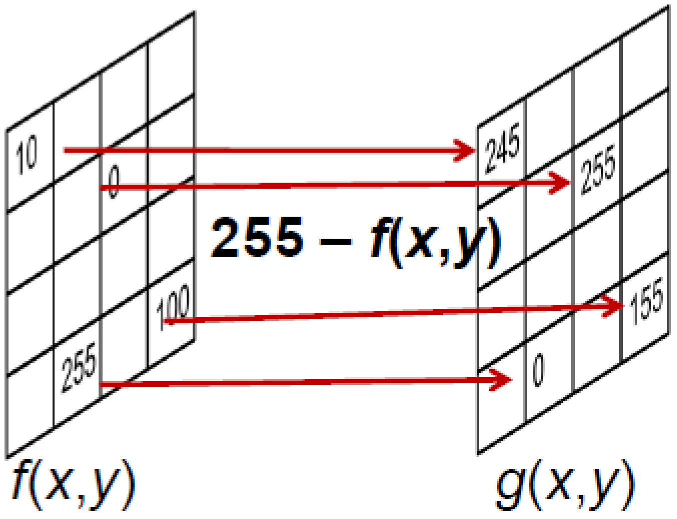
Point Processing

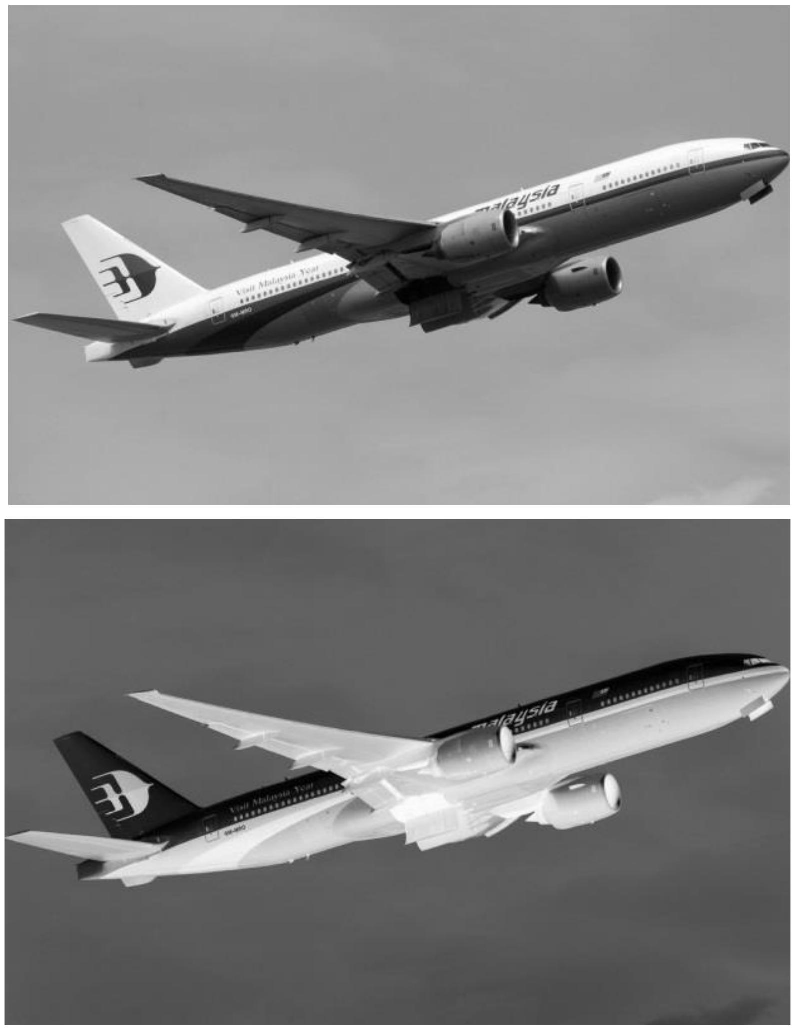
- Graphically - Example of reducing pixel brightness



Point Processing

- **Negatives**





Point Processing

Exercise

- Given the following image matrix of 4-bit 4x4 image give its negative image

3	4	12	13
2	1	14	15
3	5	11	12
12	11	13	14

input

12	11	3	2
13	14	1	0
12	10	4	3
3	4	2	1

output

UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Image Enhancement – Contrast Stretching

1) Piece-Wise Linear Transformation

- Complementary approach to gray level transformation.
- Advantage –can be arbitrarily complex thus achieve many transformation results using linear functions.
- Disadvantage –require more user input.

Image Enhancement – Contrast Stretching

- **Contrast Stretching**
  - –Simplest form of piecewise linear functions.
  - –Increase the dynamic range of the gray levels.

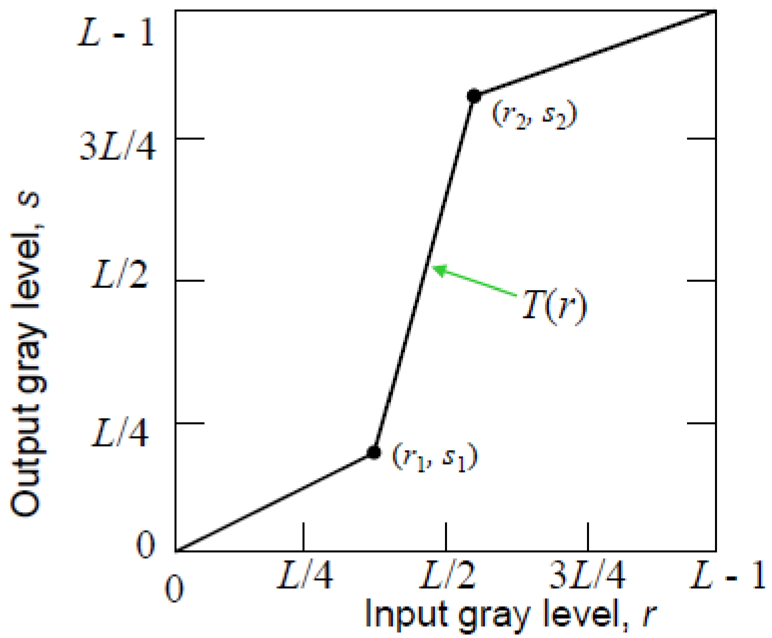


Image Enhancement – Contrast Stretching

- **Contrast Stretching**

  In the above graph

  $s1 < r1$ and $s2 > r2$

  indicates:

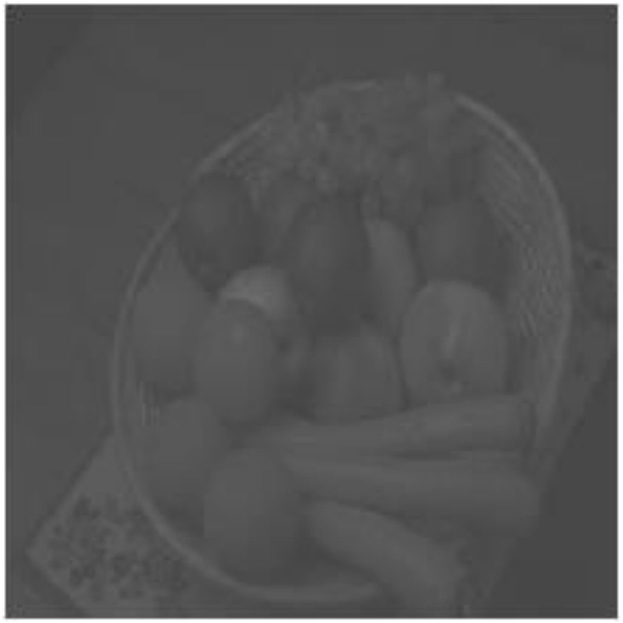
  –pixel values less than *r1 appear darker.*

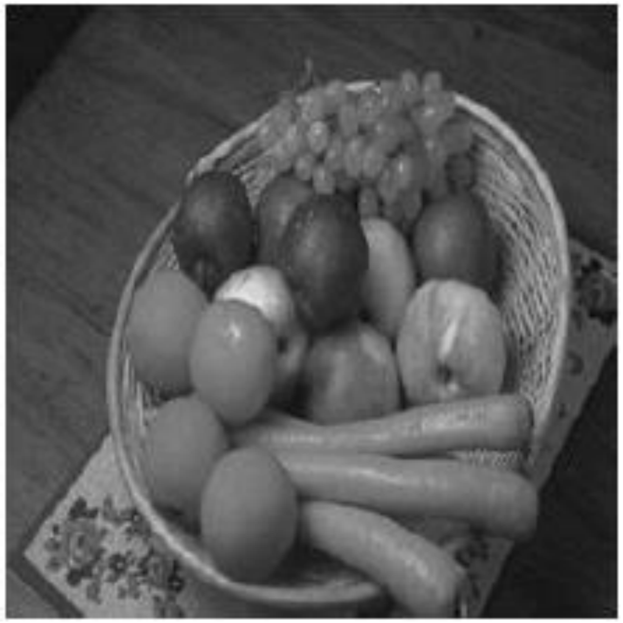
  –pixel values between *r1 & r2 will be stretched over s1& s2.*

  –pixel values above *r2 appear brighter.*

Example

Example:-
s1= 0 , r1= 67 and s2= 255 , r2= 110


Original

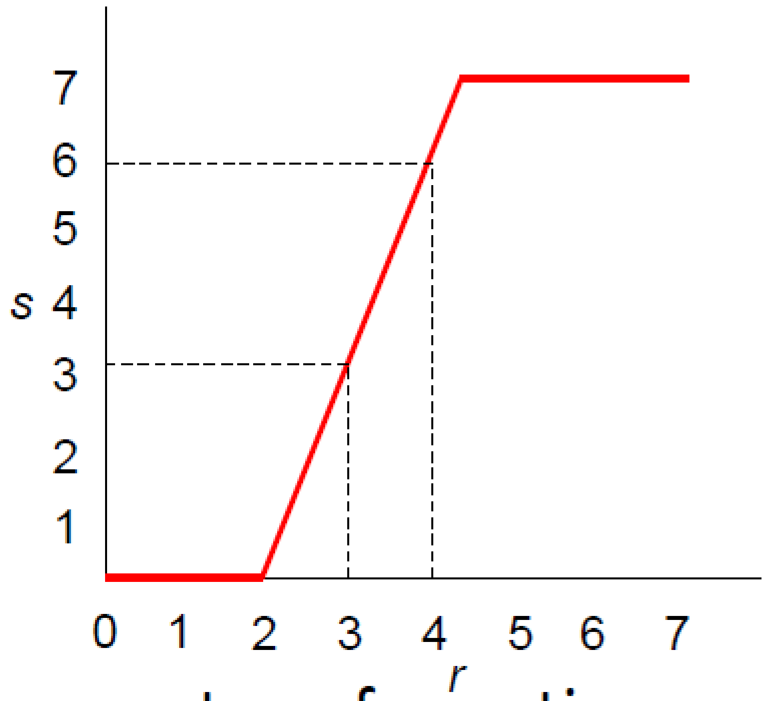

Contrast Stretched

Example

- Example – 3 bit image

0	1	1	2	2
1	1	0	0	2
3	3	5	5	4
2	1	6	6	4

input image



transformation

Example

- Result

0	0	0	0	0
0	0	0	0	0
3	3	7	7	6
0	0	7	7	6

Output

UTeM UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Image Enhancement: Histogram Processing

- Histogram is a graph that shows the frequency or occurrence of each gray level existed in an image.
- It shows pixels distribution in an image.

Example (3-bit 4x4 image):

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

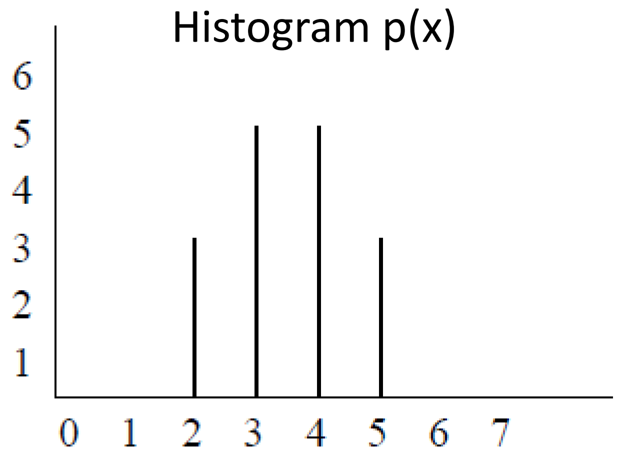
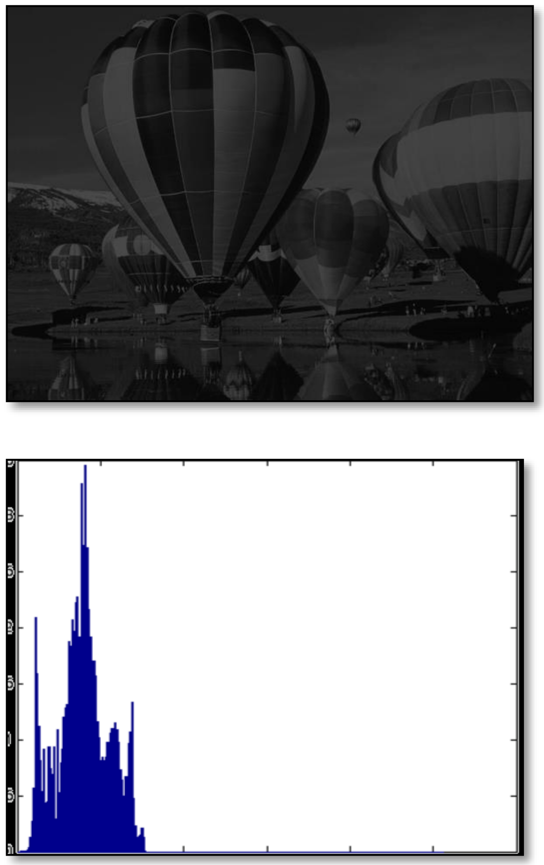
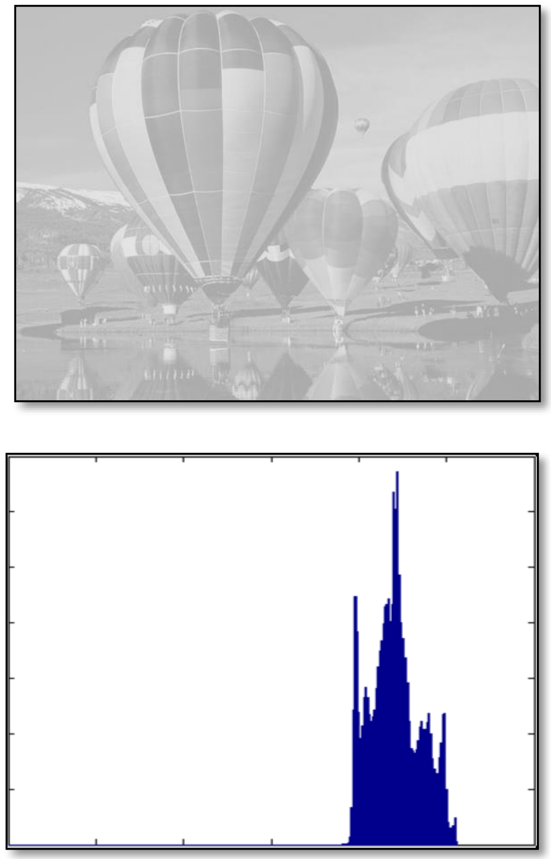
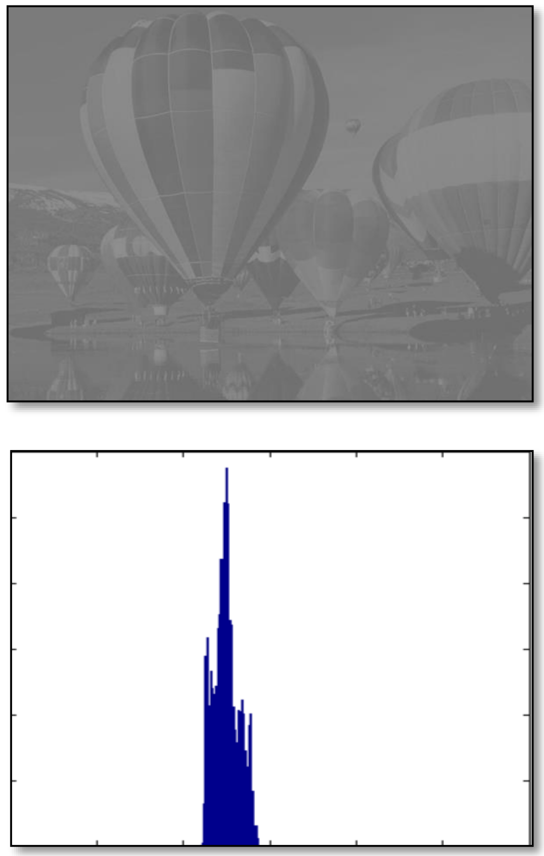


Image Enhancement: Histogram Processing


(a) Dark image


(b) Bright image


(c) Low contrast image

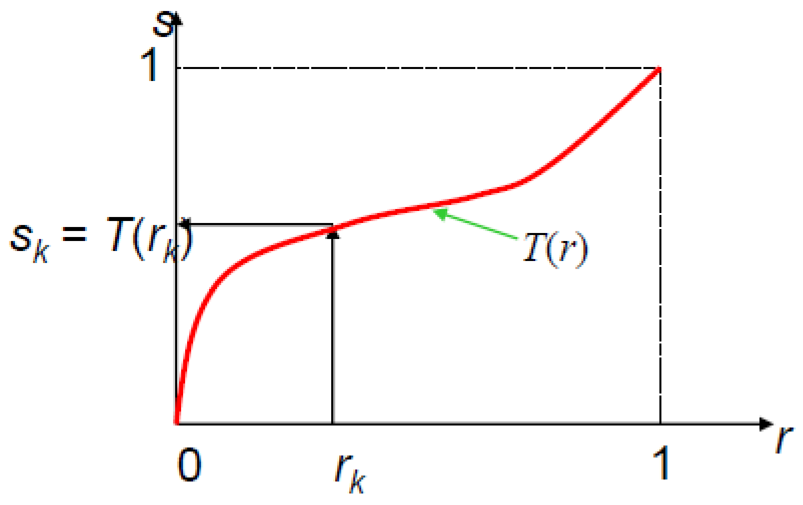
UNIVERSITI TEKNIKAL MALAYSIA MELAKA

Image Enhancement: Histogram Processing

2) Histogram Equalization

- Let pixel ***r*** represents original image and ***s*** be the processed image.
- ***s= T(r)*** ***$0 \le r \le 1$*** and **$0 \le T(r) \le 1$ *for* $0 \le r \le 1$**

•This is to guarantee that the output gray levels will be in the same range as the input levels.



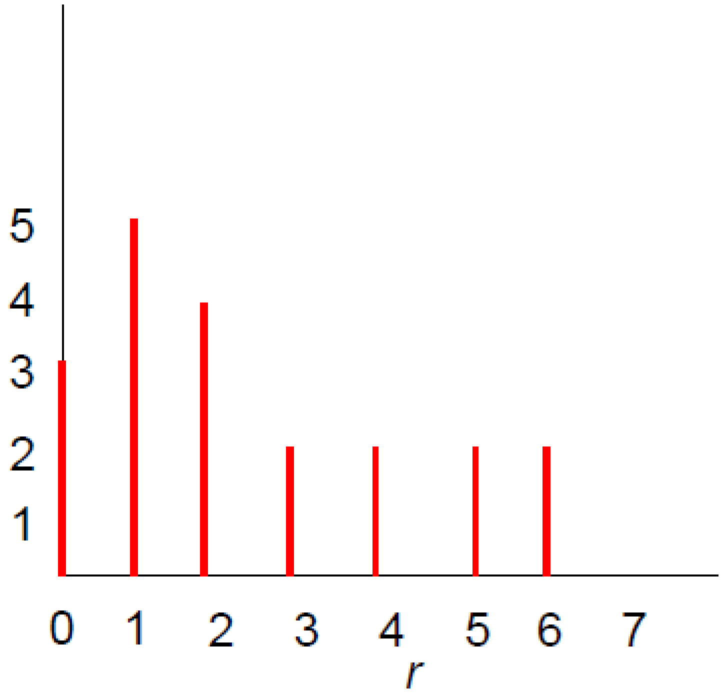
An example of such transformation

Example

Example (3-bit 4x5 image):

0	1	1	2	2
1	1	0	0	2
3	3	5	5	4
2	1	6	6	4

Input image



Histogram distributions

Example

- Using the following equation:

$$s_k = T(r_k) = \sum_{j=0}^{k} p_r(r_j) = \sum_{j=0}^{k} \frac{n_j}{n_{total}} \quad k = 0,1,2,\dots,L-1$$

- n_{total}=20

- We have

Example

Input - r_k	0	1	2	3	4	5	6	7
Output -s_k	3/20 =0.15 0.15	5/20 =0.25 0.4	4/20 =0.2 0.6	2/20 =0.1 0.7	2/20 =0.1 0.8	2/20 =0.1 0.9	2/20 =0.1 1	0 0 1
s_k *7 to obtain the actual output image	1	3	4	5	5	6	7	7

Example

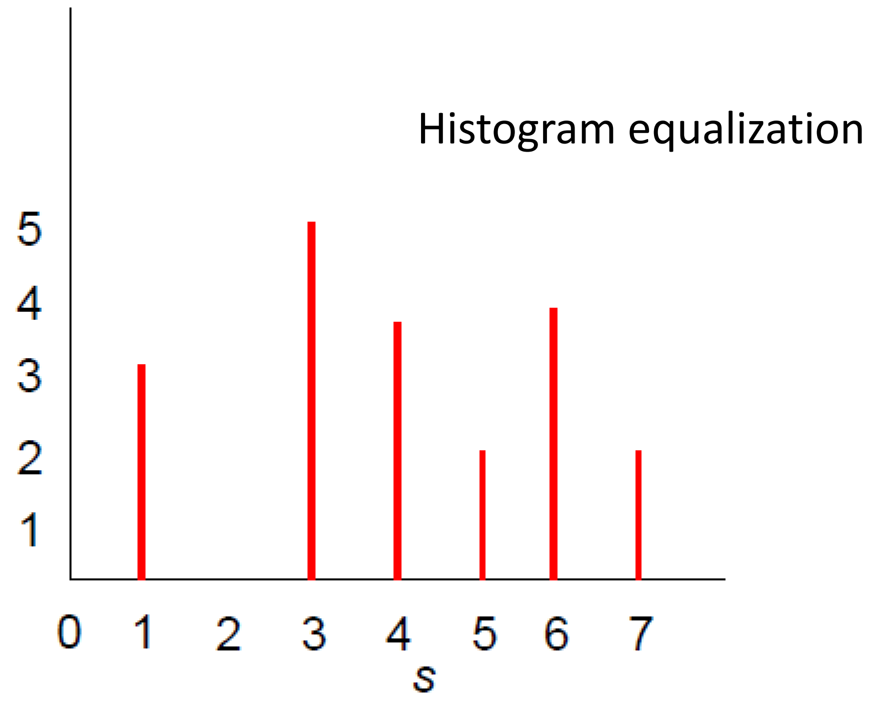
0	1	1	2	2
1	1	0	0	2
3	3	5	5	4
2	1	6	6	4

Input image

1	3	3	4	4
3	3	1	1	4
5	5	6	6	6
4	3	7	7	6

Output image

Example



•In general the discrete histogram of the output image will not be uniform as in the continuous counterpart but the histogram will spread covering full dynamic range of the gray scale.