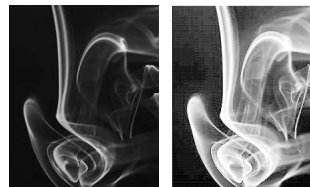


Three Classes of Image Processing Operations by Direct Manipulation of Gray Levels

- Any image-processing operation transforms the gray values of the pixels.
- Image-processing operations may be divided into three classes based on the information required to perform the transformation.
- From the simplest to the most complex, they are as follows:
 - **Point operations : chap.4**
 - **Neighborhood processing (spatial filter) : chap.5**
 - **Geometrical transforms: chap.6**



Chapter 4: Point Processing

Arithmetic Operations

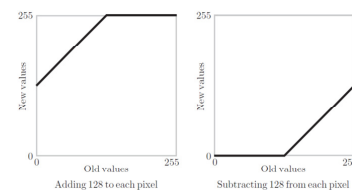
- These operations act by applying a simple function

$$y = f(x)$$

- In each case we may have to adjust the output slightly in order to ensure that the results are integers in the 0 . . . 255 range (**type uint8**)

$$y \leftarrow \begin{cases} 255 & \text{if } y > 255, \\ 0 & \text{if } y < 0. \end{cases}$$

Arithmetic Operations



```
>> b=imread('blocks.tif');
```

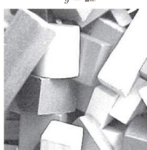
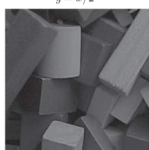
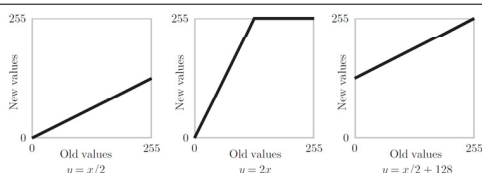
```
>> b1=imadd(b,128);
```

```
>> b2=imsubtract(b,128);
```



Arithmetic Op: Multiplication and Division

```
y = x/2;          b3=immultiply(b,0.5); or b3=imdivide(b,2)
y = 2x;          b4=immultiply(b,2);
y = x/2 + 128;   b5=imadd(immultiply(b,0.5),128);
                  or b5=imadd(imdivide(b,2),128);
```

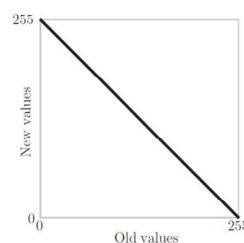


4.2 Arithmetic Op: Complements

- The **complement** of a grayscale image is its photographic negative (= **image negative**)

- **type double** (0.0~1.0) : 1-m

- **type uint8** (0~255) : 255-m



```
>> bc=imcomplement(b);
>> imshow(bc)
```



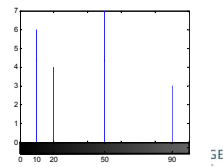
4.3 Histogram

- A graph representing pixel value vs the number of its occurrence in the image
(= 확률 밀도 함수, Probability Density Function (PDF))

$$p_r(r_k) = \frac{n_k}{n} \quad k = 0, 1, 2, \dots, L-1$$

- n : total number of pixels in an image
- n_k : number of pixel value r_k
- L : number of **intensity levels** (pixel values)
- Graph of r_k vs. $p_r(r_k)$

EX: $a = [10 \ 10 \ 10 \ 10 \ 10;$
 $20 \ 20 \ 20 \ 20 \ 10;$
 $50 \ 50 \ 50 \ 50 \ 50;$
 $90 \ 90 \ 90 \ 50 \ 50]$



Histogram Example

```
>> p=imread('pout.tif');
>> imshow(p),figure,imhist(p),axis tight
```

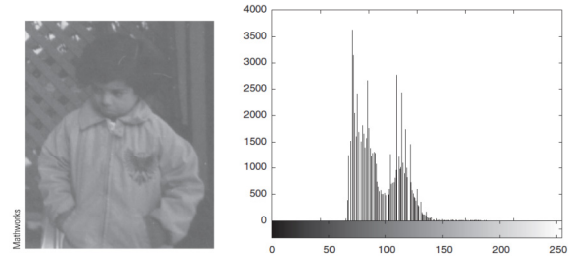
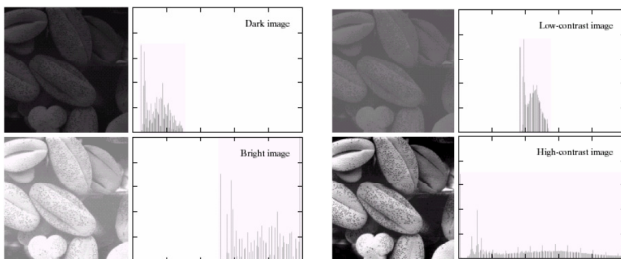


FIGURE 4.8 The image `pout.tif` and its histogram.

Histogram and Image Properties



Histogram(Contrast) Stretching

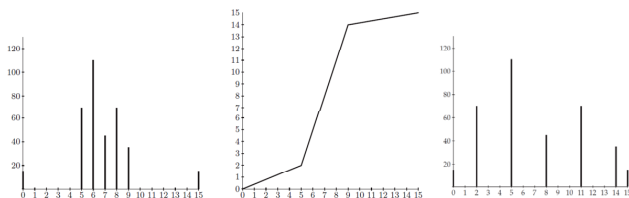
- A table of the numbers n_i of gray values

Gray level i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_i	15	0	0	0	0	70	110	45	70	35	0	0	0	0	0	15

(with $n = 360$, as before)

- We can stretch out the gray levels in the center of the range by applying the piecewise linear function

Contrast Stretching



Transformation function

$$j = \frac{14-2}{9-5}(i-5) + 2$$

i : original gray level
 j : its result after the transformation

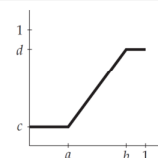
Transformation Lookup table

i	5	6	7	8	9
j	2	5	8	11	14

This function has the effect of stretching the gray levels 5-9 to gray levels 2-14

Contrast Stretching in Matlab

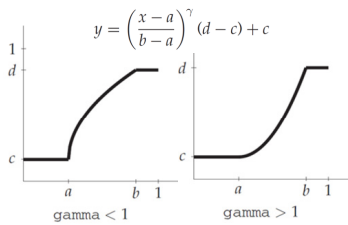
```
imadjust(im,[a,b],[c,d])
```



- imadjust** is designed to work equally well on images of type `double`, `uint8`, or `uint16`
- the values of a , b , c , and d must be between 0 and 1
- imadjust** automatically converts the image `im` (if needed) to be of type `double`
- Note** that `imadjust` does not work quite in the same way as shown in the figure.
- The `imadjust` function has one other optional parameter: the gamma value.

$$y = \left(\frac{x-a}{b-a} \right)^\gamma (d-c) + c$$

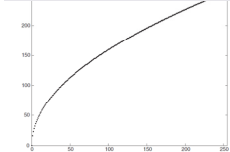
Contrast Stretching with γ



```
>> t=imread('tire.tif');
>> th=imadjust(t,[],[],0.5);
>> imshow(t),figure,imshow(th)
```



```
>> plot(t,th,'.'),axis tight
```



CENGAGE Learning

CS: Piecewise Linear Transformation Function

- How to implement in Matlab ?

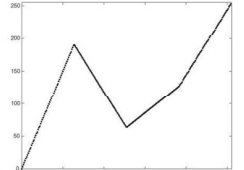
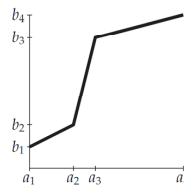
```
pix=find(im >= a(i) & im < a(i+1));
out(pix)=(im(pix)-a(i)) * (b(i+1)-b(i)) / (a(i+1)-a(i)) + b(i)
```

im : input image, **out** : output image

$$y = \frac{b_{i+1} - b_i}{a_{i+1} - a_i} (x - a_i) + b_i$$

defined in p.77

```
>> th=histpwl(t,[0 .25 .5 .75 1],[0 .75 .25 .5 1]);
>> imshow(th)
>> figure,plot(t,th,'.'),axis tight
```



CENGAGE Learning

4.3.2 Histogram Equalization

- An entirely automatic procedure
- Suppose our image has L different gray levels, $0, 1, 2, \dots, L-1$, and gray level i occurs n_i times in the image

$$\left(\frac{n_0 + n_1 + \dots + n_i}{n} \right) (L-1)$$

Where $n = n_0 + n_1 + n_2 + \dots + n_{L-1}$

CENGAGE Learning

4.3.2 Histogram Equalization

- WHY IT WORKS** If we were to treat the image as a continuous function $f(x, y)$ and the histogram as the area between different contours, then we can treat the histogram as a probability density function.

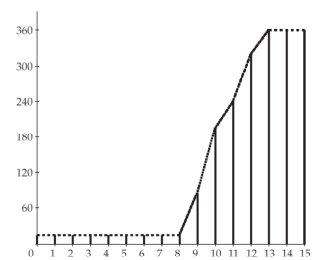


FIGURE 4.22 The cumulative histogram.

CENGAGE Learning

Histogram Equalization : How to stretch?

- Pixels $r(n_1, n_2)$ are transformed to $s(n_1, n_2)$

$$s(n_1, n_2) = T[r(n_1, n_2)]$$

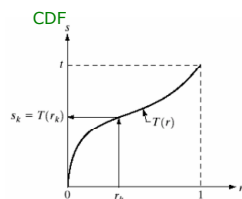
- Trans. func.(T): CDF(cumulative distribution function) of random variable r

$$S = T(r) = \int_0^r p_r(w) dw$$

(CDF) - (pdf)

WHY?

CDF makes pdf (probability density function, pdf) $p_s(s)$ after HE uniform distribution
=> purpose of HE



CENGAGE Learning

HE Example

- Suppose a 4-bit grayscale image has the histogram, associated with a table of the numbers n_i of gray values

Gray level i	0	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15
n_i	15	0	0	0	0	0	0	0	0	70	110	45	80	40	0	0

Gray level i	pdf n_i	CDF Σn_i	$(1/24)\Sigma n_i$	Rounded value
0	15	15	0.63	1
1	0	15	0.63	1
2	0	15	0.63	1
3	0	15	0.63	1
4	0	15	0.63	1
5	0	15	0.63	1
6	0	15	0.63	1
7	0	15	0.63	1
8	0	15	0.63	1
9	70	85	3.55	4
10	110	195	8.13	8
11	45	240	10	10
12	80	320	13.33	13
13	40	360	15	15
14	0	360	15	15
15	0	360	15	15

FIGURE 4.10 Another histogram indicating poor contrast.

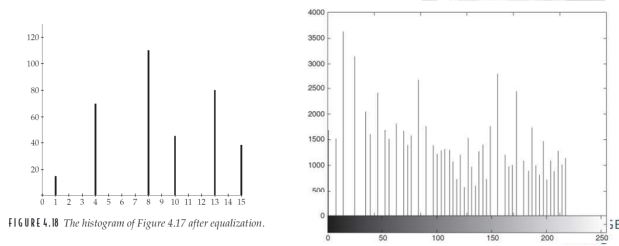
$n_i/360$

CENGAGE Learning

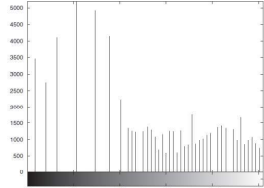
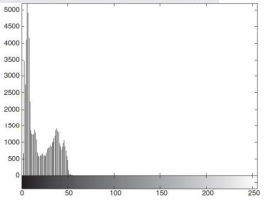
HE Example

```
>> p=imread('pout.tif');
>> ph=histeq(p);
>> imshow(ph),figure,imhist(ph),axis tight
```

For the exam, you have to be able to do it by hand as well !



```
>> en=imread('engineer.tif');
>> e=imdivide(en,4);
>> imshow(e),figure,imhist(e),axis tight
```

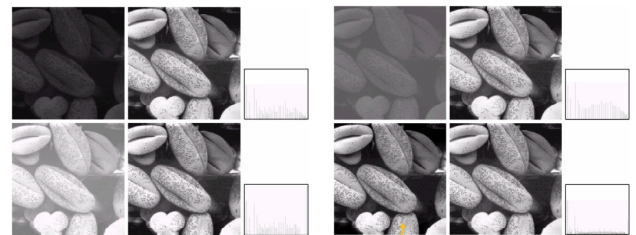


```
>> eh=histeq(e);
>> imshow(eh),figure,imhist(eh),axis tight
```

Histogram Equalization: Step by Step

1. Get histogram(pdf) of input image.
2. Get transformation function $T()$ by calculating cdf and compose a lookup table.
3. Using $T()$, change input pixel value $r \rightarrow$ output pixel value s .
4. Put the pixel value s at the corresponding pixel location of output image.

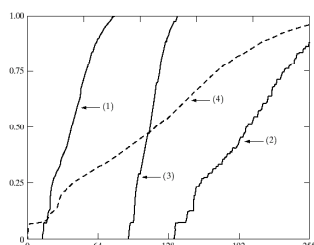
- HE 가 Histogram Equalization: Results



Drawback of HE
Transformation function is fixed.
does not work for high-contrast image

Histogram Equalization : CDFs of the previous slide

FIGURE 3.18 Transformation functions (1) through (4) were obtained from the histograms of the images in Fig. 3.17(a), using Eq. (3.3-8).



Histogram Matching(Specification)

- Histogram equalization generates a uniform histogram
 - For interactive image enhancement, the user may like to result in a customized histogram
- > Use Histogram Specification

Histogram Specification

- r : pixel values of original image
- u : pixel values of **desired** image : PDF should be pre-determined.
- s : pixel values of histogram-equalized image
- For continuous data, HE of r and u results in s . (uniform histogram)

$$s = T(r) = \int_0^r P_r(\omega) d\omega \quad \text{CDF(누적분포함수)}$$

$$v = G(u) = \int_0^u P_u(\omega) d\omega$$

$$u = G^{-1}(s) = G^{-1}(T(r)) \quad r \rightarrow u \text{ conversion}$$

- Specify a particular probability density function $G(u)$ then calculate $G^{-1}(T(r))$ for histogram specification.

Graphical Interpretation of Histogram Specification

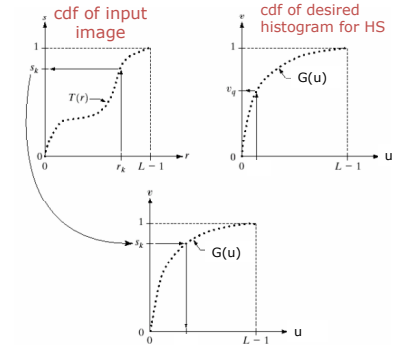
step1 : get histogram-equalized pixel values

$$s = T(r) = \int_0^r P_r(\omega) d\omega$$

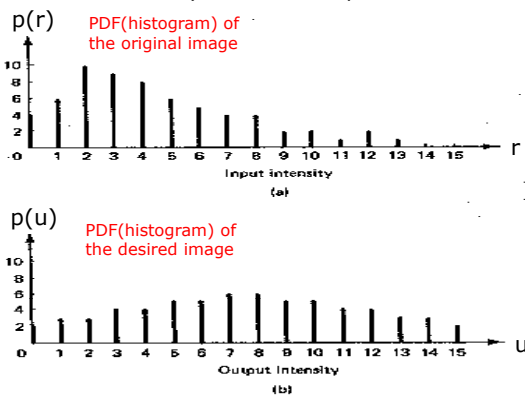
$$v = G(u) = \int_0^u P_u(\omega) d\omega$$

$$u = G^{-1}(s) = G^{-1}(T(r))$$

step2 : using histogram equalized pixel values as input, get histogram-specified pixel values



An Example of Histogram Specification with 15 intensity levels and 64 pixels



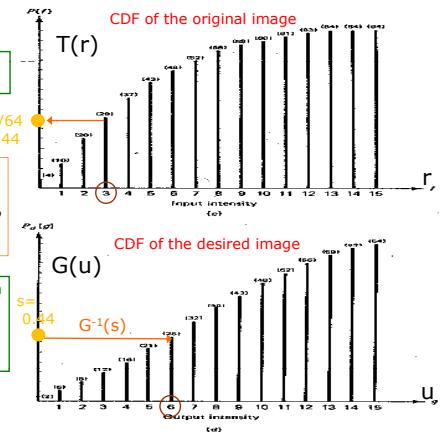
step1 : get histogram-equalized pixel values

$$s = T(r) = \int_0^r P_r(\omega) d\omega$$

$$v = G(u) = \int_0^u P_u(\omega) d\omega$$

$$u = G^{-1}(s) = G^{-1}(T(r))$$

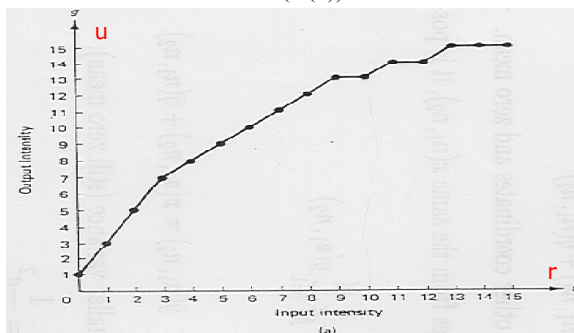
step2 : using histogram equalized pixel values as input, get histogram-specified pixel values



- histogram-equalized pixel value of pixel value 3 : floor(15*28/64) = 7
- histogram-specified pixel value of pixel value 3 : $G^{-1}(28/64) = 6$

An Example of Histogram Specification : Look-up Table

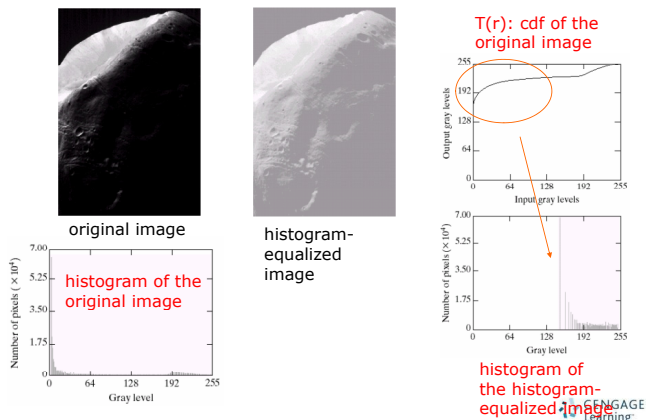
$$u = G^{-1}(T(r))$$



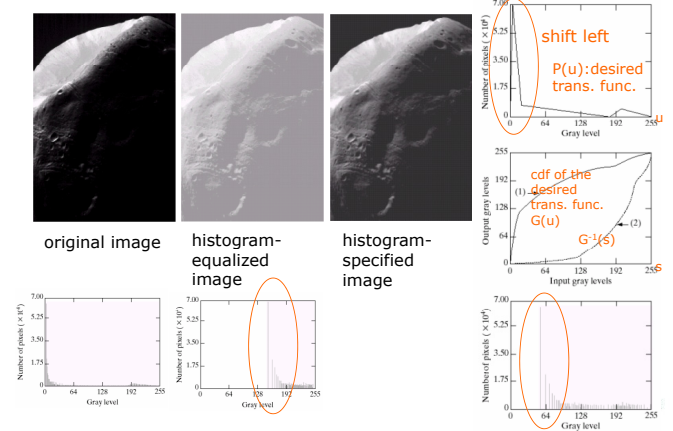
Histogram Specification: Step by Step

1. Get histogram $P(r)$ of input image.
2. Get $s=T(r)$ for histogram equalization.
3. **Design the desired histogram $P(u)$** and make transformation function $G(u)$ for HS.
(You might want to do it manually.)
4. Generate a look-up table(LUT) to convert input pixel values (r) to output values (u) by $T(r)$ and $G^{-1}(s)$.
5. Put the converted pixel values in the output image.

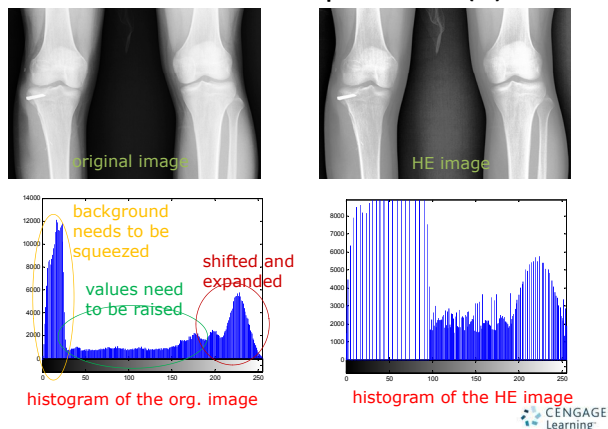
Example of Histogram Specification



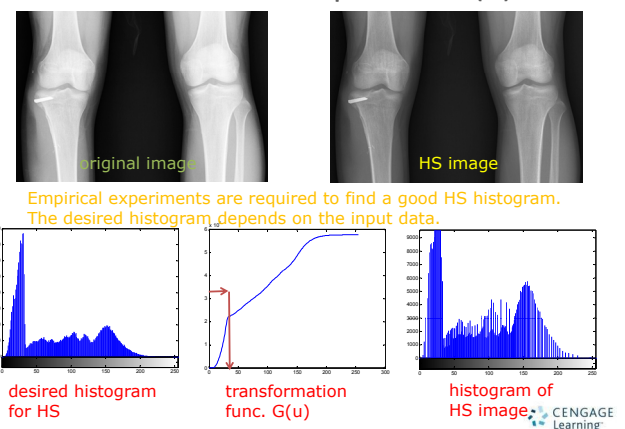
Example of Histogram Specification



A Practical Example of HS (1)



A Practical Example of HS (2)



4.4 Lookup Tables

- Point operations can be performed very effectively by using a **lookup table**, known more simply as an **LUT**
- e.g., the LUT corresponding to division by 2 looks like

Index: 0 1 2 3 4 5 ... 250 251 252 253 254 255

LUT: 0 0 1 1 2 2 ... 125 125 126 126 127 127

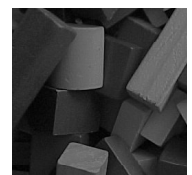
4.4 Lookup Tables

- If T is a lookup table in MATLAB and im is our image, the lookup table can be applied by the simple command

$T(im+1)$

- e.g.,

```
>> T=uint8(floor([0:255]/2));
>> b = imread('image.tif');
>> b2 = T(b);
```



4.4 Lookup Tables

- As another example, suppose we wish to apply an LUT to implement the contrast-stretching function

$$\begin{aligned} y &= \frac{64}{96}x, \\ y &= \frac{192-64}{160-96}(x-96) + 64, \\ y &= \frac{255-192}{255-160}(x-160) + 192 \end{aligned} \quad \Rightarrow \quad \begin{aligned} y &= 0.6667x, \\ y &= 2x - 128, \\ y &= 0.6632x + 85.8947 \end{aligned}$$

LUT Generation in Matlab

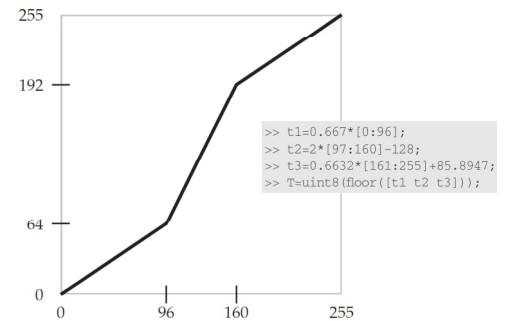


FIGURE 4.2 A piecewise linear contrast-stretching function.