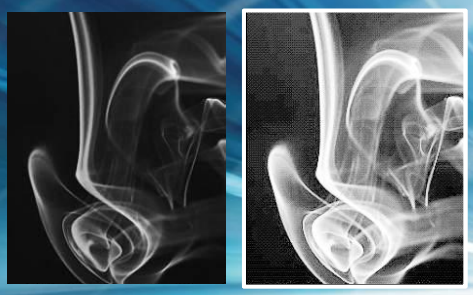
Introduction to Digital Image Processing with MATLAB® Asia Edition

McAndrew·Wang·Tseng



Chapter 4: Point Processing

CENGAGE Learning

4.1 Introduction

- 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.

CENGAGE Learning

4.1 Introduction

From the most complex to the simplest, they are as follows:

- **Transforms**
- **Neighborhood processing**
- **Point operations**

FIGURE 4.1

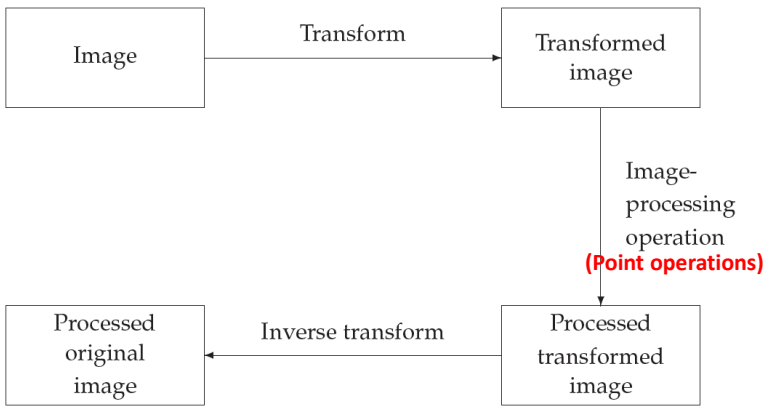


FIGURE 4.1 *Schema for transform processing.*

4.2 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)**

FIGURE 4.2

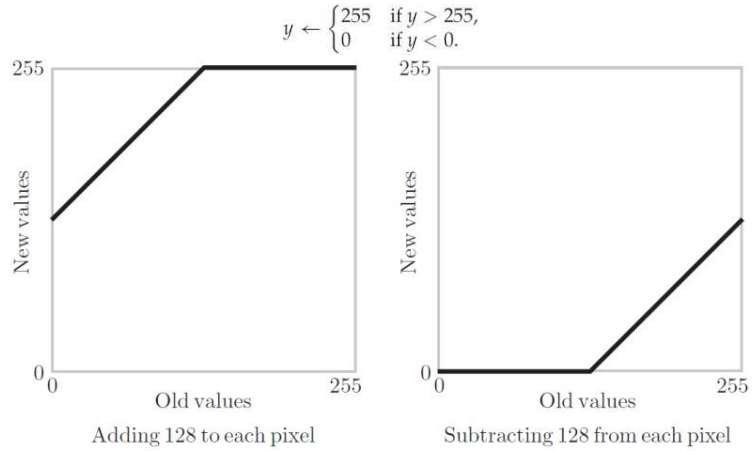


FIGURE 4.2 *Adding and subtracting a constant.*

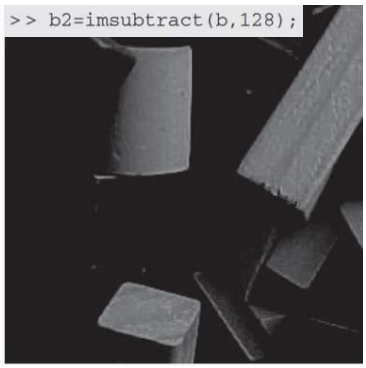
FIGURE 4.3

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

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

(a)

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



(b)

FIGURE 4.3 *Arithmetic operations on an image: adding or subtracting a constant. (a) b1: Adding 128; (b) b2: Subtracting 128.*

FIGURE 4.4

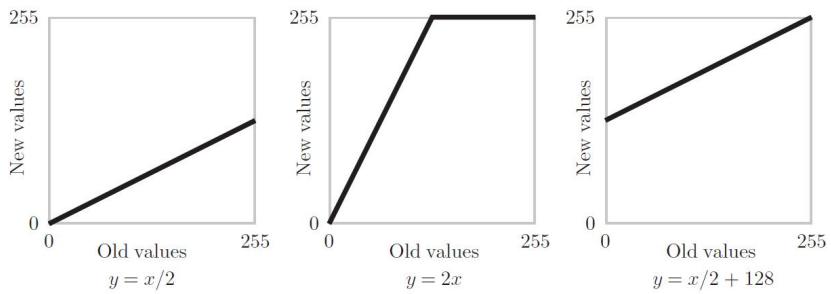


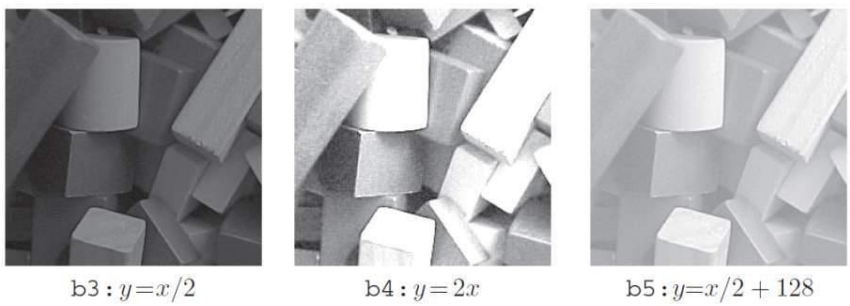
FIGURE 4.4 *Using multiplication and division.*

TABLE 4.1

TABLE 4.1 *Implementing pixel multiplication by MATLAB commands.*

$y = x/2$	`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);`

FIGURE 4.5



b3 : $y=x/2$ b4 : $y=2x$ b5 : $y=x/2+128$

FIGURE 4.5 *Arithmetic operations on an image: multiplication and division.*

4.2 Arithmetic Operations

COMPLEMENTS

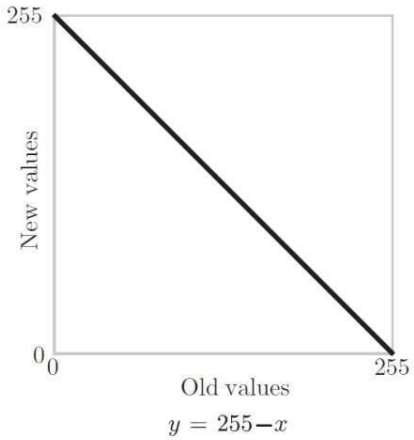
- The **complement** of a grayscale image is its photographic negative
- **type double** (0.0~1.0)

```
1-m
```

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

```
255-m
```


FIGURE 4.6




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

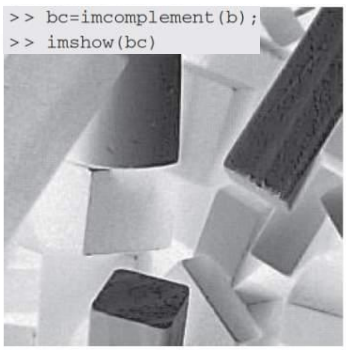


FIGURE 4.6 *Image complementation.*

4.3 Histograms

- A graph indicating the number of times each gray level occurs in the image

```
a = [10 10 10 10 10;
     20 20 20 20 10;
     50 50 50 50 50;
     90 90 90 50 50]
```

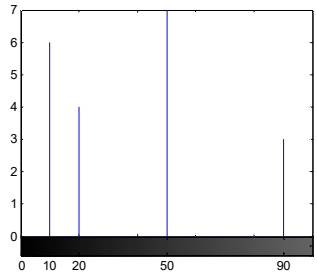


FIGURE 4.8

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

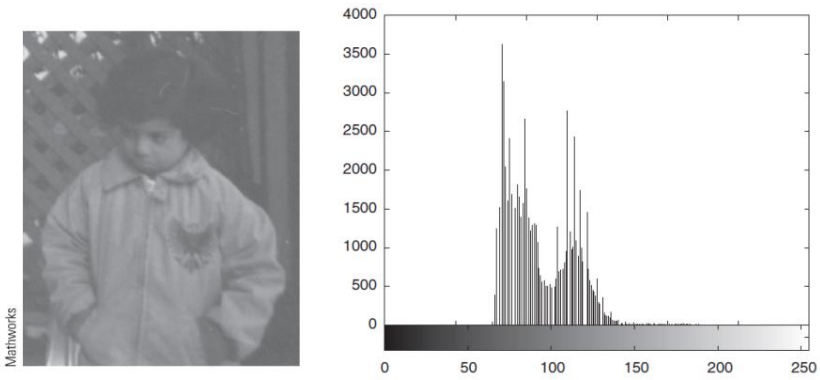


FIGURE 4.8 *The image* `pout.tif` *and its histogram.*

4.3.1 Histogram 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

FIGURE 4.9

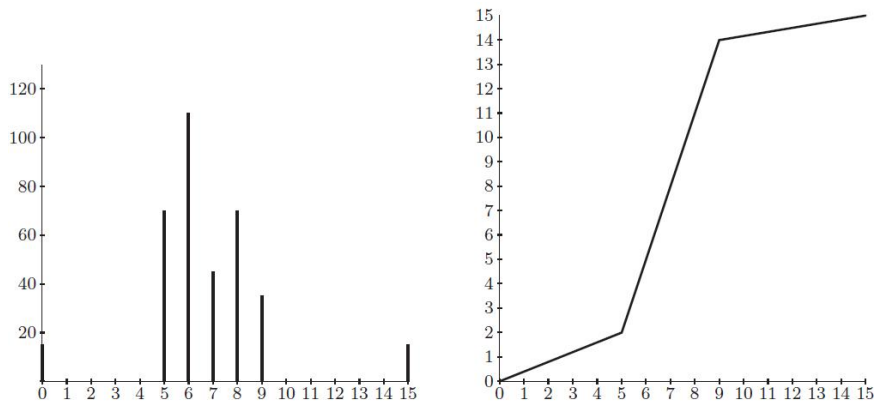


FIGURE 4.9 *A histogram of a poorly contrasted image and a stretching function.*

4.3.1 Histogram Stretching

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

- where i is the original gray level and j is its result after the transformation

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

4.3.1 Histogram Stretching

- the corresponding histogram

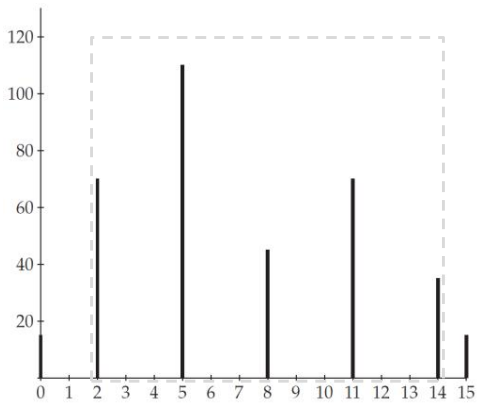


FIGURE 4.10

- **Use of `imadjust`**

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

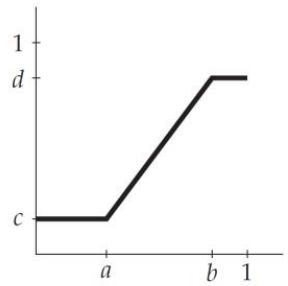


FIGURE 4.10 *The stretching function given by* `imadjust`.

4.3.1 Histogram Stretching

- `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
- the function automatically converts the image `im` (if needed) to be of type `double`

4.3.1 Histogram Stretching

- Note that `imadjust` does not work quite in the same way as shown in Figure 4.9
- The `imadjust` function has one other optional parameter: the `gamma` value

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

FIGURE 4.11

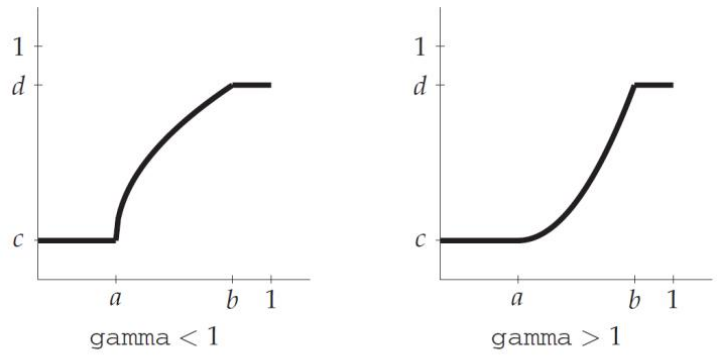
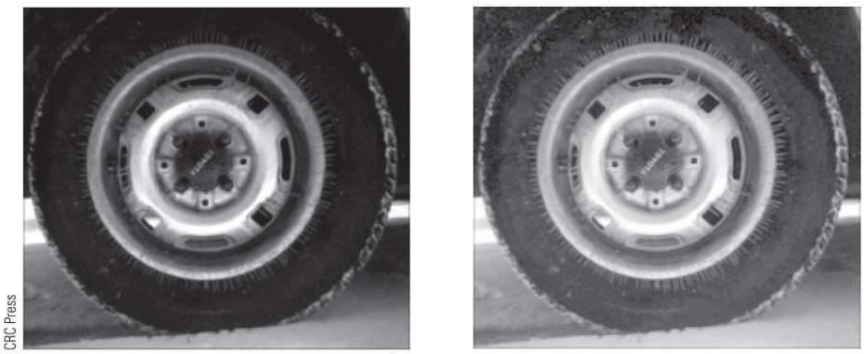


FIGURE 4.11 *The `imadjust` function with `gamma` not equal to 1.*

FIGURE 4.12

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



CRC Press

FIGURE 4.12 *The tire image before and after adjustment with the gamma value.*

Ch4-p.75

CENGAGE Learning

23

FIGURE 4.13

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

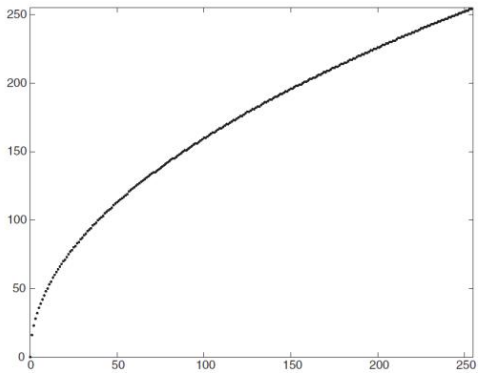


FIGURE 4.13 *The function used in Figure 4.12.*

Ch4-p.76

CENGAGE Learning

24

4.3.1 Histogram Stretching

- **A PIECE WISE LINEAR-STRETCHING FUNCTION**

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

- The heart of this function will be the lines

```
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)
```

where `im` is the input image and `out` is the output image

FIGURE 4.14

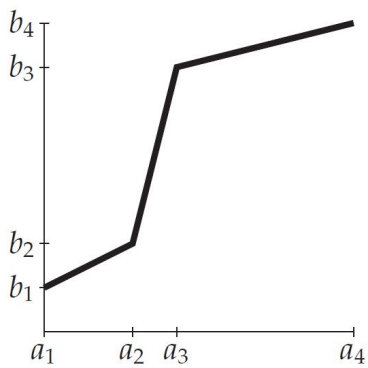


FIGURE 4.14 *A piecewise linear-stretching function.*

FIGURE 4.16

Refer to figure 4.15

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

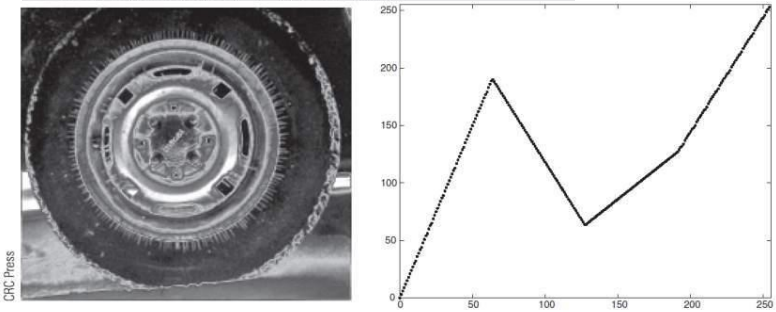


FIGURE 4.16 *The tire image after piecewise linear-stretching and piecewise linear-stretching function.*

4.3.2 Histogram Equalization

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

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

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

4.3.2 Histogram Equalization

- **EXAMPLE** Suppose a 4-bit grayscale image has the histogram shown in Figure 4.17, 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

29 Ch4-p.78

CENGAGE Learning

FIGURE 4.17

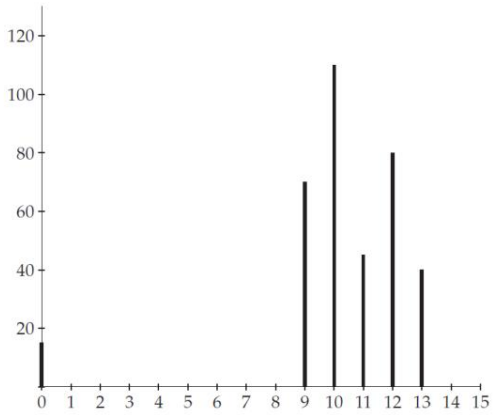


FIGURE 4.17 *Another histogram indicating poor contrast.*

30 Ch4-p.79

CENGAGE Learning

4.3.2 Histogram Equalization

$(L\text{-}1)/n$

Gray level i	n_i	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	3.65	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 4.18

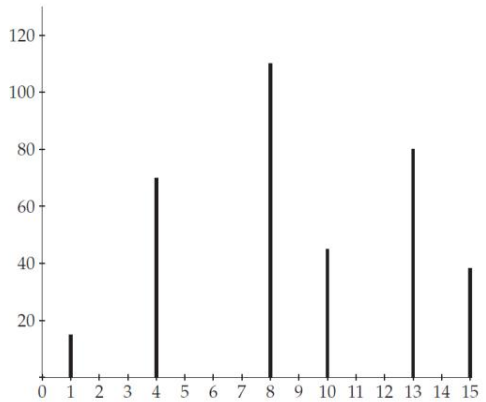


FIGURE 4.18 *The histogram of Figure 4.17 after equalization.*

FIGURE 4.19

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

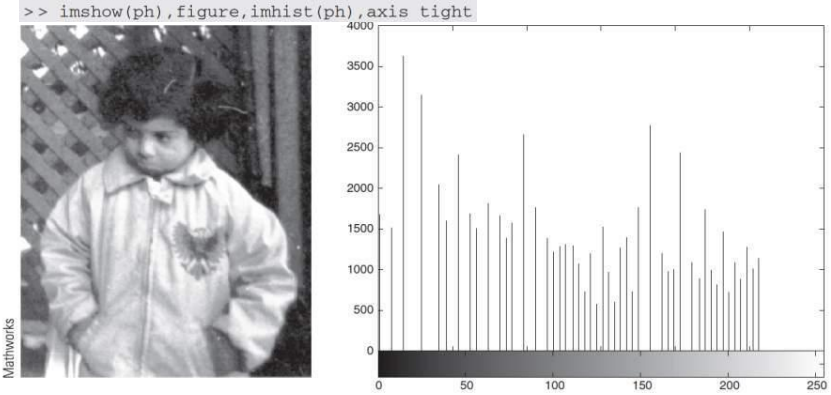


FIGURE 4.19 *The histogram of Figure 4.8 after equalization.*

Ch4-p.80

33

FIGURE 4.20

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

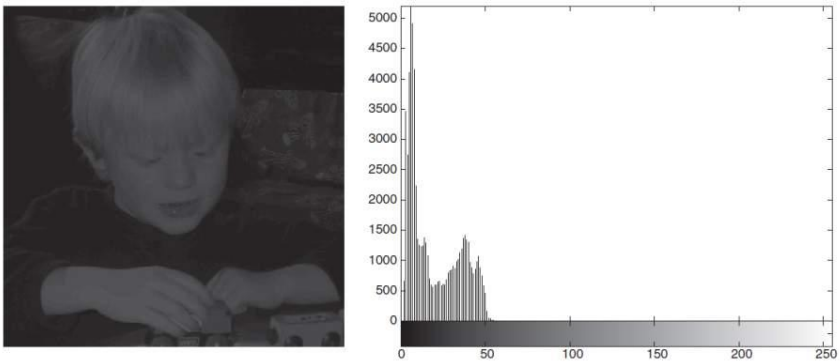


FIGURE 4.20 *The darkened version of* `engineer.tif` *and its histogram.*

Ch4-p.81

34

FIGURE 4.21

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

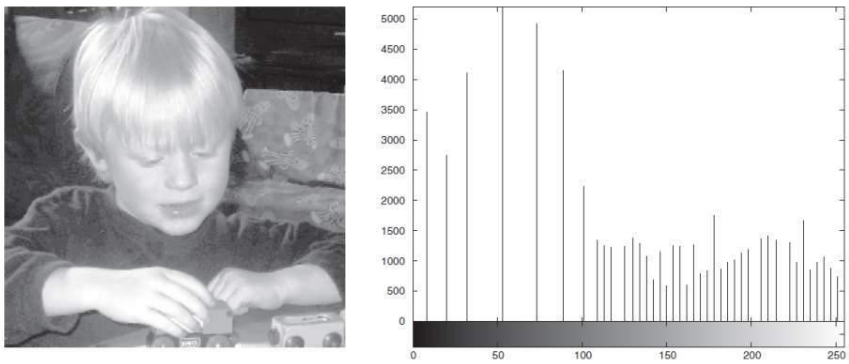


FIGURE 4.21 *The image from Figure 4.20 equalized and its histogram.*

Ch4-p.82

35

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.

Ch4-p.82-83

36

FIGURE 4.22

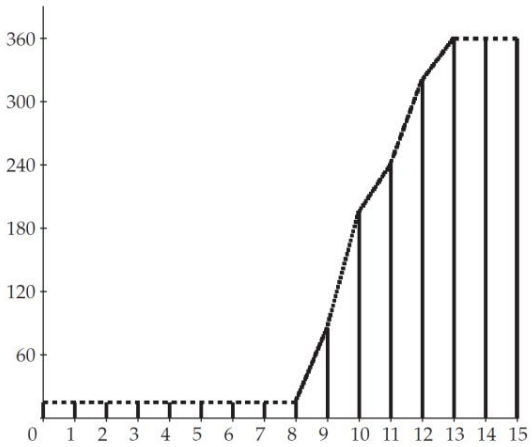


FIGURE 4.22 *The cumulative histogram.*

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

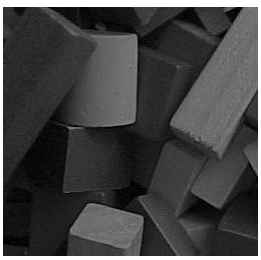
4.4 Lookup Tables

- 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 = b+1;
>> b2 = T(b+1);
```



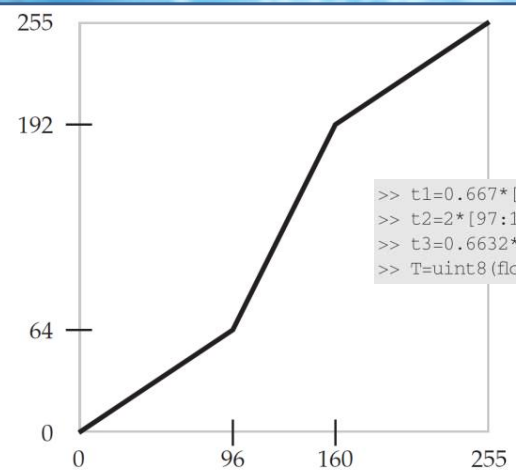
4.4 Lookup Tables

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

$$y = \frac{64}{96}x,$$
$$y = \frac{192-64}{160-96}(x-96)+64,$$
$$y = \frac{255-192}{255-160}(x-160)+192$$

$$y = 0.6667x,$$
$$y = 2x - 128,$$
$$y = 0.6632x + 85.8947$$

FIGURE 4.23




```
>> t1=0.667*[0:96];
>> t2=2*[97:160]-128;
>> t3=0.6632*[161:255]+85.8947;
>> T=uint8(floor([t1 t2 t3]));
```

FIGURE 4.23 *A piecewise linear contrast-stretching function.*

41 Ch4-p.84