Chapter 5 :Quantization and Point Processing

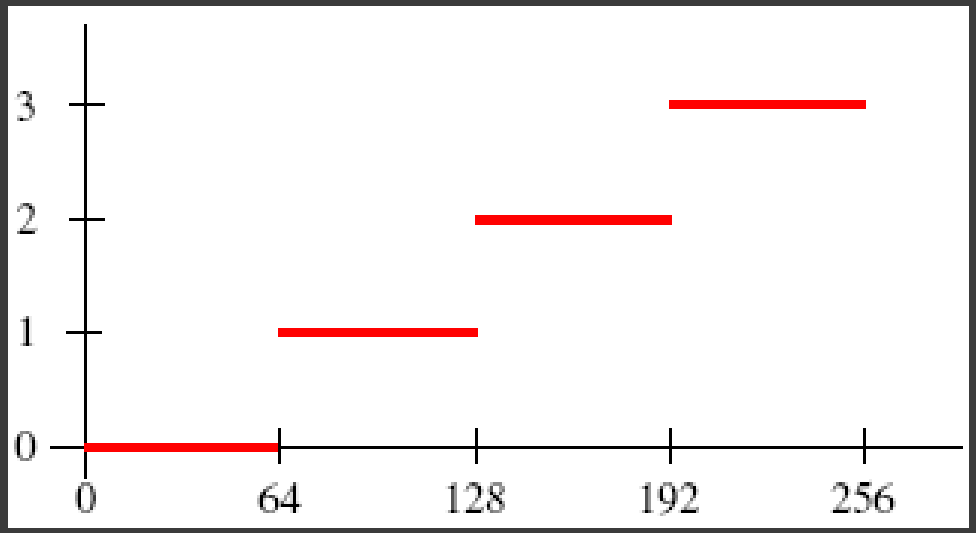
02739325 : Digital Signal and Image Processing

Aj.Kanitta Tangthaikwan

Quantization

- Quantization refers to the number of grayscales used to represent the image.
- As we have seen, most images will have 256 grayscales, which is more than enough for the needs of human vision.
- **Uniform quantization**

```
f = floor(double(x)/2)
q = uint8(f*2)
```



Uniform quantization

```
x = imread('newborn.tif');

x1 = uint8(floor(double(x)/2)*2);

x2 = uint8(floor(double(x)/4)*4);

x3 = uint8(floor(double(x)/8)*8);

x4 = uint8(floor(double(x)/16)*16);

x5 = uint8(floor(double(x)/32)*32);

x6 = uint8(floor(double(x)/64)*64);

x7 = uint8(floor(double(x)/128)*128);
```

Uniform quantization (Cont.)

Uniform quantization (Cont.)

- **grayslice** % function of reducing the number of grayscales in an image.
- Given an image matrix ***x*** and an integer ***n***

```
grayslice(x, n)
```

- Produces a matrix whose values have been reduced to the value 0, 1,…,n-1

Uniform quantization (Cont.)

```
x = imread('newborn.tif');

x1 = grayslice(x,2)*128;

x2 = grayslice(x,4)*64;

x3 = grayslice(x,8)*32;

x4 = grayslice(x,16)*16;

x5 = grayslice(x,32)*8;

x6 = grayslice(x,64)*4;

x7 = grayslice(x,128)*2;
```

Uniform quantization (Cont.)

Dithering

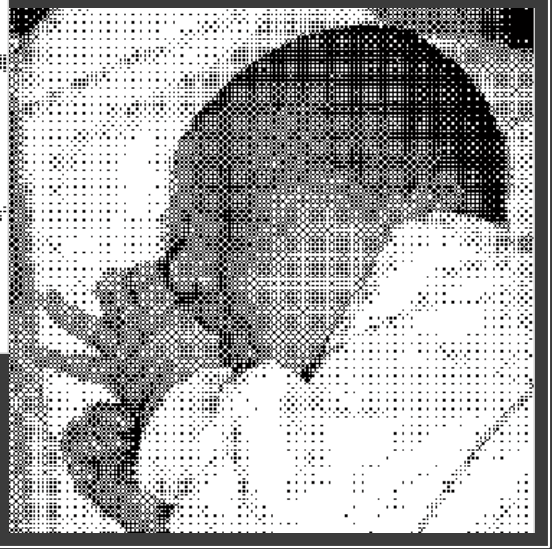
- Dithering refers to the process of reducing the number of colors in an image
- **Why?**
  - Limited number of colors in display or printing
  - Newspaper only has two grayscales – **halftoning**
- **How?**
  - Add random values to the image before quantization
  - Compare the image to a random matrix *r*
  - A darker area will contain more black than white
  - A light area will contain more white than black

Halftoning

- Using one standard matrix $D = \begin{pmatrix} 0 & 128 \\ 192 & 64 \end{pmatrix}$
- Create the matrix *r* by repeating *D*, until it is as big as the image matrix
- Compare *r* with the image matrix:
  - The halftone image is $p(i, j) = \begin{cases} 1 \text{ if } x(i, j) > r(i,j) \\ 0 \text{ if } x(i, j) <= r(i,j) \end{cases}$

Halftoning (Cont.)

```
i=imread('newborn.tif');

D1=[0 128;192 64];

D2=[0 128 32 160;192 64 224 96;48 176 16 144;240 112 208 80];

r1=repmat(D1,128,128);

r2=repmat(D2,64,64);

x1=i > r1;

x2=i > r2;

imshow(x1); figure,imshow(x2);
```



Dithering (4 Levels)

- First quantize by dividing gray value *x(i,j) by 85*

$$q(i, j) = [x(i, j) / 85] \quad \text{(because } 255/3 = 85\text{)}$$

- Suppose now our replicated dither matrix *d(i,j)* is scaled so that its values are in the range 0 – 85
- The final quantization level *p(i,j) is*

$$p(i, j) = q(i,j) + \begin{cases} 1 & \text{if } x(i,j) - 85q(i,j) > d(i,j) \\ 0 & \text{if } x(i,j) - 85q(i,j) > d(i,j) \end{cases}$$

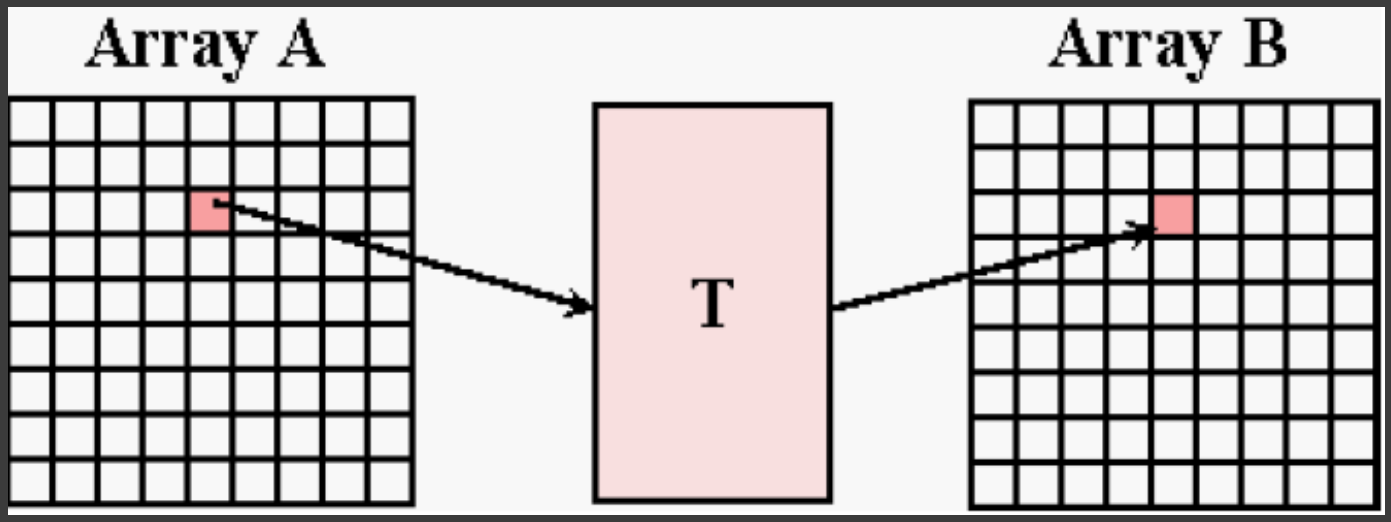
Dithering (4 Levels)

```
x = imread('newborn.tif');

D=[0 56;84 28];

r=repmat(D,128,128);

x=double(x);

q=floor(x/85);

x4=q+(x-85*q>r);

 imshow(uint8(85*x4));
```

Point processing

- Point processing is used to transform an image by operating on individual pixels. If array A represents an input image then an output array B is produced by a transformation

$$B[x, y] = T[A[x, y]]$$



Arithmetic Operations

- These operations act by applying a simple function $y = f(x)$
- To each gray value in the image. Thus $f(x)$ is a function that maps the range 0...255 onto it self
  - Add / Subtract / Multiply/ Divide

Ex. $y = x + c$

$y = cx$

Arithmetic Operations

```
b=imread('caribun.tif');

b1=imadd(b,128);   % y=x+128

b2=imsubtract(b,128);  % y=x-128

b3=immultiply(b,2);   % y=2x

b4=imdivide(b,2);   % y=x/2

b5=imadd(immultiply(b,0.5),128);   % y=x/2+128
```

Arithmetic Operations (Cont.)

Complements

- The complement of a grayscale image is its photographic negative.

```
b=imread('caribun.tif');

bc=imcomplement(b);

%% y=255-x
```

Histogram Processing

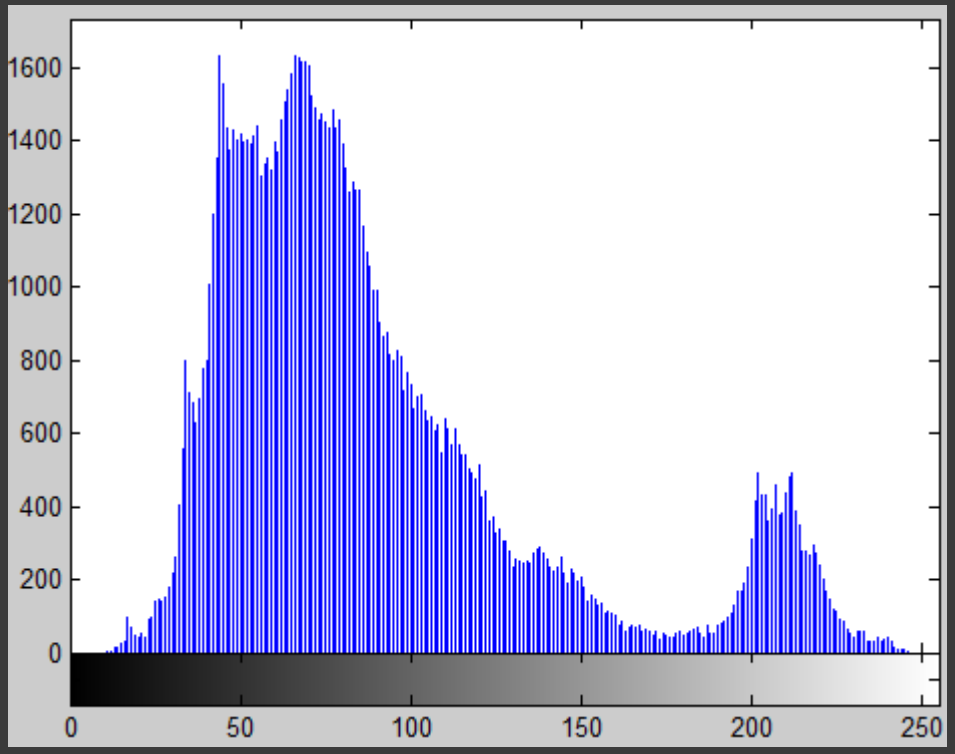
- An image will have low contrast if its brightness values are too concentrated.
- This image has low contrast, making it difficult to see some of the items in the shadows.

Histogram Processing (Cont.)

- View the histogram of an image by using the **imhist** function.

```
h=imread('logging.tif ');
imshow(h)
figure, imhist(h);
```

Histogram Processing (Cont.)



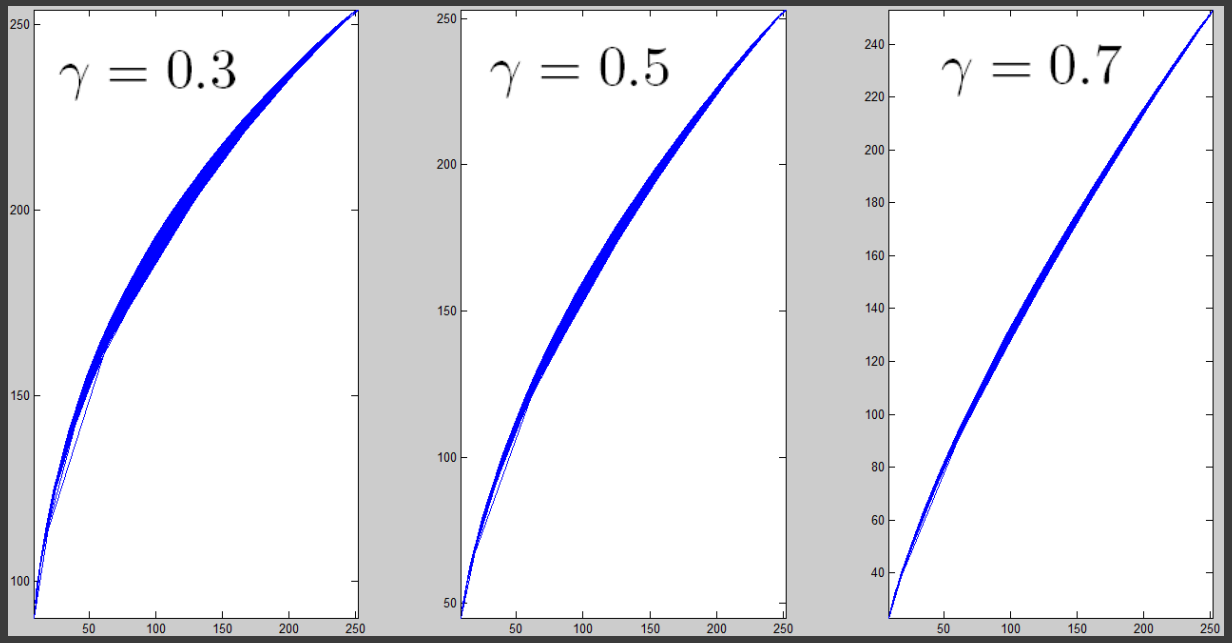
Histogram Stretching

- To perform histogram stretching the **imadjust** function.
- Use of the gamma value to substantially change the appearance of the image

```
h=imread('logging.tif ');
th1=imadjust(h,[ ],[ ],0.3);
th2=imadjust(h,[ ],[ ],0.5);
th3=imadjust(h,[ ],[ ],0.7);
```

Histogram Stretching (Cont.)

Original



Histogram Stretching (Cont.)