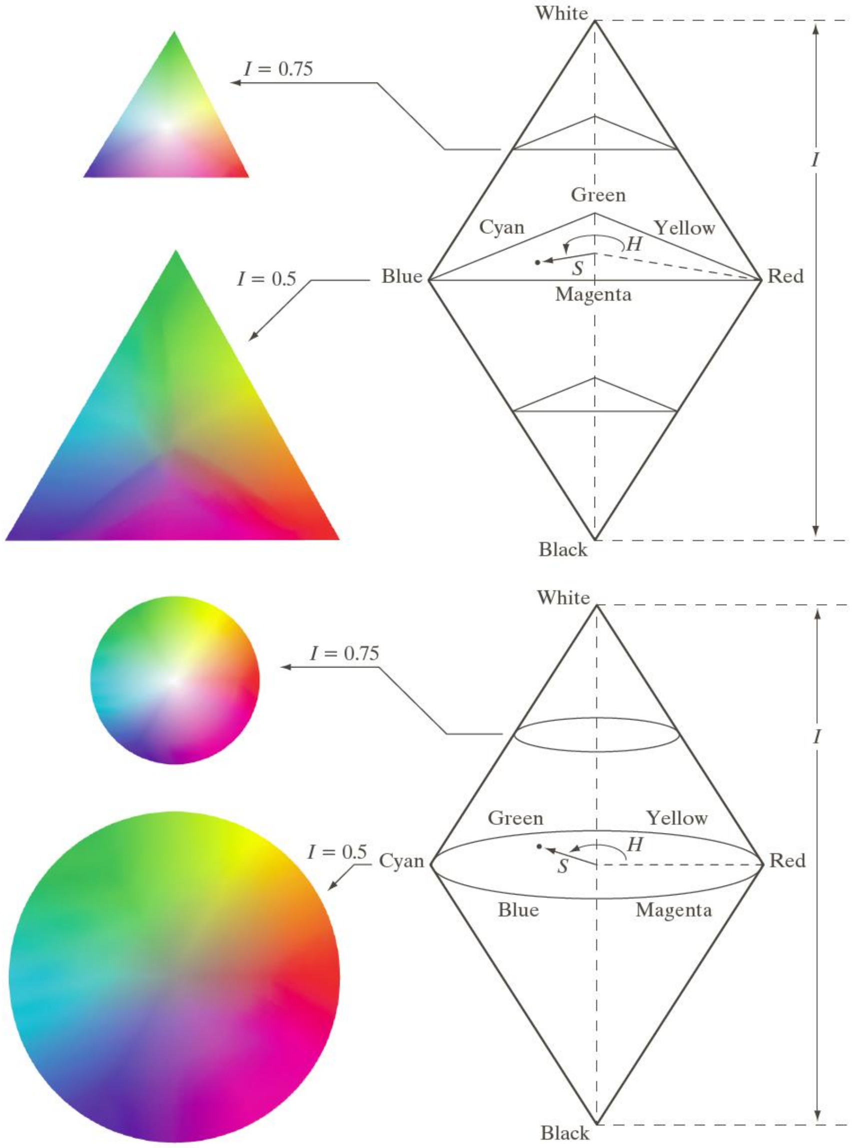
Color Image Processing

HSI Model


White
I = 0.75
Green
Cyan
Yellow
H
S
I = 0.5
Blue
Red
Magenta
I
Black
White
I = 0.75
I
Green
Yellow
H
I = 0.5
Cyan
S
Red
Blue
Magenta
Black

RGB to HSI

$$H = \begin{cases} \theta & \text{if } B \leq G \\ 360 - \theta & \text{if } B > G \end{cases}$$

$$\theta = \cos^{-1}\left\{ \frac{\frac{1}{2}[(R-G)+(R-B)]}{[(R-G)^2+(R-B)(G-B)]^{1/2}} \right\}$$

$$S = 1 - \frac{3}{(R+G+B)}[\min(R, G, B)]$$

$$I = \frac{1}{3}(R+G+B)$$

HSI to RGB

RG Sector

$$B = I(1 - S)$$

$$R = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$$

$$G = 3I - (R + B)$$

HSI to RGB

GB Sector

$$R = I(1 - S)$$

$$G = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$$

$$B = 3I - (R + G)$$

HSI to RGB

BR Sector

$$G = I(1 - S)$$

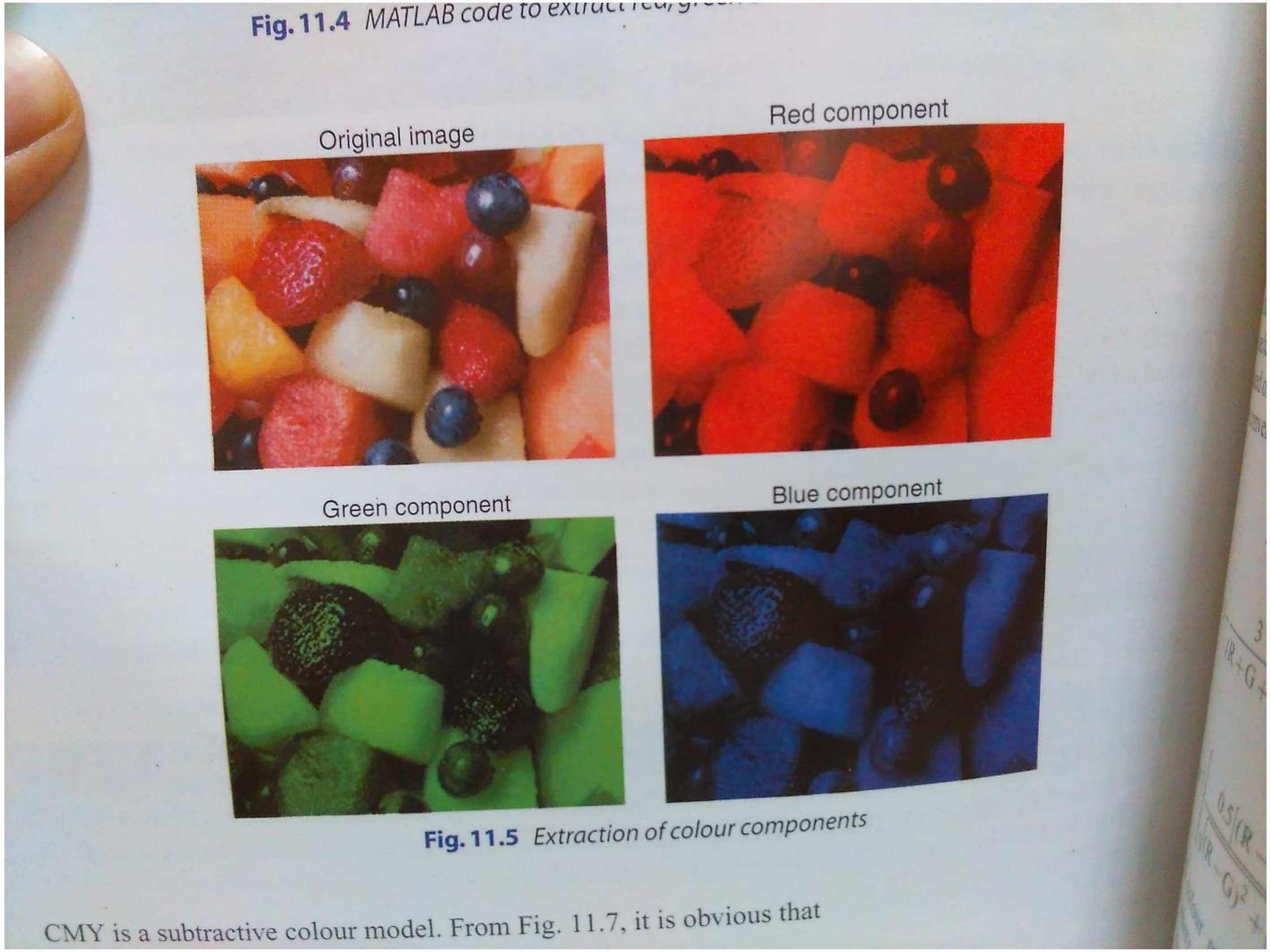
$$B = I\left[1 + \frac{S\cos H}{\cos(60^\circ - H)}\right]$$

$$R = 3I - (G + B)$$

Color Extraction

- RGB=imread('mixedfruit.bmp');
- R=RGB;
- G=RGB;
- B=RGB;
- R(:,:,2)=0;
- R(:,:,3)=0;
- G(:,:,1)=0;
- G(:,:,3)=0;
- B(:,:,1)=0;
- B(:,:,2)=0;
- subplot(2,2,1),imshow(RGB),title('original image')
- subplot(2,2,2),imshow(R),title('Red Component')
- subplot(2,2,3),imshow(G),title('Green Component')
- subplot(2,2,4),imshow(B),title('Blue Component')

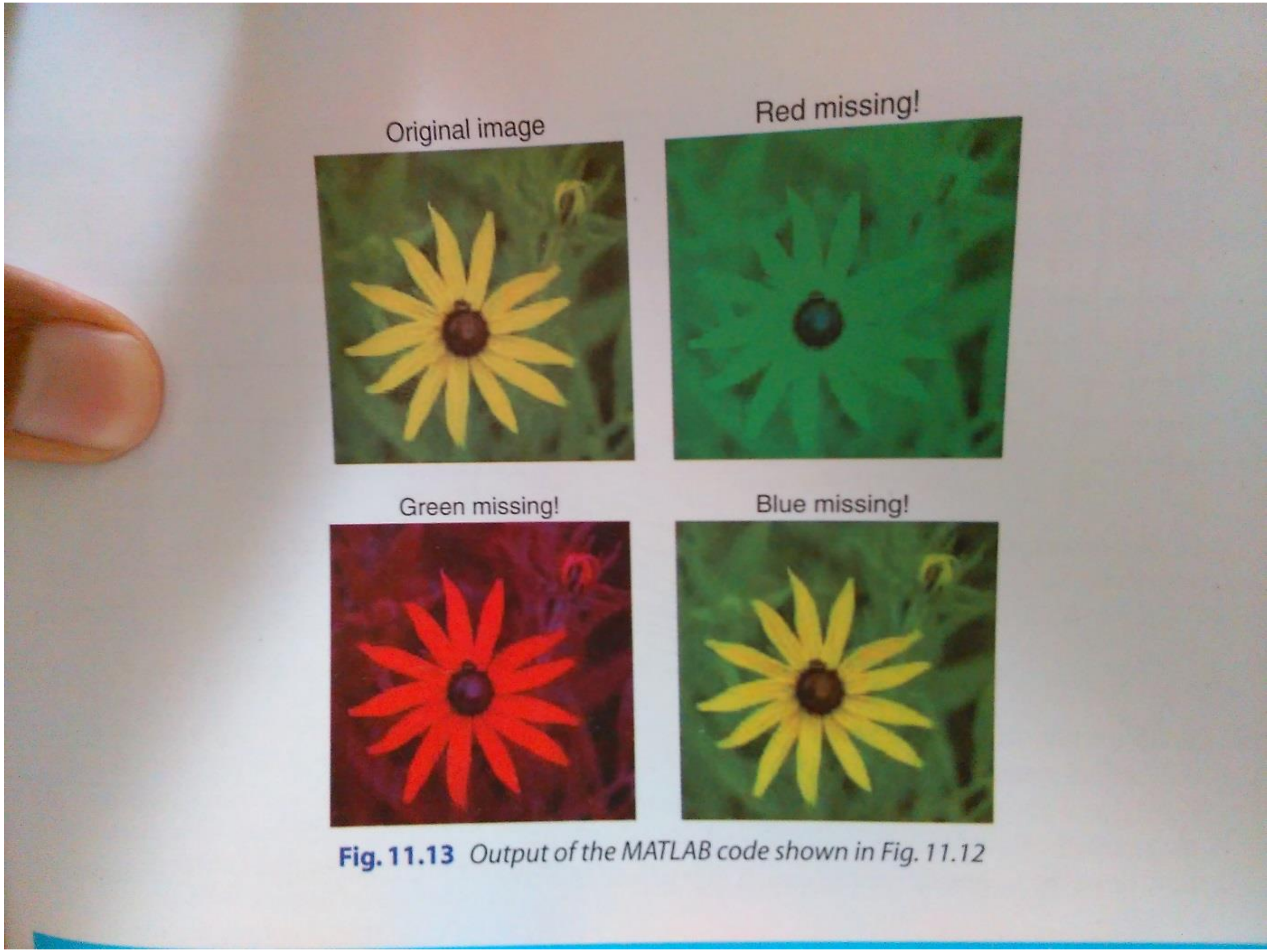
Color Extraction



RGB Plane Removal

- clc
- clear all
- close all
- a=imread('C:\Documents and Settings\esakki\My Documents\My Pictures\fl1.bmp');
- a1=a;
- b1=a;
- c1=a;
- a1(:,:,1)=0;
- b1(:,:,2)=0;
- c1(:,:,3)=0;
- imshow(a),title('original image')
- figure,imshow(a1),title('Red Missing!')
- figure,imshow(b1),title('Green Missing!')
- figure,imshow(c1),title('Blue Missing!')

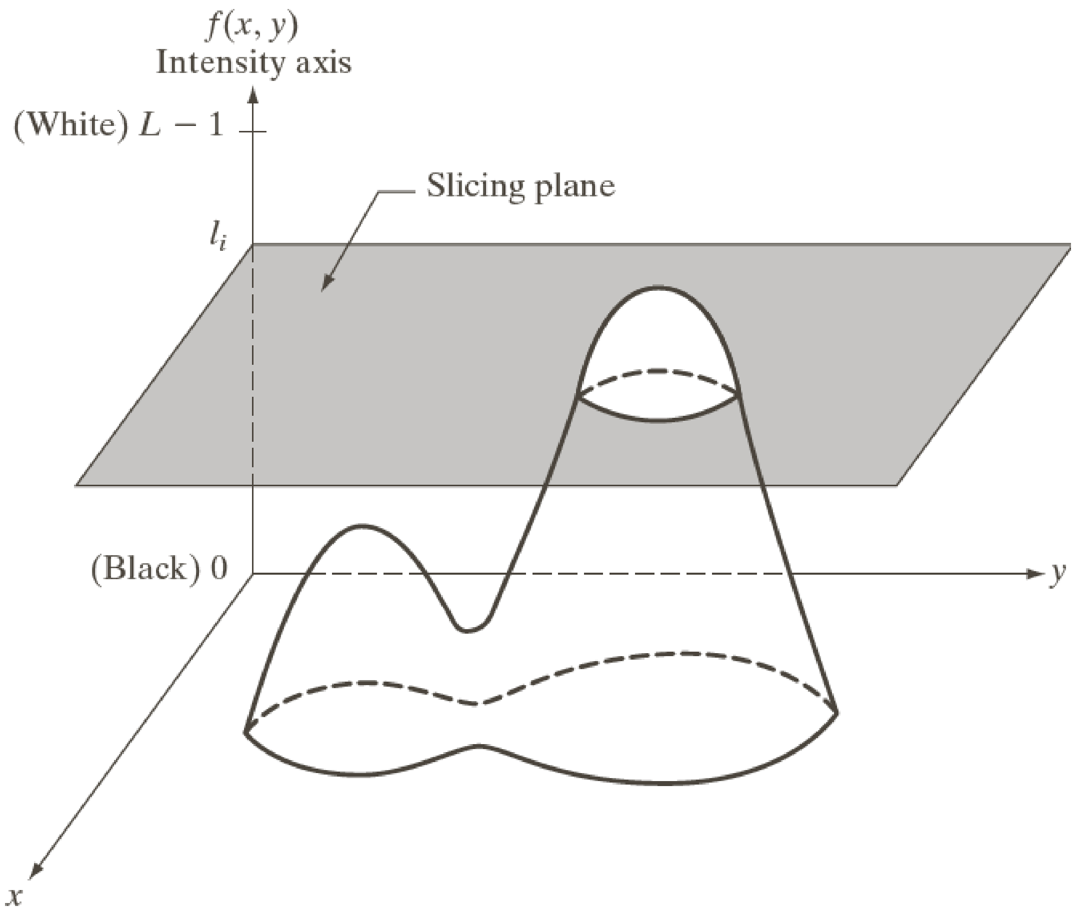
RGB Plane Removal



Pseudocolor Image Processing

- Intensity Slicing
- Intensity to Color Transformation

Intensity Slicing



Intensity Slicing

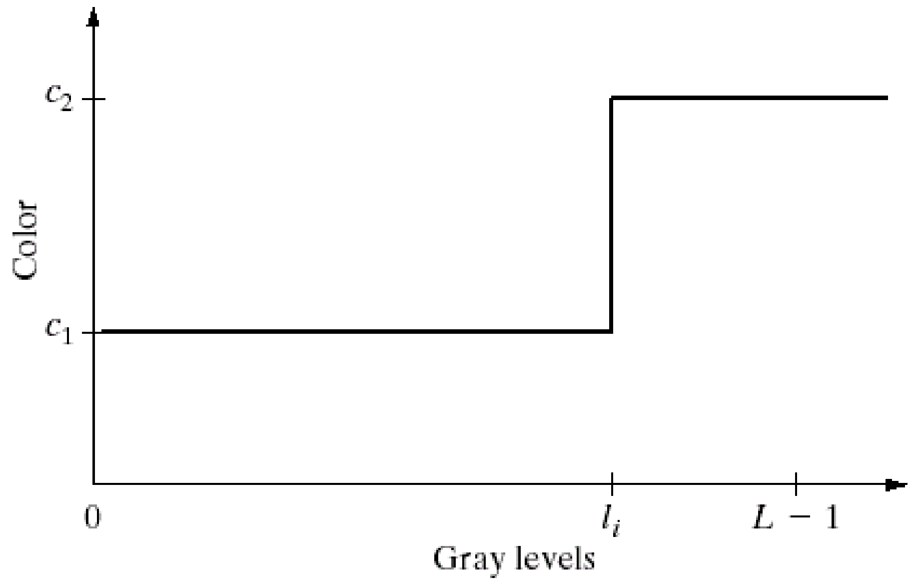
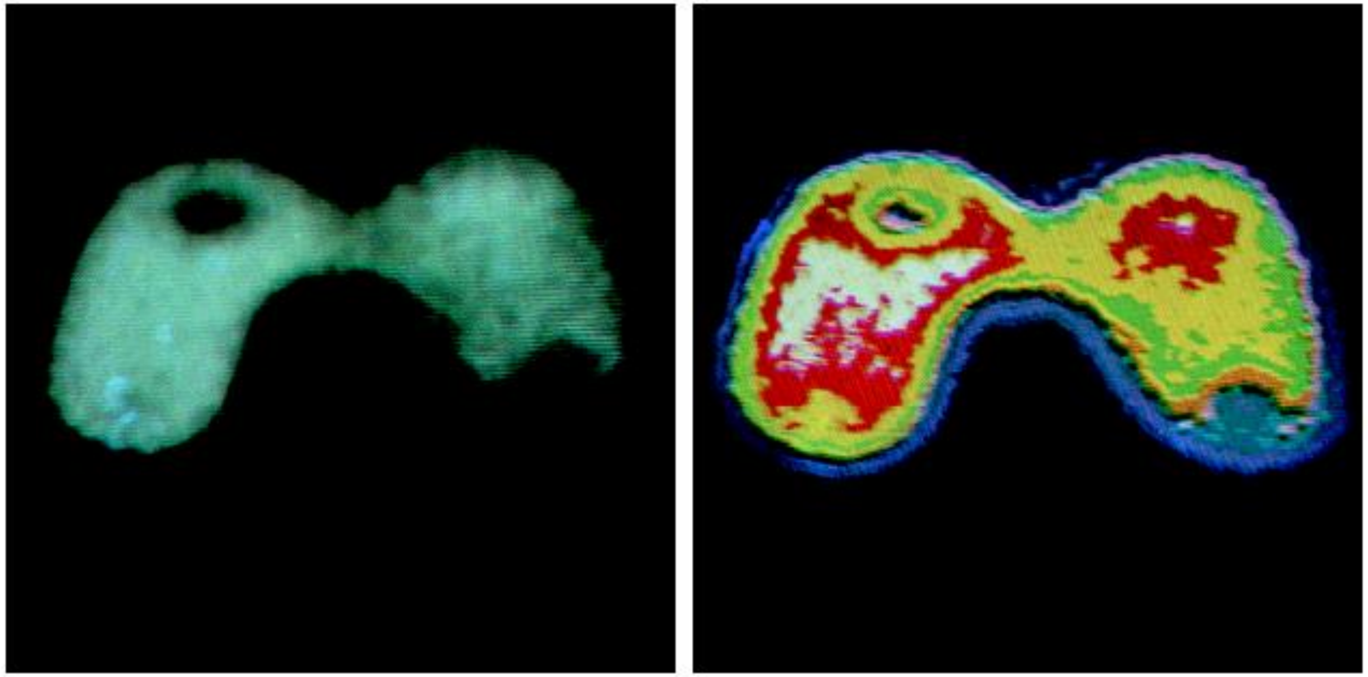


FIGURE 6.19 An alternative representation of the intensity-slicing technique.

Intensity Slicing



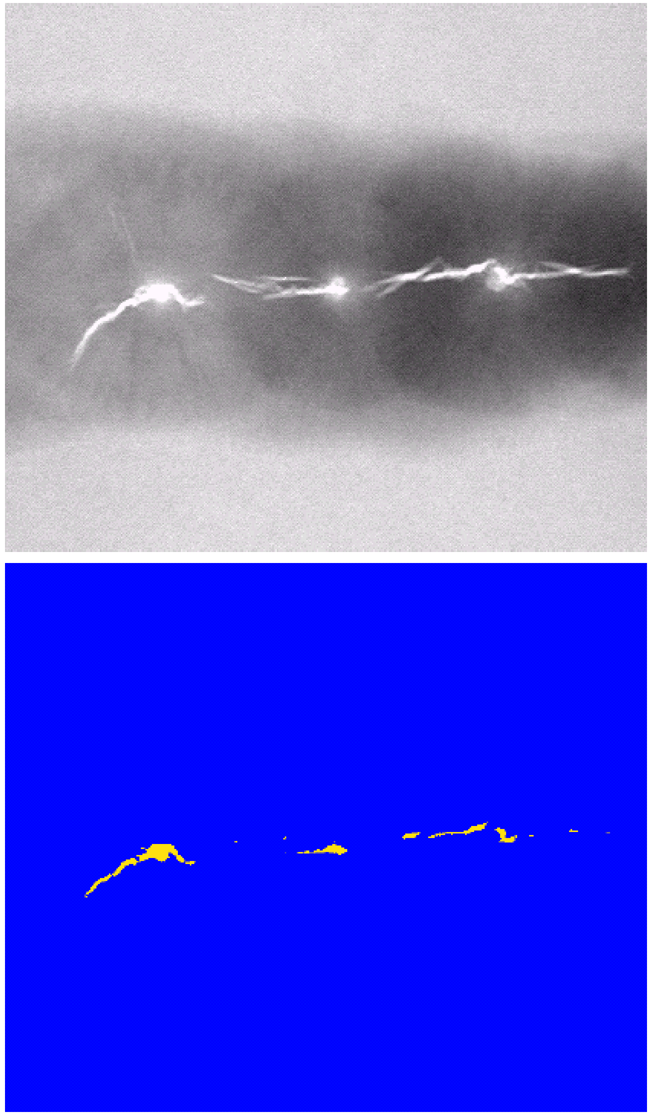
a b

FIGURE 6.20 (a) Monochrome image of the Picker Thyroid Phantom. (b) Result of density slicing into eight colors. (Courtesy of Dr. J. L. Blankenship, Instrumentation and Controls Division, Oak Ridge National Laboratory.)

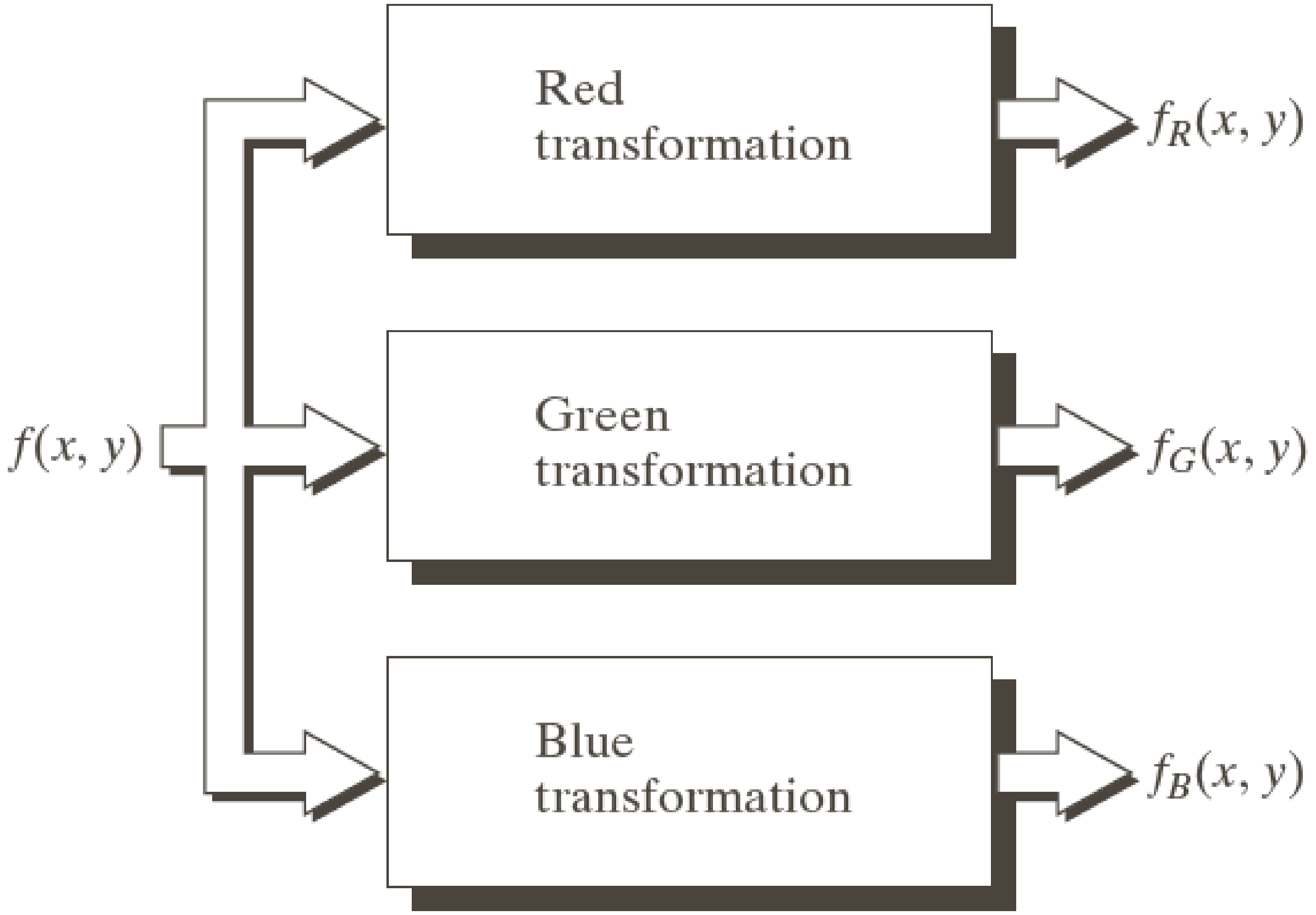
Intensity Slicing

a
b

FIGURE 6.21
(a) Monochrome X-ray image of a weld. (b) Result of color coding. (Original image courtesy of X-TEK Systems, Ltd.)



Intensity to Color Transformation



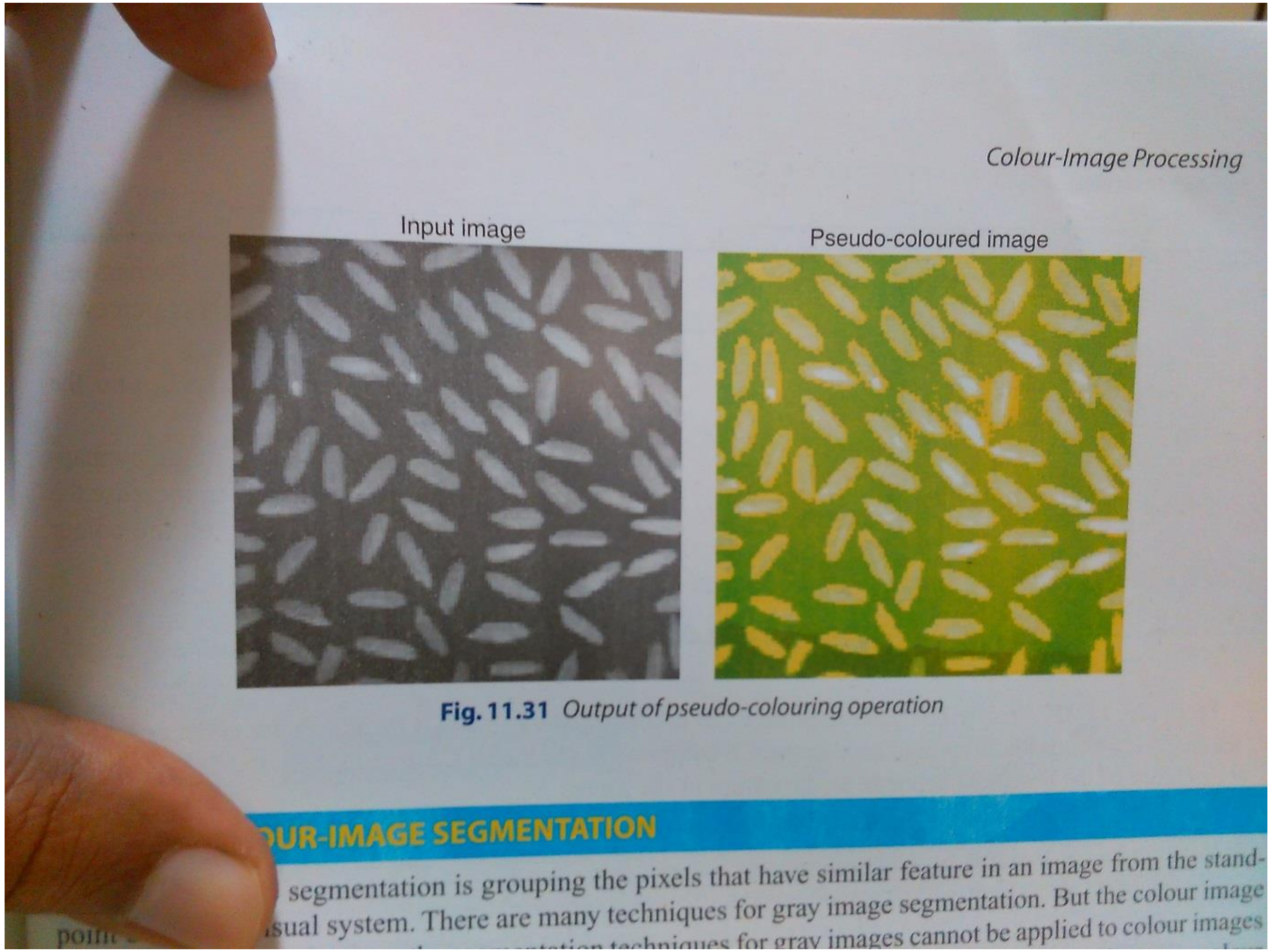
Pseudo Coloring

- clc;
- clear all;
- input_img=imread('rice.tif');
- [m n]=size(input_img);
- input_img=double(input_img);
- for i=1:m
-     for j=1:n
-         if input_img(i,j)>=0 & input_img(i,j)<50
-             output_img(i,j,1)=input_img(i,j,1)+50;
-             output_img(i,j,2)=input_img(i,j)+100;
-             output_img(i,j,3)=input_img(i,j)+10;
-         end
-         if input_img(i,j)>=50 & input_img(i,j)<100
-             output_img(i,j,1)=input_img(i,j)+35;
-             output_img(i,j,2)=input_img(i,j)+128;
-             output_img(i,j,3)=input_img(i,j)+10;
-         end

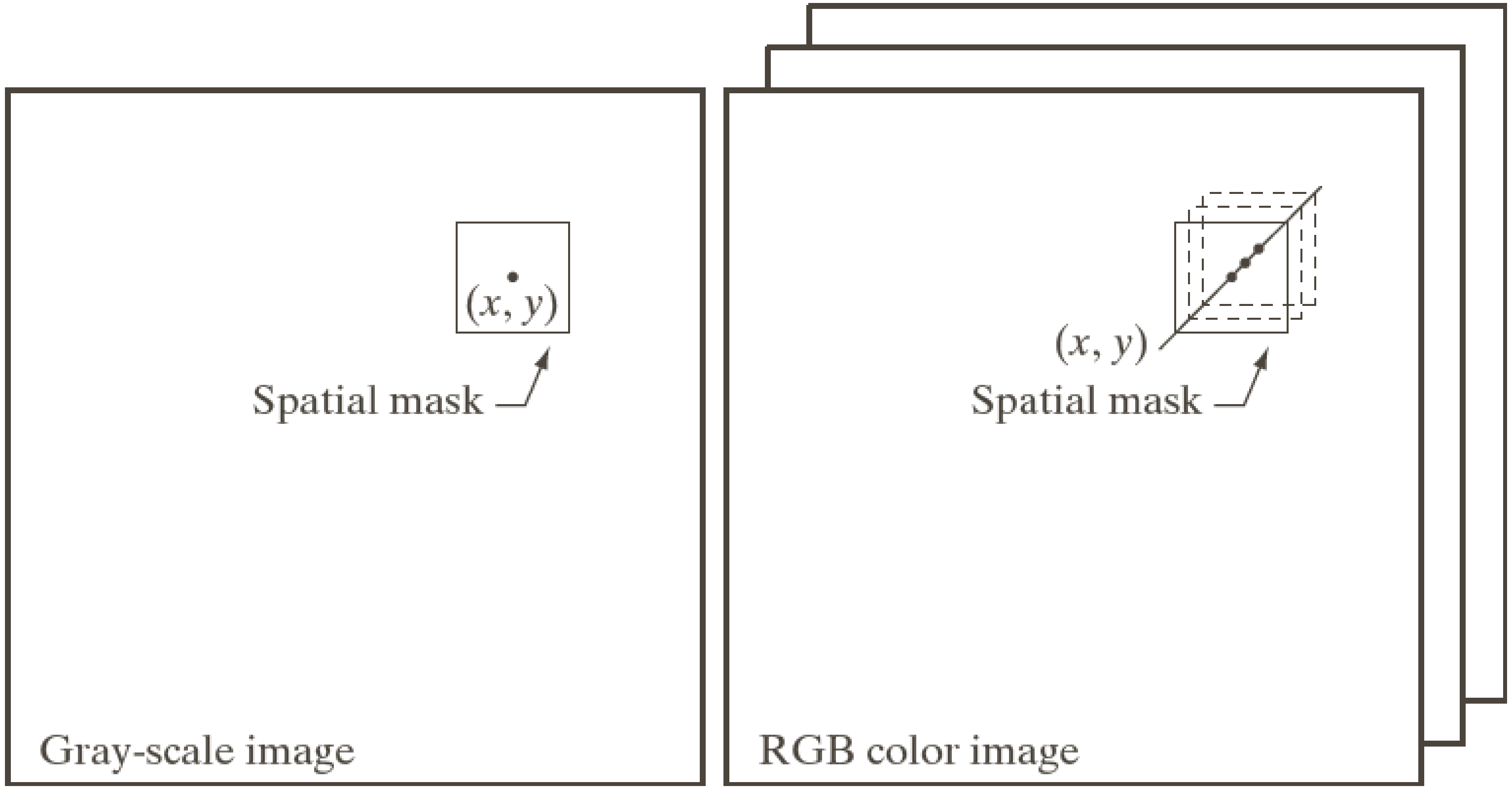
Pseudo Coloring

```
if input_img(i,j)>=100 & input_img(i,j)<150
      output_img(i,j,1)=input_img(i,j)+152;
      output_img(i,j,2)=input_img(i,j)+130;
      output_img(i,j,3)=input_img(i,j)+15;
    end
    if input_img(i,j)>=150 & input_img(i,j)<200
      output_img(i,j,1)=input_img(i,j)+50;
      output_img(i,j,2)=input_img(i,j)+140;
      output_img(i,j,3)=input_img(i,j)+25;
    end
    if input_img(i,j)>=200 & input_img(i,j)<=256
      output_img(i,j,1)=input_img(i,j)+120;
      output_img(i,j,2)=input_img(i,j)+160;
      output_img(i,j,3)=input_img(i,j)+45;
    end
  end
end
subplot(2,2,1),imshow(uint8(input_img)),title('Input Image')
subplot(2,2,2),imshow(uint8(output_img)),title('Pseudo Coloured Image')
```

Pseudo Coloring



Color Transformation



Color Transformation

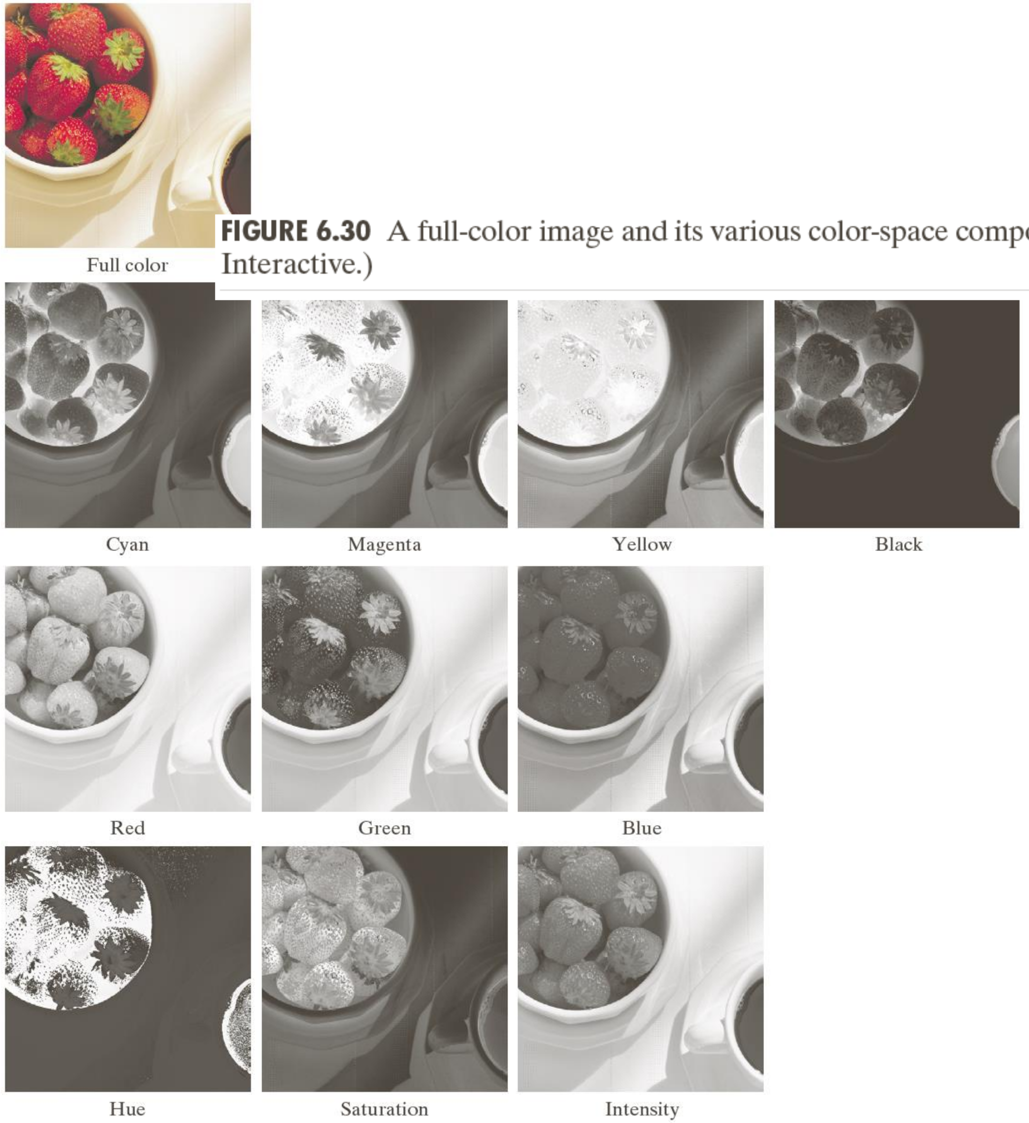


FIGURE 6.30 A full-color image and its various color-space components. Interactive.)

Color Transformation

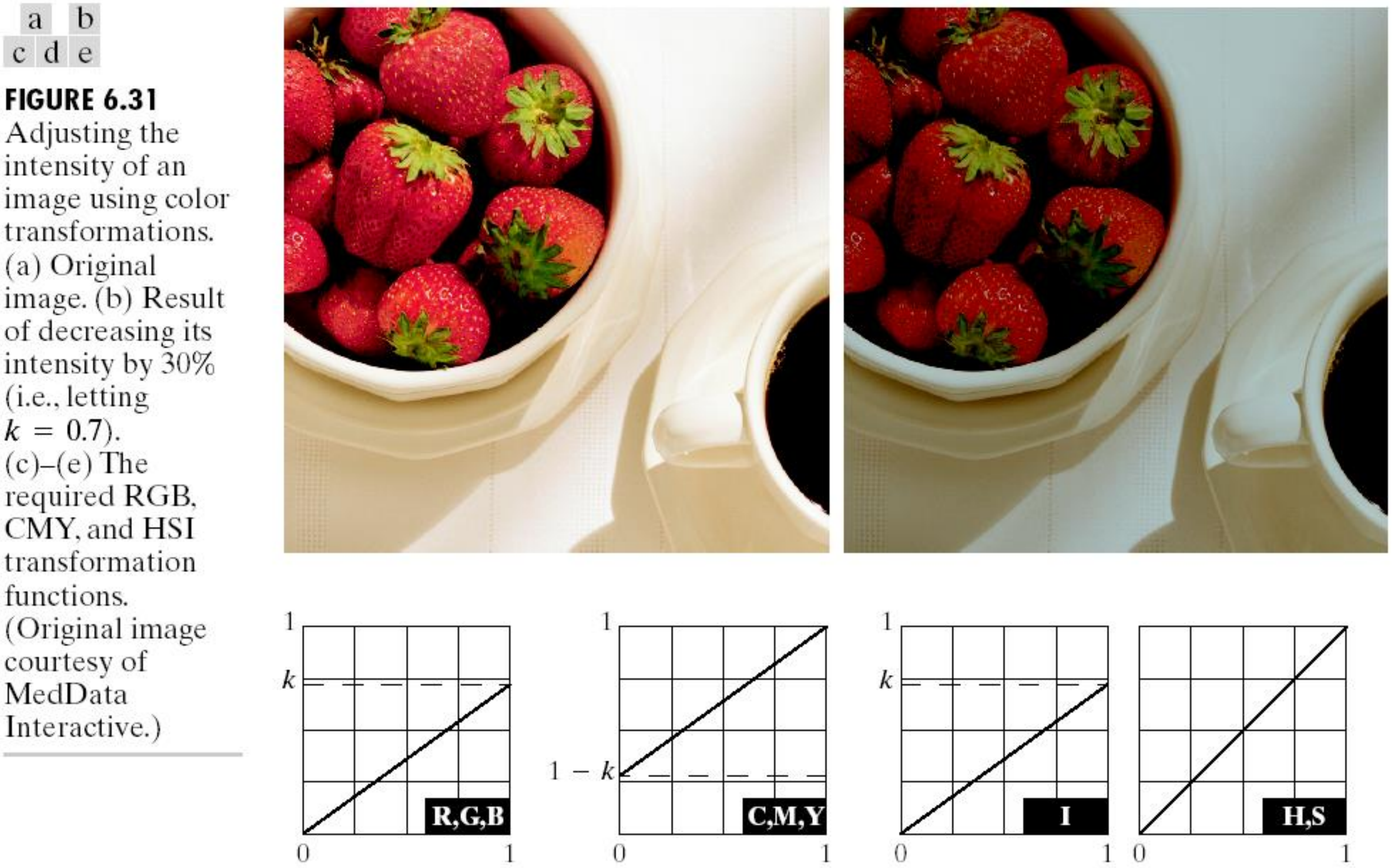
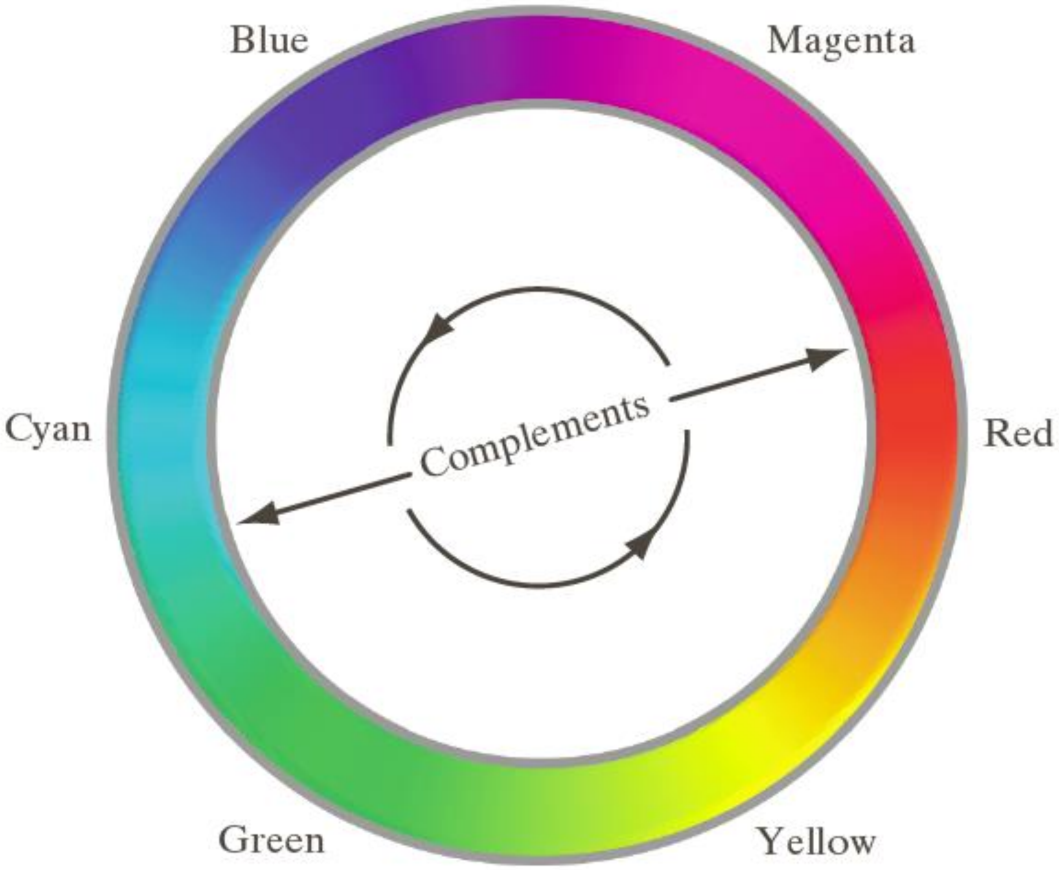


FIGURE 6.31 Adjusting the intensity of an image using color transformations. (a) Original image. (b) Result of decreasing its intensity by 30% (i.e., letting $k = 0.7$). (c)–(e) The required RGB, CMY, and HSI transformation functions. (Original image courtesy of MedData Interactive.)

Color Complement



Color Complement

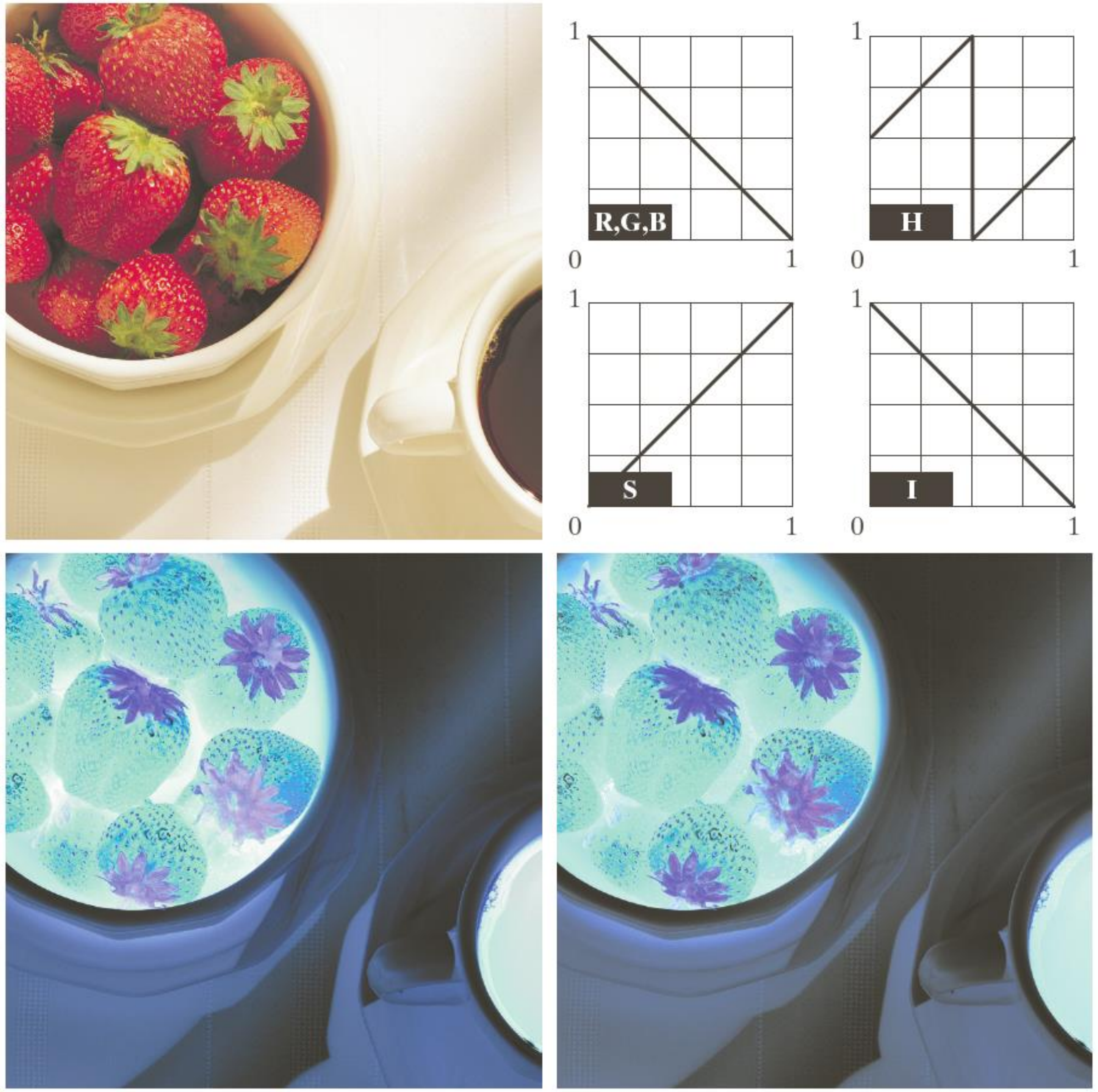
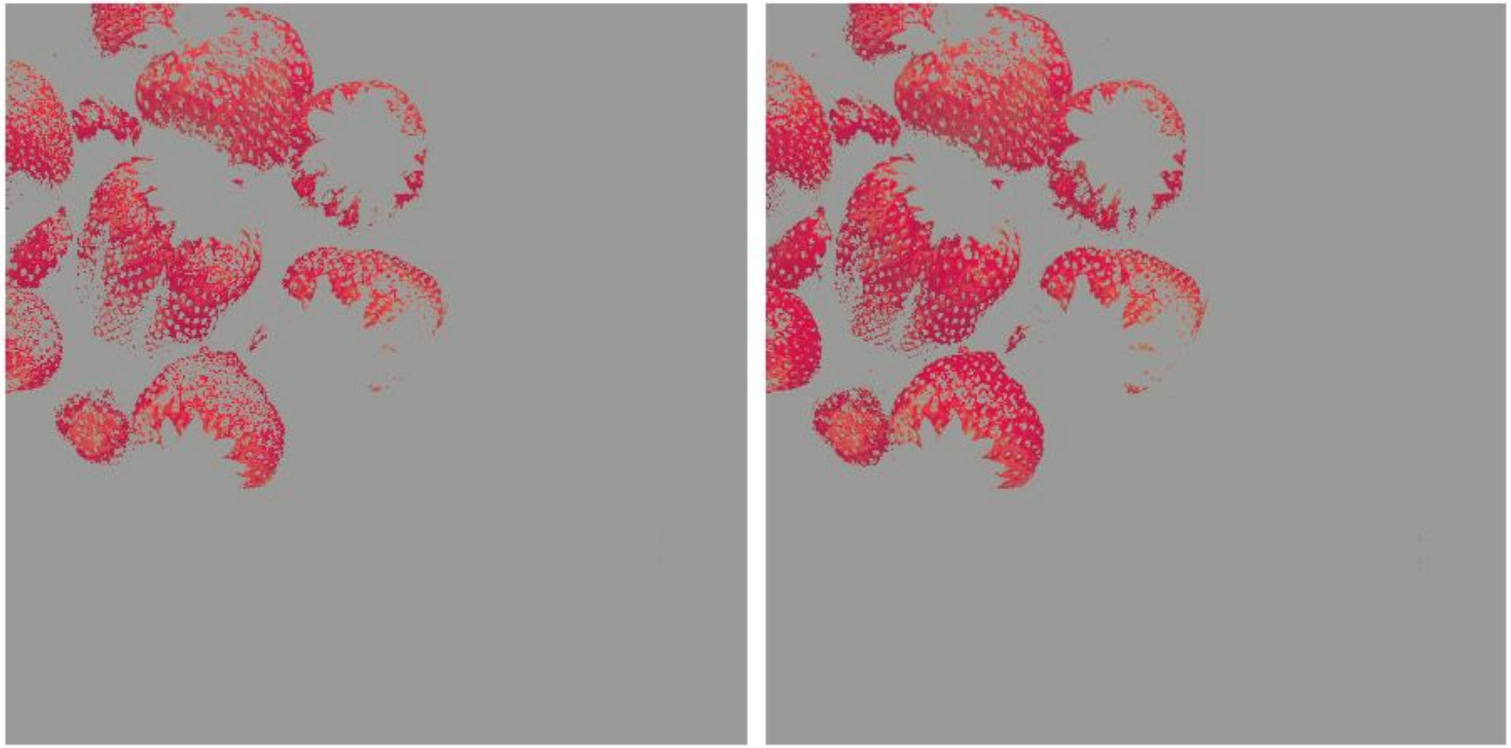


FIGURE 6.33 Color complement transformations. (a) Original image. (b) Complement transformation functions. (c) Complement of (a) based on the RGB mapping functions. (d) An approximation of the RGB complement using HSI transformations.

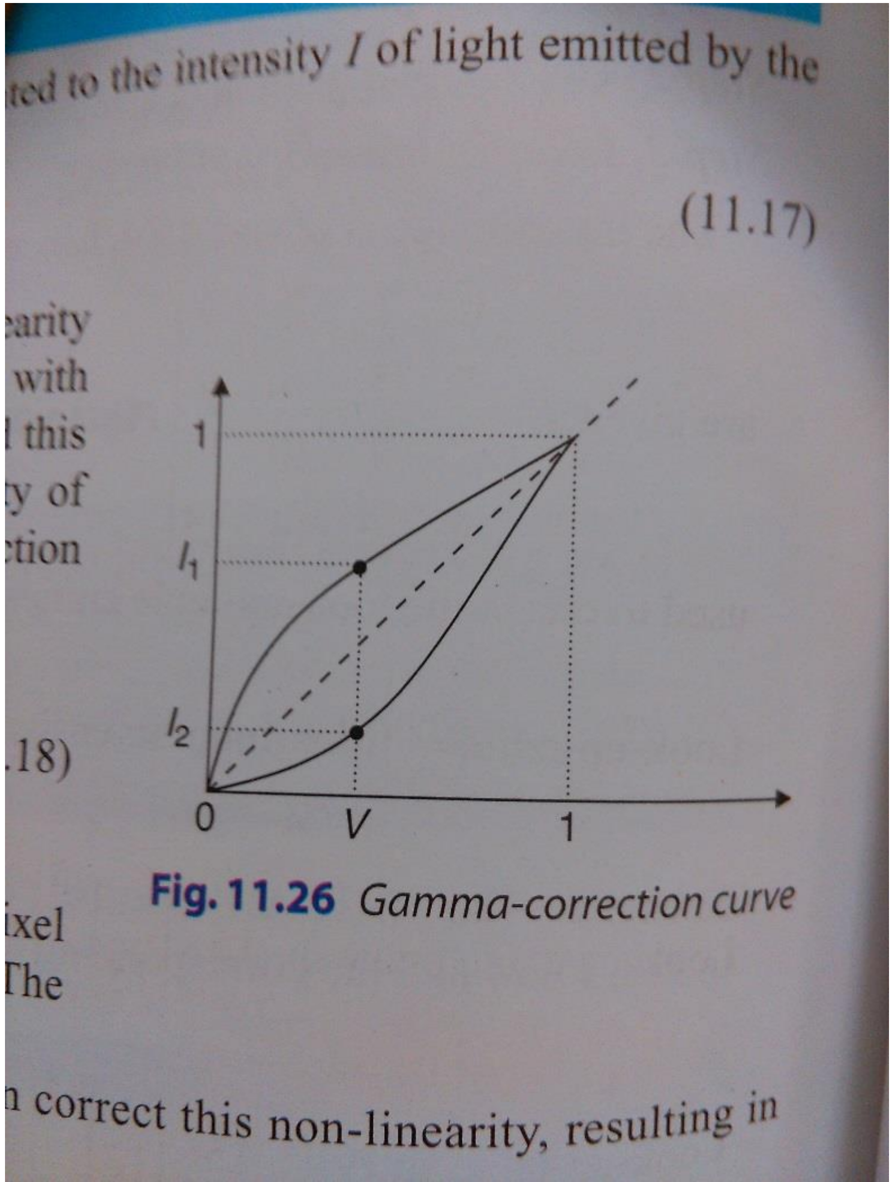
Color Slicing



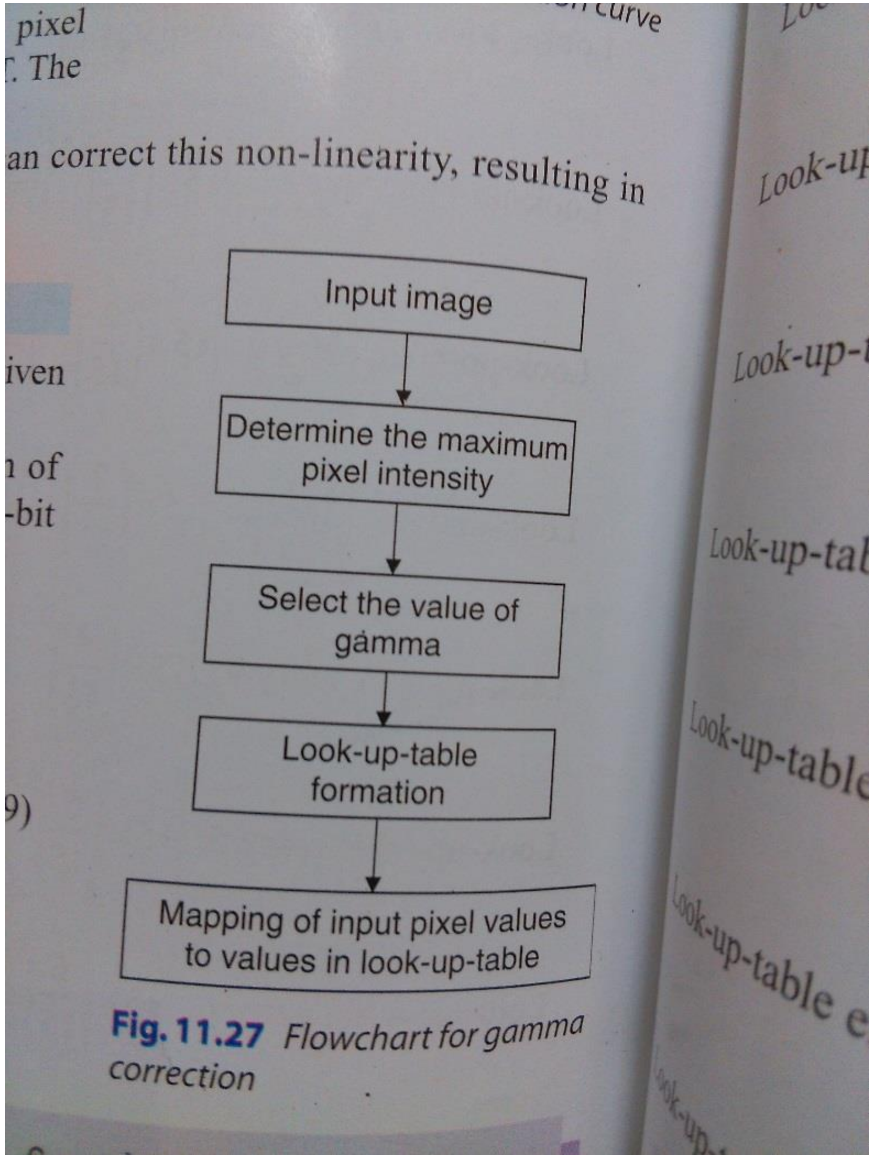
a b

FIGURE 6.34 Color-slicing transformations that detect (a) reds within an RGB cube of width $W = 0.2549$ centered at (0.6863, 0.1608, 0.1922), and (b) reds within an RGB sphere of radius 0.1765 centered at the same point. Pixels outside the cube and sphere were replaced by color (0.5, 0.5, 0.5).

Gamma Correction



Gamma Correction



Gamma Correction

11.10.1 Gamma Correction

The flow chart for gamma correction through a look-up-table is given in Fig. 11.27.

The crucial step is the look-up-table formation and the selection of the gamma value. The formula to create a look-up-table for an 8-bit image is given below.

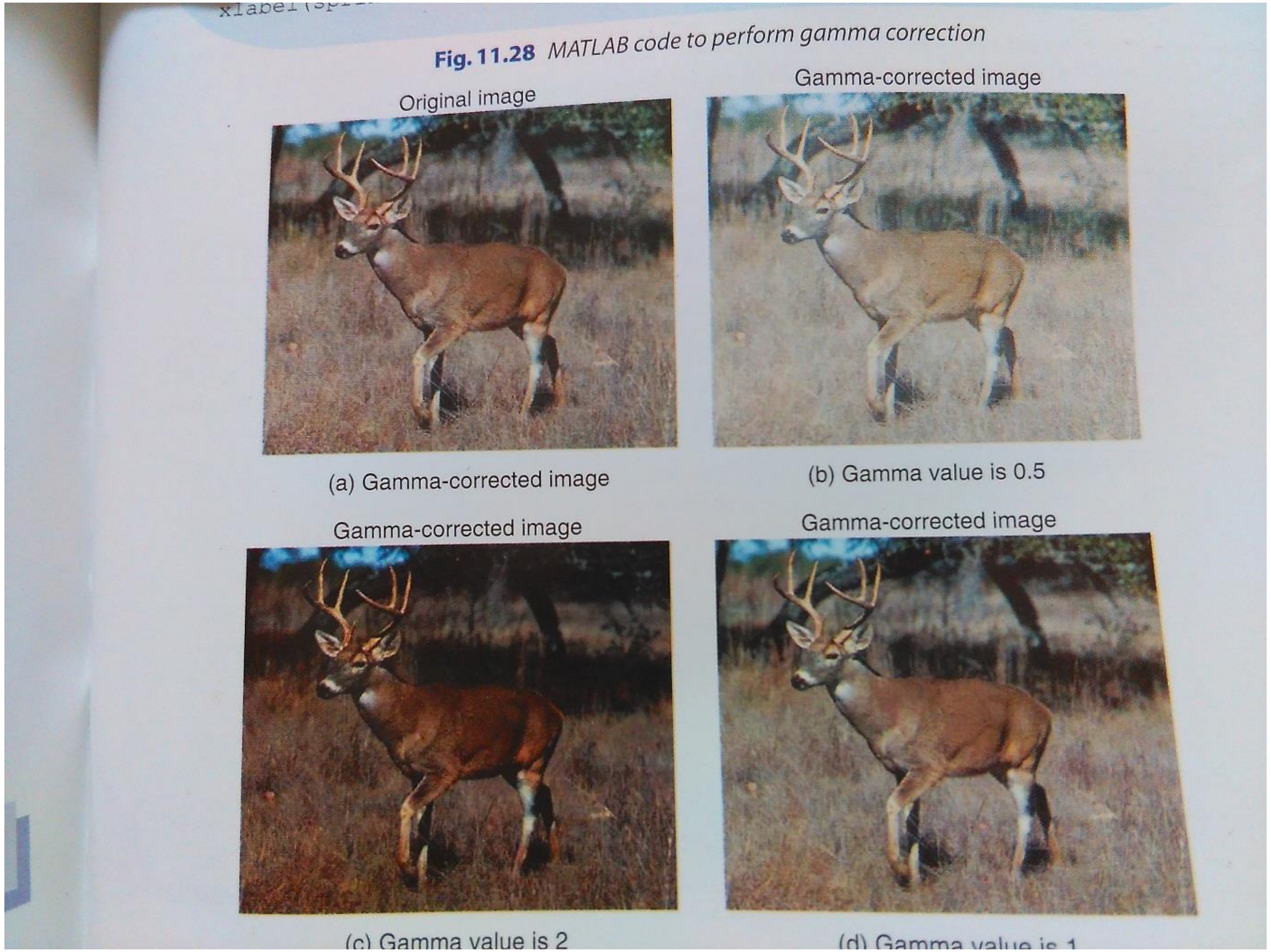
$$\text{Look-up-table} = \left\lfloor \text{Maximum intensity} \times \left(\frac{[0:\text{Maximum intensity}]}{\text{Maximum intensity}} \right)^{\gamma} \right\rfloor \quad (11.19)$$

[2 4

Gamma Correction

- close all;
- clear all;
- clc;
- I=imread('deer4.jpg');
- gamma=1;
- max_intensity =255;%for uint8 image
- %Look up table creation
- LUT = max_intensity .* (([0:max_intensity]./max_intensity).^gamma);
- LUT = floor(LUT);
- %Mapping of input pixels into lookup table values
- J = LUT(double(I)+1);
- imshow(I),title('original image');
- figure,imshow(uint8(J)),title('Gamma corrected image')
- xlabel(sprintf('Gamma value is %g', gamma))

Gamma Correction



Quantisation

- Uniform
- Non-uniform

Quantisation

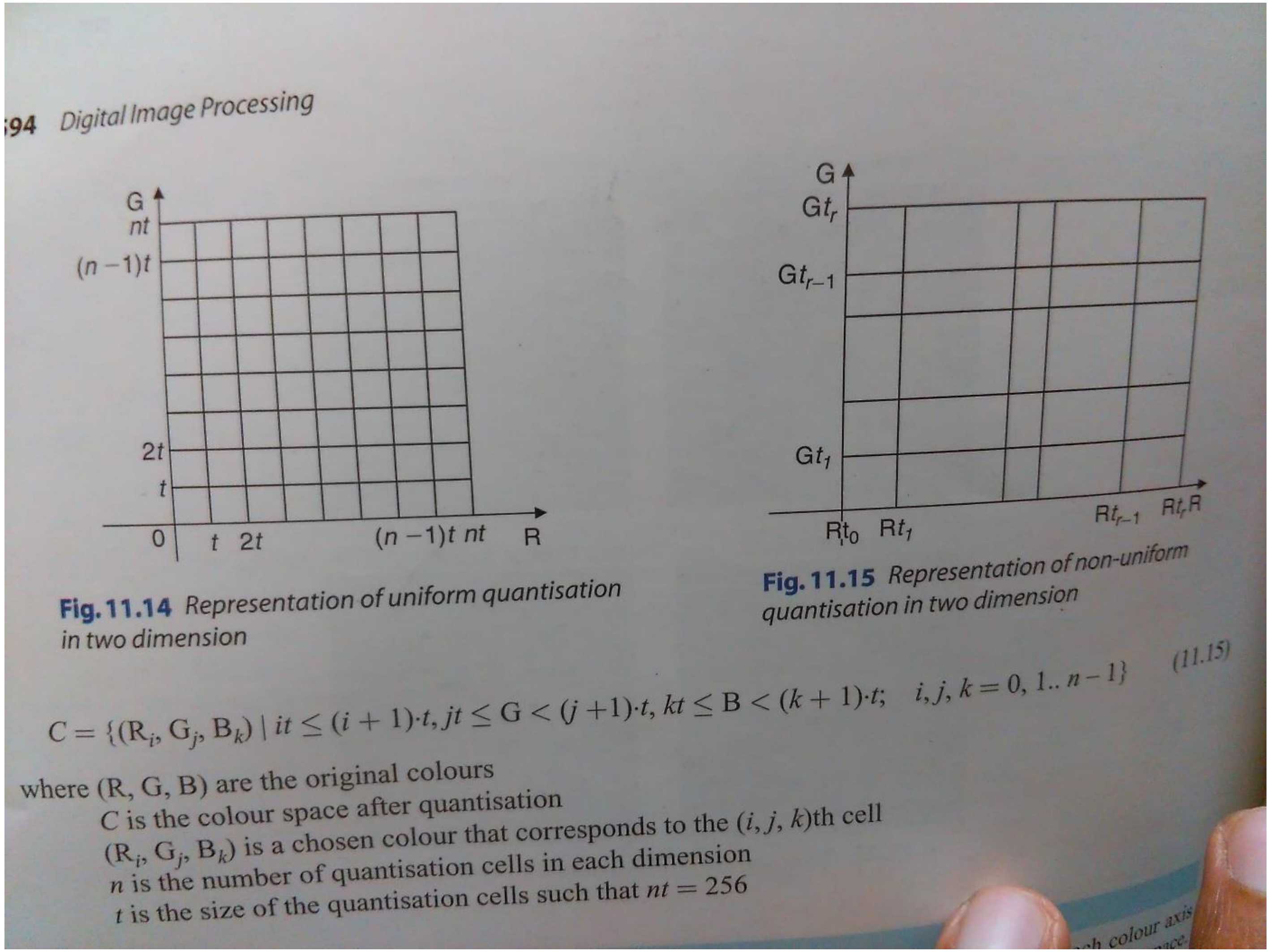


Fig. 11.14 *Representation of uniform quantisation in two dimension*

Fig. 11.15 *Representation of non-uniform quantisation in two dimension*

$$C = \{(R_i, G_j, B_k) \mid it \le (i+1)\cdot t,\ jt \le G < (j+1)\cdot t,\ kt \le B < (k+1)\cdot t; \quad i, j, k = 0, 1 .. n-1\} \tag{11.15}$$

where (R, G, B) are the original colours

C is the colour space after quantisation

(R_i, G_j, B_k) is a chosen colour that corresponds to the (i, j, k)th cell

n is the number of quantisation cells in each dimension

t is the size of the quantisation cells such that $nt = 256$

Histogram

- close all;
- clear all
- clc;
- I = imread('lavender.jpg');
- imshow(I),figure
- I = im2double(I);
- [index,map] = rgb2ind(I);
- pixels = prod(size(index));
- hsv = rgb2hsv(map);
- h = hsv(:,1);
- s = hsv(:,2);
- v = hsv(:,3);

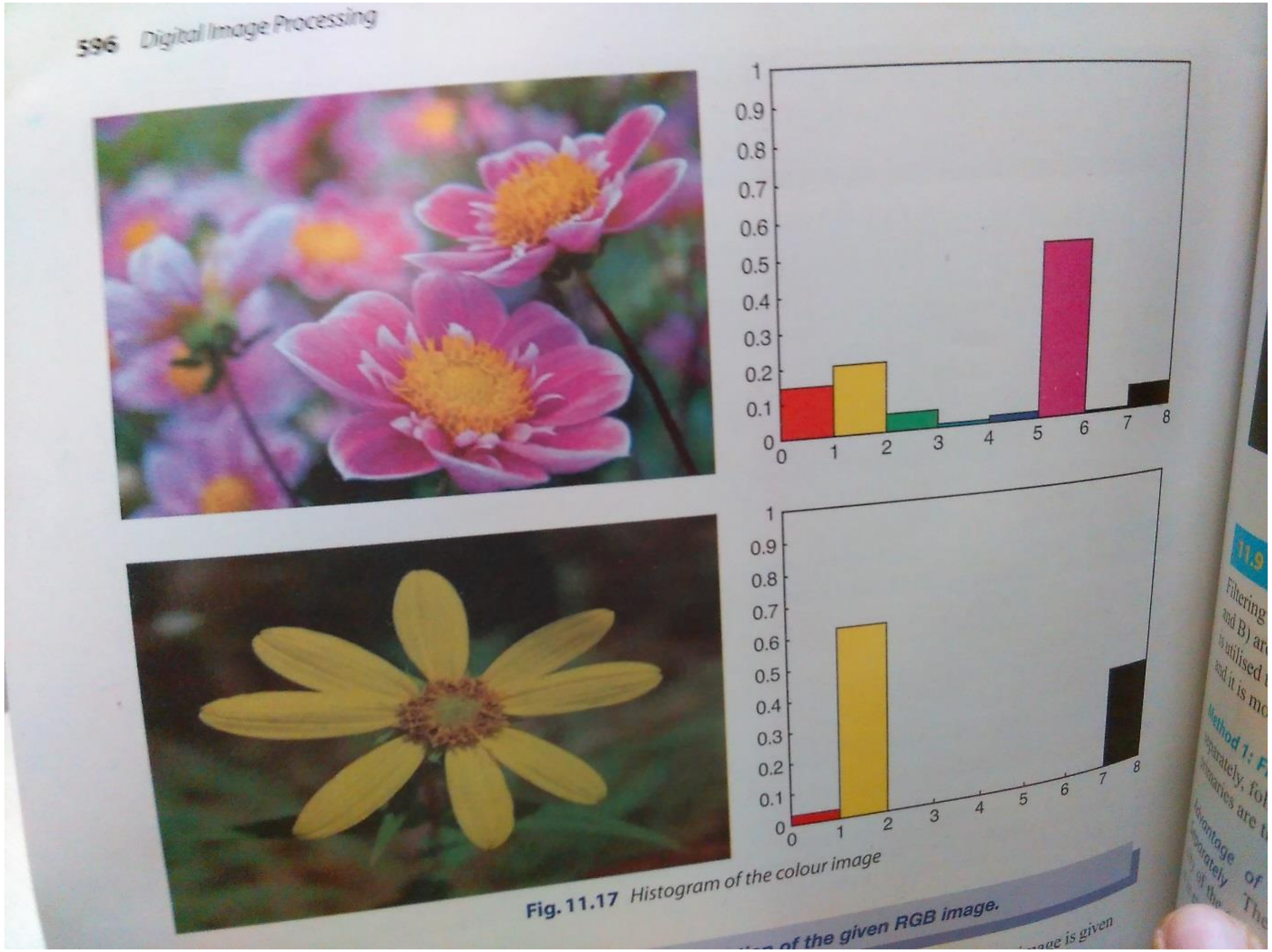
Histogram

- %Finds location of black and white pixels
- darks = find(v < .2)';
- lights = find(s < .05 & v > .85)';
- h([darks lights]) = -1;
- %Gets the number of all pixels for each color bin
- black = length(darks)/pixels;
- white = length(lights)/pixels;
- red = length(find((h > .9167 | h <= .083) & h ~= -1))/pixels;
- yellow = length(find(h > .083 & h <= .25))/pixels;
- green = length(find(h > .25 & h <= .4167))/pixels;
- cyan = length(find(h > .4167 & h <= .5833))/pixels;
- blue = length(find(h > .5833 & h <= .75))/pixels;
- magenta = length(find(h > .75 & h <= .9167))/pixels;

Histogram

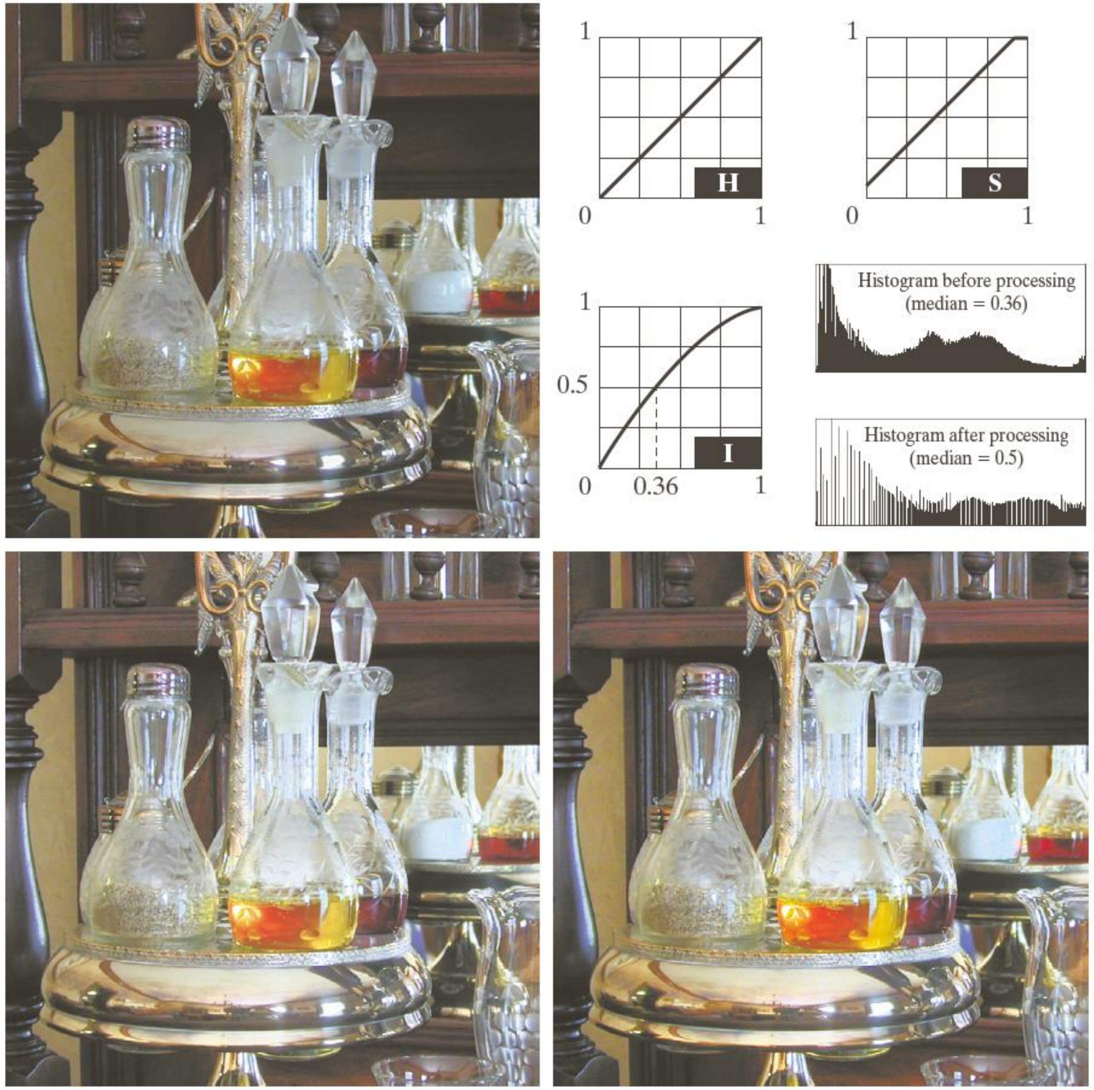
- %Plots histogram
- hold on
- fill([0 0 1 1],[0 red red 0],'r')
- fill([1 1 2 2],[0 yellow yellow 0],'y')
- fill([2 2 3 3],[0 green green 0],'g')
- fill([3 3 4 4],[0 cyan cyan 0],'c')
- fill([4 4 5 5],[0 blue blue 0],'b')
- fill([5 5 6 6],[0 magenta magenta 0],'m')
- fill([6 6 7 7],[0 white white 0],'w')
- fill([7 7 8 8],[0 black black 0],'k')
- axis([0 8 0 1])

Histogram



596 Digital Image Processing
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 1 2 3 4 5 6 7 8
1
0.9
0.8
0.7
0.6
0.5
0.4
0.3
0.2
0.1
0
0 1 2 3 4 5 6 7 8
Fig. 11.17 Histogram of the colour image
of the given RGB image.

Histogram Processing



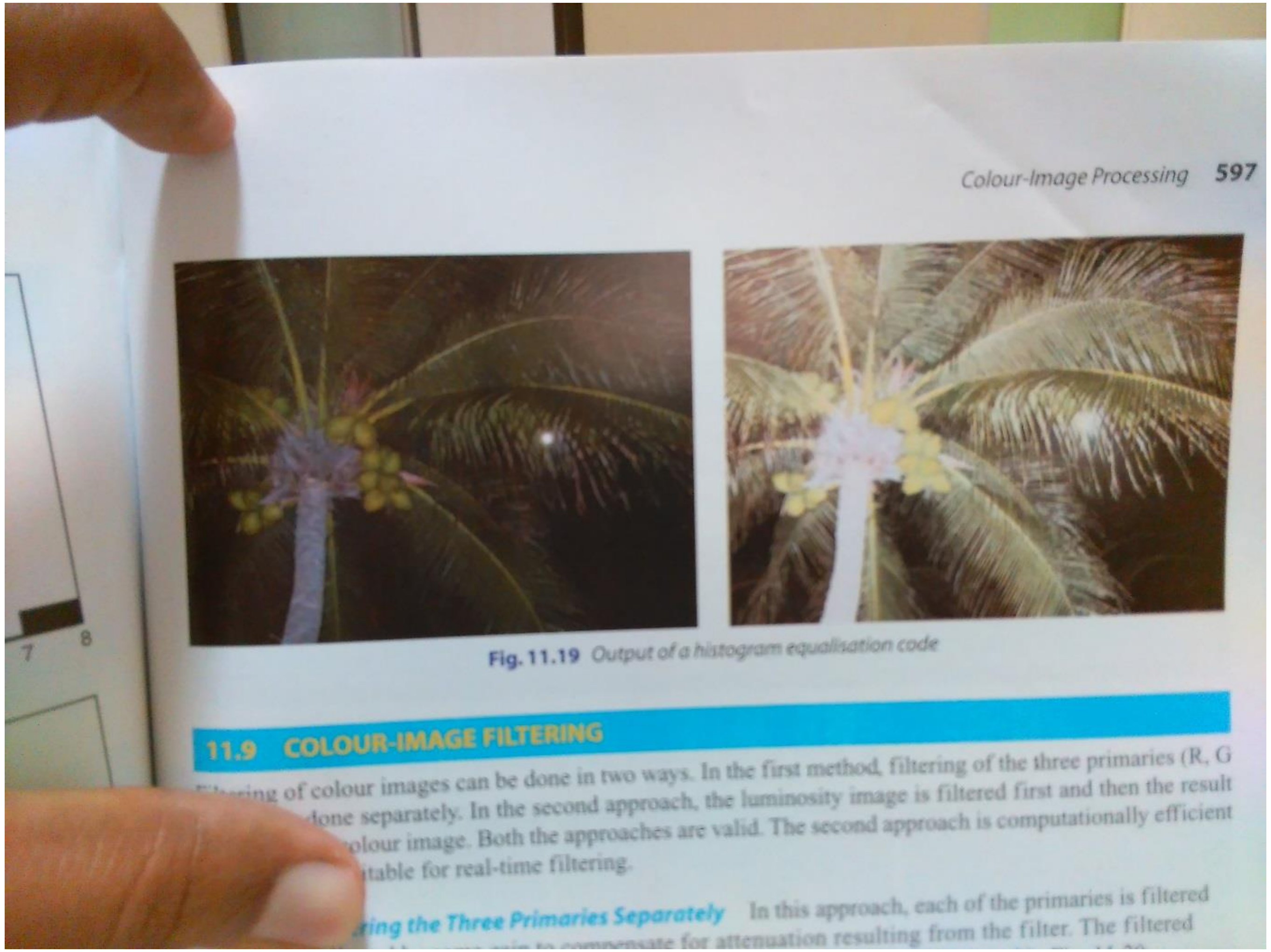
a b
c d

FIGURE 6.37 Histogram equalization (followed by saturation adjustment) in the HSI color space.

Histogram equalisation

- a=imread('coconut.bmp');
- %Conversion of RGB to YIQ format
- b=rgb2ntsc(a);
- %Histogram equalization of Y component alone
- b(:,:,1)=histeq(b(:,:,1));
- %Conversion of YIQ to RGB format
- c=ntsc2rgb(b);
- imshow(a),title('original image')
- figure,imshow(c),title('Histogram equalized image')

Histogram equalisation



Noise in Color Image

a b
c d

FIGURE 6.48 (a)–(c) Red, green, and blue component images corrupted by additive Gaussian noise of mean 0 and variance 800. (d) Resulting RGB image. [Compare (d) with Fig. 6.46(a).]

Noise in Color Image

FIGURE 6.49 HSI components of the noisy color image in Fig. 6.48(d). (a) Hue. (b) Saturation. (c) Intensity.

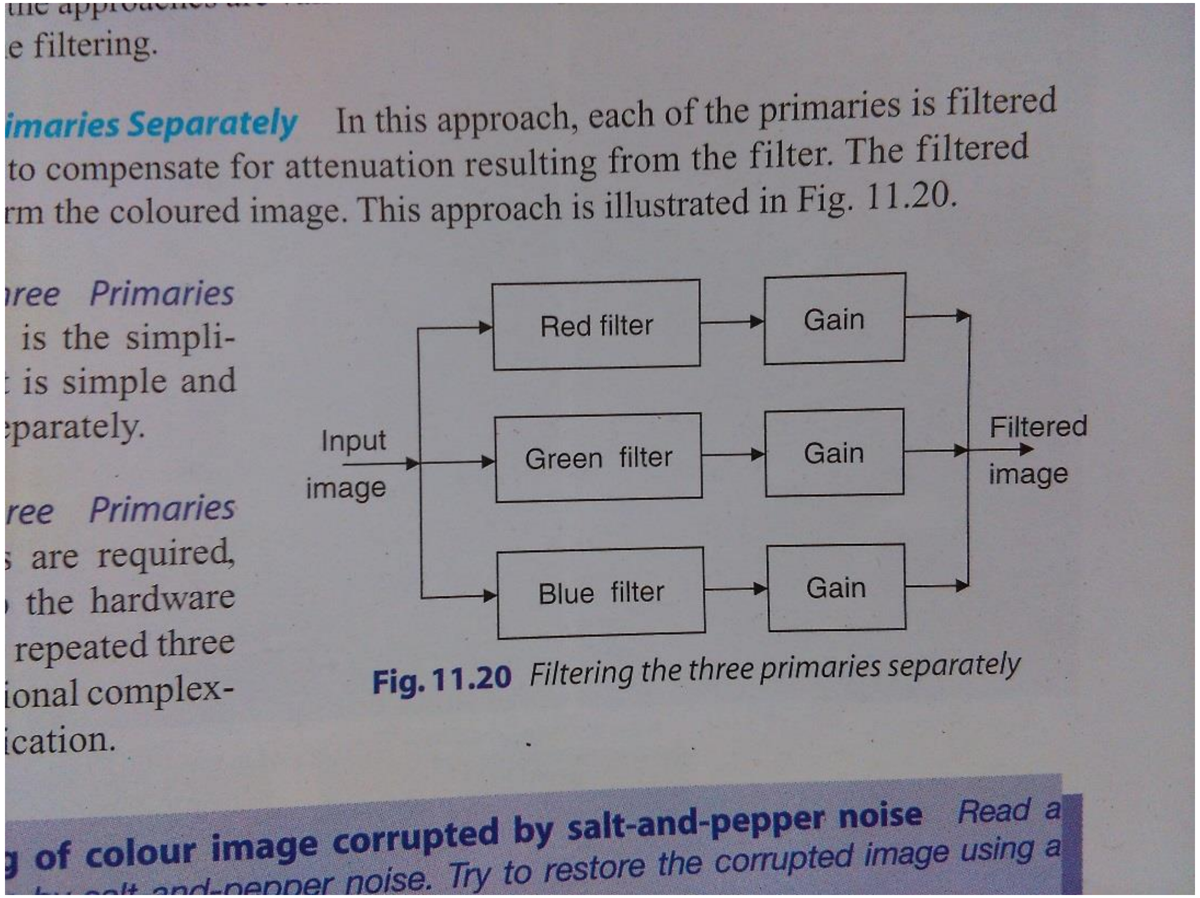
Noise in Color Image

a b
c d

FIGURE 6.50 (a) RGB image with green plane corrupted by salt-and-pepper noise. (b) Hue component of HSI image. (c) Saturation component. (d) Intensity

Color Image filtering

- Filtering the three Primaries Separately



e filtering.

imaries Separately In this approach, each of the primaries is filtered
to compensate for attenuation resulting from the filter. The filtered
rm the coloured image. This approach is illustrated in Fig. 11.20.

ree Primaries
is the simpli-
is simple and
parately.

ree Primaries
are required,
the hardware
repeated three
ional complex-
ication.

Fig. 11.20 *Filtering the three primaries separately*

of colour image corrupted by salt-and-pepper noise *Read a*
salt-and-pepper noise. Try to restore the corrupted image using a

Color Image filtering

a b
c d

FIGURE 6.38 (a) RGB image. (b) Red component image. (c) Green component. (d) Blue component.

Color Image filtering

a b c

FIGURE 6.39 HSI components of the RGB color image in Fig. 6.38(a). (a) Hue. (b) Saturation. (c) Intensity.

Color Image filtering

a b c

FIGURE 6.40 Image smoothing with a 5×5 averaging mask. (a) Result of processing each RGB component image. (b) Result of processing the intensity component of the HSI image and converting to RGB. (c) Difference between the two results.

Median Filtering

- clc
- clear all
- close all
- a=imread('C:\Documents and Settings\esakki\My Documents\My Pictures\f1.bmp');
- b=imnoise(a,'salt & pepper',0.2);
- c(:,:,1)=medfilt2(b(:,:,1));
- c(:,:,2)=medfilt2(b(:,:,2));
- c(:,:,3)=medfilt2(b(:,:,3));
- imshow(a),title('original image')
- figure,imshow(b),title('corrupted image')
- figure,imshow(c),title('Median filtered image')

Median Filtering

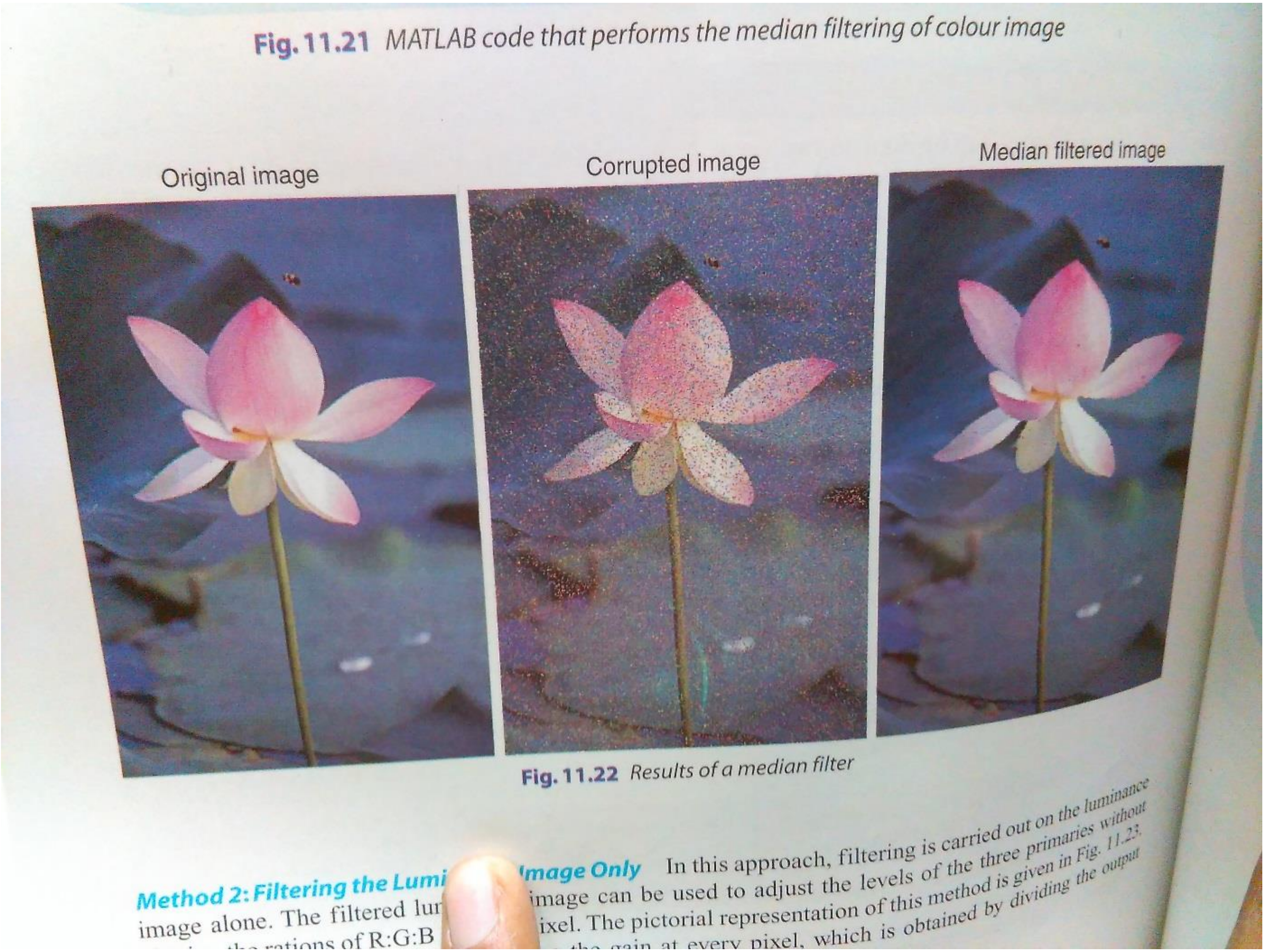


Fig. 11.21 *MATLAB code that performs the median filtering of colour image*

Original image

Corrupted image

Median filtered image

Fig. 11.22 *Results of a median filter*

Method 2: Filtering the Lumi **Image Only** In this approach, filtering is carried out on the luminance
image alone. The filtered lu image can be used to adjust the levels of the three primaries without
the rations of R:G:B ixel. The pictorial representation of this method is given in Fig. 11.23.
the gain at every pixel, which is obtained by dividing the output

Color Image Sharpening

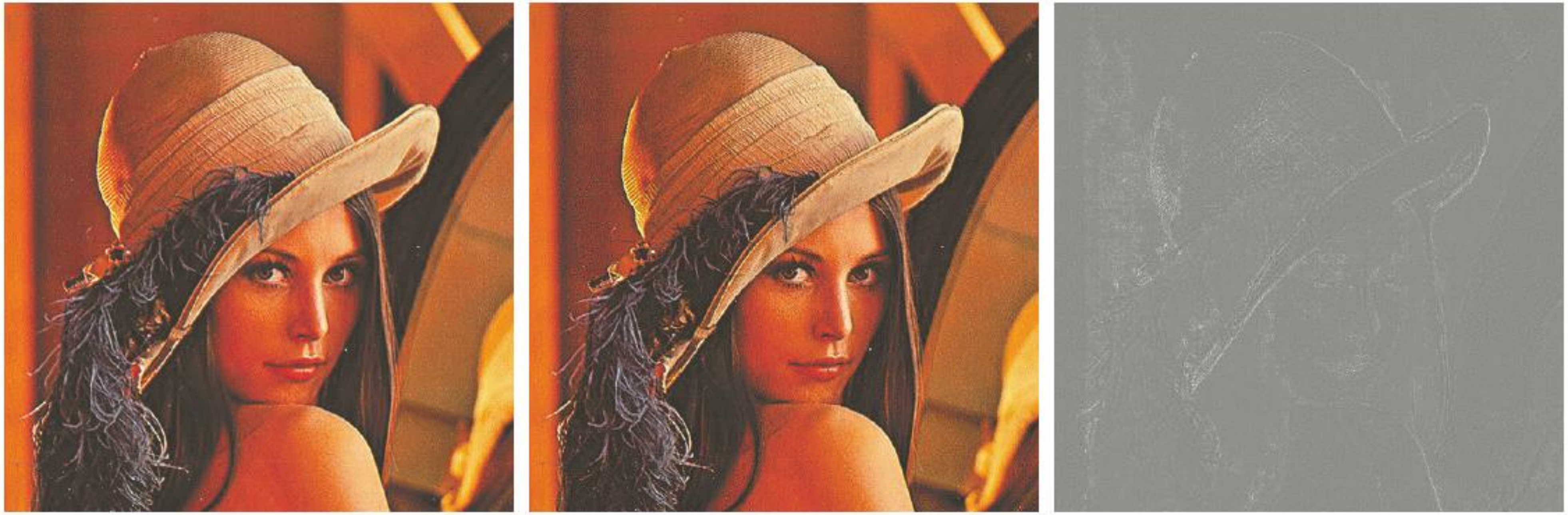


FIGURE 6.41 Image sharpening with the Laplacian. (a) Result of processing each RGB channel. (b) Result of processing the HSI intensity component and converting to RGB. (c) Difference between the two results.