فصل ۷ پردازش تصویر رنگی

تصاویر رنگی این فصل را از سایت انتشارات علـوم رایانـه بـا آدرس www.olomrayaneh.net و پیونـد "کتاب‌های الکترونیکی" دریافت نمایید.

مرور کلی

در این فصل، مبانی پردازش تصویر رنگی با استفاده از جعبه ابزار پردازش تصویر را بحث می‌کنیم و بعضی از عملکردهای آن را با توسعه توابع تبدیل و تولید رنگ، بسط می‌دهیم (برای مطالعه این فصل، بهتر است فصل پردازش تصویر رنگی، یعنی فصل ششم کتاب "پردازش تصویر دیجیتال" را مطالعه کنید که توسط اینجانب ترجمه شده است (م)).

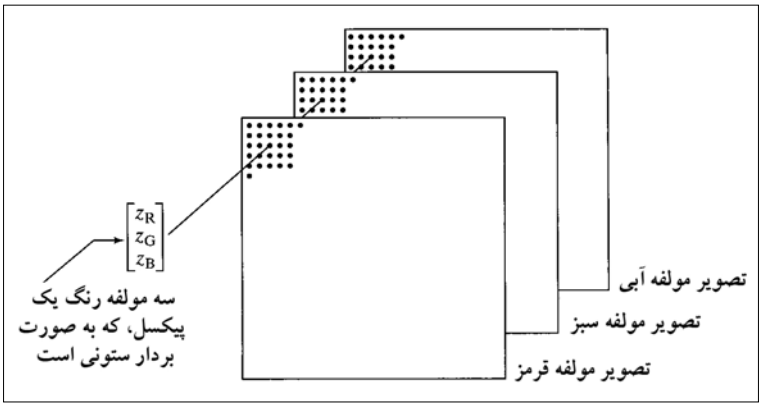
۷–۱ نمایش تصویر رنگی در MATLAB

همان‌طور که در بخش ۶–۲ گفته شد، جعبه ابزار پردازش تصویر، تصاویر رنگی را به صورت تصاویر شاخص‌دار یا تصاویر RGB (قرمز، سبز، آبی) اداره می‌کند. در این بخش این دو نوع تصویر را بحث می‌کنیم.

۷–۱–۱ تصاویر RGB

تصویر رنگی RGB ، یک آرایه $M \times N \times 3$ از *پیکسل‌های رنگی* است که هر پیکسل رنگ، یک سه‌تایی متناظر با مولفه‌های قرمز، سبز و آبیِ تصویر RGB در مکان خاصی است (شکل ۷–۱ را ببینید). تصویر RGB را می‌توان به عنوان "پشته‌ای[1]" از سه تصویر خاکستری دانست که وقتی به ورودی‌های قرمز، سبز و آبی مانیتور رنگی وارد می‌شوند، یک تصویر رنگی را در صفحه ایجاد می‌کنند. طبق قرارداد، سه تصویر سازنده‌ی تصویر رنگی RGB را *تصاویر مولفه‌ی*[2] قرمز، سبز و آبی می‌نامند. کلاس داده‌ی تصاویر مولفه، بازه‌ی مقادیر آن‌ها را تعیین می‌کند. اگر یک تصویر RGB ، از کلاس double باشد، بازه‌ی مقادیر آن [0, 1] است. به طور مشابه، بازه‌ی مقادیر [0, 255] یا [0, 65535] به ترتیب، مربوط به تصاویر RGB از کلاس uint8 و uint16 می‌باشد. تعداد بیت‌های مورد استفاده برای نمایش مقادیر پیکسل‌های تصاویر مولفه، *عمق بیت* تصویر RGB را تعیین می‌کند. برای مثال، اگر هر تصویر مولفه، یک تصویر ۸ بیتی باشد، تصویر RGB متناظر، دارای عمق ۲۴ بیت است.

1. stack 2. component images



شکل ۱–۷ نمایش چگونگی شکل‌گیری پیکسل‌های تصویر رنگی RGB از پیکسل‌های متناظر سه تصویر مولفه.

به طور کلی، تعداد بیت‌ها در تمام تصویرهای مولفه، یکسان است. در این حالت، تعداد رنگ‌های ممکن در تصویر RGB ، برابر با $(2^b)^3$ است که b تعداد بیت‌ها در هر تصویر مولفه می‌باشد. برای تصویر ۸ بیتی، این تعداد برابر با 16,777,216 رنگ است.

فرض کنید fR ، fG ، fB سه تصویر مولفه RGB باشند. تصویر RGB ، با استفاده از عملگر cat ، از این تصاویر به وجود می‌آید:

```
rgb_image = cat(3, fR, fG, fB)
```

ترتیب قرارگرفتن تصاویر در لیست عملوندها، مهم است. به طور کلی، cat(dim, A1, A2, . . .) آرایه‌ها را در امتداد بُعد مشخص‌شده توسط dim الحاق[1] می‌کند. برای مثال، اگر dim = 1 ، آرایه‌ها به طور عمود چیده می‌شوند، اگر dim = 2 ، به طور افقی چیده می‌شوند، و اگر dim = 3 ، در بُعد سوم به صورت پشته درمی‌آیند، که در شکل ۱–۷ نشان داده شده است.

اگر تمام تصاویر مولفه یکسان باشند، نتیجه یک تصویر خاکستری است. فرض کنید rgb_image یک تصویر RGB باشد. دستورات زیر، سه مولفه‌ی تصویر را استخراج می‌کنند:

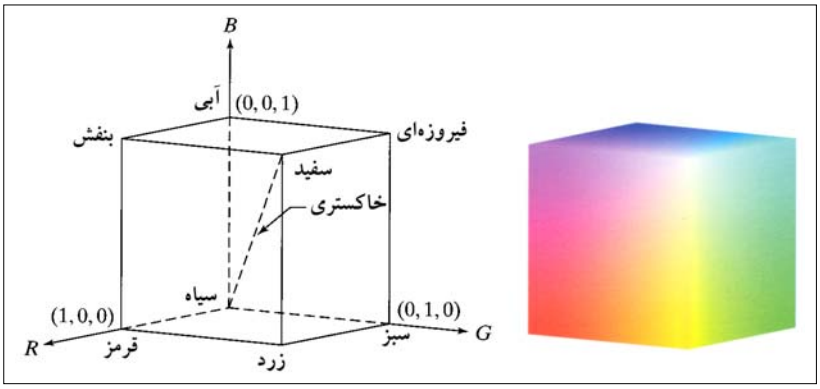
```
>> fR = rgb_image(:, :, 1);
>> fG = rgb_image(:, :, 2);
>> fB = rgb_image(:, :, 3);
```

فضای رنگ RGB معمولاً به شکل گرافیکی، به صورت مکعب رنگ RGB نمایش داده می‌شود که در شکل ۲–۷ آمده است. رئوس مکعب، رنگ‌های *اولیه* (قرمز، سبز و آبی) و *ثانویه‌ی* (فیروزه‌ای، بنفش و زرد) نور می‌باشند.

برای دیدن مکعب رنگ از هر پرسپکتیو، از تابع rgbcube استفاده کنید:

```
rgbcube(vx, vy, vz)
```

1. concatenate



شکل ۲-۷ (الف) شماتیک مکعب رنگ RGB که رنگ‌های اولیه و ثانویه را در رئوس نشان می‌دهد. نقاط در امتداد قطر اصلی، دارای مقادیر خاکستری، از سیاه در مبدأ تا سفید در نقطه (1, 1, 1) است. (ب) مکعب رنگ RGB . **ب الف**

با تایپ rgbcube(vx, vy, vz) در خط فرمان، یک مکعب RGB در دسکتاپ MATLAB ایجاد می‌شود، که از نقطه‌ی (vx, vy, vz) دیده می‌شود. تصویر حاصل می‌تواند با استفاده از تابع print در دیسک ذخیره شود، که در بخش ۴-۲ بحث شد. کد تابع rgbcube به صورت زیر است:

rgbcube

```
function rgbcube(vx, vy, vz)
%RGBCUBE Displays an RGB cube on the MATLAB desktop.
%   RGBCUBE(VX, VY, VZ) displays an RGB color cube, viewed from point
%   (VX, VY, VZ). With no input arguments, RGBCUBE uses (10,10,4) as
%   the default viewing coordinates. To view individual color
%   planes, use the following viewing coordinates, where the first
%   color in the sequence is the closest to the viewing axis, and the
%   other colors are as seen from that axis, proceeding to the right
%   right (or above), and then moving clockwise.
%
%      ----------------------------------------------------
%           COLOR PLANE                  ( vx,  vy,  vz)
%      ----------------------------------------------------
%       Blue-Magenta-White-Cyan          (  0,   0,  10)
%       Red-Yellow-White-Magenta         ( 10,   0,   0)
%       Green-Cyan-White-Yellow          (  0,  10,   0)
%       Black-Red-Magenta-Blue           (  0, -10,   0)
%       Black-Blue-Cyan-Green            (-10,   0,   0)
%       Black-Red-Yellow-Green           (  0,   0, -10)

% Set up parameters for function patch.
vertices_matrix = [0 0 0;0 0 1;0 1 0;0 1 1;1 0 0;1 0 1;1 1 0;1 1 1];
faces_matrix = [1 5 6 2;1 3 7 5;1 2 4 3;2 4 8 6;3 7 8 4;5 6 8 7];
colors = vertices_matrix;
% The order of the cube vertices was selected to be the same as
% the  order of the (R,G,B) colors (e.g., (0,0,0) corresponds to
% black, (1, 1, 1) corresponds to white, and so on.)

% Generate RGB cube using function patch.
patch('Vertices', vertices_matrix, 'Faces', faces_matrix, ...
```

```
        'FaceVertexCData', colors, 'FaceColor', 'interp', ...
        'EdgeAlpha', 0)

% Set up viewing point.
if nargin == 0
    vx = 10; vy = 10; vz = 4;
elseif nargin ~= 3
   error('Wrong number of inputs.')
end
axis off
view([vx, vy, vz])
axis square
```

۷–۱–۲ تصاویر شاخص‌دار

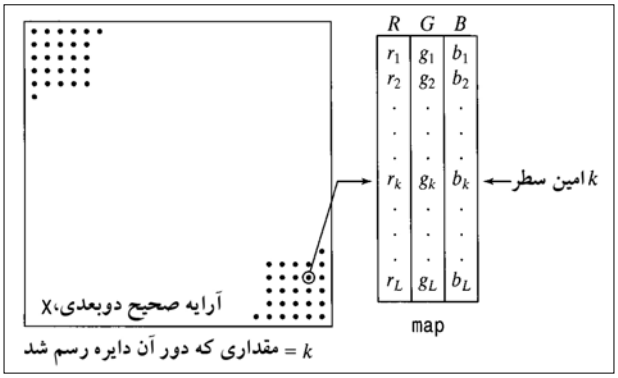
تصویر شاخص‌دار دو مولفه دارد: *ماتریس داده‌ی* صحیح X ، و *ماتریس نگاشت رنگ* map . ماتریس map یک آرایه $m\times 3$ در کلاس double است که شامل مقادیر ممیز شناور در بازه‌ی [0, 1] است. طول map برابر با تعداد رنگ‌هایی است که تعریف می‌کند. هر سطر map ، مولفه‌های قرمز، سبز و آبی یک رنگ را مشخص می‌نماید (اگر سه ستون map یکسان باشند، نگاشت رنگی تبدیل به نگاشت خاکستری می‌شود). تصویر شاخص‌دار از "نگاشت مستقیم" مقادیر شدت پیکسل به مقادیر نگاشت رنگی استفاده می‌کند. رنگ هر پیکسل با استفاده ازمقدار متناظرِ ماتریس صحیح X به عنوان شاخص map تعیین می‌شود (به همین دلیل، آن را تصویر شاخص‌دار گویند). اگر X از کلاس double باشد، آنگاه مقدار ۱ به اولین سطر در map ، مقدار ۲ به دومین سطر، و غیره اشاره می‌کند. اگر X از کلاس uint8 یا uint16 باشد، آنگاه صفر به اولین سطر در map اشاره می‌کند. این مفاهیم در شکل ۳–۷ شرح داده شدند.

برای نمایش تصویر شاخص‌دار، می‌نویسیم:

```
>> imshow(X, map)
```

یا می‌نویسیم:

```
>> image(X)
>> colormap(map)
```



شکل ۳–۷ عناصر تصویر شاخص‌دار. مقدار یک عنصر از آرایه صحیح X ، شماره سطر را در نگاشت رنگ مشخص می‌کند. هر سطر شامل یک سه تایی RGB است، و L تعداد کل سطرها است.

جدول ۱-۷ مقادیر RGB مربوط به بعضی از رنگ‌های پایه. به جای سه‌تایی عددی می‌توان از اسامی طولانی یا اسامی کوتاه استفاده کرد.

نام طولانی	طول کوتاه	مقادیر RGB
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]

نگاشت رنگی، با یک تصویر شاخص‌دار ذخیره می‌شود و هنگامی که از تابع imread برای بارکردن تصویر استفاده می‌گردد، به طور خودکار با تصویر بار می‌شود.

گاهی لازم است تصویر شاخص‌دار با رنگ‌های کمتری تقریب شود. برای این کار، از تابع imapprox به صورت زیر استفاده می‌کنیم:

```
[Y, newmap] = imapprox(X, map, n)
```

این تابع، آرایه Y را با نگاشت رنگ newmap برمی‌گرداند، که حداکثر n رنگ دارد. آرایه ورودی X می‌تواند از کلاس uint8 ، uint16 ، یا double باشد. اگر n کوچک‌تر یا مساوی ۲۵۶ باشد، خروجی Y از کلاس uint8 است. اگر n بزرگ‌تر از ۲۵۶ باشد، Y از کلاس double است.

وقتی تعداد سطرها در یک map ، کمتر از تعداد مقادیر صحیحِ مجزا در X باشد، چندین مقدار در X ، به یک رنگ در map نسبت داده می‌شوند. برای مثال، فرض کنید X شامل چهار باند عمودی با پهنای یکسان، با مقادیر 1 ، 64 ، 128 و 256 است. اگر نگاشت رنگ map = [0 0 0 ; 1 1 1] را مشخص کنیم، آنگاه تمام عناصر در X با مقدار ۱، به اولین سطر (سیاه) map و بقیه‌ی عناصر به سطر دوم (سفید) اشاره خواهند کرد. بنابراین، دستور imshow(X, map) تصویری را با باند سیاه و سپس سه باند سفید نشان می‌دهد. در حقیقت، این نکته تا زمانی درست است که طول map به ۶۵ تبدیل شود، که در آن زمان، یک نوار سیاه، سپس یک نوار خاکستری، و سپس دو نوار سفید نمایش داده می‌شود. اگر طول map از بازه‌ی مجاز عناصر X بیشتر شود، تصویر به خوبی مشاهده نمی‌شود.

روش‌های متعددی برای مشخص‌کردن نگاشت رنگ وجود دارد. یک روش استفاده از دستور زیر است:

```
>> map(k, :) = [r(k) g(k) b(k)];
```

که [r(k) g(k) b(k)] مقادیر RGB هستند که یک سطر از نگاشت رنگ را مشخص می‌کنند. نگاشت با تغییر k پر می‌شود.

جدول ۱-۷ مقادیر RGB چندین رنگ پایه را نشان می‌دهد. هر یک از سه فرمتِ نشان‌داده‌شده در جدول، می‌تواند برای مشخص‌کردن رنگ‌ها استفاده شود. برای مثال، رنگ پس‌زمینه‌ی شکل، می‌تواند با هر یک از سه دستور زیر به سبز تغییر یابد:

```
>> whitebg('g');
>> whitebg('green');
>> whitebg([0 1 0]);
```

رنگ‌های دیگر علاوه‌بر رنگ‌های جدول ۱-۷، شامل مقادیر کسری می‌باشند. برای مثال [.5 .5 .5] خاکستری، [0 .5 0] قرمز تیره، و [.83 1 .49] کبود است.

MATLAB چندین نگاشت رنگِ تعریف‌شده را فراهم می‌سازد که با دستور زیر قابل دستیابی است:

```
>> colormap(map_name);
```

که نگاشت رنگ را برابر با ماتریس map_name قرار می‌دهد؛ دستور زیر را ببینید:

```
>> colormap(copper)
```

که copper یک تابع نگاشت رنگ MATLAB است. رنگ‌ها در این نگاشت، به طور یکنواخت از سیاه به مسی روشن تغییر می‌کند. اگر آخرین تصویر نشان‌داده‌شده، یک تصویر شاخص‌دار بود، این دستور، نگاشت رنگ خود را به copper تغییر می‌دهد. از طرف دیگر، تصویر می‌تواند مستقیماً با نگاشت رنگ مطلوب نمایش داده شود:

```
>> imshow(X, copper)
```

جدول ۲-۷ نگاشت‌های رنگ از پیش‌تعریف‌شده در MATLAB را نشان می‌دهد. طول (تعداد رنگ‌های) این نگاشت‌های رنگ می‌تواند با قراردادن اعدادی در پرانتز مشخص شود. برای مثال، gray(8) یک نگاشت رنگ با ۸ سایه خاکستری را تولید می‌کند.

جدول ۲-۷ نگاشت‌های رنگ ازپیش‌تعریف‌شده در MATLAB.

تابع	شرح
Autumn	به طور یکنواخت از قرمز به نارنجی به زرد تغییر می‌کند.
bone	نگاشت رنگ خاکستری با مقدار بالاتر برای مولفه آبی. این نگاشت رنگ، برای افزودن نمای "الکترونیکی" به تصاویر خاکستری مفید است.
colorcube	دارای رنگ‌های با فاصله‌های منظم در فضای رنگ RGB است، و تلاش می‌کند مراحل بیشتری از خاکستری، قرمز خالص، سبز خالص، و آبی خالص فراهم شود.
cool	شامل رنگ‌هایی است که سایه‌های متغیر یکنواختی از فیروزه‌ای به بنفش دارد.
copper	به طور یکنواخت از سیاه به مسی روشن تغییر می‌کند.
flag	شامل رنگ‌های قرمز، سفید، آبی و سیاه است. این نگاشت رنگ، با هر افزایش شاخص، کاملاً تغییر رنگ می‌دهد.
gray	یک نگاشت رنگ خاکستری خطی را برمی‌گرداند.
hot	به طور یکنواخت از سیاه به سایه‌های قرمز، نارنجی، و زرد، به سفید تغییر می‌کند.
hsv	مولفه‌ی پرده‌ی رنگ مدل رنگ پرده – اشباع – مقدار را تغییر می‌دهد. رنگ‌ها، قرمز می‌شوند، از زرد، سبز، فیروزه‌ای، آبی، بنفش عبور می‌کنند و به قرمز برمی‌گردند. این نگاشت رنگ، برای نشان‌دادن توابع متناوب مفید است.
jet	بازه‌ی آن از آبی به قرمز است، و از رنگ‌های فیروزه‌ای، زرد و نارنجی عبور می‌کند.

جدول ۲-۷ نگاشت‌های رنگ ازپیش‌تعریف‌شده در MATLAB (ادامه).	
lines	یک نگاشت رنگ از رنگ‌های مشخص‌شده توسط محورهای خاصیت و سایه خاکستری ColorOrder تولید می‌کند.
pink	شامل سایه‌های پاستل از صورتی است. نگاشت رنگ صورتی، رنگ‌آمیزی تن قهوه‌ای تیره‌ی مربوط به عکس‌های خاکستری را فراهم می‌سازد.
prism	شش رنگ قرمز، نارنجی، زرد، سبز، آبی و بنفش را تکرار می‌کند.
spring	شامل رنگ‌هایی است که سایه‌هایی از بنفش و زرد هستند.
summer	شامل رنگ‌هایی است که سایه‌هایی از سبز و زرد هستند.
winter	شامل رنگ‌هایی است که سایه‌هایی از آبی و سبز هستند.
white	نگاشت رنگ سفید.

۳-۱-۷ توابعی برای دستکاری تصاویر RGB و شاخص‌دار

جدول ۳-۷ توابع جعبه‌ابزار را برای تبدیل بین تصاویر RGB ، شاخص‌دار، و خاکستری نشان می‌دهد. برای وضوح نمادگذاری در این بخش، فرض می‌کنیم rgb_image نشان‌دهنده‌ی تصاویر RGB ، gray_image نشان‌دهنده‌ی تصاویر خاکستری، و bw نشان‌دهنده‌ی تصاویر سیاه و سفید (دودویی)، X نشان‌دهنده‌ی مولفه‌ی ماتریس داده‌ی تصاویر شاخص‌دار باشد. به یاد داشته باشید که تصویر شاخص‌دار، مرکب از یک ماتریس داده‌ی صحیح و یک ماتریس نگاشت رنگ است.

تابع dither به هر دو تصویر خاکستری و رنگی اِعمال می‌شود. تقریب‌گیری[1]، فرآیندی است که در صنعت چاپ و نشر استفاده می‌شود تا اثر بصری به تغییرات سایه در صفحه‌ی چاپیِ شامل نقاط را ارائه کند. در مورد تصاویر خاکستری، تقریب‌گیری سعی می‌کند سایه‌های خاکستری را با تولید تصویر دودویی از نقاط سیاه روی پس‌زمینه‌ی سفید (یا برعکس) تهیه نماید. اندازه‌های نقاط، از نقاط کوچک در ناحیه‌های روشن به نقاطی بزرگ‌تر در ناحیه‌های تیره تغییر می‌کند. نکته مهم در پیاده‌سازی الگوریتم تقریب، برقراری توازن بین "دقت" دریافت بصری و پیچیدگی محاسباتی است. روش تقریب‌گیریِ مورد استفاده در جعبه‌ابزار مبتنی بر الگوریتم

جدول ۳-۷ توابع جعبه‌ابزار برای تبدیل بین تصاویر RGB ، شاخص‌دار و خاکستری.	
تابع	**شرح**
dither	یک تصویر شاخص‌دار از تصویر RGB ، به وسیله تقریب ایجاد می‌کند.
grayslice	به وسیله آستانه‌گیری، یک تصویر شاخص‌دار از تصویری با شدت خاکستری را ایجاد می‌کند.
gray2ind	یک تصویر شاخص‌دار از تصویری با شدت خاکستری ایجاد می‌کند.
ind2gray	یک تصویر خاکستری از تصویر شاخص‌دار ایجاد می‌کند.
rgb2ind	یک تصویر شاخص‌دار از تصویر RGB ایجاد می‌کند.
ind2rgb	یک تصویر RGB از تصویر شاخص‌دار ایجاد می‌کند.
rgb2gray	یک تصویر خاکستری از تصویر RGB ایجاد می‌کند.

1. dithering

Floyd-Steinberg ، و Ulichney است. تابع dither برای تصاویر خاکستری به صورت زیر به کار می‌رود:

```
bw = dither(gray_image)
```

که، همان‌طور که گفته شد، gray_image ، یک تصویر خاکستری و dw تصویر دودویی تقریبی (از کلاس logical) است.

هنگام کارکردن با تصاویر، تقریب‌گیری، همراه با تابع rgb2ind به کار می‌رود تا تعداد رنگ‌ها در تصویر کاهش یابد. این تابع در ادامه‌ی این بخش بحث می‌شود.

تابع grayslice به صورت زیر به کار می‌رود:

```
X = grayslice(gray_image, n)
```

این تابع، یک تصویر شاخص‌دار را با آستانه‌گیری تصویر خاکستری با مقادیر آستانه زیر، تولید می‌کند:

$$\frac{1}{n}, \frac{2}{n}, \ldots, \frac{n-1}{n}$$

همان‌طور که گفته شد، تصویر شاخص‌دار حاصل را می‌توان با دستور imshow(X, map) و با استفاده از نگاشتی با طول مناسب (یعنی jet(16)) مشاهده کرد. کاربرد دیگر به صورت زیر است:

```
X = grayslice(gray_image, v)
```

v برداری است (با مقادیری در بازه‌ی [0, 1]) که برای آستانه‌گیری gray_image استفاده می‌شود. تابع grayslice ابزار اصلی برای پردازش تصویر شبه‌رنگی[1] است، که باندهای شدت خاکستری معینی به رنگ‌های مختلف نسبت داده می‌شود. تصویر ورودی می‌تواند از کلاس uint8 ، uint16 ، یا double باشد. مقادیر آستانه در v باید در بازه‌ی [0, 1] باشد، حتی اگر تصویر ورودی، کلاس uint8 یا uint16 باشد. این تابع، مقیاس‌بندی لازم را انجام می‌دهد.

تابع gray2ind را با کاربرد زیر درنظربگیرید:

```
[X, map] = gray2ind(gray_image, n)
```

این تابع، تصویر gray_image را مقیاس‌بندی و گرد می‌کند تا یک تصویر شاخص‌دار X را با نگاشت رنگ gray(n) تولید نماید. اگر n حذف شده باشد، پیش‌فرض آن ۶۴ است. تصویر ورودی می‌تواند از کلاس uint8 ، uint16 یا double باشد. کلاس تصویر خروجی X در صورتی uint8 است که n کمتر یا مساوی ۲۵۶ باشد، و در صورتی از کلاس uint16 است که n بزرگ‌تر از ۲۵۶ باشد.

تابع ind2gray با روش کاربرد زیر را درنظربگیرید:

```
gray_image = ind2gray(X, map)
```

این تابع، تصویر شاخص‌دار مرکب از X و map را به تصویر خاکستری تبدیل می‌کند. آرایه X می‌تواند از کلاس uint8 ، uint16 ، یا double باشد. تصویر خروجی از کلاس double است.

---

1. pseudocolor

روش کاربرد rgb2ind که در این فصل با آن کار می‌کنیم، به صورت زیر است:

```
[X, map] = rgb2ind(rgb_image, n, dither_option)
```

که n تعداد رنگ‌های map است و dither_option می‌تواند یکی از دو مقدار را داشته باشد: 'dither' (پیش‌فرض)، که در صورت لزوم، تقریب را انجام می‌دهد تا دقت رنگ بهتری به دست آید. برعکس، 'nodither' هر رنگ موجود در تصویر را به نزدیک‌ترین رنگ در نگاشت جدید نگاشت می‌کند (برحسب مقدار n)؛ هیچ تقریبی انجام نمی‌گیرد. تصویر ورودی می‌تواند از کلاس uint8 ، uint16 یا double باشد. آرایه خروجی X ، در صورتی از کلاس uint8 است که n کمتر یا مساوی 256 باشد، وگرنه از کلاس uint16 است. مثال ۱–۷ اثری که تقریب‌زدن روی دقت رنگ دارد را نشان می‌دهد.

تابع ind2rgb با کاربرد زیر را ببینید:

```
rgb_image = ind2rgb(X, map)
```

این تابع، ماتریس X و نگاشت رنگ متناظر map را به فرمت RGB تبدیل می‌کند. X می‌تواند از نوع کلاس uint8، uint16 یا double باشد. تصویر RGB خروجی، آرایه $M \times N \times 3$ از کلاس double است.

سرانجام، تابع rgb2gray با کاربرد زیر را درنظربگیرید:

```
gray_image = rgb2gray(rgb_image)
```

این تابع، تصویر RGB را به تصویر خاکستری تبدیل می‌کند. تصویر ورودی RGB می‌تواند از کلاس uint8 ، uint16 ، یا double باشد. کلاس تصویر خروجی، مثل تصویر ورودی است.

مثال ۱–۷ : تشریح بعضی از توابع جدول ۳–۷.

تابع rgb2ind برای کاهش تعداد رنگ‌ها در تصویر RGB مفید است. برای تشریح این تابع، و امتیازات استفاده از گزینه تقریب، شکل ۴–۷ (الف) را درنظربگیرید، که تصویر ۲۴ بیتی RGB است (f). شکل‌های ۴–۷ (ب) و (پ) نتایج استفاده از دستورات زیر را نشان می‌دهند:

```
>> [X1, map1] = rgb2ind(f, 8, 'nodither');
>> imshow(X1, map1)
```

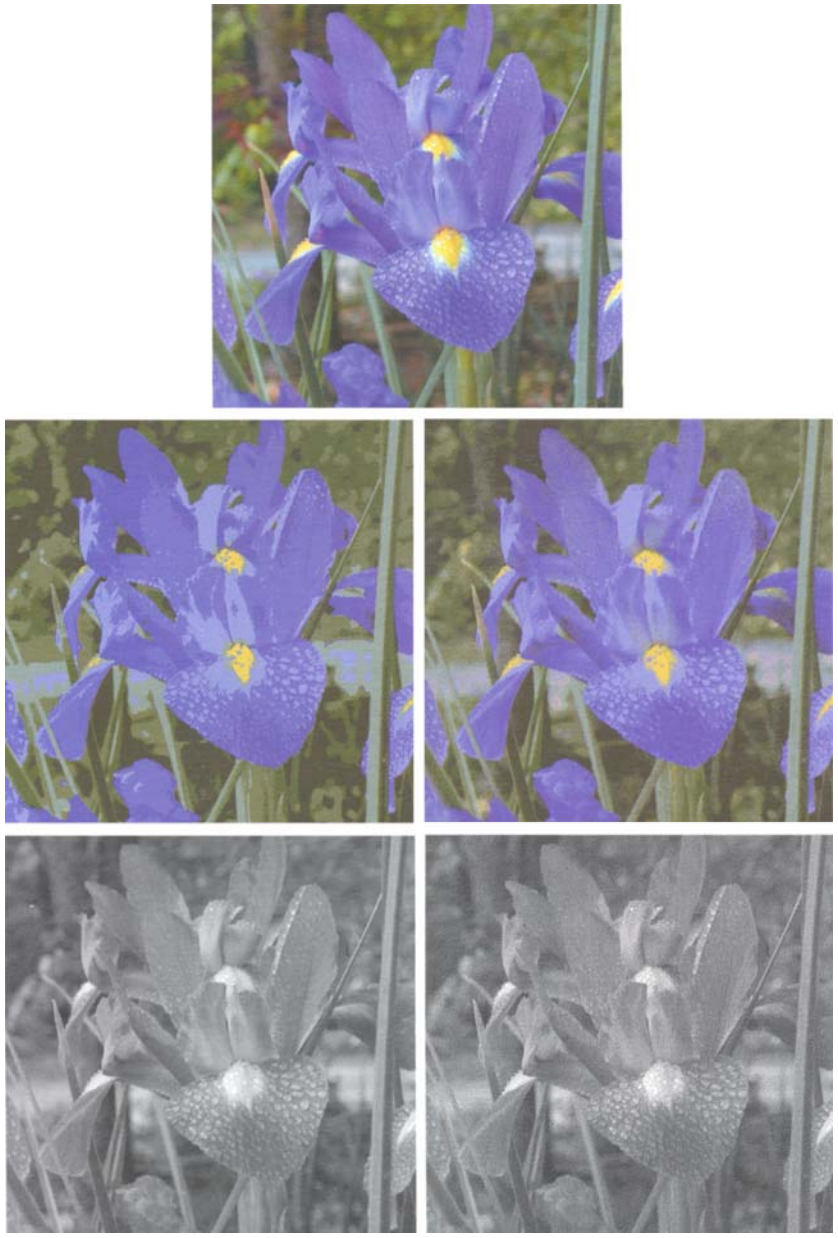
و

```
>> [X2, map2] = rgb2ind(f, 8, 'dither');
>> figure, imshow(X2, map2)
```

هر دو تصویر فقط ۸ رنگ دارند، که کاهش فاحشی در ۱۶ میلیون رنگ ممکن از تصویر f با کلاس uint8 است. شکل ۴–۷ (ب) درجه قابل توجهی از منحنی‌سازی نادرست را، مخصوصاً در مرکز گل بزرگ، نشان می‌دهد. تصویر تقریب‌زده‌شده، تونالیته‌ی بهتری را نشان می‌دهد، و اشتباه منحنی‌سازی کمتر است، اما از نظر بصری نسبت به شکل ۴–۷ (ب) بهتر است.

آثار تقریب، معمولاً با تصویر خاکستری، بهتر توضیح داده می‌شود. شکل‌های ۴–۷ (ت) و (ث) با دستورات زیر به دست می‌آیند:

```
>> g = rgb2gray(f);
>> g1 = dither(g);
>> figure, imshow(g); figure, imshow(g1)
```



الف
پ ب
ث ت

شکل ۴-۷ (الف) تصویر RGB . (ب) تعداد رنگ‌هایی که به ۸ کاهش یافتند و تقریب انجام نشده است. (پ) تعداد رنگ‌ها با عمل تقریب، به ۸ کاهش یافت. (ت) نسخه‌ی خاکستری (الف) که با تابع rgb2gray به دست آمد. (ث) تصویر خاکستری تقریب‌زده‌شده (یک تصویر دودویی است).

تصویر شکل ۴-۷ (ث) دودویی است، که درجه خوبی از کاهش داده‌ها را نشان می‌دهد. شکل‌های ۴-۷ (پ) و (ت) نشان می‌دهند که چرا تقریب‌زدن، در صنعت چاپ و نشر مفید است، به خصوص در وضعیت‌هایی (مثل روزنامه‌ها) که کیفیت کاغذ و دقت چاپ پایین است. ■

۷-۲ تبدیل بین فضاهای رنگ

همان‌طور که در بخش قبل گفته شد، جعبه‌ابزار، رنگ‌ها را به صورت مقادیر RGB ، به طور مستقیم در تصویر RGB ، یا غیر مستقیم در تصویر شاخص‌دار نمایش می‌دهد. نگاشت رنگ با فرمت RGB ذخیره می‌شود. اما، فضاهای رنگ دیگری به نام مدل‌های رنگ نیز وجود دارند که ممکن‌است کاربرد آن‌ها نسبت به RGB راحت‌تر و با معناتر باشد. این مدل‌ها، تبدیلات مدل RGB هستند و شامل فضاهای رنگ NTSC ، YCbCr ، HSV ، CMY ، CMYK و HSI است. جعبه‌ابزار، توابع تبدیل از RGB به فضاهای رنگ NTSC ، YCbCr ، HSV و CMY دارد و برعکس. توابع دیگری در ادامه‌ی این بخش، برای تبدیل فضاهای رنگ نوشته شدند.

۷-۲-۱ فضای رنگ NTSC

سیستم رنگ NTSC در تلویزیون آنالوگ استفاده می‌شود. یکی از امتیازات اصلی این فرمت این‌است‌که اطلاعات مقیاس خاکستری، جدا از داده‌ی رنگ است، و در نتیجه یک سیگنال می‌تواند هم برای تلویزیون‌های رنگی و هم سیاه و سفید استفاده شود. در فرمت NTSC ، داده‌های تصویر شامل سه مولفه است: لومینانس (Y)، پرده رنگ (I)، و اشباع (Q)، که انتخاب حروف YIQ ، قراردادی است. مولفه‌ی لومینانس، اطلاعات مقیاس خاکستری را نشان می‌دهد، و دو مولفه دیگر اطلاعات رنگ سیگنال TV را حمل می‌کنند. مولفه‌های YIQ ، با استفاده از تبدیل زیر، از مولفه‌ی RGB تصویر به دست می‌آیند:

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

توجه کنید که مجموع عناصر سطر اول برابر با 1 ، و مجموع دو سطر بعدی برابر با 0 است. علتش این‌است‌که برای تصویر خاکستری، تمام مولفه‌های RGB برابر هستند، و در نتیجه مولفه‌های I و Q برای چنین تصویری صفر هستند. تابع rgb2ntsc ، تبدیل زیر را انجام می‌دهد:

```
yiq_image = rgb2ntsc(rgb_image)
```

که تصویر RGB ورودی می‌تواند از کلاس uint8 ، uint16 یا double باشد. تصویر خروجی، یک آرایه $M \times N \times 3$ از کلاس double است. تصویر مولفه‌ی (:, :, 1)yiq_image برابر با لومینانس، (:, :, 2)yiq_image برابر با پرده رنگ، و (:, :, 3)yiq_image تصویر اشباع است.

به طور مشابه، مولفه‌های RGB ، با استفاده ازتبدیل خطی زیر از مولفه‌های YIQ به دست می‌آیند:

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

تابع جعبه‌ابزار ntsc2rgb این تبدیل را انجام می‌دهد و به صورت زیر به کار می‌رود:

```
rgb_image = ntsc2rgb(yiq_image)
```

تصاویر ورودی و خروجی از کلاس double می‌باشند.

۷-۲-۲ فضای رنگ YCbCr

فضای رنگ YCbCr به طور گسترده در ویدیوی دیجیتال استفاده می‌شود. در این فرمت، اطلاعات لومینانس با یک مولفه‌ی Y نمایش داده می‌شود و اطلاعات رنگ به صورت دو مولفه‌ی مختلف Cb و Cr ذخیره می‌گردد. مولفه‌ی Cb تفاوت بین مولفه‌ی آبی و مقدار مرجع، و مولفه‌ی Cr تفاوت بین مولفه‌ی قرمز و مقدار مرجع است. تبدیل مورد استفاده توسط جعبه‌ابزار، برای تبدیل RGB به YCbCr به صورت زیر است:

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

تابع تبدیل به صورت زیر است:

```
ycbcr_image = rgb2ycbcr(rgb_image)
```

تصویر ورودی RGB می‌تواند از کلاس uint8 ، uint16 ، یا double باشد. کلاس تصویر خروجی و ورودی یکسان است. تبدیل مشابهی، تبدیل YCbCr به RGB را انجام می‌دهد:

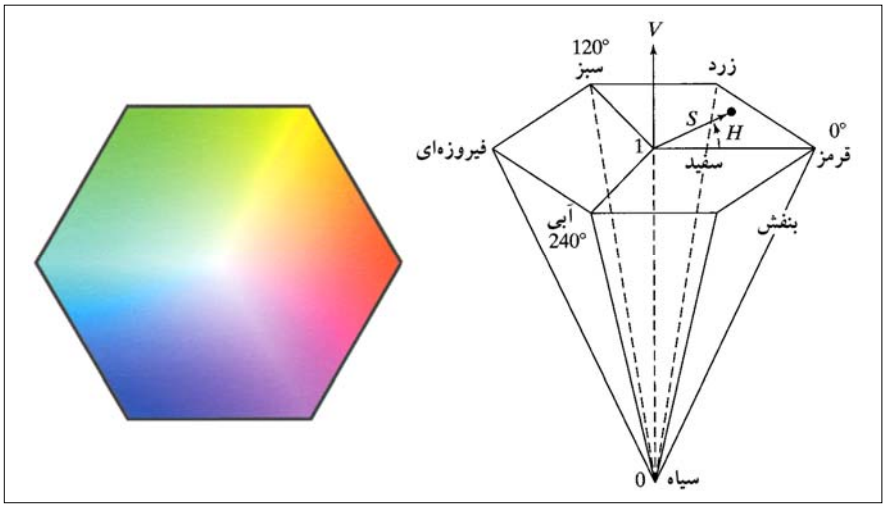
```
rgb_image = ycbr2rgb(ycbcr_image)
```

تصویر ورودی YCbCr می‌تواند از کلاس uint8 ، uint16 ، یا double باشد. کلاس تصویر خروجی مثل کلاس ورودی است.

۷-۲-۳ فضای رنگ HSV

HSV (پرده رنگ، اشباع، مقدار)، یکی از چندین سیستم رنگ است که توسط افراد برای انتخاب رنگ‌ها از جعبه یا چرخ رنگ استفاده می‌شود. این سیستم رنگ، نسبت به RGB به تجربه انسانی نزدیک‌تر است و اشباع‌های رنگ را توصیف می‌کند. از نظر هنری، پرده رنگ، اشباع، و مقدار، تقریباً برابر با tint ، سایه و تُن است.

فضای رنگ HSV با درنظرگرفتن مکعب رنگ RGB در امتداد محور خاکستری آن (محوری که رئوس سیاه و سفید را به هم متصل می‌کند) فرمول‌بندی می‌شود، که منجر به جعبه‌رنگ شش‌ضلعی در شکل ۵-۷ (الف) می‌شود. وقتی در شکل ۵-۷ (ب) در امتداد محور عمودی (خاکستری) پیش می‌رویم، اندازه صفحه‌ی شش‌ضلعی که عمود بر محور است تغییر می‌کند، و حجم نشان‌داده‌شده در شکل را ایجاد می‌کند. پرده رنگ به صورت زاویه‌ای حول شش‌ضلعی رنگ، معمولاً با استفاده از محور قرمز به عنوان محور مرجع ($0°$)، بیان می‌شود. مولفه‌ی مقدار، در امتداد محور مخروط اندازه‌گیری می‌شود. $V = 0$ در انتهای محور، سیاه است. $V = 1$ در انتهای محور، سفید است، که در مرکز شش‌ضلعی رنگی در شکل ۵-۷ (الف) قرار دارد. بنابراین، این محور تمام سایه‌های خاکستری را نشان می‌دهد. اشباع (محض‌بودن رنگ)، به عنوان فاصله‌ای از محور V سنجیده می‌شود.



ب الف **شکل ۵-۷** (الف) شش ضلعی رنگ HSV . (ب) مخروط شش ضلعی HSV .

سیستم رنگ HSV مبتنی بر مختصات استوانه‌ای است. تبدیل RGB به HSV شامل ایجاد معادلاتی برای نگاشت مقادیر RGB (در مختصات دکارتی) به مختصات استوانه‌ای است.

تابع MATLAB برای تبدیل RGB به HSV ، تابع rgb2hsv است که به صورت زیر به کار می‌رود:

```
hsv_image = rgb2hsv(rgb_image)
```

تصویر RGB ورودی می‌تواند از کلاس uint8 ، uint16 یا double باشد. تصویر ورودی از کلاس double است. تابع تبدیل ازHSV به RGB ، تابع hsv2rgb است:

```
rgb_image = hsv2rgb(hsv_image)
```

تصویر ورودی باید از کلاس double باشد و تصویر خروجی نیز از کلاس double است.

۷-۲-۴ فضاهای رنگ CMY و CMYK

فیروزه‌ای، بنفش و زرد، رنگ‌های ثانویه‌ی نور، یا رنگ‌های اولیه‌ی رنگ‌دانه[1] هستند. برای مثال، وقتی سطح پوشیده‌شده با رنگ‌دانه‌ی فیروزه‌ای، با نور سفید تابیده می‌شود، نور قرمز از این سطح منعکس نمی‌شود. یعنی، رنگ‌دانه‌ی فیروزه‌ای، نور قرمز را از نور منعکس‌شده توسط سطح، حذف می‌کند.

اغلب دستگاه‌هایی که رنگ‌دانه‌ها را روی کاغذ می‌پاشند، مثل کپی‌ها و چاپگرهای رنگی، نیاز به ورودی داده‌ی CMY دارند تا تبدیل RGB به CMY را به طور داخلی انجام دهند. یک تبدیل تقریبی می‌تواند با معادله زیر انجام شود:

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

1. pigment

که فرض می‌شود تمام مقادیر رنگ به بازه‌ی [0, 1] نرمال شدند. این معادله، این جمله‌ی پاراگراف قبلی را شرح می‌دهد که: نور منعکس‌شده از سطحی با پوشش فیروزه‌ای خالص، نمی‌تواند شامل قرمز باشد (یعنی $C = 1 - R$ در معادله). به طور مشابه، بنفش خالص، سبز را منعکس نمی‌کند، و زرد خالص، آبی را منعکس نمی‌کند. معادله قبلی نشان می‌دهد که مقادیر RGB به آسانی می‌توانند از مجموعه‌ای از مقادیر CMY به دست آیند. برای این کار، هر یک از مقادیر CMY ، از یک کم می‌شوند.

در تئوری، مقادیر یکسانِ رنگ‌دانه‌های فیروزه‌ای، بنفش، و زرد باید سیاه را تولید نماید. در عمل، ترکیب این رنگ‌ها برای چاپ، یک سیاه غلیظ را تولید می‌نماید. لذا، برای تولید سیاه کامل (که رنگ غالب در چاپ است)، رنگ چهارم، *سیاه*، اضافه می‌شود و مدل رنگ CMYK به دست می‌آید. بنابراین، وقتی ناشرین راجع به "چاپ چهاررنگ" صحبت می‌کنند، منظورشان مدل سه رنگی CMY به اضافه رنگ سیاه است.

تابع imcomplement که در بخش ۱-۲-۳ بحث شد، می‌تواند برای تبدیل تقریبی از RGB به CMY به کار می‌رود:

```
cmy_image = imcomplement(rgb_image)
```

از این تابع برای تبدیل تصاویر CMY به RGB نیز استفاده می‌کنیم:

```
rgb_image = imcomplement(cmy_image)
```

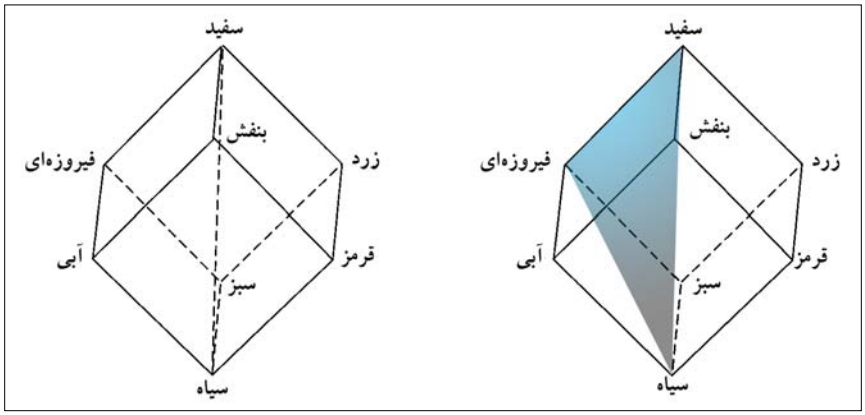
تبدیل با کیفیت بالا به CMY یا CMYK ، نیازمند دانش خاصی از جوهرها و رسانه چاپ، و روش‌های اکتشافی برای زمان استفاده از جوهر سیاه (K) به جای سه جوهر دیگر است. این تبدیل می‌تواند با استفاده از *پروفایل رنگ ICC* انجام شود که برای چاپگر خاصی ایجاد شده است (بخش ۶-۲-۷ را ببینید).

۷-۲-۵ فضای رنگ HSI

به استثنای HSV ، فضاهای رنگ بحث‌شده تاکنون، برای توصیف رنگ‌ها برحسب اصطلاحاتی که افراد با آن‌ها سروکار دارند، مناسب نیستند. برای مثال، برای رنگ اتومبیل، درصد هر کدام از رنگ‌دانه‌های موجود در آن را تعیین می‌کنیم.

وقتی انسان شیء رنگی را می‌بیند، می‌خواهیم که آن را بر اساس پرده رنگ، اشباع، و روشنی توصیف کند. پرده رنگ، صفتی از رنگ است که رنگ خالص را توصیف می‌کند، درحالی‌که اشباع، درجه‌ای از رنگ سفید را مشخص می‌کند که با رنگ خالص ترکیب می‌شود. روشنی، توصیفگر ذهنی است که به طور عملی نمی‌توان آن را اندازه‌گیری کرد. روشنی، شامل مفهوم بی‌رنگی از *شدت* است و یکی از عوامل مهم در توصیف احساس رنگ است. می‌دانیم که شدت (سطح خاکستری) مفیدترین توصیفگر تصاویر تک‌رنگ است. این کمیت قابل اندازه‌گیری و قابل دریافت است.

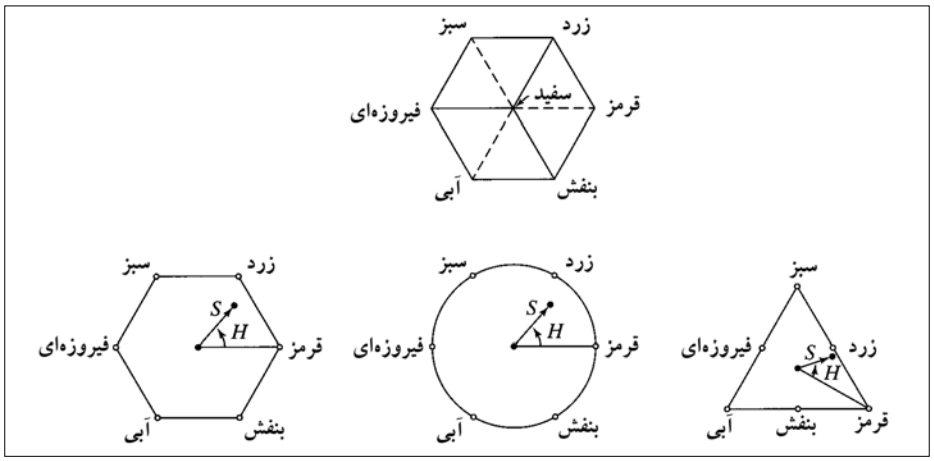
فضای رنگ مورد نظر ما، HSI (پرده رنگ، اشباع، شدت) است که مولفه شدت را از حامل اطلاعات رنگ (پرده رنگ و اشباع) در تصویر رنگی حذف می‌کند. در نتیجه، مدل HSI ابزار ایده‌آلی برای تهیه الگوریتم‌های پردازش تصویر بر اساس توصیف‌های رنگی است که برای انسان‌ها، عادی و شهودی است. فضای رنگ HSV تا حدی شبیه HSI است، اما تأکید آن بر ارائه رنگ‌هایی است که وقتی برحسب جعبه رنگ هنری تفسیر می‌شود، دارای معنا باشد.



ب الف **شکل ۶-۷** رابطه‌ی بین مدل‌های رنگ RGB و HSI .

همان‌طور که در بخش ۱-۱-۷ بحث شد، تصویر رنگی RGB مرکب از سه تصویر شدت تک‌رنگ است. بنابراین باید بتوان شدت را از تصویر RGB استخراج کرد. اگر مکعب رنگ شکل ۲-۷ را درنظربگیریم و آن را روی رأس سیاه، یعنی (0, 0, 0) قرار دهیم به طوری که رأس سفید، یعنی (1, 1, 1) مستقیماً در بالای آن قرار گیرد، مانند شکل ۶-۷ (الف)، این موضوع روشن می‌شود. همان‌طور که در رابطه با شکل ۲-۷ گفته شد، شدت، در امتداد خطی است که این دو رأس را به هم متصل می‌کند. در چیدمان شکل ۶-۷، خطی (محور شدتی) که رئوس سیاه و سفید را به هم متصل می‌کند، عمودی است، بنابراین، اگر می‌خواهیم مولفه شدت هر نقطه‌ی رنگی را در شکل ۶-۷ پیدا کنیم، صفحه‌ای را عمود بر محور شدت که حاوی آن نقطه رنگی است، عبور می‌دهیم. تقاطع این صفحه با محور شدت، یک مقدار شدت را در بازه‌ی [0, 1] ارائه می‌دهد. با کمی تفکر پی می‌بریم که اشباع رنگ، به عنوان تابعی از فاصله تا محور شدت افزایش می‌یابد. در حقیقت، اشباع نقاط در محور شدت برابر با صفر است و با این حقیقت مشهود است که تمام نقاط در امتداد این محور، سایه‌های خاکستری هستند.

برای این‌که ببینید پرده رنگ چگونه می‌تواند از یک نقطه‌ی RGB تعیین شود، شکل ۶-۷ (ب) را در نظر بگیرید، که صفحه‌ای را نشان می‌دهد که از سه نقطه (سیاه، سفید، و فیروزه‌ای) تشکیل شده است. این حقیقت که نقاط سیاه و سفید موجود در صفحه وجود دارند، به ما می‌گوید که محور شدت نیز در صفحه موجود است. علاوه‌براین، می‌بینیم که تمام نقاط موجود در قطعه‌ی صفحه، توسط محور شدت تعریف شدند و مرزهای مکعب، پرده‌ی رنگ یکسانی دارند (در این مورد، فیروزه‌ای). علتش این‌است‌که رنگ‌های موجود در مثلث رنگ، ترکیبات مختلفی از سه رنگ رأس هستند. اگر دو رأس از این سه رأس، سیاه و سفید باشند، و سومی یک نقطه‌ی رنگی باشد، تمام نقاط در مثلث باید پرده‌ی رنگ یکسانی داشته باشند، زیرا مولفه‌های سیاه و سفید، در تغییر پرده‌ی رنگ دخالت ندارند (البته، نقاط شدت و اشباع در این مثلث، تغییر می‌کنند). با دوران صفحه‌ی سایه‌دار حور محور شدت عمودی، پرده‌های رنگ مختلفی به دست می‌آوریم. از این مفاهیم نتیجه می‌گیریم که مقادیر پرده رنگ، اشباع، و شدت موردنیاز برای ایجاد فضای HSI ، می‌تواند از مکعب رنگ به دست آید. یعنی، می‌توان هر نقطه‌ی RGB را به نقطه متناظر در مدل رنگ HSI تبدیل کرد. برای این کار از فرمول‌های هندسی استفاده می‌شود که استدلال مطرح‌شده را توصیف می‌کنند.



شکل ۷-۷ پرده رنگ و اشباع در مدل رنگ HSI . نقطه، یک نقطه‌ی رنگی دلخواه است. زاویه از محور قرمز، پرده رنگ را مشخص می‌کند و طول بردار، اشباع است. شدت تمام رنگ‌ها در هر یک از این صفحات، توسط موقعیت صفحه در محور شدت عمودی تعیین می‌شود.

الف ت پ ب

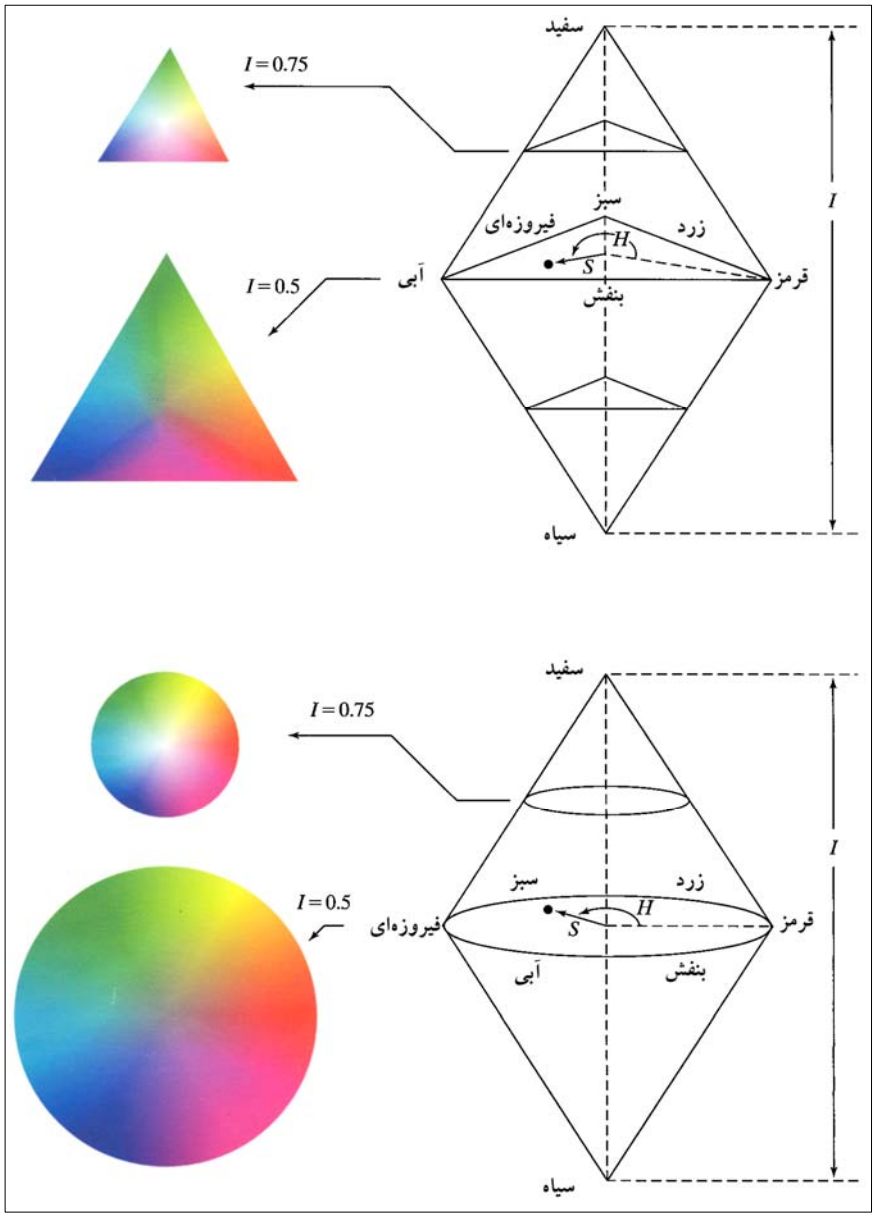
بر اساس بحث قبلی، می‌بینیم که فضای HSI شامل محور شدت عمودی و مکان هندسی نقاط رنگی است که در صفحه‌ی عمود بر این محور قرار دارد. وقتی صفحه، محور شدت را بالا و پایین می‌برد، مرزهای تعریف‌شده توسط تقاطع صفحه با وجوه مکعب، دارای شکل مثلثی یا شش‌ضلعی است. با نگاه‌کردن به مکعب زیر محور خاکستری، مثل شکل ۷-۷ (الف)، به این موضوع پی می‌برید. در این صفحه، می‌بینیم که رنگ‌های اولیه با $120°$ از هم جدا شدند. رنگ‌های ثانویه، $60°$ از رنگ‌های اولیه فاصله دارند، که به معنای این‌است‌که زاویه بین رنگ‌های ثانویه نیز $120°$ است.

شکل ۷-۷ (ب) یک شکل شش‌ضلعی و یک نقطه رنگی دلخواه را نشان می‌دهد. پرده رنگ این نقطه، با زاویه‌ای از نقطه مرجع تعیین می‌شود. معمولاً (اما نه همیشه)، زاویه صفر درجه از محور قرمز، به معنای پرده رنگ صفر است، و پرده رنگ از این‌جا در جهت عکس عقربه‌های ساعت افزایش می‌یابد. اشباع (فاصله از محورِ عمودی) برابر با طول بردار از مبدأ تا این نقطه است. توجه کنید که مبدأ توسط تقاطع صفحه‌ی رنگ با محور شدت عمودی تعریف می‌شود. مولفه‌های مهم فضای رنگ HSI عبارتنداز محور شدت عمودی، طول بردار تا نقطه رنگی، و زاویه‌ای که این بردار با محور قرمز می‌سازد. بنابراین، می‌توان دید که صفحات HSI تعریف‌شده برحسب شش‌ضلعیِ بحث‌شده، یک مثلث یا حتی یک دایره است، که شکل‌های ۷-۷ (پ) و (ت) آن را نشان می‌دهند. شکل انتخابی مهم نیست، زیرا به وسیله یک تبدیل هندسی می‌توان هر کدام از این شکل‌ها را به دیگری تبدیل کرد. شکل ۸-۷ یک مدل HSI را بر اساس مثلث‌ها و دایره‌های رنگ نشان می‌دهد.

تبدیل رنگ‌ها از RGB به HSI

در بحث زیر، معادلات تبدیل لازم را بدون مشتق ارائه می‌دهیم. با توجه به تصویر رنگی RGB ، مولفه‌ی H هر پیکسل RGB با استفاده از معادله زیر به دست می‌آید:

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



شکل ۸-۷ مدل رنگ HSI بر اساس (الف) صفحات رنگ مثلثی و (ب) دایره‌ای. مثلث‌ها و دایره‌ها بر محور شدت عمودی، عمود است.

به طوری که:

$$\theta = \cos^{-1}\left\{\frac{0.5[(R-G)+(R-B)]}{\left[(R-G)^2+(R-B)(G-B)\right]^{1/2}}\right\}$$

مولفه اشباع با فرمول زیر به دست می‌آید:

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

سرانجام، مولفه شدت با معادله زیر به دست می‌آید:

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

فرض می‌شود که مقادیر RGB به بازه‌ی [0, 1] نرمال شدند، و θ نسبت به محور قرمز فضای HSI اندازه‌گیری می‌شود که در شکل ۷–۷ آمده است. پرده رنگ نیز می‌تواند به بازه‌ی [0, 1] نرمال شود. برای این کار، تمام مقادیر حاصل از معادله‌ی مربوط به H ، بر 360° تقسیم می‌گردد. دو مولفه دیگر H در صورتی در این بازه قرار دارند که مقادیر RGB در فاصله [0, 1] باشند.

تبدیل رنگ‌ها از HSI به RGB

با توجه به مقادیر HSI که در فاصله [0, 1] قرار دارند، می‌خواهیم مقادیر RGB متناظر را در این بازه بیابیم. معادلات قابل اعمال، به مقادیر H بستگی دارند. سه قطاع[1] موردنظر وجود دارند، که متناظر با فاصله‌های بین رنگ‌های اولیه است که قبلاً شرح داده شد. با ضرب H در 360° شروع می‌کنیم، که پرده رنگ را به بازه‌ی اصلی آن، یعنی [0°,360°] برمی‌گرداند.

سکتور RG ($0° \le H < 120°$): وقتی H در این قطاع قرار دارد، مولفه‌های RGB با معادلات زیر مشخص می‌شوند:

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

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

و

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

قطاع GB ($120° \le H < 240°$): اگر مقدار H در این قطاع باشد، ابتدا 120° را از آن کم می‌کنیم:

$$H = H - 120°$$

سپس مولفه‌های RGB عبارتنداز:

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

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

و

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

1. sector

قطاع BR ($240° \le H \le 360°$): سرانجام، اگر H در این بازه باشد، $240°$ را از آن کم می‌کنیم:

$$H = H - 240°$$

آنگاه مولفه‌های RGB عبارتنداز:

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

که

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

و

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

در بخش ۱-۵-۷ نشان خواهیم داد که از این معادلات چگونه در پردازش تصویر استفاده می‌شود.

یک تابع – M برای تبدیل RGB به HSI

تابع زیر را درنظربگیرید:

```
hsi = rgb2hsi(rgb)
```

این تابع، معادلات بحث‌شده برای تبدیل RGB به HSI را پیاده‌سازی می‌کند، که rgb و hsi به ترتیب تصاویر RGB و HSI را نشان می‌دهند. مستندات کد، این تابع را شرح می‌دهد:

rgb2hsi

```
function hsi = rgb2hsi(rgb)
%RGB2HSI Converts an RGB image to HSI.
%   HSI = RGB2HSI(RGB) converts an RGB image to HSI. The input image
%   is assumed to be of size M-by-N-by-3, where the third dimension
%   accounts for three image planes: red, green, and blue, in that
%   order. If all RGB component images are equal, the HSI conversion
%   is undefined. The input image can be of class double (with
%   values in the range [0, 1]), uint8, or uint16.
%
%   The output image, HSI, is of class double, where:
%      HSI(:, :, 1) = hue image normalized to the range [0,1] by
%                     dividing all angle values by 2*pi.
%      HSI(:, :, 2) = saturation image, in the range [0, 1].
%      HSI(:, :, 3) = intensity image, in the range [0, 1].

% Extract the individual component images.
rgb = im2double(rgb);
r = rgb(:, :, 1);
g = rgb(:, :, 2);
b = rgb(:, :, 3);

% Implement the conversion equations.
num = 0.5*((r - g) + (r - b));
den = sqrt((r - g).^2 + (r - b).*(g - b));
theta = acos(num./(den + eps));

H = theta;
H(b > g) = 2*pi - H(b > g);
```

```
H = H/(2*pi);

num = min(min(r, g), b);
den = r + g + b;
den(den == 0) = eps;
S = 1 - 3.* num./den;
H(S == 0) = 0;
I = (r + g + b)/3;

% Combine all three results into an hsi image.
hsi = cat(3, H, S, I);
```

تابع M – برای تبدیل HSI به RGB

تابع زیر را درنظربگیرید:

```
rgb = hsi2rgb(hsi)
```

این تابع، معادلات مربوط به تبدیل HSI به RGB را پیاده‌سازی می‌کند. مستندات کد، جزییات تابع را شرح می‌دهد.

hsi2rgb

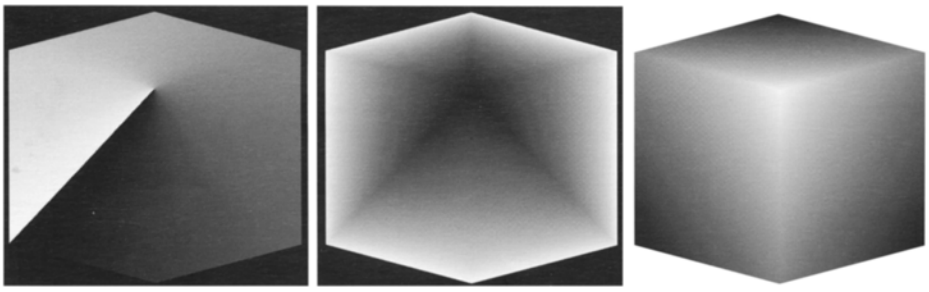
```
function rgb = hsi2rgb(hsi)
%HSI2RGB Converts an HSI image to RGB.
%   RGB = HSI2RGB(HSI) converts an HSI image RGB, where HSI is
%   assumed to be of class double with:
%      HSI(:, :, 1) = hue image, assumed to be in the range
%                     [0, 1] by having been divided by 2*pi.
%      HSI(:, :, 2) = saturation image, in the range [0, 1];
%      HSI(:, :, 3) = intensity image, in the range [0, 1].
%
%   The components of the output image are:
%      RGB(:, :, 1) = red.
%      RGB(:, :, 2) = green.
%      RGB(:, :, 3) = blue.

% Extract the individual HSI component images.
H = hsi(:, :, 1) * 2 * pi;
S = hsi(:, :, 2);
I = hsi(:, :, 3);

% Implement the conversion equations.
R = zeros(size(hsi, 1), size(hsi, 2));
G = zeros(size(hsi, 1), size(hsi, 2));
B = zeros(size(hsi, 1), size(hsi, 2));

% RG sector (0 <= H < 2*pi/3).
idx = find( (0 <= H) & (H < 2*pi/3));
B(idx) = I(idx) .* (1 - S(idx));
R(idx) = I(idx) .* (1 + S(idx) .* cos(H(idx))./ ...
                                        cos(pi/3 - H(idx)));
G(idx) = 3*I(idx) - (R(idx) + B(idx));

% BG sector (2*pi/3 <= H < 4*pi/3).
idx = find( (2*pi/3 <= H) & (H < 4*pi/3) );
```

```
R(idx) = I(idx) .* (1 - S(idx));
G(idx) = I(idx) .* (1 + S(idx) .* cos(H(idx) - 2*pi/3) ./ ...
                    cos (pi - H(idx)));
B(idx) = 3*I(idx) - (R(idx) + G(idx));

% BR sector.
idx = find( (4*pi/3 <= H) & (H <= 2*pi));
G(idx) = I(idx).* (1 - S(idx));
B(idx) = I(idx).* (1 + S(idx).* cos(H(idx) - 4*pi/3)./ ...
                                        cos(5*pi/3 - H(idx)));
R(idx) = 3*I(idx) - (G(idx) + B(idx));

% Combine all three results into an RGB image. Clip to [0, 1] to
% compensate for floating-point arithmetic rounding effects.
rgb = cat(3, R, G, B);
rgb = max(min(rgb, 1), 0);
```

مثال ۲-۷: تبدیل RGB به HSI.

شکل ۹-۷ مولفه‌های پرده رنگ، اشباع، و شدت تصویر یک مکعب RGB را در پس‌زمینه سفید، مشابه شکل ۲-۷ (ب) نشان می‌دهد. شکل ۹-۷ (الف) یک تصویر پرده رنگ است. مهمترین ویژگی متمایز آن، عدم پیوستگی در مقدار موجود در امتداد خط °45 در صفحه‌ی جلویی (قرمز) مکعب است. برای پی‌بردن به علت این عدم پیوستگی، به شکل ۲-۷ (ب) مراجعه کنید، خطی از رئوس قرمز به سفید مکعب رسم کنید و نقطه‌ای را در وسط این خط انتخاب نمایید. با شروع از آن نقطه، مسیری به راست رسم کنید. دور مکعب دور بزنید تا به نقطه اول برسید. رنگ‌های اصلی که در این مسیر دیده می‌شوند عبارتنداز: زرد، سبز، فیروزه‌ای، آبی، بنفش، و سیاه به قرمز. بر اساس شکل ۷-۷، مقدار پرده رنگ در امتداد این مسیر باید از °0 به °360 افزایش یابد (یعنی از پایین‌ترین به بالاترین مقدار پرده رنگ). این همان چیزی است که شکل ۹-۷ (الف) نشان می‌دهد، زیرا پایین‌ترین مقدار به صورت سیاه و بالاترین مقدار به صورت سفید است.

تصویر اشباع در شکل ۹-۷ (ب)، مقادیر تیره‌ی فزاینده به سمت رأس سفید مکعب RGB را نمایش می‌دهد، که مشخص می‌شود رنگ‌ها با نزدیک‌شدن به سفید، کمتر و کمتر اشباع می‌شوند. سرانجام، هر پیکسل در تصویر شکل ۹-۷ (پ)، میانگین مقادیر RGB در مکان پیکسل متناظر در شکل ۲-۷ (ب) است. توجه کنید که پس‌زمینه در این تصویر، سفید است، زیرا شدت پس‌زمینه در تصویر رنگی، سفید است. در دو تصویر دیگر، سیاه است، زیرا پرده رنگ و اشباع سفید، صفر هستند. ■



شکل ۹-۷ تصاویر مولفه HSI مربوط به تصویری از مکعب رنگ RGB. (الف) تصویر پرده رنگ، (ب) تصویر اشباع و (پ) تصویر شدت.

پ ب الف

۶-۲-۷ فضاهای رنگ مستقل از دستگاه

بحث‌های بخش ۱-۲-۷ تا ۵-۲-۷ بر روی فضاهای رنگ بود که اطلاعات رنگ را طوری نشان می‌دهند که محاسبات را آسان‌تر می‌سازند، یا رنگ‌ها را طوری نشان می‌دهند که بر کاربرد خاصی، شهودی‌تر و مناسب‌تر هستند. تمام فضاهایی که تاکنون بحث شدند، *وابسته به دستگاه*[1] هستند. برای مثال، ظاهر رنگ‌های RGB با ویژگی‌های اسکنر و مانیتور فرق می‌کند، و رنگ‌های CMYK با ویژگی‌های چاپگر، جوهر، و کاغذ فرق می‌کند.

در این بخش، بر روی فضاهای رنگ مستقل از دستگاه تأکید می‌کنیم. دستیابی به بازتولید سازگار و با کیفیت در سیستم تصویربرداری، نیازمند درک و مشخص‌کردن هر دستگاه رنگی در سیستم است. در محیط کنترل‌شده، می‌توان مولفه‌های مختلف سیستم را "تنظیم کرد" تا نتایج ارضاکننده‌ای به دست آید. برای مثال، در عملیات چاپ عکس one_shop ، می‌توان زیرسیستم‌های توسعه و چاپ رنگی را به طور دستی بهینه کرد تا نتایج بازتولید سازگاری به دست آید. از طرف دیگر، این روش، در سیستم‌های تصویربرداری دیجیتال *باز* که شامل چندین دستگاه است، یا آن‌هایی که هیچ کنترلی روی مکان یا دیدن یا پردازش تصویر وجود ندارد (مثل اینترنت) امکان‌پذیر نیست.

مرور کلی

ویژگی‌هایی که عموماً برای تمایز یک رنگ از رنگ دیگر استفاده می‌شوند، عبارتنداز *روشنایی*، *پرده رنگ*، و *اشباع*. همان‌طور که در این بخش دیدید، روشنایی شامل مفهوم بی‌رنگ[2] شدت است. پرده رنگ، صنعت مربوط به طول موج غالب در ترکیب موج‌های نور است. پرده رنگ، رنگ غالب را به عنوان رنگ دریافت‌شده توسط بیننده نمایش می‌دهد. بنابراین، وقتی شیء را قرمز، نارنجی، یا زرد می‌بینیم، منظور ما پرده رنگ است. اشباع، به معنای خالص‌بودن نسبی یا میزان نور سفید ترکیب‌شده با پرده رنگ است. رنگ‌های طیف خالص، کاملاً اشباع شده‌اند. رنگ‌هایی مثل صورتی (قرمز و سفید) و بنفش روشن (بنفش و سفید) کمتر اشباع شدند، به طوری که درجه اشباع، به طور معکوس متناسب با میزان نور سفید اضافه شده است.

پرده رنگ و اشباع، روی هم، رنگینگی[3] نام دارند، و در نتیجه، رنگ ممکن‌است توسط روشنایی و رنگینگی خود مشخص شود. میزان قرمز، سبز و آبی مورد نیاز برای ایجاد هر رنگ، مقادیر سه محرکی[4] نامیده می‌شوند و به ترتیب با X ، Y و Z نمایش داده می‌شوند. سپس رنگ با ضرایب سه رنگی[5] خود مشخص می‌شود، که به صورت زیر تعریف می‌گردد:

$$x = \frac{X}{X+Y+Z}$$

$$y = \frac{Y}{X+Y+Z}$$

و

$$z = \frac{Z}{X+Y+Z} = 1 - x - y$$

1. device dependent 2. achromatic 3. chromaticity 4. tristimulus 5. trichromatic

سپس نتیجه می‌شود که:

$$x + y + z = 1$$

که x ، y و z به ترتیب مولفه‌های قرمز، سبز و آبی را نشان می‌دهند. برای هر طول موج نور در طیف قابل رویت، مقادیر سه محرکِ مورد نیاز برای رنگ متناظر با آن طول موج، مستقیماً می‌تواند از منحنی‌ها یا جدول‌هایی به دست آید که از نتایج آزمایشی گسترده‌ای به دست آمد.

یکی از پراستفاده‌ترین فضای رنگ سه محرکی، فضای رنگ CIE XYZ 1931 است که توسط CIE ایجاد شد. در فضای رنگ CIE XYZ ، Y طوری انتخاب شد که میزان روشنایی باشد. فضای رنگ تعریف‌شده توسط Y و مقادیر رنگینگی x و y ، فضای رنگ CIE xyY نام دارد. مقادیر سه‌محرکی X و Z می‌تواند با استفاده از معادلات زیر، از x ، y و Y محاسبه شوند:

$$X = \frac{Y}{y}x$$

و

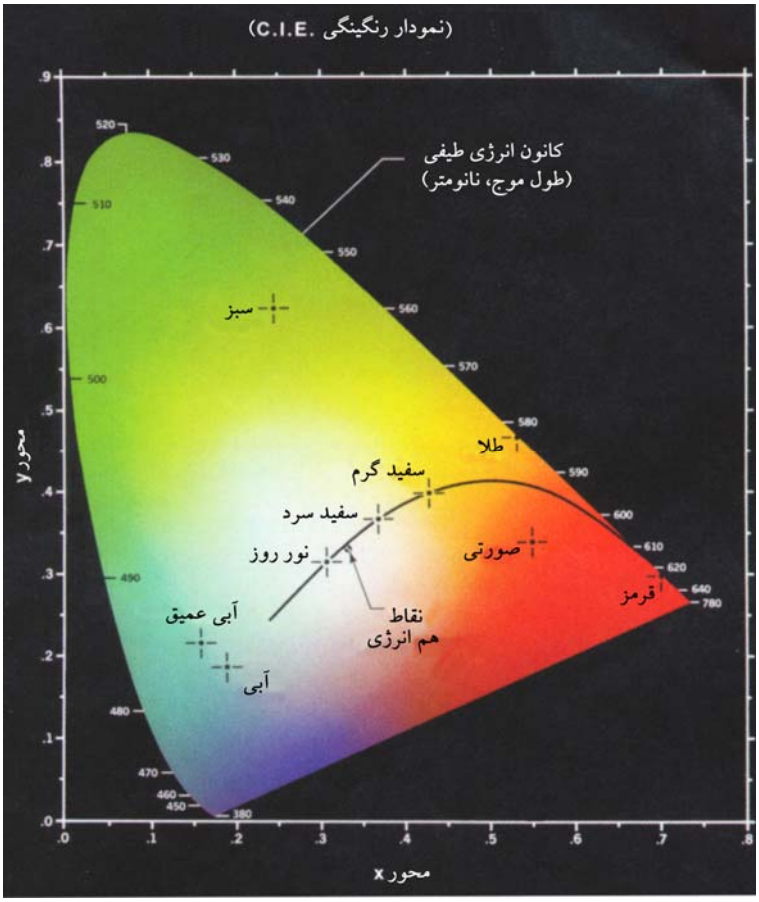
$$Z = \frac{Y}{y}(1 - x - y)$$

نموداری (شکل ۱۰–۷) که بازه‌ی رنگ دریافت‌شده توسط انسان را به صورت تابعی از x و y نمایش می‌دهد، *نمودار رنگینگی* نام دارد. برای هر مقدار x و y در این نمودار، مقدار متناظر z ، برابر با $z = 1 - (x + y)$ است. برای مثال، نقطه‌ای که با سبز در شکل ۱۰–۷ مشخص شده است، تقریباً 62% سبز و 25% قرمز دارد. لذا مولفه‌ی آبی نور برای آن رنگ، 13% است.

موقعیت‌های رنگ‌های تک‌رنگی مختلف (طیف خالص)، از بنفش در 380 nm تا قرمز در 780 nm – در حول مرز مربوط به بخش زبانی‌شکلِ نمودار رنگینگی، نشان داده شدند. بخش مستقیم مرز، *خط بنفش (ارغوانی)* نام دارد. این رنگ‌ها معادل تک‌رنگی ندارند. هر نقطه‌ای که واقعاً در مرز قرار ندارد ولی در نمودار هست، ترکیبی از رنگ‌های طیف را نشان می‌دهد. نقاط هم‌انرژی در شکل ۱۰–۷، متناظر با کسرهای مساوی از سه رنگ اصلی هستند؛ که استاندارد CIE را برای نور سفید نشان می‌دهد. هر نقطه‌ی موجود در مرز نمودار رنگینگی، کاملاً اشباع شده است. وقتی نقطه‌ای مرز را ترک می‌کند و به نقطه‌ای با انرژی یکسان می‌رود، نور سفید بیشتری به رنگ اضافه می‌شود و کمتر اشباع می‌گردد. اشباع رنگ در نقطه‌ی هم‌انرژی، صفر است.

قطعه‌خط مستقیمی که دو نقطه را در نمودار به هم متصل می‌کند، تمام تغییرات رنگ مختلفی را تعریف می‌کند که می‌تواند با ترکیب افزایشی آن دو رنگ به دست آید. برای مثال، خط مستقیمی را درنظربگیرید که نقاط قرمز و سبز را در شکل ۱۰–۷ به هم متصل می‌کند. اگر در رنگی، قرمز بیش از سبز باشد، نقطه‌ی نشان‌دهنده‌ی آن رنگ، روی خط مستقیم خواهد بود که به نقطه‌ی قرمز نزدیک‌تر است تا سبز. به طور مشابه، خط رسم‌شده از نقطه‌ی هم‌انرژی به هر نقطه‌ای در مرز نمودار، تمام سایه‌هایی با آن رنگ طیف خاص را تعریف می‌کند.

بسط این رویه به سه رنگ، ساده است. برای تعیین بازه‌ی رنگ‌هایی که می‌توانند از هر سه رنگ موجود در نمودار رنگینگی به دست آیند، خطوطی را به هر سه نقطه‌ی رنگ متصل می‌کنیم. نتیجه، یک مثلث است، و هر رنگ روی مرز یا داخل مثلث می‌تواند با ترکیبات گوناگونی از سه رنگ اولیه به دست آید. مثلثی با رئوسی درهر سه رنگ ثابت، نمی‌تواند کل ناحیه رنگ را در شکل ۱۰–۷ دربرگیرد. این مشاهدات، روشن می‌سازد که، این نکته که هر رنگی می‌تواند از سه رنگ اولیه‌ی ثابت به دست آید، اشتباه است.



شکل ۱۰-۷ نمودار رنگینگی CIE .

فضای رنگ مستقل از دستگاه خانواده CIE

در دهه‌های پس از معرفی فضای رنگ XYZ ، CIE چندین مشخصات فضای رنگ دیگر را ایجاد کرد، که سعی کردند مشخصات رنگ دیگری را فراهم نمایند که نسبت به XYZ مناسب‌تر باشند. برای مثال، CIE در سال ۱۹۷۶، L * a * b * را معرفی کرد، که در علم رنگ، هنرهای خلاق، و طراحی دستگاه‌های رنگی مثل چاپگرها، دوربین‌ها، و اسکنرها بسیار مورد استفاده قرار گرفت.* L * a * b نسبت به XYZ دو امتیاز دارد. اولاً * L * a * b اطلاعات مقیاس خاکستری را (که به صورت مقادیر *L نمایش داده می‌شوند) از اطلاعات رنگی (که با مقادیر *a و *b نمایش داده می‌شوند)، تفکیک می‌کند، ثانیاً، رنگ L * a * b طوری طراحی شد که فاصله اقلیدسی در این فضای رنگ، متناظر با تفاوت‌های دریافتی[1] بین رنگ‌ها است. به دلیل این خاصیت، می‌گوییم فضای

1. percieved

رنگ L * a * b * از نظر ادراکی یکنواخت است. از نظر رنگ، فضای رنگ L * a * b * به طور خطی با درک انسان از روشنایی، ارتباط دارد. یعنی، اگر در رنگی، مقدار L^* دو برابر L^* رنگ دیگر باشد، رنگ اول، دو برابر روشن‌تر از رنگ دوم دیده می‌شود. توجه کنید که، به دلیل پیچیدگی سیستم بصری انسان، خاصیت یکنواختی ادراکی، فقط به طور تقریبی اتفاق می‌افتد.

جدول ۴-۷ فضاهای رنگ مستقل از دستگاه CIE را که توسط جعبه‌ابزار پردازش تصویر پشتیبانی می‌شود، نشان می‌دهد.

فضای رنگ sRGB

همان‌طور که در این بخش اشاره شد، مدل رنگ RGB ، مستقل از دستگاه است، و معنایش این‌است‌که یک تفسیر رنگ نامبهم برای مقادیر معین R ، G ، و B وجود ندارد. علاوه‌براین، فایل‌های تصویر غالباً فاقد اطلاعاتی راجع‌به ویژگی‌های دستگاه تصویربرداری هستند. در نتیجه، یک فایل تصویر، در سیستم‌های کامپیوتری مختلف، می‌تواند متفاوت به نظر برسد. وقتی استفاده از کامپیوتر در دهه ۱۹۹۰ افزایش یافت، طراحان وب نمی‌دانستند وقتی تصاویر در مرورگرهای کاربران ظاهر می‌شوند، چگونه خواهند بود.

برای حل این مسئله‌ها، میکروسافت و هیولت پاکارد، یک فضای رنگ پیش‌فرض استاندارد به نام sRGB را پیشنهاد کردند. فضای رنگ sRGB طراحی شد تا با خواص مانیتورهای CRT کامپیوتر سازگار باشد. علاوه‌براین، با کامپیوترهای موجود در محیط خانه و دفتر نیز سازگار باشد. فضای رنگ sRGB مستقل از دستگاه است، به طوری که مقادیر رنگ sRGB می‌تواند به سایر فضاهای رنگ مستقل از دستگاه تبدیل شود.

استاندارد sRGB ، در صنعت کامپیوتر به طور گسترده پذیرفته شده است، مخصوصاً برای دستگاه‌های مبتنی بر مصرف‌کننده.

دوربین‌های دیجیتال، اسکنرها، نمایشگرهای کامپیوتر، و چاپگر، طوری طراحی می‌شوند که فرض می‌کنند مقادیر RGB با فضای رنگ sRGB سازگارند، مگر این‌که فایل تصویر شامل اطلاعات بیشتری راجع به دستگاه باشد.

جدول ۴-۷ فضاهای رنگ CIE مستقل از متن که توسط جعبه‌ابزار پردازش تصویر پشتیبانی می‌شوند.

فضای رنگ	شرح
XYZ	مشخصات فضای رنگ CIE 1931 اصلی.
xyY	مشخصات CIE که مقادیر رنگینگی نرمال‌شده را فراهم می‌سازد. مقدار Y لومینانس را نشان می‌دهد و همانند XYZ است.
uvL	مشخصات CIE که سعی می‌کند صفحه رنگینگی را از نظر بصری بیشتر یکنواخت کند. L لومینانس است و مثل Y در XYZ است.
u′v′L	مشخصات CIE که در آن u و v دوباره مقیاس‌بندی می‌شوند تا یکنواختی بهبود یابد.
L* a*b*	مشخصات CIE که سعی می‌کند مقیاس لومینانس را از نظر ادراکی بیشتر یکنواخت سازد. L^* مقیاس‌بندی دیگری از L است، که به یک نقطه سفید مرجع نرمال‌شده است.
L*ch	مشخصات CIE که c یک رنگ و h پرده رنگ است. این مقادیر تبدیل مختصات قطبیِ a^* و b^* در L*a*b* است.

تبدیلات فضای رنگ CIE و sRGB

توابع جعبه‌ابزار makecform و applycform می‌توانند برای تبدیل بین چندین فضای رنگ مستقل از دستگاه استفاده شوند. جدول ۵–۷ تبدیلاتی را که پشتیبانی می‌شوند، مشخص می‌کند. تابع makecform یک ساختار cform را ایجاد می‌کند که شبیه تابع maketform در ایجاد ساختار tform است (فصل ۶ را ببینید). تابع makecform به صورت زیر به کار می‌رود:

جدول ۵–۷ تبدیلات فضای رنگ مستقل از دستگاه که توسط جعبه‌ابزار پردازش تصویر پشتیبانی می‌شود.

انواع انتقال داده شده در makecform	فضاهای رنگ
‘lab2lch’, ‘lch2lab’	L * a*b* and L*ch
‘lab2srgb’, ‘srgb2lab’	L*a*b* and sRGB
‘lab2xyz’, ‘xyz2lab’	L*a*b* and XYZ
‘srgb2xyz’, ‘xyz2srgb’	sRGB and XYZ
‘upvpl2xyz’, ‘xyz2upvpl’	u′v′L and XYZ
‘uvl2xyz’, ‘xyz2uvl’	uvL and XYZ
‘xyl2xyz’, ‘xyz2xyl’	xyY and XYZ

```
cform = makecform(type)
```

که type یکی از رشته‌های نشان‌داده‌شده در جدول ۵–۷ است. تابع applycform از ساختار cform برای تبدیل رنگ‌ها استفاده می‌کند. این تابع به صورت زیر به کار می‌رود:

```
g = applycform(f, cform)
```

مثال ۳–۷: ایجاد یک مقیاس رنگ یکنواختِ ادراکی بر اساس فضای رنگ *L*a*b.

در این مثال، یک مقیاس رنگ می‌سازیم که در چاپ رنگی و مقیاس خاکستری استفاده می‌شوند. McNames [2006] چندین اصل را برای طراحی این نوع مقیاس رنگ ارائه کرده است:

۱. تفاوت دریافتی (ادراکی) بین دو مقیاس رنگ، باید متناسب با فاصله بین آن‌ها در امتداد آن مقیاس باشد.
۲. لومینانس باید به صورت تک‌رنگ افزایش یابد، به‌طوری‌که این مقیاس برای کاربردهایی با مقیاس خاکستری کار کند.
۳. رنگ‌های همجوار در سراسر مقیاس، حتی‌الامکان باید مجزا باشند.
۴. مقیاس باید بازه‌ی وسیعی از رنگ‌ها را دربرگیرد.
۵. مقیاس رنگ باید شهودی باشد.

مقیاس رنگ را طوری طراحی می‌کنیم که چهار اصل اول را برآورده کند. برای این کار، مسیری در فضای L*a*b* ایجاد می‌کنیم. اصل اول، یکنواختی مقیاس به طور ادراکی، می‌تواند با استفاده از ایجاد فاصله‌ی مساوی از رنگ‌ها در L*a*b* ، برآورده شود. اصل دوم، افزایش لومینانس تک‌رنگ، می‌تواند با ایجاد یک شیب خطیِ از مقادیر L^* (که در این‌جا بین 0 (سیاه) و 100 (روشنایی کامل) تغییر می‌کند) برآورده شود. در این‌جا شیبی با 1024 مقدار فضای مساوی بین 40 و 80 را می‌سازیم:

```
>> L = linspace(40, 80, 1024);
```

اصل سوم، رنگ‌های همجوار مجزا، می‌تواند با تغییر رنگ‌ها در پرده رنگ برآورده شود، متناظر با زاویه قطبیِ مشخصات رنگ در صفحه a^*b^* است:

```
>> radius = 70;
>> theta = linspace(0, pi, 1024);
>> a = radius * cos(theta);
>> b = radius * sin(theta);
```

شکل ۱۱-۷ یک مقیاس رنگ یکنواخت ادراکی، بر اساس فضای رنگ L*a*b* .

اصل چهارم، مربوط به استفاده از بازه‌ی وسیع از رنگ‌ها است. مجموعه بازه‌های مقادیر a^* و b^* موردنظر ما، حتی‌الامکان با هم فاصله دارند (برحسب زاویه قطبی)، بدون این‌که آخرین رنگ در مقیاس، به اولین رنگ نزدیک‌تر باشد.

سپس یک تصویر 3×1024×100 از مقیاس رنگ L*a*b* را می‌سازیم:

```
>> L = repmat(L, 100, 1);
>> a = repmat(a, 100, 1);
>> b = repmat(b, 100, 1);
>> lab_scale = cat(3, L, a, b);
```

برای نمایش تصویر مقیاس رنگی در MATLAB ، ابتدا باید به RGB تبدیل کنیم. با ایجاد ساختار cform مناسب با استفاده از تابع makecform شروع می‌کنیم و سپس از تابع applycform استفاده می‌کنیم (شکل ۱۱-۷):

```
>> cform = makecform('lab2srgb');
>> rgb_scale = applycform(lab_scale, cform);
>> imshow(rgb_scale)
```

دستیابی به اصل پنجم، شهودی‌بودن، دشوار است و به نوع کاربرد بستگی دارد. مقیاس‌های رنگ مختلف می‌توانند با استفاده از رویه مشابهی ساخته شود، ولی از مقادیر شروع و پایان متفاوتی در L^* استفاده گردد. مقیاس‌های رنگ جدیدِ حاصل، برای بعضی از کاربردها باید شهودی‌تر باشد. ■

پروفایل‌های رنگ ICC

رنگ‌های سند می‌تواند در مانیتور و چاپ با هم متفاوت باشد. یا رنگ سند، وقتی در چاپگرهای مختلف چاپ می‌شود، ممکن‌است متفاوت باشد. برای بازتولید رنگ با کیفیت بالا بین دستگاه‌های ورودی، خروجی، و نمایش، لازم است تبدیلی برای نگاشت رنگ‌ها از یک دستگاه به دستگاه دیگر ایجاد شود. به طور کلی، برای هر جفت از دستگاه‌ها، به تبدیل رنگ جداگانه‌ای نیاز است. تبدیلات دیگری برای شرایط چاپ مختلف، تنظیمات کیفیت دستگاه، و غیره لازم است. هر یک از این تبدیلات باید با استفاده از شرایط تجربی منظم و کنترل‌شده تهیه شوند. بدیهی است که این روش، در مواردی غیرممکن و در مواردی دشوار است.

کنسرسیوم رنگ بین‌المللی (ICC)، که یک گروه صنعتی است که در سال ۱۹۹۳ ایجاد شد، روش دیگری را استانداردسازی می‌کند. هر دستگاه فقط دو تبدیل مربوط به خود دارد، و این ربطی به تعداد دستگاه‌های موجود در سیستم ندارد. یکی از تبدیلات، رنگ‌های دستگاه را به یک فضای رنگ مستقل از دستگاه استاندارد، به نام *فضای اتصال پروفایل*[1] (PCS) تبدیل می‌کند. این تبدیل رنگ، معکوس اولی است؛ یعنی رنگ‌های PCS را به رنگ‌های دستگاه تبدیل می‌کند. روی هم رفته، این دو تبدیل، *پروفایل رنگ ICC* را برای دستگاه می‌سازند.

1. Profile Connection Space

هدف اولیه ICC ، ساخت، استانداردسازی، نگهداری، و ارتقای استاندارد پروفایل رنگ ICC بود. تابع جعبه‌ابزار پردازش تصویر iccread ، فایل پروفایل را می‌خواند که به صورت زیر به کار می‌رود:

```
p = iccread(filename)
```

خروجی p ، ساختاری است که شامل اطلاعات سرآیند فایل و ضرایب عددی و جدول‌های مورد نیاز برای محاسبه تبدیلات فضای رنگ بین رنگ‌های PCS و دستگاه است.

تبدیل رنگ‌ها با استفاده از پروفایل‌های ICC ، با استفاده از makecform و applycform انجام می‌شود. روش کاربرد پروفایل ICC برای makecform به صورت زیر است:

```
cform = makecform('icc', src_profile, dest_profile)
```

که src_profile نام فایل پروفایل دستگاه منبع است، و dest_profile نام فایل پروفایل دستگاه مقصد است.

استاندارد پروفایل رنگ ICC شامل راهکارهایی برای اداره‌کردن تبدیل رنگ حیاتی، به نام *نگاشت گام*[1] است. *گام رنگ*[2]، حجمی در فضای رنگ است که بازه‌ای از رنگ‌ها را تعریف می‌کند که دستگاهی می‌تواند تولید نماید. گام رنگ، از دستگاهی به دستگاه دیگر فرق می‌کند. برای مثال، مانیتور معمولی می‌تواند رنگ‌هایی را نشان دهد که نمی‌تواند با استفاده از چاپگر بازتولید شود. بنابراین، هنگام نگاشت رنگ‌ها از دستگاهی به دستگاه دیگر، لازم است گام‌های مختلفی در نظر گرفته شوند. فرآیند جبران تفاوت‌های بین گام‌های منبع و مقصد، *نگاشت گام* نام دارد.

روش‌های مختلفی برای نگاشت گام وجود دارد. بعضی از روش‌ها، برای اهداف ما نسبت به روش‌های دیگر مفیدتر هستند. استاندارد پروفایل رنگ ICC ، چهار "هدف" (به نام اهداف اجرا) را برای نگاشت گام تعریف می‌کند. این اهداف اجرا در جدول ۶-۷ توصیف شدند. کاربرد makecform برای مشخص‌کردن اهداف اجرا، به صورت زیر است:

```
cform = makecform('icc', src_profile, dest_profile, ...
        'SourceRenderingIntent', src_intent, ...
          'DestRenderingIntent', dest_intent)
```

که src_intent و dest_intent از رشته‌های 'Preceptual' (پیش‌فرض)، 'AbsoluteColorimetric' ، 'RelativeColorimetric' و 'Saturation' انتخاب می‌شود.

مثال ٤-٧ : نمونه‌خوانی نرم با استفاده از پروفایل‌های رنگ ICC .

در این مثال، از پروفایل‌های رنگ ICC ، makecform ، و applycform برای پیاده‌سازی فرآیندی به نام *نمونه‌خوانی نرم*[3] استفاده می‌کنیم. نمونه‌خوانی نرم، در یک مانیتور کامپیوتر، نمایی را که تصویر می‌تواند پس از چاپ در کاغذ داشته باشد، شبیه‌سازی می‌کند. از نظر ادراکی، نمونه‌خوانی نرم، یک فرآیند دو مرحله‌ای است:

۱. رنگ‌های مانیتور (معمولاً sRGB) را به رنگ‌های دستگاه خروجی تبدیل کنید. این کار معمولاً با استفاده از اهداف اجرایی انجام می‌شود.
۲. رنگ‌های دستگاه خروجیِ محاسبه‌شده را به رنگ‌های مانیتور برگردانید. این کار با هدف اجرایی اندازه‌گیری رنگ مطلق انجام می‌گیرد.

1. gamut mapping 2. color gamut 3. soft proofing

جدول ۶-۷ اهداف اجرایی پروفایل ICC .	
هدف اجرایی	**توصیف**
ادراکی	نگاشت گام را بهینه‌سازی می‌کند تا نتیجه‌ی جذاب هنری به دست آید. رنگ‌های درونی گام ممکن‌است نگهداری نشوند.
اندازه‌گیری رنگ مطلق	رنگ‌های خارج از گام را به نزدیک‌ترین سطح گام نگاشت می‌کند. رابطه‌ی رنگ‌های داخلی گام را حفظ می‌کند.
اندازه‌گیری رنگ نسبی	رنگ‌های خارج از گام را به نزدیک‌ترین سطح گام نگاشت می‌کند. رابطه رنگ‌های داخل گام را حفظ می‌کند. رنگ‌ها را نسبت به نقطه سفید دستگاه رسانه‌ی خروجی، تنظیم می‌کند.
اشباع	اشباع رنگ‌های دستگاه را ماکزیمم می‌کند، که هزینه آن احتمالاً جابه‌جایی پرده رنگ است. به جای تصاویر، برای گراف‌ها و نمودارهای گرافیکی مناسب است.

برای پروفایل ورودی خود، از sRGB.icm استفاده می‌کنیم، پروفایلی که فضای رنگ sRGB را نشان می‌دهد که همراه جعبه‌ابزار وجود دارد. پروفایل خروجی ما SNAP2007.icc است، پروفایل چاپ جدیدی که در رجیستری پروفایل ICC قرار دارد (www.color.org/registry). تصویر نمونه‌ی ما مانند شکل ۴-۷ (الف) است.

ابتدا، تصویر را با افزودن یک مرز سفید ضخیم و یک مرز خاکستری نازک به دور تصویر، پیش‌پردازش می‌کنیم. مرزها باعث می‌شوند "سفید" شبیه‌سازی‌شده‌ی کاغذ روزنامه، بهتر دیده شود:

```
>> f = imread('Fig0704(a).tif');
>> fp = padarray(f, [40 40], 255, 'both');
>> fp = padarray(fp, [4 4], 230, 'both');
>> imshow(fp)
```

شکل ۱۲-۷ (الف) تصویر همراه با اضافات را نشان می‌دهد.

سپس دو پروفایل را می‌خوانیم و از آن‌ها برای تبدیل تصویر زنبق از sRGB به رنگ‌های کاغذ روزنامه استفاده می‌کنیم:

```
>> p_srgb = iccread('sRGB.icm');
>> p_snap = iccread('SNAP2007.icc');
>> cform1 = makecform('icc', p_srgb, p_snap);
>> fp_newsprint = applycform(fp, cform1);
```

سرانجام، از یک ساختار cform دیگر، با استفاده از هدف اجرایی اندازه‌گیری رنگ مطلق، برای تبدیل به sRGB جهت نمایش، استفاده می‌کنیم:

```
>> cform2 = makecform('icc', p_snap, p_srgb, ...
        'SourceRenderingIntent', 'AbsoluteColorimetric', ...
        'DestRenderingIntent', 'AbsoluteColorimetric');
>> fp_proof = applycform(fp_newsprint, cform2);
>> imshow(fp_proof)
```

شکل ۱۲-۷ (ب) نتیجه را نشان می‌دهد. این شکل، تقریبی از تصویری است که در مانیتور دیده می‌شود، زیرا گام رنگ این کتابِ چاپ‌شده، مثل گام رنگ مانیتور نیست. ■

شکل ۱۲–۷ مثالی از نمونه‌خوانی نرم. (الف) تصویر اصلی با مرز سفید. (ب) شبیه‌سازی ظاهر تصویر وقتی که بر روی کاغذ روزنامه چاپ می‌شود.
ب الف

۷–۳ مبانی پردازش تصویر رنگی

در این بخش، مطالعه تکنیک‌های پردازشی را شروع می‌کنیم که برای تصاویر تمام‌رنگی مناسب هستند. گرچه این مطالعات جامع نیستند، تکنیک‌های مطرح‌شده در این بخش، به چگونگی اداره‌کردن تصاویر تمام‌رنگی می‌پردازند. برای بحث‌های بعدی، پردازش تصویر رنگی، را به سه دسته تقسیم می‌کنیم: (۱) *تبدیلات رنگی* (که *نگاشت‌های رنگی* نیز نامیده می‌شود)، (۲) *پردازش مکانی هر یک از صفحات رنگ*، و (۳) *پردازش بردار رنگ*. دسته‌ی اول با پردازش پیکسل‌های هر صفحه‌ی رنگ به طور اکید مبتنی بر مقادیر آن‌ها است و مبتنی بر مختصات مکانی نیست. این دسته مشابه مطالب بخش ۲–۳ است که با تبدیلات شدت سروکار دارد. دسته‌ی دوم با فیلترینگ (همسایگی) مکانی هر یک از صفحات رنگ سروکار دارد و مشابه با بحث در بخش‌های ۴–۳ و ۵–۳ درباره فیلترینگ مکانی است.

دسته‌ی سوم با تکنیک‌های مبتنی بر پردازش تمام مولفه‌های یک تصویر رنگی به طور همزمان سروکار دارد. چون تصاویر تمام‌رنگی حداقل سه مولفه دارند، با پیکسل‌های رنگی می‌توان مثل بردارها رفتار کرد. برای مثال، در سیستم RGB ، هر نقطه رنگی می‌تواند به عنوان برداری تفسیر شود که از مبدأ به آن نقطه در سیستم مختصات RGB بسط می‌یابد (شکل ۲–۷ را ببینید).

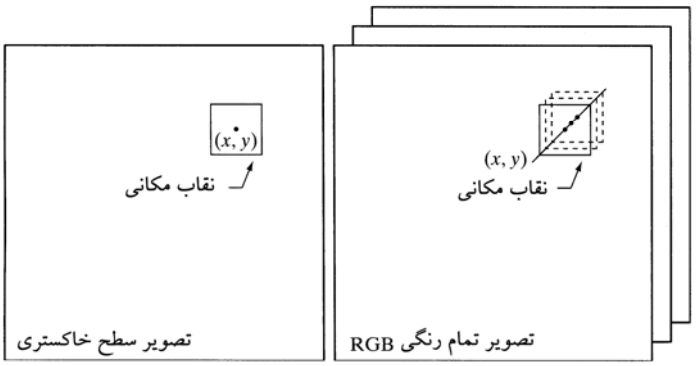
فرض کنید $\mathbf{c}$ ، بردار دلخواهی در فضای رنگ RGB باشد:

$$\mathbf{c} = \begin{bmatrix} c_R \\ c_G \\ c_B \end{bmatrix} = \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

این معادله نشان می‌دهد که مولفه‌های $\mathbf{c}$ ، مولفه‌های RGB مربوط به یک تصویر رنگی در یک نقطه است. این حقیقت را درنظرمی‌گیریم که مولفه‌های رنگی، تابعی از مختصات، با استفاده از نمادگذاری زیر هستند:

$$\mathbf{c}(x, y) = \begin{bmatrix} c_R(x, y) \\ c_G(x, y) \\ c_B(x, y) \end{bmatrix} = \begin{bmatrix} R(x, y) \\ G(x, y) \\ B(x, y) \end{bmatrix}$$

برای تصویری به اندازه $M \times N$ ، تعداد MN بردار از نوع $\mathbf{c}(x, y)$ ، برای $x = 0, 1, 2, \ldots, M-1$ و $y = 0, 1, 2, \ldots, N-1$ وجود دارد.



ب الف **شکل ۱۳-۷** نقاب‌های مکانی برای تصاویر رنگی RGB و سطح خاکستری.

گاهی نتایج حاصل از پردازش تصویر رنگی به صورت صفحه به صفحه یا به صورت کمیت‌های برداری، یکسان است. اما، همان‌طور که در بخش ۶-۷ با جزییات بیشتری شرح داده شد، همیشه این‌طور نیست. برای این‌که این دو روش، معادل هم باشند، دو شرط باید برقرار باشد: اولاً، پردازش باید به بردارها و اسکالرها قابل اعمال باشد. ثانیاً، عملیات روی هر مولفه‌ی بردار باید مستقل از مولفه‌های دیگر باشد. همان‌طور که شرح داده شد، شکل ۱۳-۷ ، پردازش همسایگی مکانی تصاویر رنگی و خاکستری را نشان می‌دهد. فرض کنید، پردازش، میانگین‌گیری همسایگی است. در شکل ۱۳-۷ (الف)، میانگین‌گیری از طریق مجموع سطوح خاکستری همسایگی انجام می‌گیرد. در شکل ۱۳-۷ (ب) میانگین‌گیری از طریق مجموع تمام بردارهای موجود در همسایگی و تقسیم هر مولفه بر تعداد کل بردارها در آن همسایگی به دست می‌آید. اما هر مولفه‌ی بردار میانگین، برابر با مجموع پیکسل‌های موجود در تصویر متناظر با آن مولفه است، که همانند نتیجه‌ای است که میانگین‌گیری، در همسایگی هر تصویر مولفه‌ی رنگی به طور جداگانه انجام می‌دهد، و سپس بردار تشکیل می‌گردد.

۷-۴ تبدیلات رنگ

تکنیک‌های توصیف‌شده در این بخش، مبتنی بر پردازش مولفه‌های رنگ یک تصویر رنگی یا مولفه شدت تصویر تک‌رنگ در زمینه یک مدل رنگی است. برای تصاویر رنگی، به تبدیلاتی به شکل زیر می‌پردازیم:

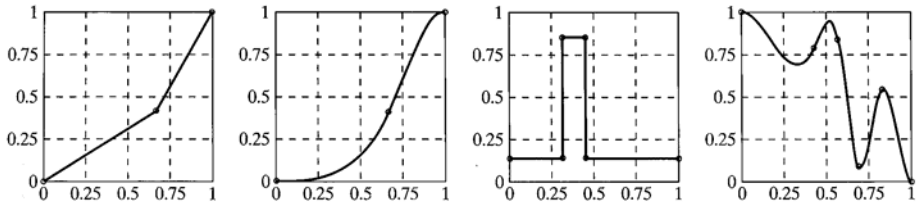
$$s_i = T_i(r_i) \qquad i = 1, 2, \ldots, n$$

که r_i و s_i مولفه‌های رنگ تصاویر ورودی و خروجی، n ، بُعد (یا تعداد مولفه‌های رنگ در) فضای رنگ r_i است و T_i به عنوان توابع تبدیل تمام رنگی (یا نگاشت) در نظر گرفته می‌شوند.

اگر تصاویر ورودی، تک‌رنگ باشند، آنگاه معادله‌ای به شکل زیر می‌نویسیم:

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

که r نشان‌دهنده‌ی مقادیر سطح خاکستری است، s_i و T_i همانند قبل، و n تعداد مولفه‌های رنگ در s_i است. این معادله، نگاشت سطوح خاکستری به رنگ‌های دلخواه را توصیف می‌کند، و فرآیندی که غالباً به نام *تبدیل شبه‌رنگی* یا *نگاشت شبه‌رنگی* نامیده می‌شود. توجه کنید که معادله اول در صورتی می‌تواند برای پردازش تصاویر تک‌رنگ به کار رود که قرار دهیم $r_1 = r_2 = r_3 = r$. در هر حال، معادلاتی که در این‌جا آمده‌اند، بسط‌های ساده‌ای از معادله تبدیل شدت هستند که در بخش ۲-۳ معرفی شدند. تمام n تابعِ تبدیل شبه‌رنگی یا تمام رنگی $\{T_1, T_2, \ldots, T_n\}$ مستقل از مختصات (x, y) تصویر مکانی هستند.



شکل ۱۴-۷ مشخص‌کردن توابع نگاشت با استفاده از سه نقطه: (الف) و (پ) درونیابی خطی و (ب) و (ت) درونیابی محور خطی.

ت پ ب الف

بعضی از تبدیلات مقیاس خاکستری معرفی‌شده در فصل ۳، مثل imcomplement ، که نگاتیو (منفی) تصویر را محاسبه می‌کند، مستقل از محتوای سطح خاکستری تصویر در حال تبدیل است. تبدیلات دیگری مثل histeq ، که به تبدیل سطح خاکستری بستگی دارند، *وفقی*[1] هستند، اما وقتی پارامترهای آن برآورده شدند، این تبدیل ثابت است. حتی، بعضی دیگر، برای حالت محاوره‌ای مفیدتر هستند. هنگام کارکردن با نگاشت‌های رنگی و تمام‌رنگی، مخصوصاً وقتی که دیدن و تفسیر انسان‌ها (برای توازن رنگ) مطرح است، وضعیت مشابهی وجود دارد. در این کاربردها، انتخاب توابع نگاشت مناسب، با دستکاری مستقیم نمایش گرافیکی توابع کاندید و مشاهده‌ی اثر ترکیبی آن‌ها (بی‌درنگ[2]) در تصویر در حال پردازش، بهتر صورت می‌گیرد.

شکل ۱۴-۷ یک روش ساده ولی قدرتمند را برای مشخص‌کردن توابع نگاشت نشان می‌دهد. شکل ۱۴-۷ (الف) تبدیلی را نشان می‌دهد که با درونیابی خطی سه *نقطه‌ی کنترلی* (مختصاتی که دور آن‌ها در شکل، دایره کشیده شده است) ایجاد شد. شکل ۱۴-۷ (ب) تبدیلی را نشان می‌دهد که از درونیابی محوری مکعبی همان سه نقطه به دست آمد. شکل‌های ۱۴-۷ (پ) و (ت) به ترتیب، درونیابی‌های خطی و محوری مکعبی پیچیده‌تری را نشان می‌دهند. در MATLAB از هر دو درونیابی پشتیبانی می‌شود. درونیابی خطی، به صورت زیر پیاده‌سازی می‌شود:

```
z = interp1q(x, y, xi)
```

که یک بردار ستونی را برمی‌گرداند که شامل مقادیری از تابع یک‌بعدی با درونیابی خطی z در نقطه xi است. بردارهای ستونی x و y ، مختصات نقاط کنترلی موردنظر را مشخص می‌کنند. عناصر x باید به صورت یکنواخت افزایش یابند. طول z برابر با طول xi است. مثال زیر را درنظربگیرید:

```
>> z = interp1q([0 255]', [0 255]', [0: 255]')
```

این دستور، یک نگاشت یک به یک ۲۵۶ عنصری را به وجود می‌آورد که نقاط کنترلی (0, 0) و (255, 255) را به هم متصل می‌کند – یعنی z = [0 1 2 . . . 255]' .

به طور مشابه، درونیابی محوری مکعبی، با تابع محوری زیر پیاده‌سازی می‌شود:

```
z = spline(x, y, xi)
```

که متغیرهای z ، x ، y و xi همانند تابع interp1q است که شرح آن گذشت. اما، xi باید جدا از استفاده در تابع spline باشد. علاوه‌براین، اگر y دو عنصر بیشتر از x داشته باشد، فرض می‌شود که عناصر اول و آخر آن باید شیب‌های پایانی محور مکعبی باشند. تابع نشان‌داده‌شده در شکل ۱۴-۷ (ب)، با استفاده از شیب‌های پایانی با مقدار صفر تولید شده است.

---

1. adaptive 2. real time

جدول ۷–۷ ورودی‌های معتبر برای تابع ice .	
نام خاصیت	مقدار خاصیت
‘image’	تصویر ورودی f تک‌رنگ یا RGB که باید با استفاده از نگاشتی که به طور محاوره‌ای مشخص می‌گردد، تبدیل شود.
‘space’	فضای رنگ مولفه‌هایی که باید تغییر کند. مقادیر ممکن عبارتنداز ’rgb‘ ، ’cmy‘ ، ’hsi‘ ، ’hsv‘ ، ’ntsc‘ (یا ’yiq‘) و ’ycbcr‘ . پیش‌فرض ’rgb‘ است.
‘wait’	اگر ’on‘ باشد (پیش‌فرض)، q تصویر ورودیِ نگاشت‌شده است. اگر ’off‘ باشد، g دستگیره‌ی تصویر ورودی نگاشت‌شده است.

مشخصات توابع تبدیل، با دستکاری گرافیکی نقاط کنترلی که ورودی توابع interp1q و spline هستند، و نمایش بی‌درنگ تصاویر در حال پردازش، می‌توانند محاوره‌ای شوند. تابع ice (ویرایش محاوره‌ای رنگ)، این کار را دقیقاً انجام می‌دهد. این تابع به صورت زیر به کار می‌رود:

ice

```
g = ice('Property Name', 'Property Value', . . .)
```

که ’PropertyName‘ و ’PropertyValue‘ باید به صورت جفتی ظاهر شوند، و نقاط، تکرارهای الگوی شامل جفت‌های ورودی متناظر را نشان می‌دهند. جدول ۷–۷ جفت‌های معتبر را برای استفاده در تابع ice نشان می‌دهد. مثال‌هایی در ادامه‌ی این بخش ارائه خواهد شد.

با مراجعه به پارامتر ’wait‘ ، وقتی گزینه ’on‘ به طور صریح یا پیش‌فرض انتخاب شده است، خروجی g ، تصویر پردازش شده است. در این حالت، ice کنترل فرآیند، از جمله مکان‌نما[1] را به دست می‌گیرد، و در نتیجه، نمی‌توان چیزی در پنجره‌ی فرمان تایپ کرد، تا این‌که تابع خاتمه یابد، که در این زمان، نتیجه‌ی نهایی، تصویری با دستگیره g (یا هر شیء گرافیکی در حالت کلی) است. وقتی گزینه ’off‘ انتخاب می‌شود، g یک دستگیره از تصویر پردازش‌شده است و کنترل فوراً به پنجره فرمان برمی‌گردد. بنابراین، فرمان‌های جدیدی می‌تواند با تابع ice تایپ شود. برای به دست آوردن خواص شیء گرافیکی، از تابع get استفاده می‌کنیم:

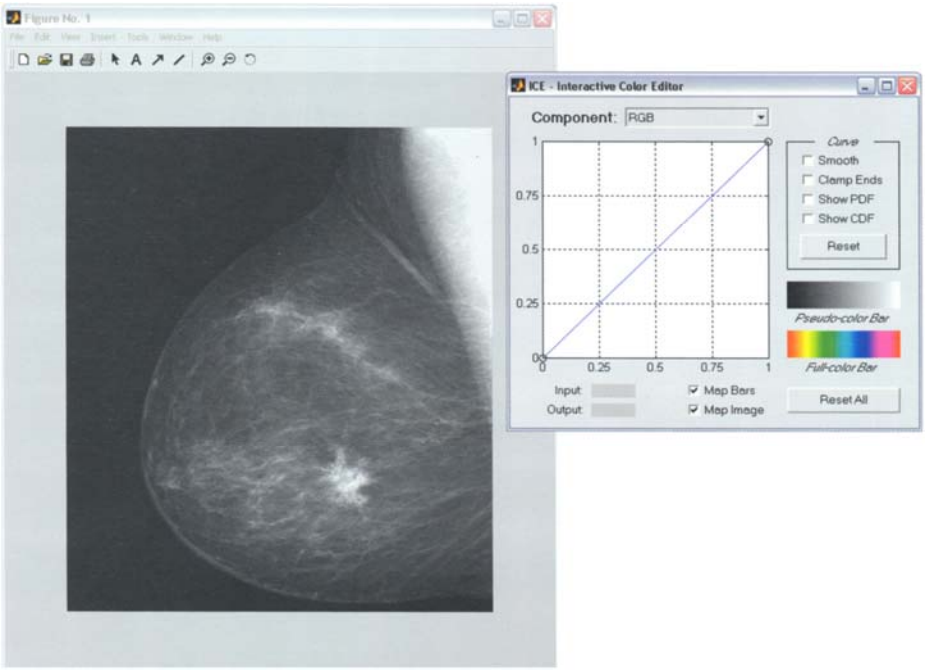
```
h = get(g)
```

این تابع، تمام خواص و مقادیر فعلیِ قابل اِعمال شیء گرافیکی مشخص‌شده توسط دستگیره g را برمی‌گرداند. این خواص در ساختار h ذخیره می‌شود، و در نتیجه تایپ h در خط فرمان، تمام خواص تصویر پردازش‌شده را نمایش می‌دهد. برای استخراج یک خاصیت ویژه، دستور h.PropertyName را تایپ می‌کنیم.

اگر f یک تصویر تک‌رنگ RGB باشد، تابع ice به صورت زیر به کار می‌رود:

```
>> ice                                    % Only the ice
                                          % graphical
                                          % interface is
                                          % displayed.
>> g = ice('image', f);                   % Shows and returns
                                          % the mapped image g.
>> g = ice('image', f, 'wait', 'off')     % Shows g and returns
                                          % the handle.
>> g = ice('image', f, 'space', 'hsi')    % Maps RGB image f in
                                          % HSI space.
```

1. cursor



شکل ۱۵-۷ نمونه‌ای از پنجره‌های تابع ice .

توجه کنید که وقتی فضای رنگی غیر از RGB مشخص شود، تصویر ورودی (تک‌رنگ یا RGB)، قبل از هر نگاشتی، به فضای مشخص‌شده تبدیل می‌گردد. سپس تصویرِ نگاشت‌شده، برای خروجی به RGB تبدیل می‌شود. خروجی ice همیشه RGB است. ورودی آن تک‌رنگ یا RGB است. اگر g = ice('image', f) را تایپ کنیم، یک تصویر و واسط گرافیکی (GUI) مانند شکل ۱۵-۷ در دسکتاپ MATLAB ظاهر می‌شود. در آغاز، منحنی تبدیل، خط مستقیمی است که نقطه کنترلی در هر انتهای آن وجود دارد. نقاط کنترلی، با ماوس دستکاری می‌شوند و جدول‌های ۸-۷ و ۹-۷ تابع سایر مولفه‌های GUI را نشان می‌دهد. مثال‌های زیر، بعضی از کاربردهای تابع ice را نشان می‌دهند.

مثال ۵-۷ : نگاشت‌های معکوس، منفیِ تک‌رنگ و مکمل‌های رنگی.

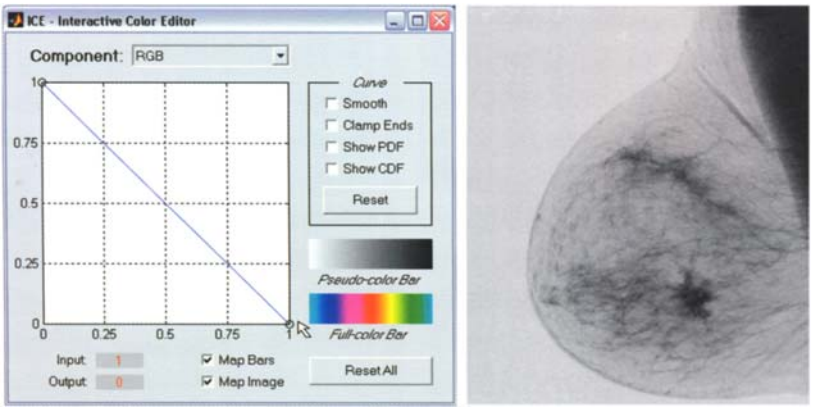
شکل ۱۶-۷ (الف) واسط ice را نشان می‌دهد، به طوری که منحنی RGB پیش‌فرضِ شکل ۱۵-۷ تغییر یافت تا یک تابع معکوس یا منفی تولید شود. برای ایجاد تابع نگاشت معکوس، نقطه کنترلی (0, 0) به (0, 1) منتقل می‌شود (با کلیک‌کردن و حرکت آن به گوشه‌ی بالای چپ) و نقطه کنترلی (1, 1) به مختصات (1, 0) منتقل می‌شود.

جدول ۸-۷ دستکاری نقاط کنترلی با ماوس.

عمل ماوس	نتیجه
دکمه چپ	نقطه کنترلی را با فشاردادن و حرکت‌دادن، انتقال می‌دهد.
دکمه چپ + کلید شیفت	نقطه کنترلی را اضافه می‌کند. مکانِ نقطه کنترلی می‌تواند با فشاردادن کلید shift و حرکت ماوس، تغییر کند.
دکمه چپ + کلید Ctrl	نقطه کنترلی را حذف می‌کند.

جدول ۹-۷ تابع کادرهای انتخاب و دکمه‌های فشاری در iceGUI .	
عنصر GUI	شرح
Smooth	برای درونیابی محور مکعبی (منحنی یکنواخت) انتخاب می‌شود. اگر انتخاب نشده باشد، از درونیابی خطی تکه‌ای (piecewise) استفاده می‌شود.
Clamp Ends	برای این‌که شیب‌های منحنی شروع و پایان در درونیابی محور مکعبی صفر باشد، انتخاب می‌شود. درونیابی خطی تکه‌ای تأثیری ندارد.
Show PDF	تابع چگالی احتمالِ مولفه‌های تصویر تحت تأثیر تابع نگاشت را نشان می‌دهد.
Show CDF	تابع توزیع تجمیع احتمال را به جای PDF ها نشان می‌دهد (توجه کنید که PDF ها و CDF ها همزمان نمی‌توانند نمایش داده شوند).
Map Image	اگر انتخاب شده باشد، نگاشت تصویر فعال است، وگرنه فعال نیست.
Map Bars	اگر انتخاب شده باشد، نگاشت شبه رنگی و تمام رنگی فعال خواهد بود. وگرنه نوارهای نگاشت‌نشده نمایش داده می‌شوند.
Reset	تابع نگاشت نشان‌داده‌شده را مقداردهی می‌کند، و تمام پارامترهای منحنی را از حالت انتخاب خارج می‌کند.
Reset All	توابع نگاشت را مقدار اولیه می‌دهد.
Input/Output	مشخصات نقطه کنترلی انتخاب‌شده در منحنی تبدیل را نشان می‌دهد. ورودی به محور افقی، و خروجی به محور عمودی اشاره دارد.
Component	یک تابع نگاشت را برای دستکاری محاوره‌ای انتخاب می‌کند. در فضای RGB ، انتخاب‌های ممکن شامل R ، G ، B ، و RGB است. در فضای HSI ، گزینه‌ها عبارتنداز H ، S ، I ، HSI و غیره.

توجه کنید که چگونه مختصات مکان‌نما با رنگ قرمز در کادرهای ورودی و خروجی نمایش داده می‌شود. نقطه نگاشت RGB تغییر می‌یابد. هر یک از نگاشت‌های R ، G و B در حالت‌های پیش‌فرض 1:1 باقی می‌مانند (وارده‌ی Component را در جدول ۶-۷ ببینید) برای ورودی‌های تک‌رنگ، تضمین می‌شود که خروجی‌ها تک‌رنگ باشند. شکل ۱۶-۷ (ب) منفیِ تک‌رنگ را نشان می‌دهد که ناشی از نگاشت معکوس است. توجه کنید که معادل شکل ۳-۳ (ب) است، که با استفاده از تابع imcomplement به دست آمد. نوار شبه‌رنگی در شکل ۱۶-۷ (الف)، " نگاتیو عکاسی" مربوط به نوار مقیاس خاکستریِ اصلی در شکل ۱۵-۷ است.



ب الف **شکل ۱۶-۷** (الف) تابع نگاشت منفی و (ب) اثر آن بر تصویر تک‌رنگ شکل ۱۵-۷.

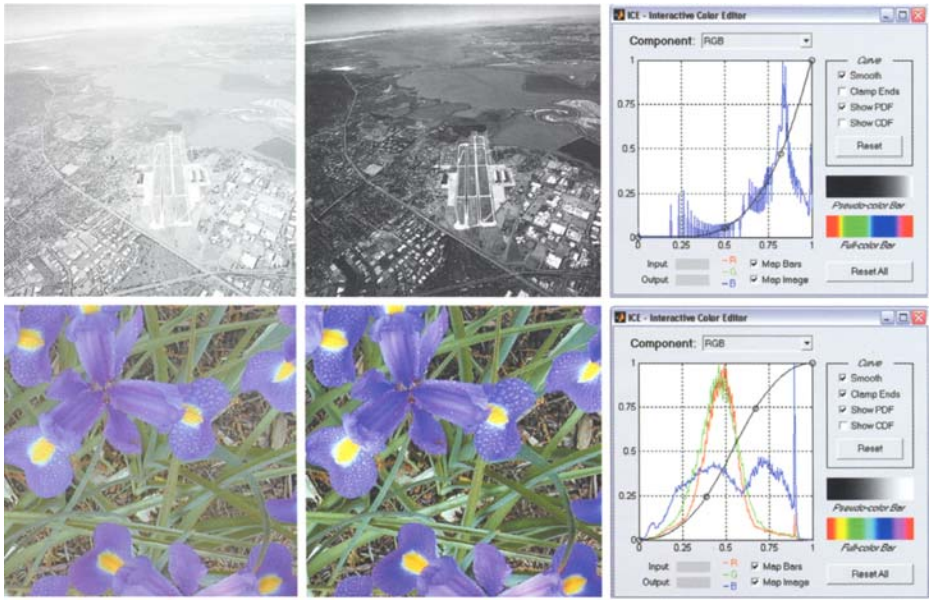
ب الف **شکل ۱۷–۷** (الف) تصویر کاملاً رنگی. (ب) نگاتیو آن (متمم رنگ).

توابع معکوس یا منفی، در پردازش رنگ نیز مفید هستند. همان‌طور که در شکل‌های ۱۷–۷ (الف) و (ب) مشاهده می‌کنید، نتیجه‌ی نگاشت، شبیه نگاتیوهای فیلم رنگی معمولی است. برای نمونه، نوار قرمز در سطر پایینی شکل ۱۷–۷ (الف) به فیروزه‌ای در شکل ۱۷–۷ (ب) تبدیل می‌شود – *متمم رنگ قرمز*. متمم رنگ اولیه، مخلوطی از دو رنگ اولیه دیگر است (مثلاً فیروزه‌ای مخلوطی از آبی و سبز است). همانند حالت مقیاس خاکستری، متمم‌های رنگ، برای ارتقای جزییاتی مفید است که اندازه‌ی آن غالب است. توجه کنید که *نوار کاملاً رنگی* در شکل ۱۶–۷ (الف) شامل متمم پرده‌های رنگ در نوار کاملاً رنگی شکل ۱۵–۷ است. ■

مثال ۶–۷: ارتقای کنتراست تک‌رنگی و رنگی.

استفاده از تابع ice را برای دستکاری کنتراست رنگی و تک‌رنگ، درنظرمی‌گیریم. شکل‌های ۱۸–۷ (الف) تا (پ) اثربخشی ice را در پردازش تصویر تک‌رنگ نشان می‌دهند. تصاویر شکل‌های ۱۸–۷ (ت) تا (ج) اثربخشی مشابهی را برای ورودی‌های رنگی نشان می‌دهند. همانند مثال قبل، توابع نگاشتی که نشان داده نشدند، در حالت پیش‌فرض یا 1:1 باقی می‌مانند. در هر دو دنباله پردازش، کادر انتخاب Show PDF فعال است. بنابراین، هیستوگرام عکس هوایی در (الف)، زیرِ تابع نگاشتِ گاماشکل در شکل (پ) نمایش داده می‌شود و سه هیستوگرام در (ج) برای تصویر رنگی در (پ) نشانَ داده شده‌اند. برای هر کدام از سه مولفه رنگ، یک هیستوگرام. گرچه تابع نگاشت Sشکل در (ج)، کنتراست تصویر در (ت) را افزایش می‌دهد (آن را با (ث) مقایسه کنید)، تأثیر اندکی در پرده رنگ نیز دارد. تغییر اندکی در رنگ‌ها، به طور مجازی در (ث) قابل دریافت نیست، اما نتیجه‌ی بدیهی از نگاشت است، که در نوار مرجع تمام‌رنگی در (ج) قابل مشاهده می‌باشد. از مثال قبلی به یاد بیاورید که تغییرات یکسان در سه مولفه‌ی تصویر RGB می‌تواند اثر شگرفی روی رنگ داشته باشد (نگاشت متمم رنگ در شکل ۱۷–۷ را ببینید). ■

مولفه‌های قرمز، سبز و آبی تصاویر ورودی در مثال‌های ۵–۷ و ۶–۷ به طور یکسان نگاشت می‌شوند – یعنی، از یک تابع تبدیل استفاده می‌کنند. برای اجتناب از مشخصات سه تابع یکسان، تابع ice یک تابع "تمام‌مولفه" را فراهم می‌سازد که برای نگاشت تمام مولفه‌های ورودی استفاده می‌شود. بقیه مثال‌های این بخش، تبدیلاتی را شرح می‌دهند که در آن‌ها، سه مولفه، به صورت‌های مختلفی پردازش می‌شوند.



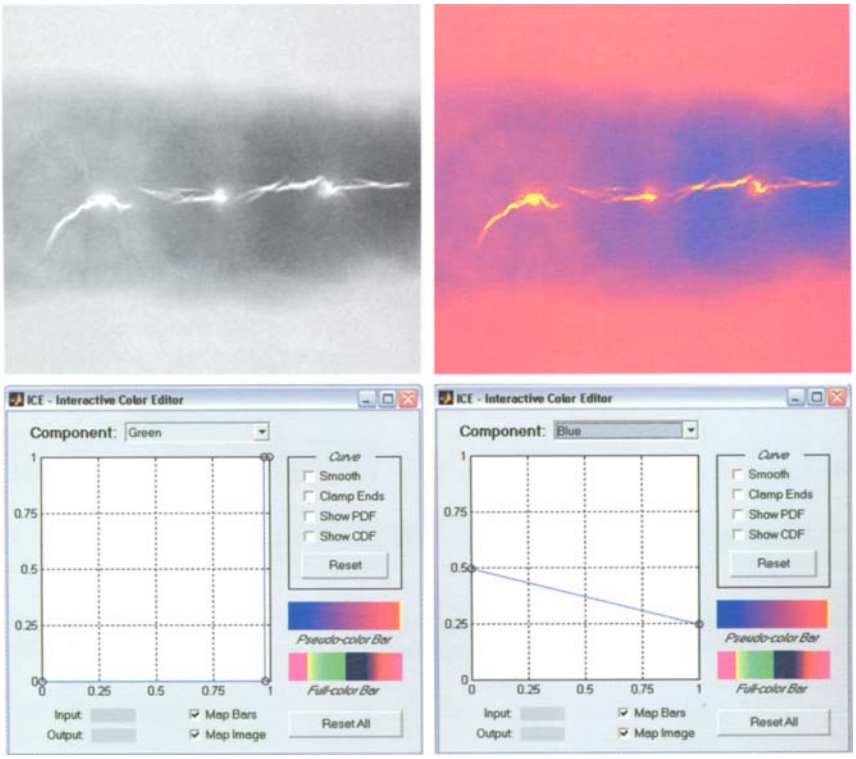
پ ب الف
ج ث ت **شکل ۱۸–۷** استفاده از تابع ice برای ارتقای کنتراست تمام‌رنگی و تک‌رنگ: (الف) و (ت) تصاویر ورودی هستند که هر کدام، ظاهری "washed-out" دارند؛ (ب) و (ث) نتایج پردازش‌شده‌اند. (پ) و (ج) نمایش‌های ice هستند.

مثال ۷–۷ : نگاشت‌های شبه‌رنگی.

همان‌طور که گفته شد، وقتی تصویر تک‌رنگ در فضای رنگ RGB نمایش داده می‌شود و مولفه‌های حاصل، به طور مستقل نگاشت می‌شوند، نتیجه‌ی تبدیل‌یافته، یک تصویر شبه‌رنگی است که در آن، سطوح خاکستری تصویر ورودی، با رنگ‌های دلخواه جایگزین می‌شود. تبدیلاتی که این کار را انجام می‌دهند، مفید هستند، زیرا چشم انسان می‌تواند بین میلیون‌ها رنگ تمایز قائل شود – اما سایه‌های اندکی از خاکستری را تشخیص می‌دهد. بنابراین، نگاشت‌های شبه‌رنگی غالباً برای انجام تغییرات اندکی در سطح خاکستریِ قابل مشاهده توسط چشم انسان، به کار می‌روند، یا ناحیه‌های مقیاس خاکستری مهم را برجسته می‌کنند. در حقیقت، استفاده اصلی شبه‌رنگی، بصری‌سازی انسانی است.

شکل ۱۹–۷ (الف) یک تصویر اشعه ایکس از جوش[1] (ناحیه تیره‌ی افقی) است که شامل چندین ترک (شیار) و منفذ (نقاط سفید در وسط تصویر) است. نسخه‌ی شبه‌رنگی تصویر در شکل ۱۹–۷ (ب) آمده است. این تصویر، با نگاشت مولفه‌های سبز و آبی، بر روی تصویر ورودیِ RGB که با استفاده از توابع نگاشت در شکل‌های ۱۹–۷ (پ) و (ت) تبدیل شده، به دست آمده است. به تفاوت بصری شگرفی که نگاشت شبه‌رنگی ایجاد می‌کند، توجه کنید. نوار مرجع شبه‌رنگی GUI ، یک راهنمای بصری آسانی را برای نگاشت مرکب فراهم می‌سازد. همان‌طور که می‌توانید در شکل‌های ۱۹–۷ (پ) و (ت) ببینید، توابع نگاشتی که به طور محاوره‌ای مشخص شدند، مقیاس *سیاه به سفید* را به پرده‌های رنگ بین آبی و قرمز تبدیل می‌کنند، به طوری که زرد برای سفید رزرو شده است. البته، زرد متناظر با شیارها و منفذهای جوش است که ویژگی‌های مهمی در این مثال هستند. ■

1. weld



ب الف
ت پ

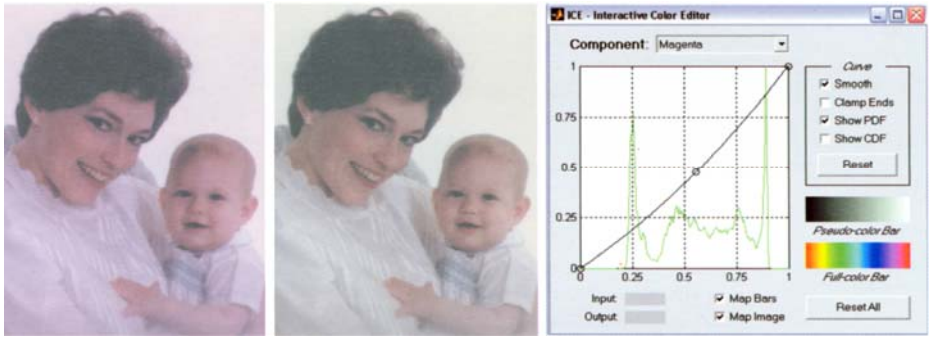
شکل ۱۹-۷ (الف) اشعه ایکس یک جوش نامناسب. (ب) نسخه‌ی شبه‌رنگی جوش. (پ) و (ت) توابع نگاشت برای مولفه‌های سبز و آبی.

مثال ۸-۷ : توازن رنگ[1].

شکل ۲۰-۷ کاربردی را نشان می‌دهد که شامل تصویر تمام‌رنگی است، که بهتر است، مولفه‌های رنگ تصویر، مستقل از یکدیگر نگاشت شوند. تبدیل *توازن رنگ*[2] یا *تصحیح رنگ*، در سیستم‌های بازتولید رنگ استفاده می‌شد، ولی امروزه در اغلب کامپیوترهای رومیزی انجام می‌گیرد. یک استفاده مهم، ارتقای عکس است. گرچه عدم توازن‌های رنگ می‌تواند از طریق تحلیل یک رنگ معین در تصویر به دست آید – با استفاده از *طیف‌سنج*[3] – برآورد بصری دقیق وقتی امکان‌پذیر است که ناحیه‌های سفیدی وجود داشته باشند، که در آن‌جا، مولفه‌های RGB یا CMY باید برابر باشند. همان‌طور که در شکل ۲۰-۷ می‌بینید، تُن‌های پوست نیز برای برآورد بصری مفید هستند، زیرا انسان‌ها دریافت‌کننده خوبی از رنگ پوستِ مناسب هستند.

شکل ۲۰-۷ (الف) یک اسکن CMY از مادر و فرزندش را با تأکید بر بنفش نشان می‌دهد (توجه کنید که تنها نسخه‌ی RGB از تصویر می‌تواند در MATLAB نمایش داده شود). برای سادگی و سازگاری با MATLAB ، تابع ice فقط ورودی‌های RGB (و تک‌رنگ) را می‌پذیرد– ولی می‌تواند ورودی را در فضاهای رنگ متعددی

1. color balancing 2. color correction 3. spectrometer



شکل ۲۰-۷ استفاده از تابع ice برای توازن رنگ: (الف) تصویری با رنگ زیاد بنفش. (ب) تصویر درست‌شده؛ و (پ) تابع نگاشتی که برای تصحیح عدم توازن به کار می‌رود.
پ ب الف

پردازش کند که در جدول ۷-۷ آمده‌اند. برای تغییر مولفه‌های CMY تصویر f1 با فرمت RGB به طور محاوره‌ای، ice به صورت زیر فراخوانی می‌شود:

```
>> f2 = ice('image', f1, 'space', 'CMY');
```

همان‌طور که شکل ۲۰-۷ نشان می‌دهد، کاهش اندکی در بنفش، اثر شگرفی در رنگ تصویر دارد. ■

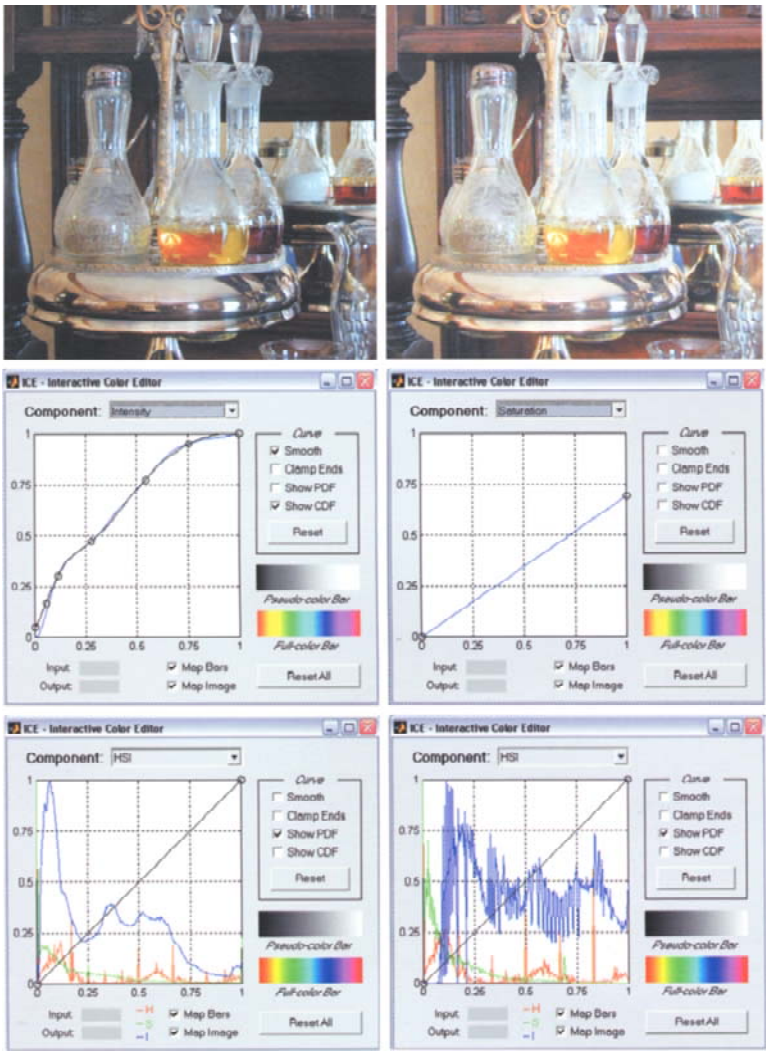
مثال ۹-۷ : نگاشت‌های مبتنی بر هیستوگرام.

تعدیل هیستوگرام، یک فرآیند نگاشت سطح خاکستری است که سعی می‌کند تصاویر تک‌رنگ را با هیستوگرام‌هایی با شدت یکنواخت تولید نماید. همان‌طور که در بخش ۲-۳-۳ گفته شد، تابع نگاشت مورد نیاز، تابع توزیع تجمیع (CDF) از سطوح خاکستری در تصویر ورودی است. چون تصاویر رنگی چندین مولفه دارد، تکنیک مقیاس خاکستری باید اصلاح شود تا بیش از یک مولفه و هیستوگرام مربوط را اداره کند. همان‌طور که انتظار می‌رود، تعدیل هیستوگرامِ مولفه‌های تصویر رنگی به طور مستقل، عقلانی نیست. نتیجه، معمولاً همراه با خطای رنگی است. یک روش منطقی‌تر، توزیع یکنواخت شدت‌های رنگ است، به طوری که خود رنگ‌ها (یعنی پرده رنگ) تغییر نکنند.

شکل ۲۱-۷ (الف) یک تصویر رنگی از چرخی است که شامل جای نمک و فلفل و همزن‌ها است. تصویر تبدیل‌شده در شکل ۲۱-۷ (ب)، که با استفاده از تبدیلات شکل‌های ۲۱-۷ (پ) و (ت) تولید شد، بسیار روشن‌تر است. اکنون چندین رگه و تزیینات میز چوبی که چرخ روی آن قرار دارد، مشهود است. مولفه شدت، با استفاده از تابع شکل ۲۱-۷ (پ) نگاشت شده است، که برآورد نزدیکی از CDF آن مولفه است (که در شکل نشان داده شده است). تابع پرده نگاشت در شکل ۲۱-۷ (ت) انتخاب شد تا کل دریافت رنگِ نتیجه با تعدیل شدت را بهبود بخشد. توجه کنید که هیستوگرام‌های مربوط به مولفه‌های پرده رنگ، اشباع و شدتِ تصاویر ورودی و خروجی، به ترتیب در شکل‌های ۲۱-۷ (ت) و (ج) نشان داده شده‌اند. مولفه‌های پرده رنگ، به طور مجازی یکسان هستند (که مطلوب است)، درحالی‌که مولفه‌های شدت و اشباع تغییر کردند. سرانجام، توجه کنید که برای پردازش RGB در فضای رنگ HSI ، جفت نام / مقدار 'hsi' / 'space' را در فراخوانی ice می‌گنجانیم. ■

تصاویر خروجی تولیدشده در مثال‌های قبلی در این بخش، از نوع RGB و کلاس uint8 هستند. برای نتایج تک‌رنگی، همانند مثال ۵-۷، هر سه مولفه خروجی RGB یکسان هستند. نمایش فشرده‌تری را می‌توان از طریق تابع rgb2gray در جدول ۳-۷ یا با استفاده از دستور زیر به دست آورد، که f2 یک تصویر RGB است که توسط ice تولید شد و f3 تصویر تک‌رنگ است:

```
>> f3 = f2(:, :, 1);
```



ب الف
ت پ
ج ث

شکل ۲۱-۷ تعدیل هیستوگرام و تنظیم اشباع در فضای رنگ HSI . (الف) تصویر ورودی. (ب) نتیجه‌ی نگاشت‌شده. (پ) تابع نگاشت مولفه شدت و تابع توزیع تجمیعی. (ت) تابع نگاشت مولفه اشباع. (ث) هیستوگرام‌های مولفه تصویر ورودی. (ج) هیستوگرام‌های مولفه‌ی نتیجه‌ی نگاشت‌شده.

۷-۵ فیلترینگ مکانی تصاویر رنگی

مطالب بخش ۴-۷ با تبدیلات رنگی سروکار دارد که بر روی پیکسل‌های یک‌تصویری از صفحات مولفه‌ی یک‌رنگی اجرا می‌شوند. سطح بعدی پیچیدگی، شامل اجرای پردازش همسایگی مکانی، باز هم روی صفحات یک‌تصویری است. این عیب، مثل بحث روی تبدیلات شدت در بخش ۲-۳، و بحث روی فیلترینگ مکانی در بخش‌های ۴-۳ و ۵-۳ است. فیلترینگ مکانی تصاویر رنگی را، متمرکز بر روی تصاویر RGB معرفی می‌کنیم،

ولی مفاهیم اساسی، برای سایر مدل‌های رنگی نیز قابل اِعمال است. پردازش مکانی تصاویر رنگی را با دو مثال از فیلترینگ خطی شرح می‌دهیم: هموارکردن و تیزکردنِ تصویر.

۷–۵–۱ هموارکردن تصویر رنگی

با مراجعه به شکل ۷–۱۳ (الف) و بحث در بخش‌های ۴–۳ و ۵–۳، یک روش هموارکردن تصویر تک‌رنگ، تعریف یک نقاب فیلتر از یک‌ها است، که تمام مقادیر پیکسل را با ضرایب موجود در نقاب مکانی ضرب می‌کند و نتیجه را بر مجموع عناصر موجود در نقاب جمع می‌کند. فرآیند هموارسازی تصویر تمام‌رنگی با استفاده از نقاب‌های مکانی، در شکل ۷–۱۳ (ب) آمده است.

این فرآیند (مثلاً در فضای RGB) همانند تصاویر مقیاس خاکستری فرمول‌بندی می‌شود، با این تفاوت که به جای هر یک از پیکسل‌ها، اکنون با مقادیر بردار به شکلی که در بخش ۷–۳ نشان داده شد، سروکار داریم. فرض کنید S_{xy} مجموعه‌ای از مختصات را نشان می‌دهد که یک همسایگی به مرکز (x,y) را در تصاویر رنگی تعریف می‌کند. میانگین بردارهای RGB در این همسایگی به صورت زیر است:

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

که K تعداد پیکسل‌ها در همسایگی است. از بحث بخش ۷–۳ و خواص جمع بردارها، نتیجه می‌شود که:

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

هر مولفه‌ی این بردار را به عنوان نتیجه‌ای می‌دانیم که با اجرای میانگین‌گیری همسایگی روی هر تصویر مولفه، با استفاده از نقاب فیلتری که شرح آن گذشت، به دست خواهیم آورد. بنابراین، نتیجه می‌گیریم که هموارسازی به وسیله میانگین‌گیری همسایگی، می‌تواند بر اساس هر تصویر در صفحه انجام گیرد. نتایج یکسان خواهد بود، به طوری که گویی میانگین‌گیری همسایگی مستقیماً در فضای بردار رنگ انجام شده است.

همان‌طور که در بخش ۱–۵–۳ بحث شد، فیلترینگ هموارسازی مکانی، از نوعی که در پاراگراف قبلی بحث شد، با استفاده از تابع fspecial با گزینه 'average' انجام می‌گیرد. وقتی فیلتری ایجاد شد، فیلترینگ با استفاده از تابع imfilter انجام می‌گیرد که در بخش ۱–۴–۳ معرفی شد. از نظر ادراکی، هموارسازی یک تصویر رنگی RGB مثل fc ، با فیلتر خطی شامل مراحل زیر است:

۱. سه مولفه تصویر را استخراج کنید:

```
>> fR = fc(:, :, 1);
>> fG = fc(:, :, 2);
>> fB = fc(:, :, 3);
```

۲. هر یک از تصاویر مولفه را به طور جداگانه فیلتر کنید. برای مثال، اگر w فیلتر هموارسازی باشد که توسط fspecial تولید شد، تصویر مولفه قرمز را به صورت زیر هموار کنید:

```
>> fR_filtered = imfilter(fR, w, 'replicate');
```

برای دو مولفه تصویر دیگر نیز به همین صورت عمل می‌کنیم.

۳. تصویر RGB فیلترشده را بازسازی کنید:

```
>> fc_filtered = cat(3, fR_filtered, fG_filtered, fB_filtered);
```

اما، چون فیلترینگ خطی تصاویر RGB را، مستقیماً در MATLAB ، با استفاده از روش تصاویر تک‌رنگ انجام می‌دهیم، سه مرحله‌ی قبلی می‌توانند در یک مرحله با هم ادغام شوند:

```
>> fc_filtered = imfilter(fc, w, 'replicate');
```

مثال ۱۰-۷ : هموارسازی تصویر رنگی.

شکل ۲۲-۷ (الف) یک تصویر RGB به اندازه 1197×1197 پیکسل را نشان می‌دهد و شکل‌های ۲۲-۷ (ب) تا (ت) تصاویر مولفه RGB آن هستند که با استفاده از رویه شرح‌داده‌شده در پاراگراف قبلی استخراج شدند.

شکل ۲۲-۷ (الف) تصویر RGB . (ب) تا (ت) به ترتیب تصاویر مولفه قرمز، سبز و آبی.

پ ب الف **شکل ۲۳-۷** از چپ به راست: مولفه‌های پرده رنگ، اشباع، و شدتِ شکل ۲۲-۷ (الف).

از نتیجه‌ی بحث قبلی می‌دانیم که هموارسازی هر یک از تصاویر مولفه و تشکیل یک تصویر رنگی مرکب، شبیه هموارسازی تصویر RGB اصلی با استفاده از فرمان انتهای پاراگراف قبلی است. شکل ۲۴-۷ (الف) نتیجه‌ی به دست آمده با استفاده از فیلتر میانگین به اندازه 25×25 پیکسل را نشان می‌دهد.

سپس، آثار هموارسازی فقط مولفه‌ی شدت نسخه‌ی HSI شکل ۲۲-۷ (الف) را بررسی می‌کنیم. شکل‌های ۲۳-۷ (الف) تا (پ)، سه تصویر مولفه‌ی HSI به دست آمده با استفاده از تابع rgb2hsi را نشان می‌دهند، که fc شکل ۲۲-۷ (الف) است:

```
>> h = rgb2hsi(fc);
>> H = h(:, :, 1);
>> S = h(:, :, 2);
>> I = h(:, :, 3);
```

سپس، مولفه شدت را با استفاده از همان فیلتر به اندازه 25×25 پیکسل، فیلتر می‌کنیم. فیلتر میانگین، برای تولید ماتی زیاد، به اندازه کافی بزرگ بود. فیلتری با این اندازه انتخاب شد تا تفاوت بین هموارسازی در فضای RGB و تلاش برای دستیابی به نتیجه‌ای مشابه با استفاده از فقط مولفه شدت تصویر پس از تبدیل به HSI ، تشریح شود. شکل ۲۴-۷ (ب) با دستورات زیر به دست آمد:

```
>> w = fspecial('average', 25);
>> I_filtered = imfilter(I, w, 'replicate');
>> h = cat(3, H, S, I_filtered);
>> f = hsi2rgb(h); % Back to RGB for comparison.
>> imshow(f);
```

روشن است که، نتایج فیلترشده کاملاً متفاوت هستند. برای مثال، علاوه‌بر این‌که تصویر کمتر مات می‌شود، به مرز سبز در بخش بالایی گل در شکل ۲۴-۷ (ب) توجه کنید. علتش این‌است‌که مولفه‌های پرده رنگ و اشباع تغییر کردند درحالی‌که تغییرپذیری مقادیر مولفه‌های شدت، با فرآیند هموارسازی، به شدت کاهش یافت. یک نکته‌ی منطقی این‌است‌که هر سه مولفه HSI با استفاده از یک فیلتر هموار شود. اما، این کار موجب تغییر رابطه‌ی نسبی بین مقادیر پرده رنگ و اشباع می‌شود و نتایج بدتری تولید می‌کند (مانند شکل ۲۴-۷ (پ)). مخصوصاً مشاهده کنید که مرز سبز حول گل‌ها، در این تصویر روشن‌تر است. این اثر، همچنین در حول مرزهای ناحیه زرد مرکزی کاملاً مشهود است.

به طور کلی، هر چه اندازه نقاب کاهش می‌یابد، تفاوت‌های به دست آمده در هنگام فیلترینگ تصاویر مولفه RGB ، و مولفه‌ی شدت تصویر معادلِ HSI نیز کاهش می‌یابد. ■

شکل ۷-۲۴ (الف) تصویر هموارشده‌ی RGB که با هموارسازی جداگانه‌ی صفحات تصویر R ، G و B به دست آمد. (ب) نتیجه‌ی هموارسازی فقط مولفه شدت تصویر معادل HSI . (پ) نتیجه‌ی هموارسازی یکسانِ هر سه مولفه HSI .

پ ب الف

۷-۵-۲ تیزکردن تصویر رنگی

تیزکردن تصویر رنگی RGB با فیلتر مکانی خطی، از همان رویه بخش قبل پیروی می‌کند، ولی به جای آن از فیلتر تیزکردن استفاده می‌شود. در این بخش، تیزکردن تصویر را با استفاده از لاپلاسین (بخش ۱-۵-۳) در نظر می‌گیریم. از تحلیل بردار می‌دانیم که لاپلاسین بردار، به عنوان برداری تعریف می‌شود که مولفه‌های آن برابر با لاپلاسین هر یک از مولفه‌های اسکالر بردار ورودی است. در سیستم رنگ RGB ، لاپلاسین بردار **c** که در بخش ۳-۷ معرفی شد، برابر است با:

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

که همانند بخش قبل، می‌گوید که می‌توان لاپلاسین تصویر تمام‌رنگی را با محاسبه لاپلاسین هر تصویر مولفه به طور جداگانه، محاسبه کرد.

مثال ۱۱-۷ : تیزکردن تصویر رنگی.

شکل ۷-۲۵ (الف) نسخه‌ی مات تصویر شکل ۷-۲۲ را نشان می‌دهد (fb)، که با استفاده از فیلتر میانگین 5×5 به دست آمد. برای تیزکردن این تصویر، از نقاب فیلتر لاپلاسین استفاده کردیم:

```
>> lapmask = [1 1 1; 1 -8 1; 1 1 1];
```

آنگاه، تصویر ارتقایافته محاسبه و نمایش داده شد:

```
>> fb = tofloat(fb);
>> fen = fb - imfilter(fb, lapmask, 'replicate');
>> imshow(fen)
```

همانند بخش قبل، توجه کنید که تصویر RGB مستقیماً با استفاده از imfilter فیلتر شد. شکل ۷-۲۵ (ب) نتیجه را نشان می‌دهد. به افزایش چشمگیر در تیزی ویژگی‌هایی مثل قطره‌های آب، رگبرگ‌ها در برگ، مراکز زرد گل‌ها، و گیاهان سبز در پیش‌زمینه توجه کنید. ■

ب الف **شکل ۲۵-۷** (الف) تصویر ماتشده. (ب) تصویری که با لاپلاسین تغییر کرده است.

۷-۶ کارکردن مستقیم در فضای بردار RGB

همان‌طور که در بخش ۳-۷ گفته شد، مواردی وجود دارد که در آن‌ها، فرآیندهای مبتنی بر هر یک از صفحات، معادل کارکردن مستقیم در فضای بردار RGB نیست. این نکته در این بخش تشریح شده است، که پردازش بردار را با درنظرگرفتن دو کاربرد مهم در پردازش تصویر رنگی، شرح می‌دهیم: تشخیص لبه رنگ و بخش‌بندی ناحیه.

۷-۶-۱ تشخیص لبه رنگ با استفاده از گرادیان

گرادیان تابع دوبُعدی $f(x, y)$ به عنوان یک بردار تعریف می‌شود:

$$\nabla \mathbf{f} = \begin{bmatrix} g_x \\ g_y \end{bmatrix} = \begin{bmatrix} \dfrac{\partial f}{\partial x} \\ \dfrac{\partial f}{\partial y} \end{bmatrix}$$

بزرگی این بردار به صورت زیر است:

$$\nabla f = \text{mag}(\nabla \mathbf{f}) = \left[g_x^2 + g_y^2 \right]^{1/2}$$
$$= \left[(\partial f/\partial x)^2 + (\partial f/\partial y)^2 \right]^{1/2}$$

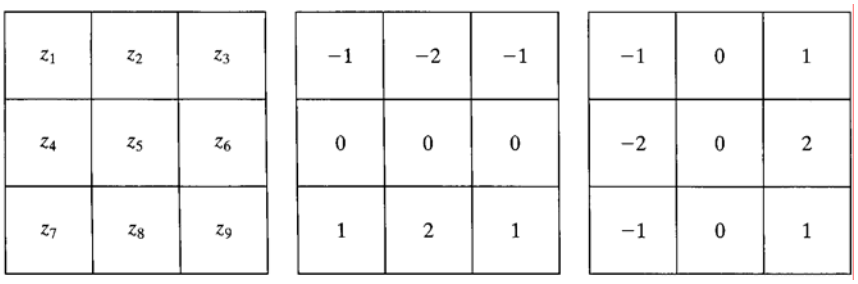
اغلب، این کمیت توسط مقادیر مطلق زیر تخمین زده می‌شود:

$$\nabla f \approx |g_x| + |g_y|$$

این تخمین، از محاسبات مربع و ریشه دوم اجتناب می‌کند، ولی هنوز به صورت مشتق عمل می‌کند. معمولاً، بزرگی گرادیان را همان "گرادیان" می‌نامیم.

خاصیت اساسی بردار گرادیان این‌است‌که به جهت نرخ ماکزیمم تغییر f در مختصات (x, y) اشاره می‌کند. زاویه‌ای که در آن، این نرخ ماکزیمم تغییر می‌کند، به صورت زیر است:

$$\alpha(x, y) = \tan^{-1}\left[\frac{g_y}{g_x} \right]$$



شکل ۲۶-۷ (الف) یک همسایگی کوچک. (ب) و (پ) نقاب‌های سوبل که برای محاسبه گرادیان در جهت‌های x (عمودی) و y (افقی)، نسبت به نقطه‌ی مرکزی همسایگی، محاسبه شدند.
پ ب الف

خوب است که مشتق، با تفاضل‌های مقادیر مقیاس خاکستری روی همسایگی‌های کوچک در تصویر، تخمین زده شود. شکل ۲۶-۷ (الف) یک همسایگی به اندازه 3×3 را نشان می‌دهد، که z ها مقادیر شدت هستند. تخمینی از مشتق‌های جزیی در جهت x (عمودی) در نقطه مرکزی ناحیه، با تفاضل زیر محاسبه می‌شود:

$$g_x = (z_7 + 2z_8 + z_9) - (z_1 + 2z_2 + z_3)$$

به طور مشابه، مشتق در جهت y با تفاضل زیر تخمین زده می‌شود:

$$g_y = (z_3 + 2z_6 + z_9) - (z_1 + 2z_4 + z_7)$$

این دو کمیت در تمام نقاط تصویر، به آسانی با استفاده از فیلترینگ جداگانه‌ی تصویر به کمک دو نقاب نشان‌داده‌شده در شکل‌های ۲۶-۷ (ب) و (پ)، قابل محاسبه هستند. سپس، تخمینی از تصویر گرادیان متناظر، به وسیله مجموع دو تصویر فیلترشده به دست می‌آید. نقاب‌های بحث‌شده، نقاب‌های سوبل (Sobel) هستند که در جدول ۵-۳ آمده‌اند، و در نتیجه می‌توانند با استفاده از تابع fspecial تولید شوند.

گرادیانی که به این روش محاسبه می‌شود، یکی از پراستفاده‌ترین روش‌ها برای تشخیص لبه‌ی تصاویر مقیاس خاکستری است که در فصل ۱۱ بحث می‌شود. فعلاً علاقه‌مند به محاسبه گرادیان در فضای رنگ RGB هستیم. اما، این روش، در فضای دوبُعدی قابل اِعمال است، ولی به ابعاد بالاتر بسط داده نمی‌شود. تنها روش اِعمال آن به تصاویر RGB ، محاسبه گرادیان هر تصویر رنگیِ مولفه، و سپس ترکیب نتایج است. متأسفانه، همان‌طور که در ادامه‌ی این بخش خواهید دید، مثل محاسبه مستقیم لبه‌ها در فضای رنگ RGB نیست. سپس، مسئله این‌است‌که گرادیان بردار**c** تعریف شود. آنچه که در زیر آمده است، یکی از چندین راه مختلف است که در آن، مفهوم گرادیان می‌تواند به توابع برداری بسط داده شود.

فرض کنید **r** ، **g** و **b** بردارهای واحدی در امتداد کادرهای R ، G و B مربوط به فضای رنگ RGB (شکل ۲-۷ را ببینید) باشند، و بردارهای زیر را تعریف می‌کنیم:

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

و

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

فرض کنید کمیت‌های g_{xx} ، g_{yy} ، g_{xy} برحسب حاصلضرب نقاط (درونی) این بردارها تعریف شوند:

$$g_{xx} = \mathbf{u} \bullet \mathbf{u} = \mathbf{u}^T\mathbf{u} = \left|\frac{\partial R}{\partial x}\right|^2 + \left|\frac{\partial G}{\partial x}\right|^2 + \left|\frac{\partial B}{\partial x}\right|^2$$

$$g_{yy} = \mathbf{v} \bullet \mathbf{v} = \mathbf{v}^T\mathbf{v} = \left|\frac{\partial R}{\partial y}\right|^2 + \left|\frac{\partial G}{\partial y}\right|^2 + \left|\frac{\partial B}{\partial y}\right|^2$$

و

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

به یاد داشته باشید که R ، G و B و در نتیجه g ها توابعی از x و y هستند. با این نمادگذاری، می‌توان نشان داد که جهت نرخ ماکزیمم تغییر $\mathbf{c}(x, y)$ به عنوان تابع (x, y) با زاویه زیر مشخص می‌گردد:

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

و مقدار نرخ تغییر در این جهت‌ها، با عناصر $\theta(x, y)$ ، به صورت زیر تعیین می‌شود:

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

آرایه‌های $\theta(x, y)$ و $F_\theta(x, y)$ تصاویری با اندازه‌ی ورودی هستند. عناصر $\theta(x, y)$ زاویه‌هایی در هر نقطه هستند که گرادیان محاسبه می‌شود، و $F_\theta(x, y)$ تصویر گرادیان است.

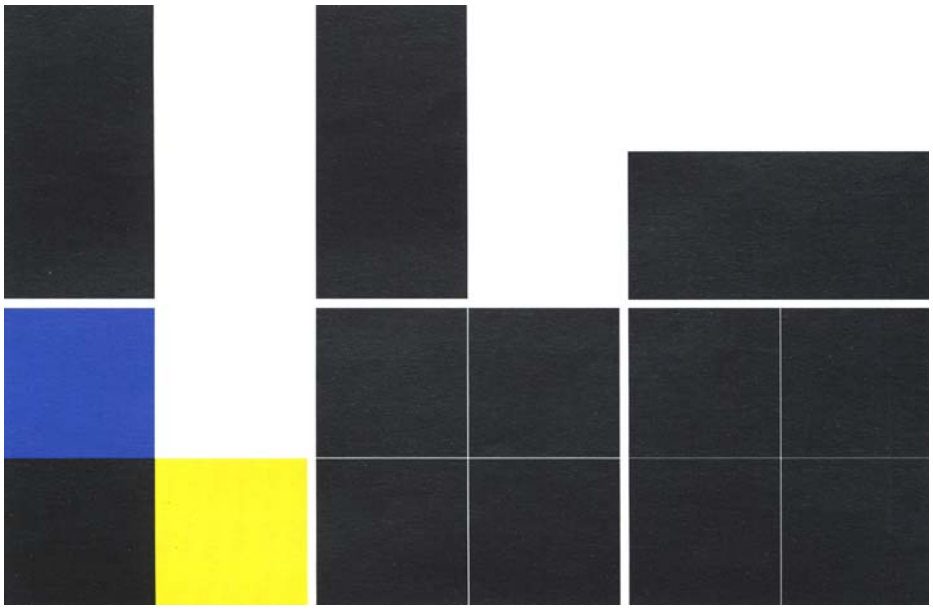
چون $\tan(\alpha) = \tan(\alpha \pm \pi)$ ، اگر θ_0 جواب معادله آرک تانژانت قبلی باشد، $\theta_0 \pm \pi/2$ نیز هست. علاوه‌براین، $F_\theta(x, y) = F_{\theta+\pi}(x, y)$ ، و در نتیجه لازم است F فقط برای مقادیر θ در فاصله‌ی نیمه‌باز $[0, \pi]$ محاسبه شود. این حقیقت که معادله آرک تانژانت، دو مقدار با فاصله‌ی 90° را فراهم می‌سازد، به معنای این‌است‌که این معادله برای هر نقطه (x, y) ، جفتی از جهت‌های عمود بر هم را تولید می‌کند. F در امتداد یکی از آن جهت‌ها ماکزیمم است، و در امتداد دیگری می‌نیمم است. نتیجه‌ی نهایی، با انتخاب ماکزیمم در هر نقطه تولید می‌شود. به دست آوردن این نتایج، بسیار طولانی است و با توضیح آن در این‌جا، چیز زیادی به دست نمی‌آوریم.

تابع زیر، گرادیان رنگ را برای تصاویر RGB پیاده‌سازی می‌کند (کد آن در پیوست ۳ آمده است):

colorgrad

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

که f یک تصویر RGB و T یک آستانه انتخابی در بازه‌ی [0, 1] است (پیش‌فرض صفر است). VG گرادیان بردار RGB مربوط به $F_\theta(x, y)$ است. A تصویر زاویه‌ای $\theta(x, y)$ برحسب رادیان است و PPG تصویر گرادیانی است که با جمع‌کردن تصاویر گرادیان دوبُعدی هر یک از صفحات رنگ به دست آمد. تمام مشتق‌های لازم برای پیاده‌سازی معادلات قبلی، در تابع colorgrad ، با استفاده از عملگرهای سوبل پیاده‌سازی شدند. خروجی‌های VG و PPG به بازه‌ی [0, 1] نرمال شدند و طوری آستانه‌گیری شدند که برای مقادیر کمتر یا مساوی T ، VG(x, y) = 0 و در بقیه موارد VG(x, y) = VG(x, y) . توضیحات مشابهی برای PPG درست است.



شکل ۲۷-۷ (الف) تا (پ) تصاویر مولفه RGB . (ت) تصویر رنگی متناظر. (ث) گرادیانی که مستقیماً در فضای بردار RGB محاسبه شد. (ج) گرادیان مرکبی که با محاسبه گرادیان دوبُعدی هر یک از تصاویر مولفه به طور جداگانه و با جمع‌کردن نتایج، به دست آمد.

پ ب الف
ج ث ت

مثال ۱۲-۷ : تشخیص لبه RGB با استفاده از تابع colorgrad .

شکل‌های ۲۷-۷ (الف) تا (پ) سه تصویر تک‌رنگ را نشان می‌دهد که وقتی به عنوان صفحات RGB استفاده شدند، تصویر رنگی شکل ۲۷-۷ (ت) را تولید کردند. اهداف این مثال عبارتنداز: تشریح استفاده از تابع colorgrad ، و (۲) نشان دهیم که محاسبه گرادیان تصویر رنگی به وسیله ترکیب گرادیان‌های هر یک از صفحات رنگ آن، کاملاً متفاوت از محاسبه مستقیم گرادیان در فضای بردار RGB با استفاده از روش شرح داده شده است.

فرض کنید f تصویر RGB شکل ۲۷-۷ (ت) را نشان می‌دهد. دستور زیر را درنظربگیرید:

```
>> [VG, A, PPG] = colorgrad(f);
```

این دستور، تصاویر VG و PPG را در شکل‌های ۲۷-۷ (ت) و (ج) تولید کرده است. مهمترین تفاوت بین این دو نتیجه این‌است‌که لبه افقی در شکل ۲۷-۷ (ج) چقدر ضعیف‌تر از لبه متناظر در شکل ۲۷-۷ (ت) است. علتش ساده است: گرادیان‌های صفحات قرمز و سبز (شکل‌های ۲۷-۷ (الف) و (ب)) دو لبه عمودی را تولید می‌کند، درحالی‌که گرادیان صفحه‌ی آبی، یک یال افقی را تولید می‌نماید. جمع‌کردن این سه گرادیان برای ایجاد PPG ، یک یال عمودی ایجاد می‌کند که شدت آن دو برابر یال افقی است.

از طرف دیگر، وقتی گرادیان تصویر رنگی مستقیماً در فضای بردار محاسبه می‌شود (شکل ۲۷-۷ (ت))، نسبت مقادیر لبه‌های عمودی و افقی، به جای 2 ، برابر با $\sqrt{2}$ است. علتش ساده است. با مراجعه به مکعب رنگ در شکل ۲-۷ (الف) و تصویر شکل ۲۷-۷ (ت)، می‌بینیم که لبه عمودی در تصویر رنگی، بین مربع آبی و سفید و مربع سیاه و زرد است. فاصله‌ی بین این رنگ‌ها در مکعب رنگ، $\sqrt{2}$ است اما فاصله بین سیاه و آبی و

زرد و سفید (لبه افقی) فقط ۱ است. بنابراین، نسبت تفاضل‌های عمودی و افقی برابر با $\sqrt{2}$ است. اگر دقت لبه مهم باشد، و مخصوصاً وقتی که از آستانه استفاده شود، آنگاه تفاوت بین این دو روش چشمگیر است. برای مثال، اگر از آستانه 0.6 استفاده می‌کردیم، خط افقی در شکل ۲۷–۷ (ج) از بین می‌رفت.

اگر دقت در تشخیص لبه چندان مهم نباشد، دو روش بحث‌شده، معمولاً نتایج قابل مقایسه‌ای تولید می‌کنند. برای مثال، شکل‌های ۲۸–۷ (ب) و (پ) مشابه شکل‌های ۲۷–۷ (ت) و (ج) هستند. آن‌ها با اِعمال تابع colorgrad به تصویر شکل ۲۸–۷ (الف) به دست آمدند. شکل ۲۸–۷ (ت) تفاضل دو تصویر گرادیانِ است که به بازه‌ی [0, 1] نرمال شده است. حداکثرِ تفاضل مطلق بین این دو تصویر 0.2 است، که به 51 سطح خاکستری در بازه‌ی ۸ بیتی [0, 255] تبدیل می‌شود. اما، این دو تصویر گرادیان، در نمای بصری به هم نزدیک هستند، به طوری که شکل ۲۸–۷ (ب) در بعضی از مکان‌ها روشن‌تر است. بنابراین، برای این نوع تحلیل، روش ساده‌تر، محاسبه گرادیان هر مولفه به طور جداگانه است. در وضعیت‌های دیگری که دقت مهم است، روش برداری ضروری است. ■

شکل ۲۸–۷ (الف) تصویر RGB . (ب) گرادیان محاسبه‌شده در فضای بردار RGB . (پ) گرادیان محاسبه‌شده همانند شکل ۲۷–۶ (ج). (ت) تفاضل مطلق بین (ب) و (پ)، که به بازه‌ی [0, 1] مقیاس‌بندی شد.

۷-۶-۲ بخش‌بندی تصوی در فضای بردار RGB

بخش‌بندی، فرآیندی است که تصویر را به چند ناحیه تقسیم می‌کند. گرچه بخش‌بندی موضوع فصل ۱۱ است، در این‌جا بخش‌بندی ناحیه رنگ را به طور مختصر بحث می‌کنیم تا پیوستگی مطالب حفظ شود.

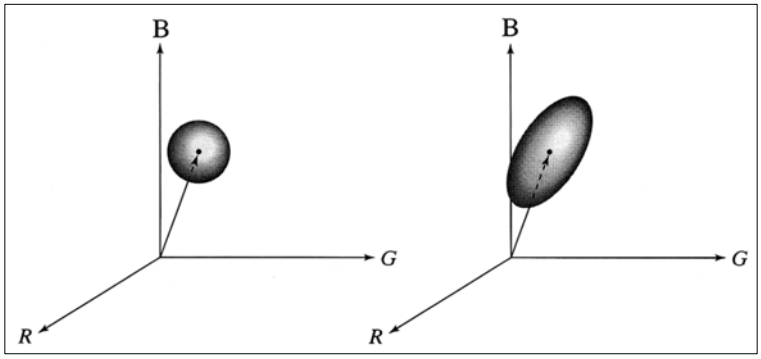
بخش‌بندی ناحیه رنگ با استفاده از بردارهای رنگ RGB ، ساده است. فرض کنید هدف، بخش‌بندی اشیایی از بازه‌ی رنگ معین در یک تصویر RGB است. با توجه به مجموعه‌ای از نقاط رنگی نمونه به نمایندگی رنگ موردنظر، برآوردی از "میانگین" یا "میانه‌ی" رنگی را به دست می‌آوریم که می‌خواهیم بخش‌بندی کنیم. فرض کنید این رنگ میانگین با بردار RGB به نام $\mathbf{m}$ نشان داده شود. هدف بخش‌بندی، دسته‌بندی هر پیکسل RGB در یک تصویر است، به طوری که، دارای رنگی در بازه‌ی مشخص‌شده هست یا خیر. به منظور انجام این مقایسه، لازم است معیاری از شباهت را داشته باشیم. یکی از ساده‌ترین معیارها، فاصله‌ی اقلیدسی است. فرض کنید $\mathbf{z}$ نقطه دلخواهی در فضای سه بعدی RGB باشد. می‌گوییم $\mathbf{z}$ شبیه $\mathbf{m}$ است اگر فاصله بین آن‌ها کمتر از آستانه‌ی مشخص‌شده، یعنی T باشد. فاصله‌ی اقلیدسی بین $\mathbf{z}$ و $\mathbf{m}$ به صورت زیر تعیین می‌شود:

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

که $\|\cdot\|$ نُرم آرگومان است، و زیرنویس‌های R ، G و B مولفه‌های RGB مربوط به بردارهای $\mathbf{z}$ و $\mathbf{m}$ را نشان می‌دهند. مکان هندسی نقاط به طوری که $D(\mathbf{z},\mathbf{m}) \le T$ یک کره پُر با شعاع T است، که در شکل ۲۹-۷ (الف) شرح داده شده است. بنا به تعریف، نقاط موجود در داخل یا سطح کره، معیار رنگ مشخص‌شده را برآورده می‌کند. نقاط خارج از کره، این معیار را برآورده نمی‌کند. با کدکردن این دو مجموعه از نقاط در تصویر، مثلاً با سیاه و سفید، یک تصویر دودویی بخش‌بندی‌شده تولید می‌شود.

تعمیم مفید معادله قبلی، یک معیار فاصله به شکل زیر است:

$$D(\mathbf{z},\mathbf{m}) = \left[(\mathbf{z}-\mathbf{m})^T\mathbf{C}^{-1}(\mathbf{z}-\mathbf{m})\right]^{1/2}$$



ب الف **شکل ۲۹-۷** دو روش برای احاطه‌کردن داده‌ها در فضای بردار RGB به منظور بخش‌بندی.

که **C** ماتریس کوواریانس نمونه‌های نماینده‌ی رنگی است که می‌خواهیم بخش‌بندی کنیم. این فاصله را *فاصله ماهالانوبیس*[1] گویند. مکان هندسی نقاط مثل $D(\mathbf{z},\mathbf{m}) \le T$، یک بدنه بیضوی پُر (شکل ۲۹–۷ (ب)) با این خاصیت مهم را توصیف می‌کند که محورهای اصلی آن در جهتی که حداکثر داده‌ها پخش شدند، قرار دارند. وقتی $\mathbf{C} = \mathbf{I}$، که ماتریس همانی است، فاصله ماهالانوبیس، به فاصله اقلیدسی تبدیل می‌شود. بخش‌بندی همانند پاراگراف قبلی انجام می‌شود، با این تفاوت که اکنون داده‌ها به جای بدنه کروی، در بدنه بیضوی قرار دارند.

بخش‌بندی به روشی که شرح آن گذشت، با تابع colorseg انجام می‌شود (کد آن در پیوست ۳ آمده است)، که به صورت زیر به کار می‌رود:

colorseg

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

که method برابر با 'euclidean' یا 'mahalanobis' است، f تصویر رنگی RGB است که باید بخش‌بندی شود، و T آستانه‌ای است که شرح آن گذشت. اگر 'euclidean' انتخاب شود، پارامتر ورودی برابر با m ، و اگر 'mahalanobis' انتخاب شود، برابر با m یا C است. پارامتر m برابر با میانگین، یعنی **m** ، و پارامتر C برابر با ماتریس کوواریانس یعنی **C** است. خروجی **S** ، یک تصویر دو سطحی (به اندازه تصویر اصلی) است که در نقاطی که آستانه را رعایت نمی‌کنند برابر با صفر و در آن‌هایی که آستانه را رعایت می‌کنند، برابر با یک است. یک‌ها، ناحیه‌هایی را نشان می‌دهند که از f بر اساس محتوای رنگ بخش‌بندی شدند.

مثال ۱۳–۷: بخش‌بندی تصویر رنگی RGB.

شکل ۳۰–۷ (الف) یک تصویر شبه‌رنگی از ناحیه‌ای از یو، یکی از قمرهای سیاره مشتری است. در این تصویر، رنگ‌های قرمز نشان‌دهنده‌ی موادی هستند که به تازگی از یک آتشفشان فعال به بیرون پرتاب شده‌اند و مواد زرد اطراف، نشت‌های گوگردی قدیمی‌ترند. این تصویر، بخش‌بندی ناحیه‌ی قرمزرنگ را با استفاده از هر دو گزینه‌ی موجود در تابع colorseg ، برای مقایسه نشان می‌دهد.

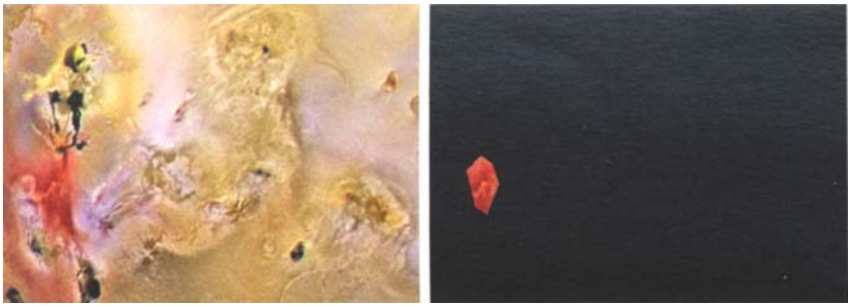
ابتدا نمونه‌هایی را به دست می‌آوریم که بازه‌ای از رنگ‌ها را نمایش می‌دهد که باید بخش‌بندی شود. یک روش ساده برای به دست آوردن ناحیه موردنظر (ROI)، استفاده از تابع roipoly است که در بخش ۴–۲–۵ شرح داده شد (مثل ۲–۱۳)، که یک نقاب از ناحیه‌ای را تولید می‌کند که به طور محاوره‌ای انتخاب شده است. بنابراین، اگر f تصویر شکل ۳۰–۷ (الف) باشد، ناحیه شکل ۳۰–۷ (ب) با دستورات زیر به دست می‌آید:

```
>> mask = roipoly(f); % Select region interactively.
>> red = immultiply(mask, f(:, :, 1));
>> green = immultiply(mask, f(:, :, 2));
>> blue = immultiply(mask, f(:, :, 3));
>> g = cat(3, red, green, blue);
>> figure, imshow(g);
```

که mask یک تصویر دودویی (هم‌اندازه‌ی f) است که با استفاده از roipoly به دست آمد.

سپس، بردار میانگین و ماتریس کوواریانس نقاط در ROI را به دست می‌آوریم، اما ابتدا مختصات نقاط در ROI باید استخراج شوند:

1. mahalanobis distance



ب الف **شکل ۳۰-۷** (الف) تصویر شبه رنگی از یو، قمر مشتری. (ب) ناحیه موردنظر که با تابع roipoly استخراج شده است.

```
>> [M, N, K] = size(g);
>> I = reshape(g, M * N, 3);
>> idx = find(mask);
>> I = double(I(idx, 1:3));
>> [C, m] = covmatrix(I);
```

دستور دوم، پیکسل‌های رنگ در g را به صورت سطرهای I می‌چیند، و دستور سوم اندیس‌های سطر پیکسل ماتریس را می‌یابد که سیاه نیستند. این‌ها پیکسل‌هایی غیر از پس‌زمینه‌ی تصویر نقاب‌زده در شکل ۳۰-۷ (ب) هستند.

آخرین محاسبات مقدماتی، تعیین مقداری برای T است. یک نقطه‌ی شروع خوب این‌است‌که فرض کنیم T مضربی از انحراف استاندارد یکی از مولفه‌های رنگ باشد. قطر اصلی C شامل واریانس‌های مولفه‌های RGB است، و در نتیجه کاری که باید انجام دهیم، استخراج عناصر و محاسبه ریشه‌های مربع آن‌ها است:

```
>> d = diag(C);
>> sd = sqrt(d)'
    22.0643   24.2442   16.1806
```

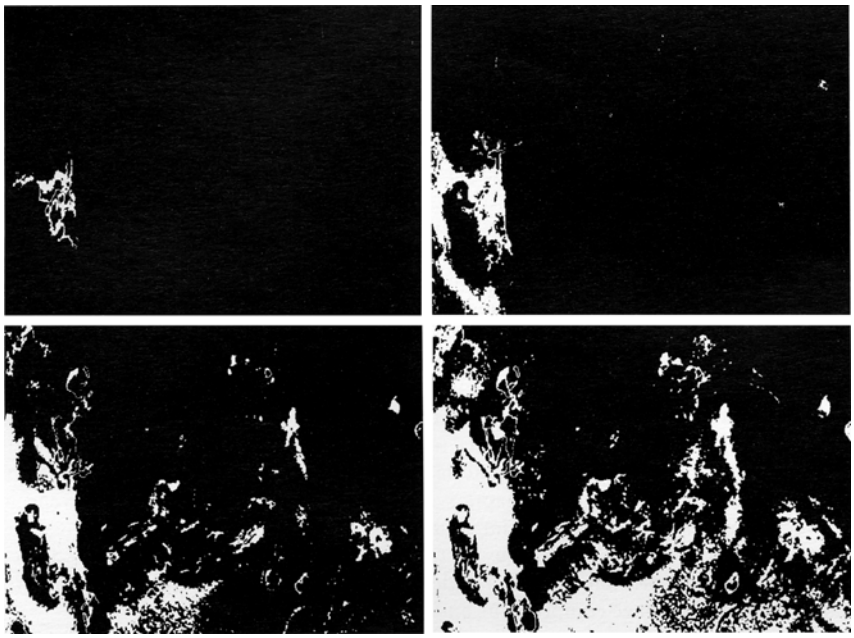
اولین عنصر sd ، انحراف استاندارد مولفه قرمز پیکسل‌های رنگ در ROI است، و برای دو مولفه دیگر نیز همین‌طور است.

اکنون تصویر را با استفاده از مقادیر T ، مساوی با مضرب‌های ۲۵ بخش‌بندی می‌کنیم، که تقریبی از بزرگ‌ترین انحراف استاندارد است: T = 25, 50, 75, 100 . برای گزینه ’euclidean‘ با T = 25 ، به صورت زیر عمل می‌کنیم:

```
>> E25 = colorseg('euclidean', f, 25, m);
```

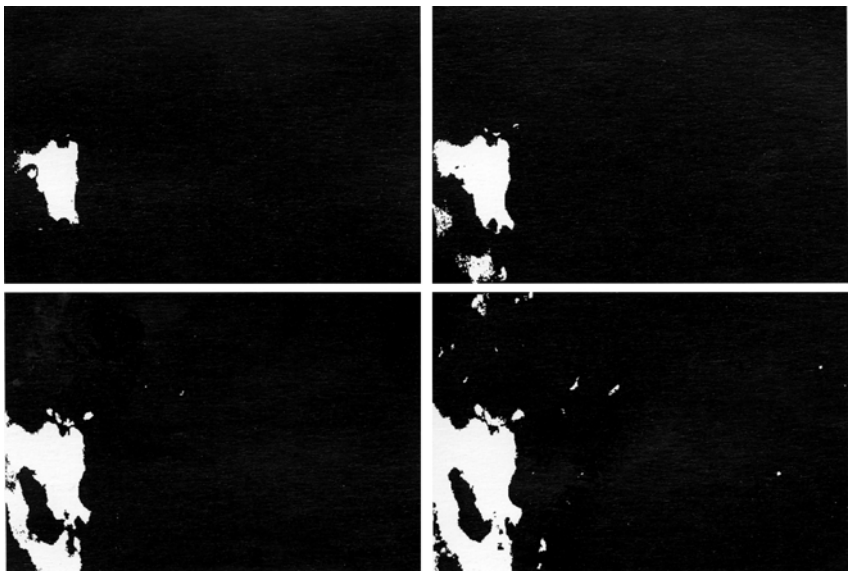
شکل ۳۱-۷ (الف) نتیجه را نشان می‌دهد، و شکل‌های ۳۱-۷ (ب) تا (ت) نتایج بخش‌بندی با T = 50, 75, 100 را نشان می‌دهند. به طور مشابه، شکل‌های ۳۲-۷ (الف) تا (ت) نتایج به دست آمده با استفاده از گزینه ’mahalanobis‘ با همان دنباله از مقادیر آستانه را نشان می‌دهد.

نتایج بامعنا (بسته به این‌که چه چیزی را به عنوان قرمز در شکل ۳۰-۷ (الف) درنظربگیریم)، با گزینه ’euclidean‘ با استفاده از T = 25, 50 به دست آمدند، ولی 75 و 100 بخش‌بندی بیش از اندازه را تولید کردند. از طرف دیگر، نتایج مربوط به گزینه ’mahalanobis‘ با استفاده از همان مقادیر T ، دقیق‌تر بودند، که در شکل ۳۲-۷ آمده‌اند. علتش این‌است‌که پخش داده‌ی رنگ به صورت سه‌بعدی در ROI ، در این مورد، در حالت بیضوی بهتر از حالت کروی است. توجه کنید که در هر دو روش، افزایش T موجب شد سایه‌های ضعیف‌تری از قرمز در ناحیه‌های بخش‌بندی‌شده گنجانده شود، که انتظار نیز همین است. ■



ب الف
ت پ

شکل ۳۱-۷ (الف) تا (ت) بخش‌بندی شکل ۳۰-۷ (الف) با استفاده از گزینه 'euclidean' در تابع colorseg با T = 25, 50, 75, 100 .



ب الف
ت پ

شکل ۳۲-۷ (الف) تا (ت) بخش‌بندی شکل ۳۰-۷ (الف) با استفاده از گزینه 'mahalanobis' در تابع colorseg با T = 25, 50, 75, 100 . با شکل ۳۱-۷ مقایسه کنید.