Image Processing and Analysis
lecture 8、Color Image Processing

Weiqiang Wang
School of Computer Science and Technology, UCAS

November 1, 2023

Outline

Color Image Representation in MATLAB

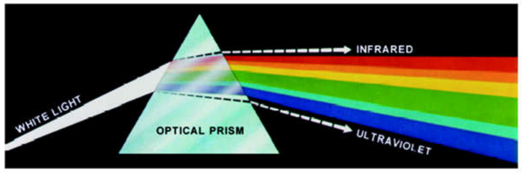


FIGURE 6.1 Color spectrum seen by passing white light through a prism. (Courtesy of the General Electric Co., Lamp Business Division.)

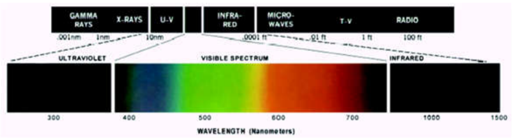


FIGURE 6.2 Wavelengths comprising the visible range of the electromagnetic spectrum. (Courtesy of the General Electric Co., Lamp Business Division.)

Color Image Representation in MATLAB(cont.)

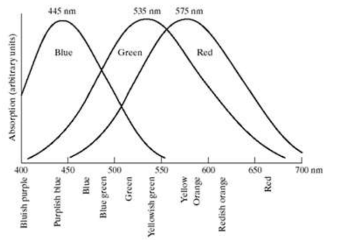
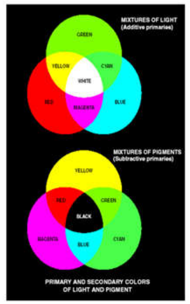


FIGURE 6.3 Absorption of light by the red, green, and blue cones in the human eye as a function of wavelength.

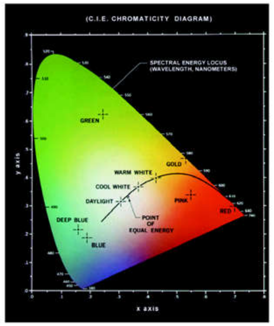


a
b

FIGURE 6.4 Primary and secondary colors of light and pigments. (Courtesy of the General Electric Co., Lamp Business Division.)

Color Image Representation in MATLAB(cont.)

FIGURE 6.5 Chromaticity diagram. (Courtesy of the General Electric Co., Lamp Business Division.)



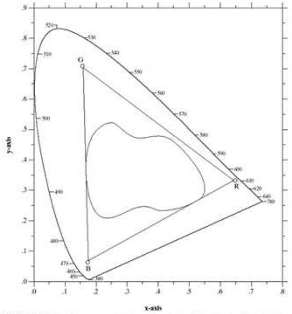
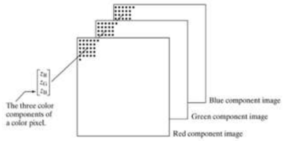
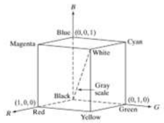


FIGURE 6.6 Typical color gamut of color monitors (triangle) and color printing devices (irregular region).

Color Image Representation in MATLAB(cont.)

- RGB: M×N×3 array of color pixels





- Cat operator to stack the images:

  rgb_image = cat(3,fr,fg,fb);

  fr= rgb_image(:,:,1);

  fg= rgb_image(:,:,2);

  fb= rgb_image(:,:,3);

- View the color cubic

  rgbcube(vx,vy,vz)

Color Image Representation in MATLAB(cont.)

- An indexed image consists of a data matrix of integers (two dimensions): X and Colormap matrix :map

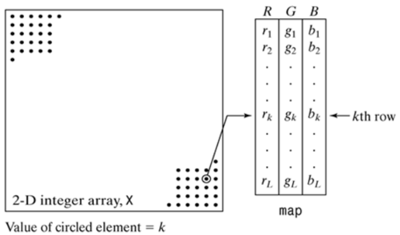


FIGURE 6.3 Elements of an indexed image. Note that the value of an element of integer array X determines the row number in the colormap. Each row contains an RGB triplet, and L is the total number of rows.

- *Display it using matlab:*

  imshow(x,map) Or image(X);colormap(map)

Color Image Representation in MATLAB(cont.)

- Approximate an indexed image by one with fewer colors

  [Y,newmap]= imapprox(x,map,n)

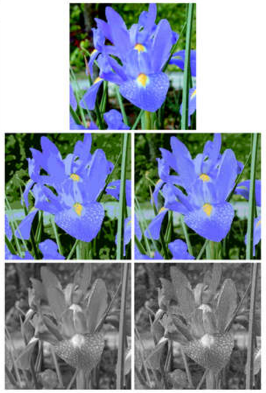
Long name	Short name	RGB values
Black	k	[0 0 0]
Blue	b	[0 0 1]
Green	g	[0 1 0]
Cyan	c	[0 1 1]
Red	r	[1 0 0]
Magenta	m	[1 0 1]
Yellow	y	[1 1 0]
White	w	[1 1 1]

Name	Description
autumn	Varies smoothly from red, through orange, to yellow.
bone	A gray-scale colormap with a higher value for the blue component. This colormap is useful for adding an "electronic" look to gray-scale images.
colorcube	Contains as many regularly spaced colors in RGB color space as possible, while attempting to provide more steps of gray, pure red, pure green, and pure blue.
cool	Consists of colors that are shades of cyan and magenta. It varies smoothly from cyan to magenta.
copper	Varies smoothly from black to bright copper.
flag	Consists of the colors red, white, blue, and black. This colormap completely changes color with each index increment.
gray	Returns a linear gray-scale colormap.
hot	Varies smoothly from black, through shades of red, orange, and yellow, to white.
hsv	Varies the hue component of the hue-saturation-value color model. The colors begin with red, pass through yellow, green, cyan, blue, magenta, and return to red. The colormap is particularly appropriate for displaying periodic functions.
jet	Ranges from blue to red, and passes through the colors cyan, yellow, and orange.
lines	Produces a colormap of colors specified by the ColorOrder property and a shade of gray. Consult online help regarding function ColorOrder.
pink	Contains pastel shades of pink. The pink colormap provides sepia tone colorization of grayscale photographs.
prism	Repeats the six colors red, orange, yellow, green, blue, and violet.
spring	Consists of colors that are shades of magenta and yellow.
summer	Consists of colors that are shades of green and yellow.
white	This is an all white monochrome colormap.
winter	Consists of colors that are shades of blue and green.

Color Image Representation in MATLAB(cont.)

Function	Purpose
dither	Creates an indexed image from an RGB image by dithering.
grayslice	Creates an indexed image from a gray-scale intensity image by multilevel thresholding.
gray2ind	Creates an indexed image from a gray-scale intensity image.
ind2gray	Creates a gray-scale intensity image from an indexed image.
rgb2ind	Creates an indexed image from an RGB image.
ind2rgb	Creates an RGB image from an indexed image.
rgb2gray	Creates a gray-scale image from an RGB image.

TABLE 6.3 IPT functions for converting between RGB, indexed, and gray-scale intensity images.



a
b c
d e

FIGURE 6.4 (a) RGB image. (b) Number of colors reduced to 8 without dithering. (c) Number of colors reduced to 8 with dithering. (d) Gray-scale version of (a) obtained using function rgb2gray. (e) Dithered gray-scale image (this is a binary image).

Converting to Other Color Spaces

- RGB→NTSC

$$\begin{bmatrix} Y \\ I \\ Q \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ 0.596 & -0.274 & -0.322 \\ 0.211 & -0.523 & 0.312 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

yiq_image=rgb2ntsc(rgb_image)

- NTSC → RGB

$$\begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1.000 & 0.956 & 0.621 \\ 1.000 & -0.272 & -0.647 \\ 1.000 & -1.106 & 1.703 \end{bmatrix} \begin{bmatrix} Y \\ I \\ Q \end{bmatrix}$$

rgb_image=ntsc2rgb(yiq_image)

Converting to Other Color Spaces(cont.)

- RGB→YCbCr

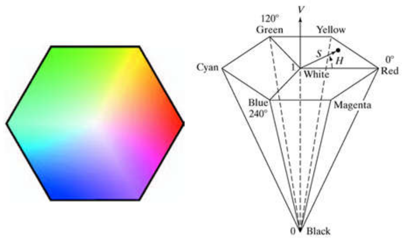
$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 16 \\ 128 \\ 128 \end{bmatrix} + \begin{bmatrix} 65.481 & 128.553 & 24.966 \\ -37.797 & -74.203 & 112.000 \\ 112.000 & -93.786 & -18.214 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix} /256(about)$$

Ycbcr_image=rgb2ycbcr(rgb_image)

rgb_image=ycbcr2rgb(ycbcr_image)

Converting to Other Color Spaces(cont.)

- HSV



a b

FIGURE 6.5 (a) The HSV color hexagon. (b) The HSV hexagonal cone.

Converting to Other Color Spaces(cont.)

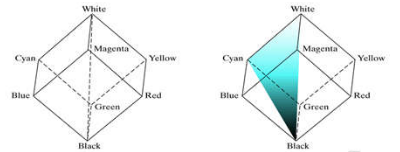
- CMY & CMYK

$$\begin{bmatrix} C \\ M \\ Y \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad \begin{bmatrix} R \\ G \\ B \end{bmatrix} = \begin{bmatrix} 1 \\ 1 \\ 1 \end{bmatrix} - \begin{bmatrix} C \\ M \\ Y \end{bmatrix}$$

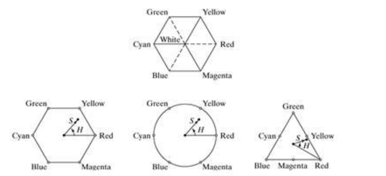
Cmy_image=imcomplent(rgb_image) Rgb_image=imcomplent(cmy_image)

Converting to Other Color Spaces(cont.)

- HIS



a b
FIGURE 6.6 Relationship between the RGB and HSI color models.



a
b c d
FIGURE 6.7 Hue and saturation in the HSI color model. The dot is an arbitrary color point. The angle from the red axis gives the hue, and the length of the vector is the saturation. The intensity of all colors in any of these planes is given by the position of the plane on the vertical intensity axis.

Converting to Other Color Spaces(cont.)

- RGB→HSI

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

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

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

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

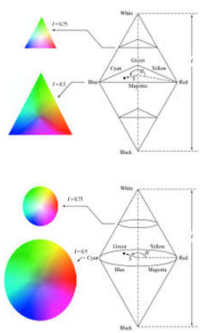
- HSI → RGB
  - BR sector$(240° \le H \le 360°)$

$$H = H - 240° \quad G = I(1-S)$$

$$B = I[1 + \frac{ScosH}{\cos(60° - H)}]$$

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

a
b
FIGURE 6.8 The HSI color model based on (a) triangular and (b) circular color planes. The triangles and circles are perpendicular to the vertical intensity axis.



Converting to Other Color Spaces(cont.)

- HSI→RGB
  - BR sector$(240° \le H \le 360°)$

$$H = H - 240° \quad G = I(1-S)$$

$$B = I[1 + \frac{ScosH}{\cos(60° - H)}]$$

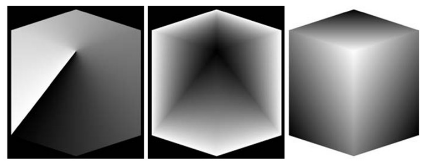
$$R = 3I - (R+B)$$

- HSI→ RGB
  - GB sector$(120° \le H \le 240°)$

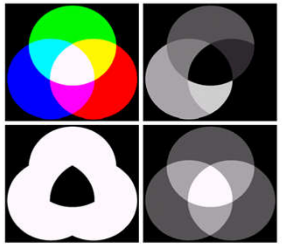
$$R = I(1-S) \quad H = H - 120°$$

$$G = I[1 + \frac{ScosH}{\cos(60° - H)}]$$

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

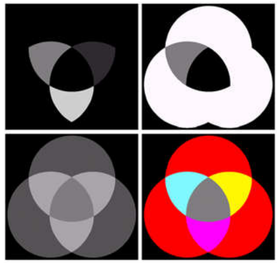


a b c
FIGURE 6.9 HSI component images of an image of an RGB color cube. (a) Hue, (b) saturation, and (c) intensity images.



a b
c d

FIGURE 6.16 (a) RGB image and the components of its corresponding HSI image: (b) hue, (c) saturation, and (d) intensity.

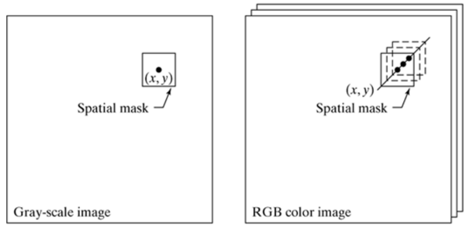


a b
c d

FIGURE 6.17 (a)–(c) Modified HSI component images. (d) Resulting RGB image. (See Fig. 6.16 for the original HSI images.)

The Basics of Color Image Processing

$$c = \begin{bmatrix} cr \\ cg \\ cb \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix} \qquad c = \begin{bmatrix} cr(x,y) \\ cg(x,y) \\ cb(x,y) \end{bmatrix} = \begin{bmatrix} R(x,y) \\ G(x,y) \\ B(x,y) \end{bmatrix}$$



a b

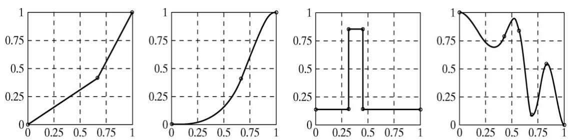
FIGURE 6.10 Spatial masks for gray-scale and RGB color images.

Color Transformations

- $s_i = T_i(r), i = 1, 2, \ldots, n$

  monochrome

  $s_i = T_i(r), i = 1, 2, \ldots, n$
- Interpolation



a b c d

FIGURE 6.11 Specifying mapping functions using control points: (a) and (c) linear interpolation, and (b) and (d) cubic spline interpolation.

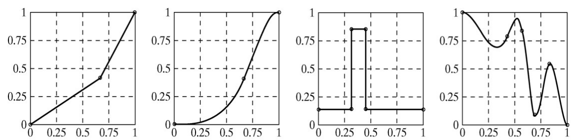
Color Transformations

- z=interp1q(x,y,xi);
- Figure 6.11(a)

  z=interp1q([0 0.6 1]',[0 0.4 1]',linspace(0,1,10)');
  plot(linspace(0,1,10),z);
  grid;

- z=spline(x,y,xi);
- Figure 6.11(b)

  z=spline([0 0.6 1]',[0 0.4 1]',linspace(0,1,10)');
  plot(linspace(0,1,10),z);
  grid;



a b c d

FIGURE 6.11 Specifying mapping functions using control points: (a) and (c) linear interpolation, and (b) and (d) cubic spline interpolation.

Color Transformations

- ICE (interactive color editing)

g=ice('Property Name' ,' property value' ,⋯)

Property Name	Property Value
'image'	An RGB or monochrome input image, f, to be transformed by interactively specified mappings.
'space'	The color space of the components to be modified. Possible values are 'rgb', 'cmy', 'hsi', 'hsv', 'ntsc' (or 'yiq'), and 'ycbcr'. The default is 'rgb'.
'wait'	If 'on' (the default), g is the mapped input image. If 'off', g is the handle of the mapped input image.

```
>>ice
>>f=imread('Fig0303(a)(breast).tif');
>>g=ice('image',f);
>>g=ice('image',f, 'wait', 'off');
>>g=ice('image',f, 'space', 'hsi');
```


Color Transformations

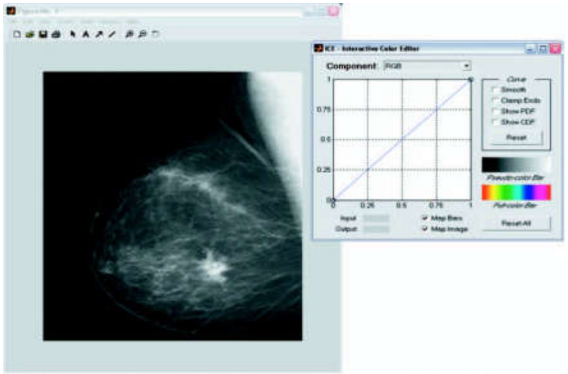
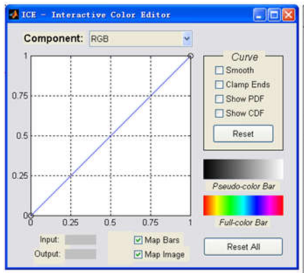


FIGURE 6.12 The typical opening windows of function ice. (Image courtesy of G. E. Medical Systems.)

ICE

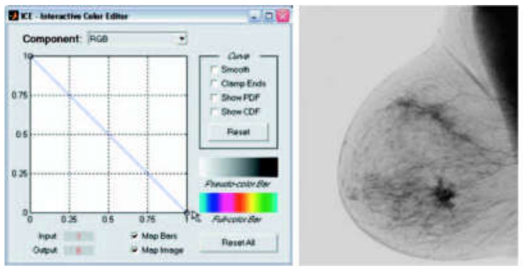


Mouse Action[†]	Result
Left Button	Move control point by pressing and dragging.
Left Button + Shift Key	Add control point. The location of the control point can be changed by dragging (while still pressing the Shift Key).
Left Button + Control Key	Delete control point.

[†] For three button mice, the left, middle, and right buttons correspond to the move, add, and delete operations in the table.

GUI Element	Function
Smooth	Checked for cubic spline (smooth curve) interpolation. If unchecked, piecewise linear interpolation is used.
Clamp Ends	Checked to force the starting and ending curve slopes in cubic spline interpolation to 0. Piecewise linear interpolation is not affected.
Show PDF	Display probability density function(s) [i.e., histogram(s)] of the image components affected by the mapping function.
Show CDF	Display cumulative distribution function(s) instead of PDFs. (Note: PDFs and CDFs cannot be displayed simultaneously.)
Map Image	If checked, image mapping is enabled; otherwise it is not.
Map Bars	If checked, pseudo- and full-color bar mapping is enabled; otherwise the unmapped bars (a gray wedge and hue wedge, respectively) are displayed.
Reset	Initialize the currently displayed mapping function and uncheck all curve parameters.
Reset All	Initialize all mapping functions.
Input/Output	Shows the coordinates of a *selected* control point on the transformation curve. Input refers to the horizontal axis, and Output to the vertical axis.
Component	Select a mapping function for interactive manipulation. In RGB space, possible selections include R, G, B, and RGB (which maps all three color components). In HSI space, the options are H, S, I, and HSI, and so on.

Inverse Mappings: Monochrome negatives



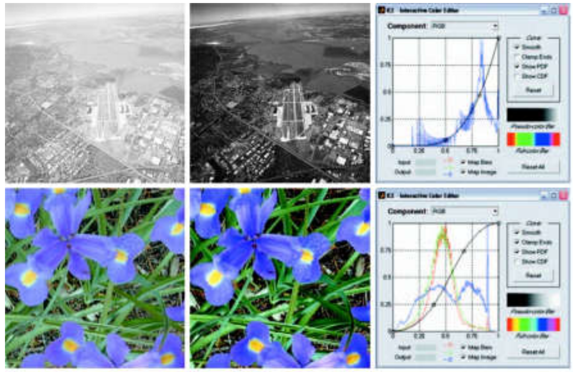
a b

FIGURE 6.13 (a) A negative mapping function, and (b) its effect on the monochrome image of Fig. 6.12.

Inverse Mappings: Color Complement

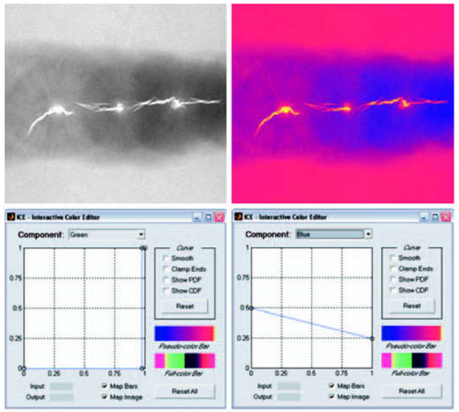
a b
FIGURE 6.14 (a) A full color image, and (b) its negative (color complement).

Color Transformations



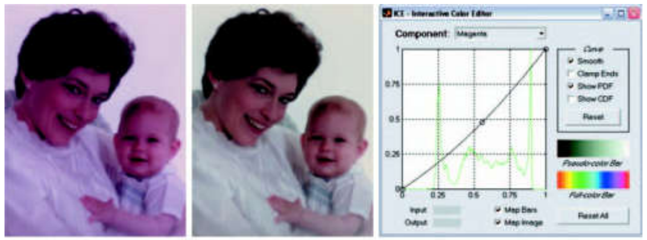
a b c
d e f
FIGURE 6.15 Using function ice for monochrome and full color contrast enhancement: (a) and (d) are the input images, both of which have a "washed-out" appearance; (b) and (e) show the processed results; (c) and (f) are the ice displays. (Original monochrome image for this example courtesy of NASA.)

Pseudocolor Mappings



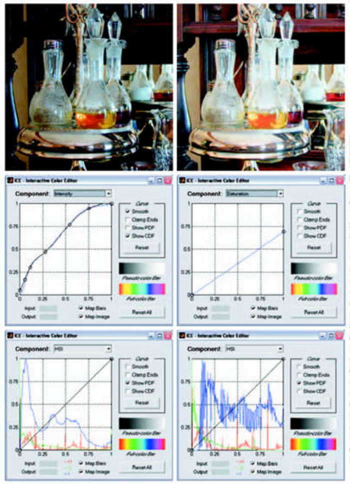
a b
c d
FIGURE 6.16 (a) X-ray of a defective weld; (b) a pseudo-color version of the weld; (c) and (d) mapping functions for the green and blue components. (Original image courtesy of X-TEK Systems, Ltd.)

Color Balancing



a b c
FIGURE 6.17 Using function ice for color balancing: (a) an image heavy in magenta; (b) the corrected image; and (c) the mapping function used to correct the imbalance.

Histogram Equalization for Color Images



a b
c d
e f

FIGURE 6.18 Histogram equalization followed by saturation adjustment in the HSI color space: (a) input image; (b) mapped result; (c) intensity component mapping function and cumulative distribution function; (d) saturation component mapping function; (e) input image's component histograms; and (f) mapped result's component histograms.

Spatial Filtering of Color Images

- The average of the RGB vectors in its neighborhood is

$$\bar{c}(x,y) = \frac{1}{K}\sum_{(s,t)\in S_{xy}} c(s,t)$$

$$\bar{c}(x,y) = \begin{bmatrix} \frac{1}{K}\sum_{(s,t)\in S_{xy}} R(s,t) \\ \frac{1}{K}\sum_{(s,t)\in S_{xy}} G(s,t) \\ \frac{1}{K}\sum_{(s,t)\in S_{xy}} B(s,t) \end{bmatrix}$$

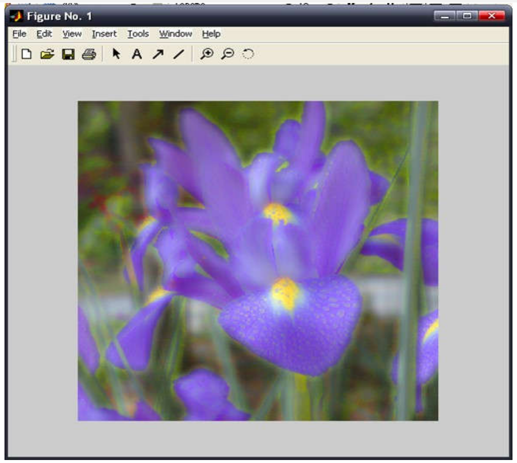
Spatial Filtering of Color Images

- Smoothing an RGB color image,fc, with a linear spatial filter consists of the following steps:
  - 1.Extract
    fR=fc(:,:,1);···
  - 2.Filter each component image individually
    fR_filtered=imfilter(fR,w);···
  - 3.Reconstruct
    fc_filtered=cat(3,fR_filtered,..);

Spatial Filtering of Color Images

- >> fc=imread('Fig0619(a)(RGB_iris).tif');
- >> h=rgb2hsi(fc);
- >> H=h(:,:,1);S=h(:,:,2);I=h(:,:,3);
- >> w=fspecial('average',25);
- >> I_filtered=imfilter(I,w,'replicate');
- >> h=cat(3,H,S,I_filtered);
- >> f=hsi2rgb(h);
- >> f=min(f,1);
- >> imshow(f);

Spatial Filtering of Color Images



Spatial Filtering of Color Images

a b
c d

FIGURE 6.19 (a) RGB image; (b) through (d) are the red, green and blue component images, respectively.

Spatial Filtering of Color Images

a b c

FIGURE 6.20 From left to right: hue, saturation, and intensity components of Fig. 6.19(a).

a b c

FIGURE 6.21 (a) Smoothed RGB image obtained by smoothing the R, G, and B image planes separately. (b) Result of smoothing only the intensity component of the HSI equivalent image. (c) Result of smoothing all three HSI components equally.

Spatial Filtering of Color Images

- Color image sharpening

$$\nabla^2[c(x,y)] = \begin{bmatrix} \nabla^2 R(x,y) \\ \nabla^2 G(x,y) \\ \nabla^2 B(x,y) \end{bmatrix}$$

  - fb=imread('Fig0619(a)(RGB_ iris).tif');
  - lapmask=[1 1 1;1 -8 1;1 1 1];
  - fen=imsubtract(fb,imfilter(fb,lapmask,'replicate'));
  - imshow(fen);

Spatial Filtering of Color Images

a b
FIGURE 6.22 (a) Blurred image. (b) Image enhanced using the Laplacian, followed by contrast enhancement using function ice.

Working Directly in RGB Vector Space

- Color Edge Detection Using the Gradient

gradient

$$\nabla f = \begin{bmatrix} G_x \\ G_y \end{bmatrix} = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix}$$

magnitude

$$|\nabla f| = mag(\nabla f) = [G_x^2 + G_y^2]^{1/2} = [(\frac{\partial f}{\partial x})^2 + (\frac{\partial f}{\partial y})^2]^{1/2}$$

angle

$$\alpha\,(x,y) = tan^{-1}(\frac{G_x}{G_y})$$

z_1	z_2	z_3
z_4	z_5	z_6
z_7	z_8	z_9

−1	−2	−1
0	0	0
1	2	1

−1	0	1
−2	0	2
−1	0	1

a b c
FIGURE 6.23 (a) A small neighborhood. (b) and (c) Sobel masks used to compute the gradient in the x (vertical) and y (horizontal) directions, respectively, with respect to the center point of the neighborhood.

Working Directly in RGB Vector Space

- r,g and b are the *unit vectors* along the R,G and B axis.

$$u = \frac{\partial R}{\partial x}r + \frac{\partial G}{\partial x}g + \frac{\partial B}{\partial x}b$$

and

$$v = \frac{\partial R}{\partial y}r + \frac{\partial G}{\partial y}g + \frac{\partial B}{\partial y}b$$

- Quantities

$$g_{xx} = u \cdot u = u^T u = |\frac{\partial R}{\partial x}|^2 + |\frac{\partial G}{\partial x}|^2 + |\frac{\partial B}{\partial x}|^2$$

$$g_{yy} = v \cdot v = v^T v = |\frac{\partial R}{\partial y}|^2 + |\frac{\partial G}{\partial y}|^2 + |\frac{\partial B}{\partial y}|^2$$

$$g_{xy} = u \cdot v = u^T v = |\frac{\partial R}{\partial x}\frac{\partial R}{\partial y}| + |\frac{\partial G}{\partial x}\frac{\partial G}{\partial y}| + |\frac{\partial B}{\partial x}\frac{\partial B}{\partial y}|$$

Working Directly in RGB Vector Space

- Angle=the direction of max rate of change of c(x,y)

$$\theta(x,y) = \frac{1}{2}tan^{-1}[\frac{2g_{xy}}{g_{xx} - g_{yy}}]$$

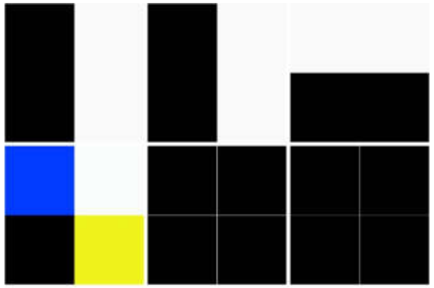
- The value of the rate of the change

$$F_\theta(x,y) = \{\frac{1}{2}[(g_{xx} + g_{yy}) + (g_{xx} - g_{yy})\cos 2\theta + 2g_{xy}sin2\theta]\}^{1/2}$$

Working Directly in RGB Vector Space

- The IPT for color gradient for RGB images

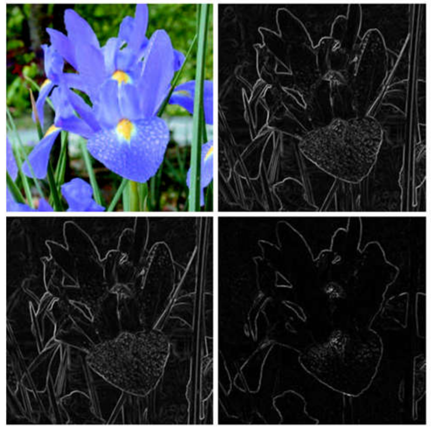
  [VG,A,PPG]=colorgrad(f,T)



a b c
d e f

FIGURE 6.24 (a) through (c) RGB component images (black is 0 and white is 255). (d) Corresponding color image. (e) Gradient computed directly in RGB vector space. (f) Composite gradient obtained by computing the 2-D gradient of each RGB component image separately and adding the results.

Working Directly in RGB Vector Space



a b
c d

FIGURE 6.25
(a) RGB image.
(b) Gradient computed in RGB vector space.
(c) Gradient computed as in Fig. 6.24(f).
(d) Absolute difference between (b) and (c), scaled to the range [0, 1].

Working Directly in RGB Vector Space

- Image Segmentation in RGB Vector Space

$$D(z,m) = ||z - m||$$
$$= [(z-m)'(z-m)]^{1/2}$$
$$= [(z_R - m_R)^2 + (z_G - m_G)^2 + (z_B - m_B)^2]^{1/2}$$

- A useful generalization of the preceding equation is a distance measure of the form.

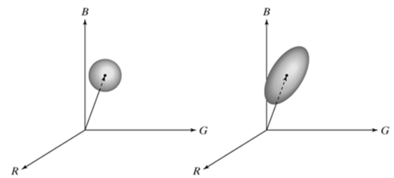
$$D(z,m) = [(z-m)'C^{-1}(z-m)]^{1/2}$$

The matrix C can be defined any format(if you like)

Working Directly in RGB Vector Space

- IPT for segmentaion

  S=colorseg(method,f,T,parameters)



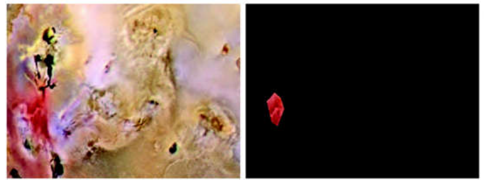
a b

FIGURE 6.26 Two approaches for enclosing data in RGB vector space for the purpose of segmentation.

Working Directly in RGB Vector Space

- >> f=imread('Fig0627(a)(jupitermoon_original).tif');
- >> mask=roipoly(f);
- >> imshow(mask);
- >> red=immultiply(mask,f(:,:,1));
- >> green=immultiply(mask,f(:,:,2));
- >> blue=immultiply(mask,f(:,:,3));
- >> g=cat(3,red,green,blue);
- >> imshow(g);

Working Directly in RGB Vector Space



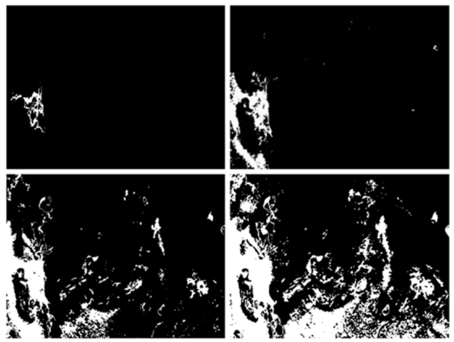
a b

FIGURE 6.27 (a) Pseudocolor of the surface of Jupiter's Moon Io. (b) Region of interest extracted interactively using function `roipoly`. (Original image courtesy of NASA.)

Working Directly in RGB Vector Space

- >> [M,N,K]=size(g);
- >> I=reshape(g,M*N,3);
- >> idx=find(mask);
- >> I=double(I(idx,1:3));
- >> [C,m]=covmatrix(I);
- >> d=diag(C);
- >> sd=sqrt(d)';
- >> e25=colorseg('euclidean',f,25,m);

Working Directly in RGB Vector Space



a b
c d

FIGURE 6.28 (a) through (d) Segmentation of Fig. 6.27(a) using option `'euclidean'` in function `colorseg` with $T = 25, 50, 75$, and 100, respectively.

Working Directly in RGB Vector Space

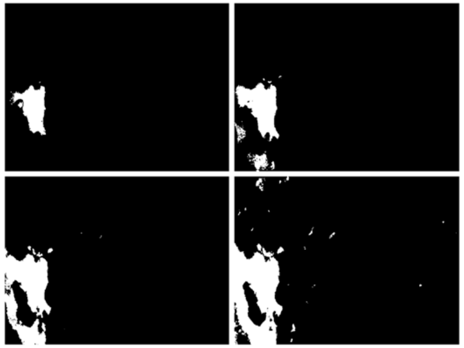


FIGURE 6.29 (a) through (d) Segmentation of Fig. 6.27(a) using option `'mahalanobis'` in function `colorseg` with $T = 25, 50, 75,$ and 100, respectively. Compare with Fig. 6.28.