第一章　图像载体的基本知识

信息隐藏这门学科是一个新兴学科，到目前为止，被认为成熟的信息隐藏算法基本上都是以图像作为载体的。我们学习信息隐藏，也是从图像载体下的信息隐藏出发，然后才能不断深入。只有了解了载体本身的特点，才能进一步学习和运用具体的隐藏算法。本章主要介绍图像的文件格式、图像的类型、图像的颜色模型和图像的存储方式以及利用 MATLAB 对图像进行类型转换和颜色模型转换的有关知识。这些都是我们在今后的实验中涉及到和必须清楚认识的。

简而言之，图像就是用各种观测系统以不同形式和手段观测客观世界而获得的，可以直接或间接作用于人眼而产生视知觉的实体。人类的大部分信息都是从图像中获得的。图像是人们从出生以来体验到的最重要、最丰富、信息量获取最大的对象。就图像本质来说，可以将图像分为两大类：模拟图像和数字图像。一幅二维(2-D) 平面图像可用一个二元函数 I = f(x，y) 来表示。(x，y) 表示 2-D 空间坐标系中一个坐标点的位置，f 则表示相应实际物体在该点的某个性质的度量值，所有点的度量值的有序集合构成图像 I。例如，对于一幅灰度图像，f 表示灰度值，即相应物体在每个坐标点的明暗程度。一般认为，I = f(x，y) 所表示的图像是连续的，如一幅照片、一幅绘画等。为了能用计算机对图像 I 进行处理，则将 f，x，y 的值域从实数域映射到整数域。离散化后的图像就是数字图像。离散化的方法就是从水平和竖直两个方向上同时进行采样。这些采样点称为像素(pixel) 。因此，通常用二维矩阵来表示一幅数字图像，矩阵的各个元素代表一个像素的色彩信息。作为信息隐藏的载体，涉及到的图像都是数字图像。数字图像以其信息量大、处理和传输方便、应用范围广等一系列优点已成为现代信息化社会的重要支柱，是人类获取信息的重要来源和利用信息的重要手段。随着计算机科学技术的飞速发展，数字图像处理技术在近年来得到了快速发展，并得到广泛应用。

1.1　图像的基本类型

1.1.1　图像类型的引入

图形图像文件可以分为两大类：一类为位图(Bitmap) 文件，另一类为矢量(Vector) 文件。前者以点阵形式描述图形图像，后者是以数学方法描述的一种由几何元

素组成的图形图像。位图文件在有足够的文件量的前提下,能真实细腻地反映图像的层次、色彩,缺点是文件体积较大。一般说来,适合描述照片。矢量类图像文件的特点是文件量小,并且能任意缩放而不会改变图像质量,适合描述图形。位图的映射存储模式是将图像的每一个像素点转换为一个数据,并存放在以字节为单位的一、二维矩阵中。当图像是单色时,一个字节可存放 8 个像素点的图像数据;16 色图像每两个像素点用一个字节存储;256 色图像每一个像素点用一个字节存储。以此类推,就能够精准地描述各种不同颜色模式的图像画面。所以位图文件较适合于内容复杂的图像和真实的照片(位图正符合作为信息隐藏载体的最基本要求)。但位图也有缺点:随着分辨率以及颜色数的提高,位图图像所占用的磁盘空间会急剧增大,同时在放大图像的过程中,图像也会变得模糊而失真。

图像离不开色彩。大家都知道,在物理光学中,红、绿、蓝(Red, Green, Blue,即 RGB)被称为光学三原色。大千世界的自然景色丰富多彩,但任何色彩(严格地说是绝大多数色彩)都可以用红、绿、蓝这三种颜色按一定的比例混合而得。例如:

红 + 绿 + 蓝→白色

红 + 绿→黄色

红 + 蓝→品红……

由此,我们可以用一个由 R, G, B 为坐标轴定义的单位立方体来描述这样一个符合视觉理论的颜色模型(图 1.1(c))。坐标原点代表黑色,(1, 1, 1)代表白色,坐标轴上的顶点称为基色(Primitive Colour)点。立方体中的每一种颜色由一个三元组(R, G, B)表示,每一个分量的数值均在[0,1]区间。

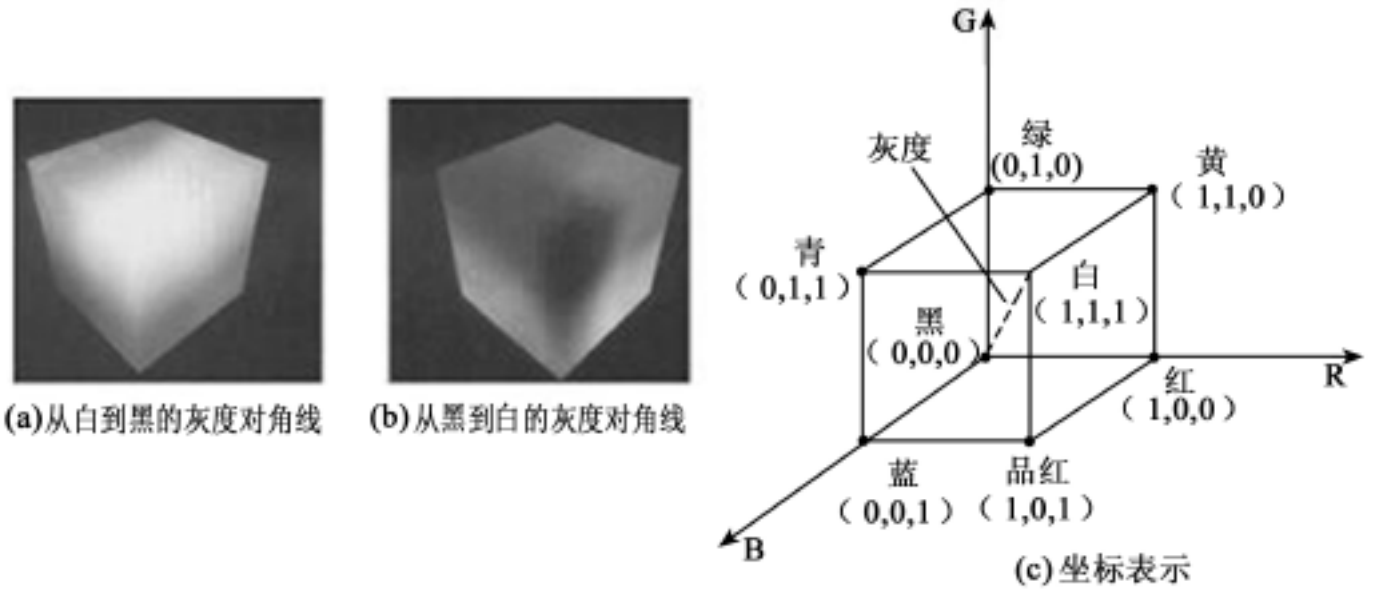

图 1.1　RGB 颜色模型

那么,位图中的颜色是如何在矩阵中体现的呢?我们首先引入一个重要概念:调色板。调色板是包含不同颜色的颜色表,每种颜色以红、绿、蓝三种颜色的组合来表示,图像的每一个像素对应一个数字,而该数字对应调色板中的一种颜色,如某像素

值为 1, 则表示该颜色为调色板的编号为 1 的颜色。调色板的单元个数是与图像的颜色数相一致的, 256 色图像的调色板就有 256 个单元。有些图像(如 RGB 图像) 的每个像素值直接用 R, G, B 三个字节来表示颜色, 不需要单独的调色板。值得注意的是, 对于 16 色或 256 色图像并非全部的图像都采用相同的 16 种或 256 种颜色, 由于调色板中定义的颜色不同, 则不同图像用到的颜色是千差万别的, 所谓 16 色或 256 色图像, 只是表示该幅图像最多只能有 16 种或 256 种颜色。不同的图像有不同的调色板。

一个图像的调色板所含有的色彩个数取决于数字图像的量化方式。将像素点上的灰度值离散为整数, 称之为量化。量化的结果反映了图像容纳的所有颜色数据。量化决定使用多大范围的数值来表示图像采样之后的每一个点, 这个数值范围确定了图像能使用的颜色总数。例如, 以 4 个 bit 存储一个点, 就表示图像只能有 16 种颜色。数值范围越大, 表示图像可以拥有更多的颜色, 自然就可以产生更为精细的图像效果。通常所说的量化等级, 是指每幅图像样本量化后一共可取多少个像素点(离散的数值) 或用多少个二进制数位来表示。量级越高, 图像质量就越高, 存储空间要求就越大。图 1.2 就是我们在本书中将大量使用到的 lenna 图像的二维矩阵。

lenna 图像

Array Editor: woman

File Edit View Web Window Help

Numeric format: shortG　Size: 256 by 256

	1	2	3	4	5	6	7	8	9	10
1	125	79	111	98	111	79	125	98	91	112
2	98	112	91	112	79	111	112	79	112	79
3	98	125	98	111	112	98	79	111	112	111
4	106	98	79	112	79	125	98	94	98	79
5	79	125	98	91	98	98	79	111	112	125
6	98	111	112	98	94	111	112	79	98	79
7	112	79	111	94	98	79	111	112	91	112
8	111	112	98	111	112	91	112	79	98	91
9	106	98	79	112	79	98	111	112	111	112
10	111	112	111	91	112	111	94	111	79	98
11	79	98	125	98	112	91	98	112	111	94
12	111	94	111	98	91	112	111	79	112	111

lenna 图像的二维矩阵

图 1.2　位图的二维矩阵表示

矢量文件只存储图像内容的轮廓部分, 而不存储图像数据的每一点。例如, 对于一个圆形图案, 只要存储圆心的坐标位置和半径长度, 以及圆形边线和内部的颜色即可。该存储方式的缺点是经常耗费大量的时间做一些复杂的分析演算工作, 但图像的缩放不会影响显示精度, 即图像不会失真, 而且图像的存储空间较位图文件要少得多。所以, 向量处理比较适合存储各种图表和工程设计图。

总体来看,位图文件是记录每一个像素的颜色值,再把这些像素点组合成一幅图像,而矢量图文件是依靠保存图形对象的位置、曲线、颜色的算法来生成的。位图占用的存储空间较矢量图要大得多,而矢量图的显示速度较位图慢。

我们针对的对象基本上是位图。前文已述,位图都可以看做一个二维数据矩阵,根据其图像调色板的存在方式及矩阵数值与像素颜色之间的对应关系,我们定义了4种基本的图像类型:二值图像、索引图像、灰度图像、RGB图像。下面一一介绍其性质。

1.1.2 二值图像

二值图像顾名思义就是图像像素只存在0,1两个值,也叫做二进制图像。一个二值图像显然是纯黑白的。每一个像素值将取两个离散值(0或1)中的一个,0表示黑,1表示白。二进制图像能够使用无符号8位整型(uint8)或双精度类型(double)的数组来存储。

图1.3 二进制的lenna图像

uint8类型的数组通常比双精度类型的数组性能更好,因为uint8数组使用的内存要小得多。在MATLAB图像处理工具箱中,任何返回一幅二进制图像的函数都使用uint8逻辑数组存储该图像。图1.3给出了二进制的lenna图像。

1.1.3 索引图像

索引图像是一种把像素值直接作为RGB调色板下标的图像。一幅索引图包含一个数据矩阵data和一个调色板矩阵map,数据矩阵可以是uint8, uint16或双精度类型的,而调色板矩阵则总是一个m×3的双精度类型矩阵(其中,m表示颜色数目),该矩阵的元素都是[0,1]范围内的浮点数。map矩阵的每一行指定一个颜色的红、绿、蓝颜色分量。索引图像可以把像素值直接映射为调色板数值,每一个像素的颜色通过使用data的数值作为map的下标来获得:数值1表示map的第一行,数值2表示map的第二行,依此类推。

图1.4是MATLAB自带的woman信号构成的图像的像素索引矩阵和调色板矩阵。woman图像是一幅典型的索引图像。其图像矩阵大小为256×256,表示有65535个像素点构成。调色板矩阵大小为256×3,表示有256种颜色。我们看到图像索引矩阵的(1,1)单元的内容为125,也就是说这一点像素的颜色就是调色板矩阵的第125行所定义的颜色。可以看到调色板矩阵的第125行为[0.60536 ,0.60536, 0.60536],表示RGB三个分量的比重都比较重且在图像中的地位相同,对照图1.1的RGB色彩模型可以推断出这一点是灰白色的。图1.4左下的woman图像证实了这一推断。

图像矩阵中的数值与调色板的关系依赖于图像矩阵的类型: 如果图像矩阵是双精度类型的, 那么数值 1 将指向调色板的第一行, 数值 2 将指向调色板的第二行, 依此类推; 如果图像矩阵是 uint8 或 uint16 类型的, 那么将产生一个偏移量: 数值 0 表示调色板的第一行, 数值 1 表示调色板的第二行, 依此类推。

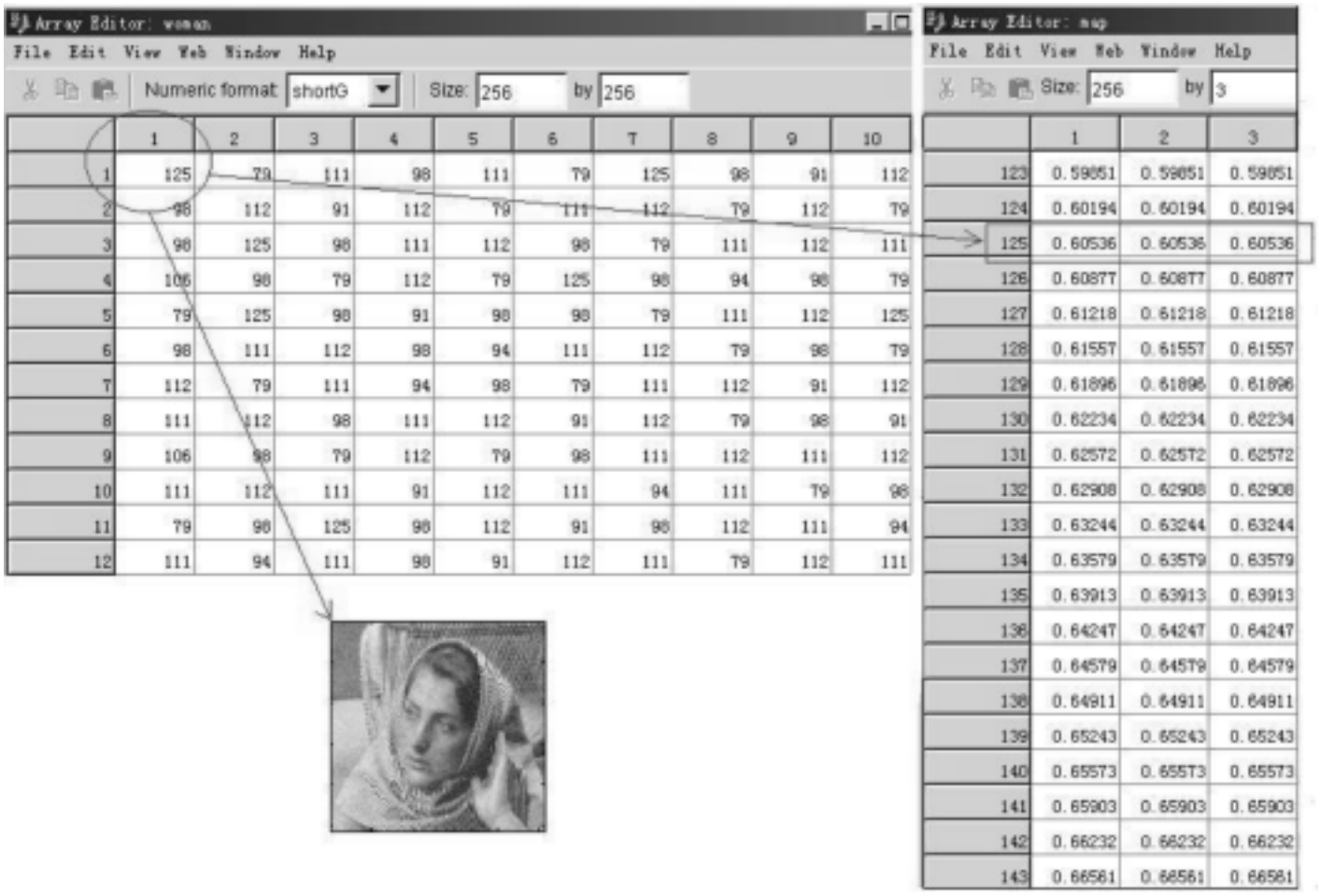

图 1.4　索引图像

在实际操作中应该注意到, 调色板通常应与索引图像存储在一起; 装载图像时, 调色板将和图像数据一同自动装载。在 MATLAB 中以下三条对索引图像的操作语句是我们今后会大量使用的:

读取索引图像: [data, map] = imread(filename, permission) ;

显示索引图像: image(data) , colormap(map) ;

存储索引图像: imwrite(data, map, filename, permission) ;

其中, data 为像素索引矩阵, map 为调色板矩阵, filename 为图像文件路径, permission 为图像文件格式。例如[data, map] = imread('c: \woman. bmp', 'bmp') ;

1.1.4　灰度图像

灰度图像是包含灰度级(亮度) 的图像。灰度就是我们通常说的亮度。与二值图像不同, 灰度图像虽然在感观上给人感觉仍然是“ 黑白 ”的, 但实际上它的像素并

不是纯黑(0)和纯白(1)那么简单,所以相应的其一个像素也绝不是1bit就可以表征的。

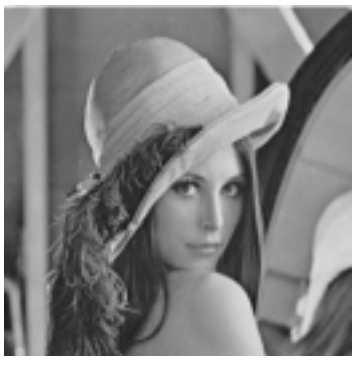

图1.5　灰度lenna图像

在MATLAB中,灰度图像由一个uint8, uint16或双精度类型的数组来描述。灰度图像实际上是一个数据矩阵I,该矩阵的每一个元素对应于图像的一个像素点,元素的数值代表一定范围内的灰度级,通常0代表黑色,1, 255或65535(不同存储类型)代表白色。

数据矩阵I可以是双精度、uint8或uint16类型。由于灰度图像存储时不使用调色板,因而MATLAB将使用一个默认的系统调色板来显示图像。二值图像可以看成是灰度图像的一个特例。联系到后面我们将阐述的YCbCr颜色模型,我们可以发现所谓灰度图像的像素值就是YCbCr中每个像素的亮度分量值。二者与RGB像素有同样的转换关系。图1.5是一幅灰度lenna图像。

1.1.5　RGB图像

需要说明一点的是,RGB图像显然是符合RGB颜色模型的,但不是说只有RGB图像才符合RGB颜色模型,事实上前面我们已经看到,我们一般意义上说的图像都是符合这一颜色模型的。所谓RGB图像仅是一类图像的总称。这类图像不使用单独的调色板,每一个像素的颜色由存储在相应位置的红、绿、蓝颜色分量共同决定。RGB图像是24位图像,红、绿、蓝分量分别占用8位,理论上可以包含16M种不同颜色,由于这种颜色精度能够再现图像的真实色彩,所以又称RGB图像为真彩图像。

在MATLAB中,一幅RGB图像由一个uint8, uint16或双精度类型的m×n×3数组(通常称为RGB数组)来描述,其中,m和n分别表示图像的宽度和高度。

在一个双精度类型的RGB数组中,每一个颜色分量都是一个[0, 1]范围内的数值,颜色分量为(0, 0, 0)的像素将显示为黑色,颜色分量为(1, 1, 1)的像素将显示为白色。每一个像素三个颜色分量都存储在数据数组的第三维中。例如,像素(10, 5)的红、绿、蓝色分量都存储在RGB(10,5,1), RGB(10,5,2), RGB(10,5,3)中。

为了更好地说明在RGB图像中所使用的3个不同颜色分量的作用效果,我们在MATLAB中创建一个简单的RGB图像,该图像包含某一范围内不中断的红、绿、蓝颜色分量。同时,提取每一个颜色分量各创建一幅灰度图像来加以对比,输入命令为:

```
>> RGB = reshape( ones( 64,1) *  reshape( jet( 64) , 1, 192) , [ 64, 64, 3] ) ;
>> R = RGB( : , : , 1) ;
>> G = RGB( : , : , 2) ;
>> B = RGB( : , : , 3) ;
>> subplot( 141) , imshow( R) , title( 红色分量) ;
```

>> subplot(142) , imshow(G) , title(绿色分量) ;

>> subplot(143) , imshow(B) , title(蓝色分量) ;

>> subplot(144) , imshow(RGB) , title(原始图像) ;

四幅图像的显示结果如图 1. 6 所示。

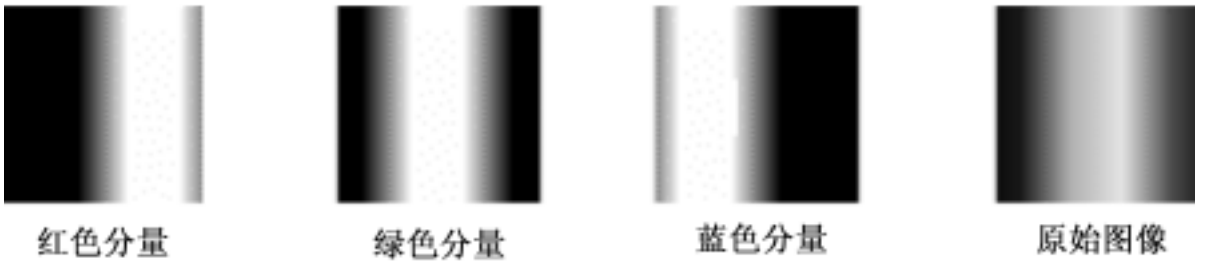

图 1. 6　RGB 颜色色谱的分层显示

注意到图 1. 6 中每一个单独的颜色项对应的灰度图像都包含一个白色区域, 这个白色区域相应于每一个颜色项的最高值。例如, 在红色分量中, 白色代表纯红色数值浓度最高的区域。当红色与绿色或蓝色混合时将会出现灰像素。图像中的黑色区域说明该区域不包含任何红色数值, 即 R = 0。

我们再来看一个将输入的 RGB 图像分层, 并将图像的指定层加强的实验。对应的函数为 rgbanalysis. m, 代码如下:

% 文件名: rgbanalysis. m

% 程序员: 郭迟

% 编写时间: 2004. 2. 8

% 函数功能: 将输入的 RGB 图像分层, 并将图像的指定层加强

% 输入格式举例: [imageRGB, imageR, imageG, imageB, result] = rgbanalysis('c: \ lenna. jpg', 'jpg', 1)

% 参数说明:

% image 输入的原始 RGB 图像的地址

% permission 为图像文件类型

% level 为要加强的层: 1 为 R, 2 为 G, 3 为 B

% imageRGB 为输出的 RGB 图像的 RGB 矩阵

% imageR 为 R 层分量的矩阵

% imageG 为 G 层分量的矩阵

% imageB 为 B 层分量的矩阵

% result 为色彩加强的 RGB 矩阵

function [imageRGB, imageR, imageG, imageB, result] = rgbanalysis (image, permission, level) ;

% 读取图像信息并转换为 double 型

```
imageRGB = imread( image, permission) ;
imageRGB = double( imageRGB) /255;
result = imageRGB;
% 对图像进行分层提取
imageR = imageRGB( : , : , 1) ;
imageG = imageRGB( : , : , 2) ;
imageB = imageRGB( : , : , 3) ;
% 显示结果
subplot( 321) , imshow( imageRGB) , title( '原始图像') ;
subplot( 322) , imshow( imageR) , title( 'R 层灰度图像') ;
subplot( 323) , imshow( imageG) , title( 'G 层灰度图像') ;
subplot( 324) , imshow( imageB) , title( 'B 层灰度图像') ;
% 对相应的层进行颜色加强
if level == 1
    imageR = imageR + 0. 2;
end
if level == 2
    imageG = imageG + 0. 2;
end
if level == 3
    imageB = imageB + 0. 2;
end
result( : , : , 1) = imageR;
result( : , : , 2) = imageG;
result( : , : , 3) = imageB;
% 通过图像写回保存将超出范围的像素值自动调整为最大
imwrite( result, 'temp. jpg', 'jpg') ;
result = imread( 'temp. jpg', 'jpg') ;
% 显示结果
subplot( 325) , imshow( result) , title( '色彩增强的结果') ;
```

图 1. 7 是执行[imageRGB, imageR, imageG, imageB, result] = rgbanalysis('c: \ lenna. jpg', 'jpg', 3) 的图像结果, 图 1. 8 是相应的矩阵表示。可以看到, 对 B 层进行颜色加强后, lenna 图像明显出现了我们常说的"颜色泛蓝"的现象。

最后, 通过 MATLAB 要想在一幅 RGB 图像中获得某个特定像素(x, y) 的颜色, 可以键入如下命令: >> RGB(x, y, :) ;

图 1.7　lenna 的分层显示及蓝色加强

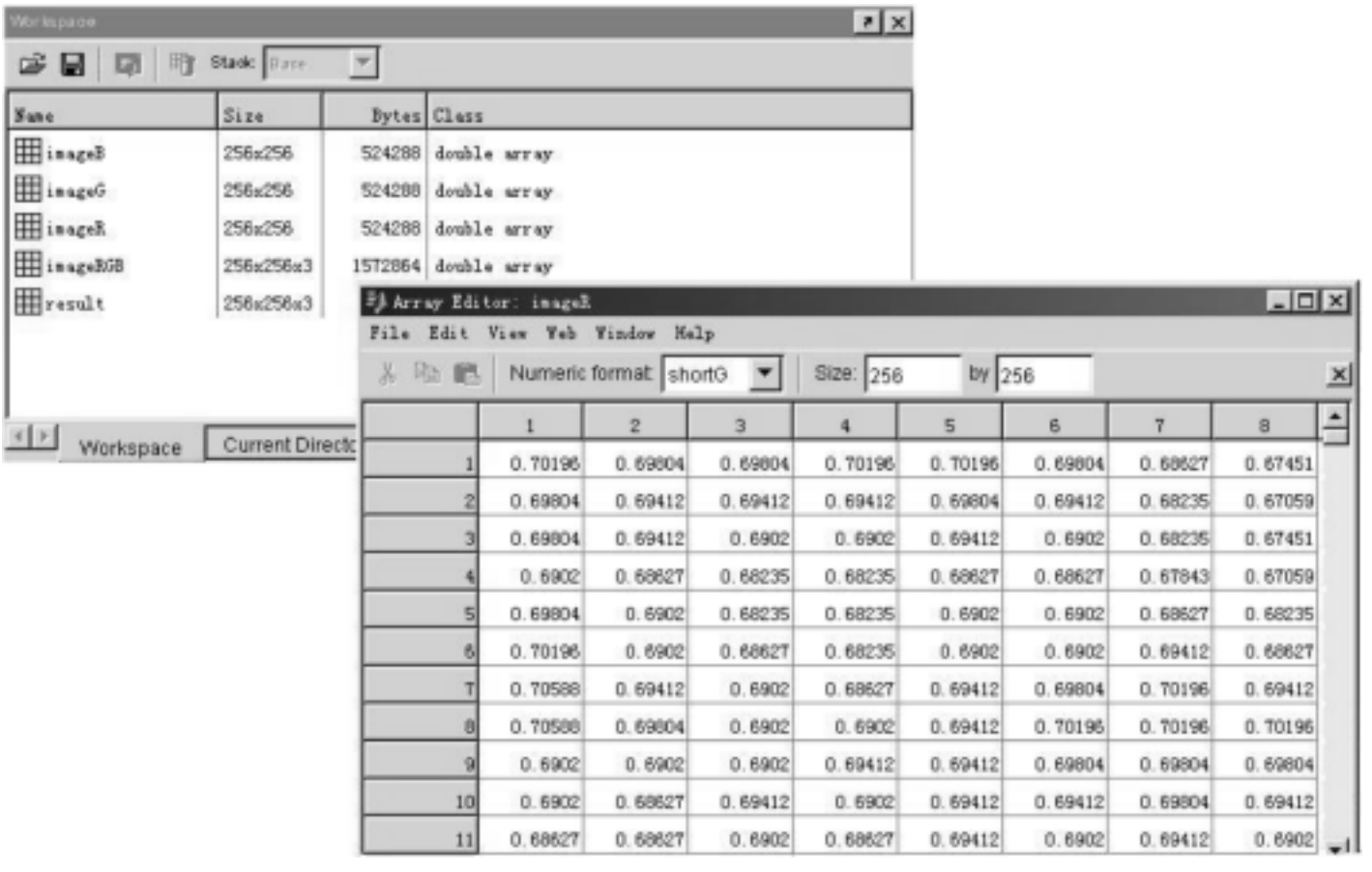

图 1.8　lenna 的图像矩阵信息和 R 层矩阵数值示例

1.2 图像类型的相互转换

有些时候,图像类型的转换是非常有用的。例如,如果用户希望对一幅存储为索引图像的彩色图像进行滤波时,那么应该首先将该图像转换为 RGB 格式,此时再对 RGB 图像使用滤波器,MATLAB 将恰当地滤掉图像中的部分灰度值。如果用户企图对一幅索引图进行直接滤波,那么 MATLAB 只能简单地对索引图像矩阵的下标进行滤波,这样得到的结果将是毫无意义的。再比如在变换域的数字水印算法中,对于索引图像的载体必须将其先转换为 RGB 图像再加水印,否则将破坏载体。在 MATLAB 中,各种图像类型之间的转换关系如图 1.9 所示。其图像处理工具箱中所有的图像类型转换函数如表 1.1 所示。

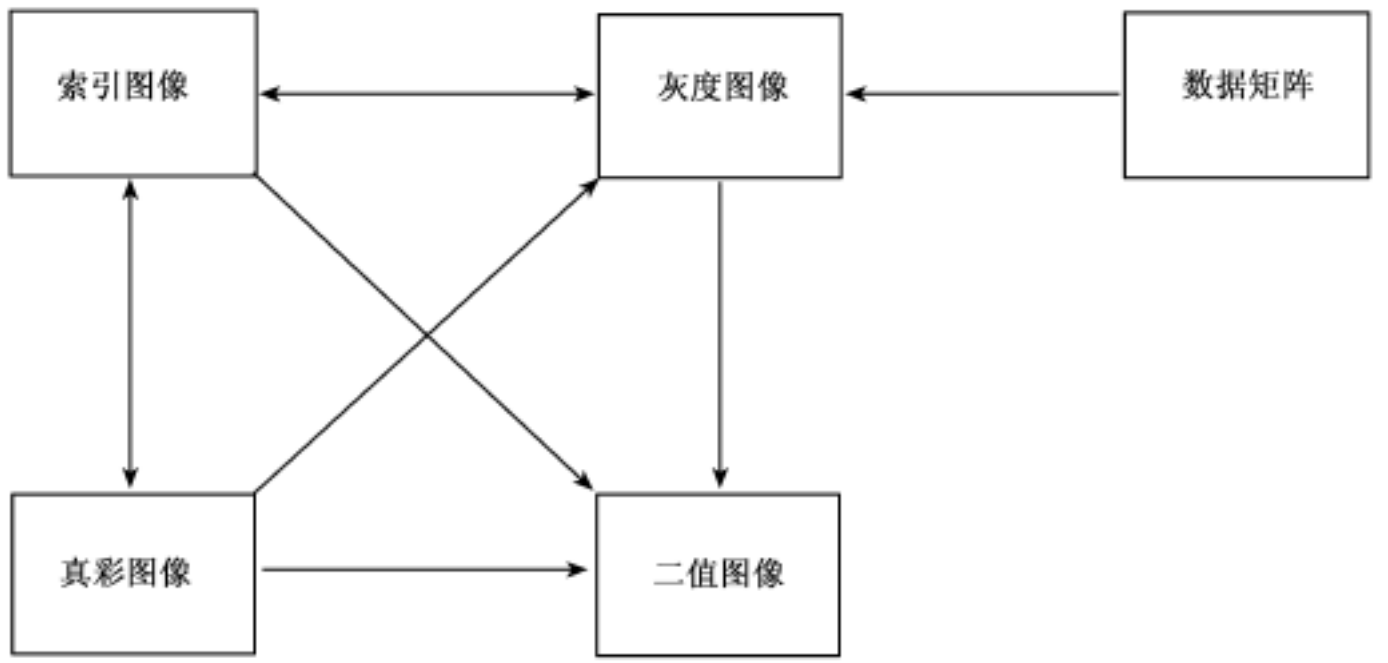

图 1.9 图像类型转换的关系

表 1.1 MATLAB 图像类型转换函数及其功能

函 数	功 能
dither	使用抖动方法,根据灰度图像创建二进制图像或根据 RGB 图像创建索引图像
gay2ind	根据一幅灰度图像创建索引图像
grayslice	使用阈值截取方法,根据一幅灰度图像创建索引图像
im2bw	使用阈值截取方法,根据一幅灰度图像、索引图像或 RGB 图像创建二进制图像
ind2gray	根据一幅索引图像创建一幅灰度图像
ind2rgb	根据一幅索引图像创建一幅 RGB 图像
mat2gray	通过数据缩放,再根据矩阵数据创建一幅灰度图像
rgb2gray	根据一幅 RGB 图像创建一幅灰度图像
rgb2ind	根据一幅 RGB 图像创建一幅索引图像

表 1.1 中几乎所有的函数都具有类似的调用格式: 函数的输入是图像数据矩阵(如果是索引图像, 那么输入参数还包括调色板矩阵) , 返回值是转换后的图像(包括索引图像的调色板) 。

1.2.1　灰度图像的二值化方法

所谓灰度图像的二值化方法实际上解决的就是将灰度图像转换为二值图像这一问题。转换的方法可用伪 C 语言描述为:

```
设 (x, y)G为灰度图像 G 的像素
float threshold;  //定义一个转换阈值
if ((x, y)G >= threshold)
  (x, y)B = 1;
else
  (x, y)B = 0;
```

则图像 B 为 G 的二值转换图

可以发现, 灰度图像二值化的关键因素便是阈值 threshold 的大小。取阈值不同, 得到的转换图像也不尽相同。MATLAB 中函数 im2bw 的输入参数中就包括一个截取阈值。我们先直观地感受一下 threshold 与转换效果的关系。在 MATLAB 中输入:

```
>> rgb = imread('c:\lenna.jpg','jpg');
>> rgb = double(rgb)/255;
>> binary1 = im2bw(rgb, 0.7);
>> binary2 = im2bw(rgb, 0.5);
>> binary3 = im2bw(rgb, 0.4);
>> binary4 = im2bw(rgb, 0.2);
>> subplot(221), imshow(binary1); title('Threshold = 0.7');
>> subplot(222), imshow(binary2); title('Threshold = 0.5');
>> subplot(223), imshow(binary3); title('Threshold = 0.4');
>> subplot(224), imshow(binary4); title('Threshold = 0.2');
```

得到结果如图 1.10 所示。

可以发现, 当转换阈值取 0.4 时, 二值化的效果是最好的。这仅是就一幅 lenna 图像在简单比较后得出的结论, 只是一个个案。那么, 对于普遍情况又是如何确定阈值的呢? 这里, 我们介绍一下最简单的全局阈值法(global threshold method) 。

全局阈值法是与灰度直方图紧密联系的。灰度直方图的横坐标表示图像的灰度, 纵坐标表示该灰度在图像全体像素中出现的频度。图 1.11 是一个简单图像的灰度直方图。

对照发现, 原始图像只有两个灰度, 所以灰度直方图呈现了两个峰值。在二值化

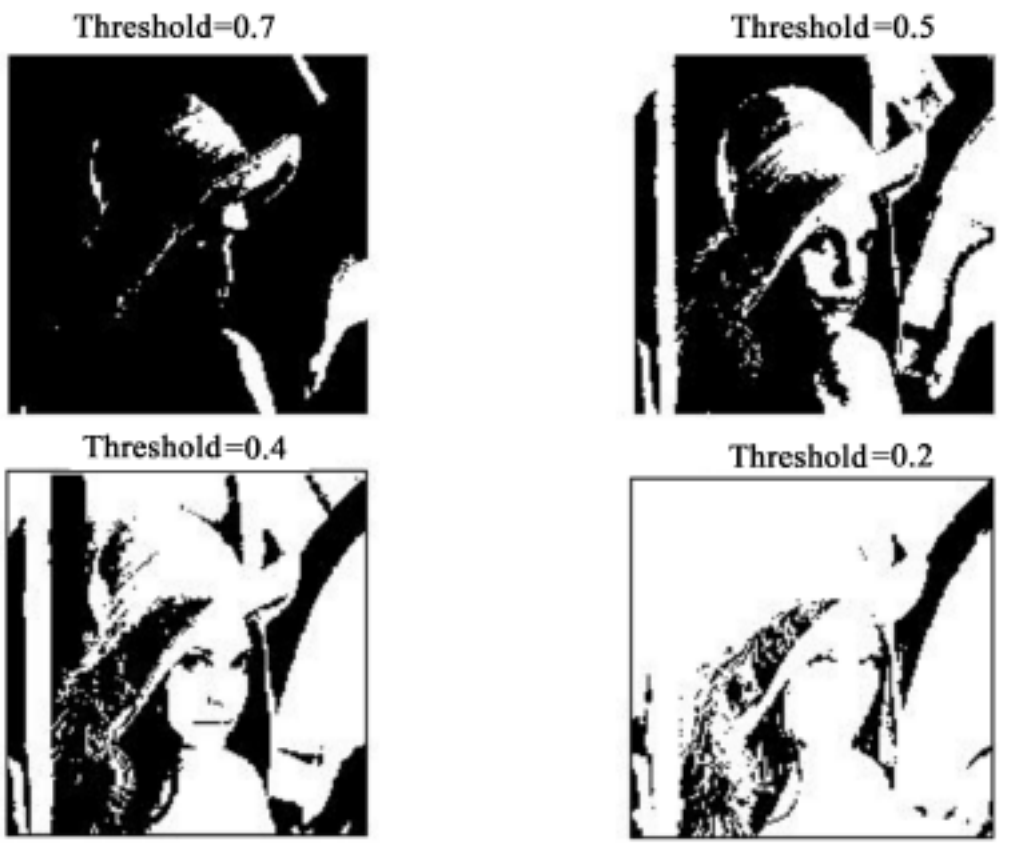

图 1.10　灰度图像二值化阈值的作用

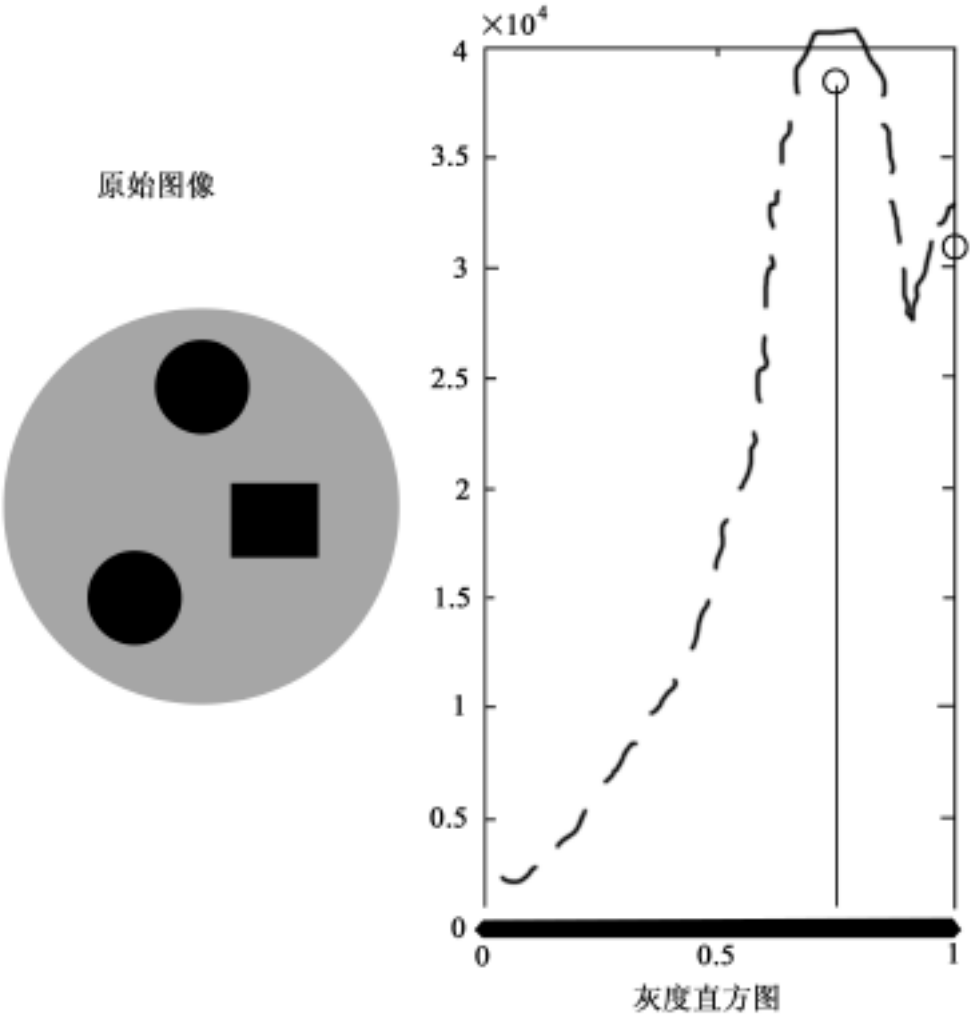

图 1.11　灰度直方图

的时候,如果将阈值取在两峰之间,自然可以得到良好的二值化效果,否则转换出来的图像就是漆黑一片。在图 1.11 中,我们很容易理解将几个黑色的图形称为对象,将灰色的大圆称做背景。灰度直方图明显地区分了对象与背景。事实上,大多数的图像都有一定数量的对象和背景。也就是说,对象和背景部分的灰度区域集中了大量的像素,从而使灰度直方图呈现峰(modal) 与谷(bottom) 交替出现的情况,我们称这一现象为灰度直方图的多峰性(multi-modal) 。将二值化阈值取在两个主要的峰之间,就可以保证较好的转换效果。这个阈值就是全局阈值。lenna 的灰度直方图如图 1.12 所示。

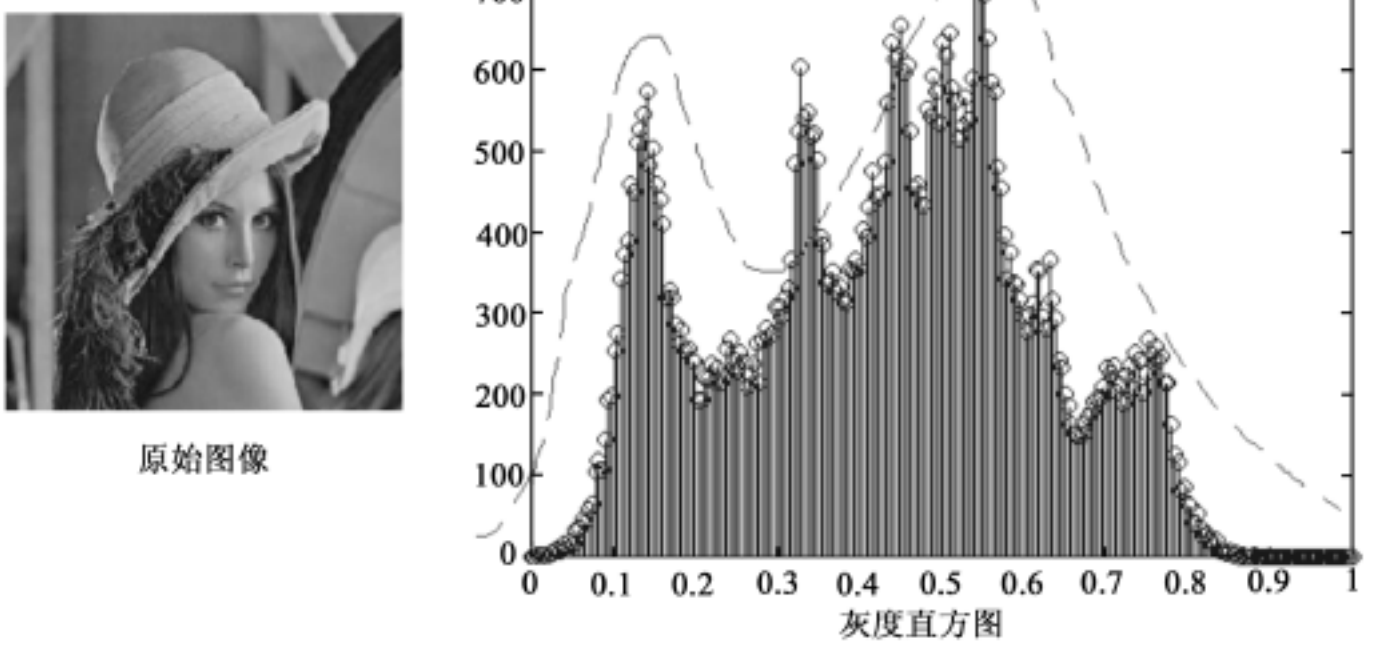

图 1.12　lenna 的灰度直方图

全局阈值法是最简单的灰度图像二值化方法。由于其只能设立一个绝对阈值,显然它并不是一个优秀的二值化方法。有兴趣去了解其他二值化方法的读者可以参考有关图像图形和模式识别方面的资料。此外,用 MATLAB 的 imhist. m 函数可以方便地画出一幅灰度图像的灰度直方图。

由于 RGB 与索引图可以转换为灰度图像,所以它们也可以转换为二值图像。这里我们就不赘述了。很容易想到,非二值图像二值化是可以由一个映射关系去描述的,但反之,则不是一个映射,也是无法实现的。

1.2.2　RGB 图像与索引图像的互换

下面我们重点介绍 RGB 图像和索引图像之间的转换,这在今后的信息隐藏实验中会用到。RGB 图像转换到索引图像使用的函数是 rgb2ind,该函数的一般使用格式为:

[data, map] = rgb2ind(rgbimage, tol) 或

[data, map] = rgb2ind(rgbimage, n)

引入 tol 和 n 两个参数是因为 RGB 图像的色彩非常丰富, 而索引图像无法全部显示, 故利用这两个参数控制转换的图像色彩数量。tol 是一个(0, 1) 区间的实数, 相应转换的索引图像的调色板矩阵包含$\left(\frac{1}{\text{tol}}\right)^3$种色彩。n 是一个[0, 65535] 的整数, 直接表示转换后的索引图像的色彩数量。图 1. 13 是将 lenna 图像(RGB) 转换为索引图像的效果, 取 tol = 0. 1。

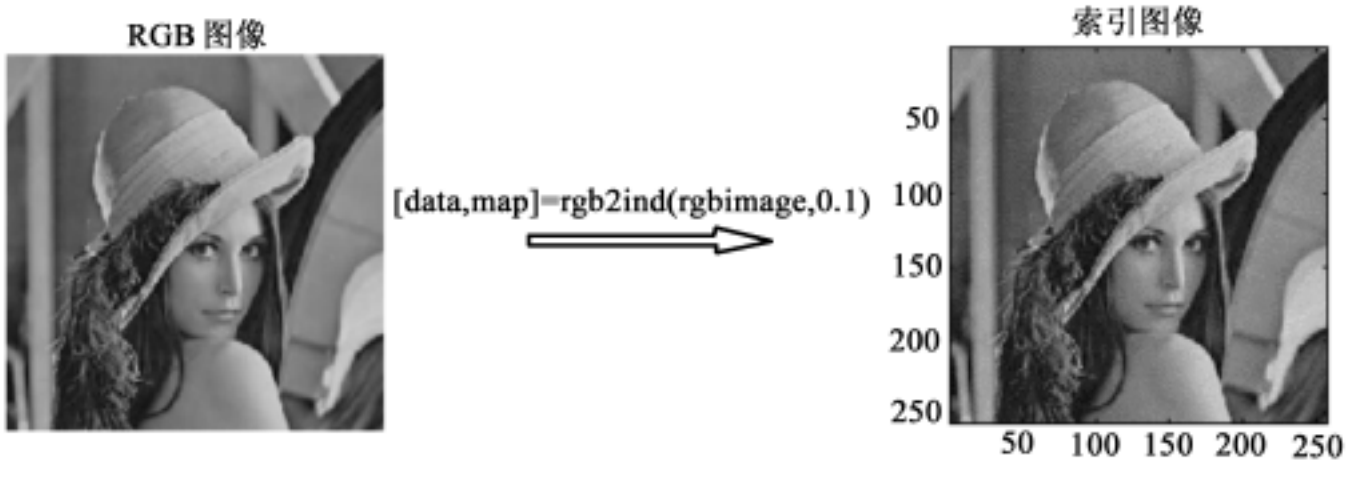

图 1. 13　RGB 图像转换为索引图像

索引图像转换为 RGB 图像使用的函数是 ind2rgb。该函数使用就非常简单了。使用格式为:

rgbimage = ind2rgb(data, map)

MATLAB 自带的 wbarb 信号用 load 命令得到的图像是一个典型的彩色索引图像。图 1. 14 是将它转换为 RGB 图像的效果。

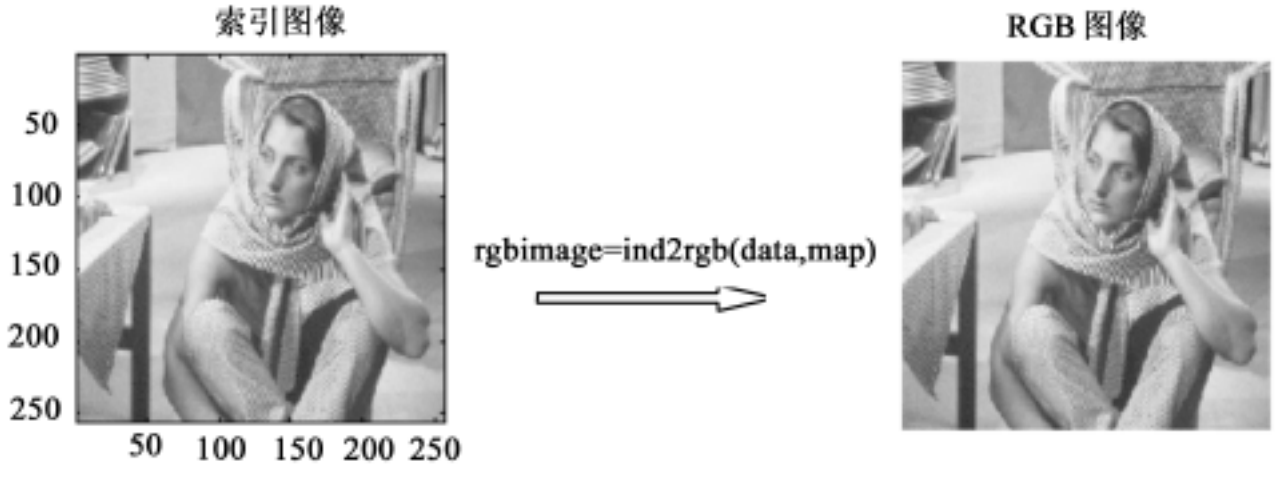

图 1. 14　索引图像转换为 RGB 图像

比较图 1. 13 和图 1. 14 可以发现, 索引图像转换为 RGB 图像色彩不会失真, 而

RGB 图像转换为索引图像一般会出现色彩丢失而导致图像效果变差。

1.2.3　其他转换

由于将 RGB 图像转换为灰度图像与提取 RGB 图像各像素的亮度分量是一样的，所以灰度图像与 RGB 图像的互换我们将在后面的 RGB 颜色模型与 YCbCr 模型互换中阐述。

另外，还可以使用 MATLAB 的一些基本语句实现某些转换操作。例如，在灰度图像矩阵 I 上连接自身的三个拷贝构成第三维，就可以得到一幅 RGB 图像。如 RGB = GRAY(3，I，I，I) ;

除了以上的标准转换方法外，还可以利用某些函数返回的图像类型与输入的图像类型之间不同这一特点进行类型转换。例如，基于区域的操作函数总是返回二进制图像，用这些函数可以实现索引图像或灰度图像向二进制图像的转换。

1.3　数字图像的基本文件格式

图像格式与图像类型不同，指的是存储图像采用的文件格式。不同的操作系统、不同的图像处理系统，所支持的图像格式都有可能不同。在实际应用中常用到以下几种图像格式。

(1) BMP 文件

BMP 文件是 Microsoft Windows 所定义的图像文件格式，最早应用在 Microsoft 公司的 Microsoft Windows 视窗系统中。BMP 图像文件的特点是:

① 每个文件只能存放一幅非压缩图像。

② 只能存储四种图像数据: 单色、16 色、256 色、真彩色。

③ 图像数据有压缩或不压缩两种处理方式。

④ BMP 图像文件的文件结构可分为三部分: 表头、调色板和图像数据。表头长度固定为 54 个字节。图 1.15 给出一种定义的表头格式。

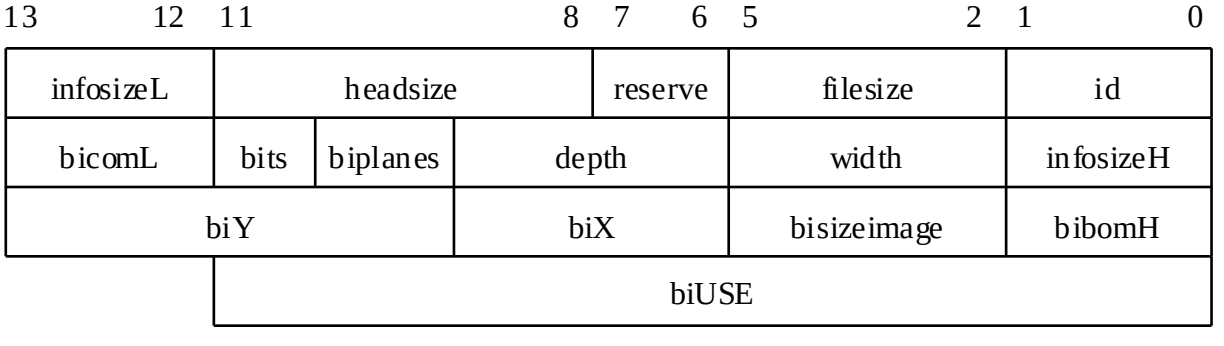

图 1.15　BMP 图像的表头结构

对照图 1.15 简单说明一下: id 域包括 2 Bytes，为字符“ BM ”。filesize 域存放了图

像文件的大小。headsize 域是一个图像数据起始地址的指针，实际反映了表头和调色板的大小。width 和 depth 域定义了图像的宽度和高度各是多少。bits 表示像素点的颜色位数，对应特点②中的 4 种数据类型，相应的为 1，4，8 和 24。bisizeimage 为图像数据的字节数。我们在以 BMP 图像为载体的信息隐藏实验中曾经遇到过这样一个问题，就是隐藏信息的开头几个字总是不能有效地提取出来。后来发现这就是由于对 BMP 文件格式不了解导致的。所以，若要以 BMP 文件为载体做信息隐藏，请务必回避这 54 字节的表头（起码应该回避前两个字节，即 id 域）。对于空域隐藏，则从第 55 个字节开始存放信息。当然，MATLAB 的 imread 函数所读取的就是数据部分，所以在 MATLAB 上做相关实验则无需考虑上述问题。

⑤ 调色板的数据存储结构较为特殊，其存储格式不是固定的，而是与文件头某些参数有关。只有真彩色 BMP 图像文件内没有调色板数据，其余不超过 256 种颜色的图像文件都必须设定调色板信息。

⑥ 以真彩色（24 位）BMP 文件为例子，其数据部分的排列顺序是以图像左下角为起点，每 3 个连续字节表示一个像素点的颜色信息。这三个字节分别指示 B，G，R 的分量值。如某像素对应的三个字节内容为 FF0000H，则意味着它是一个纯蓝点。

需要强调的是，大家不要将 BMP 文件格式与前文所述的位图类型混淆了。一个是图像类型，一个是文件格式，尽管二者有相似的英文表示。

（2）GIF 文件

GIF（Graphics Interchange Format）文件是由 CompuServe 公司为了方便网络用户传送图像数据而制定的一种图像文件格式。GIF 图像文件经常用于网页的动画、透明等特技制作。GIF 文件有这样一些特点：文件具有多元化结构，能够存储多张图像；调色板数据有通用调色板和局部调色板之分；图像数据一个字节存储一点；文件内的各种图像数据区和补充区多数没有固定的数据长度和存放位置，为了方便程序寻找数据区，就以数据的第一个字节作为标识符，以使程序能够判断所读到的是哪种数据区；图像数据有两种排列方式：顺序排列和交叉排列；图像最多只能存储 256 色图像。GIF 图像文件结构一般由 7 个数据单元组成，它们分别是：表头、通用调色板、图像数据区以及 4 个补充区。表头和图像数据是文件不可缺少的单元，通用调色板和其余的 4 个补充区是可选内容。GIF 图像文件可以有多个图像数据区，而每个图像数据区存储 1 幅图像，通过软件处理和控制，将这些分离的图像形成连续的有动感的图示效果。

（3）TIFF 文件

TIFF（Tag Image File Format）文件是由 Aldus 公司与微软公司共同开发设计的图像文件格式。它有这样一些特点：善于应用指针的功能，可以存储多幅图像；文件内数据区没有固定的排列顺序，但规定表头必须在文件前端，标识信息区和图像数据区在文件中可以随意存放；可制定私人用的标识信息；除了一般图像处理常用的 RGB 模式外，TIFF 图像文件还能够接受 CMYK，YCbCr 等多种不同颜色模式；可存储多份

调色板数据;调色板的数据类型和排列顺序较为特殊;能提供多种不同压缩数据的方法,以方便使用者的选择;图像数据可分割成几个部分分别存档。TIFF 图像文件主要由三部分组成:表头、标识信息区和图像数据区。文件内固定只有一个表头,且一定位于文件前端。表头有一个标识参数指出标识信息区在文件中的存储地址,标识信息区有多组标识信息用于存储图像数据区的地址。每组标识信息长度固定在 12 个字节,前 8 个字节分别代表标识信息的代号(两个字节)、数据类型(两个字节)、数据量(4 个字节),最后 4 个字节则存储数据值或标识参数。文件末尾有时还存放一些标识信息区内容容纳不了的数据,例如,调色板数据就是其中一项。

(4) PCX 文件

PCX 文件是由 Zsoft 公司在 20 世纪 80 年代初期设计的,专用于存储该公司开发的 PC Paintbrush 绘图软件所产生的图像画面数据。目前 PCX 文件已成为较为流行的图像文件。PCX 图像文件具有这样一些特点:一个 PCX 图像文件只能存放一张图像画面;PCX 图像文件有多个版本,能处理多种不同模式下的图像数据;4 色和 16 色 PCX 图像文件有可设定或不设定调色板数据的两种选项;16 色图像数据可由 1 个或 4 个 bit Plane(颜色的 RGB 等级)来处理。

(5) JPEG 格式

它是由 Joint Photographic Expert Group 制定的图像压缩格式,其正式名称为“连续色调静态图像的数字压缩和编码”,是一种基于离散余弦变换(DCT)或离散小波变换(DWT)的图像压缩编码标准。JPEG 压缩技术十分先进,它采用最少的磁盘空间来得到较好的图像质量。此图像格式原理将在第三章中讨论,这里就不再阐述了。

(6) PSD 格式

这是由 Adobe 公司开发的图像处理软件 Photoshop 中自建的标准图像文件格式,由于 Photoshop 软件被广泛地应用,因而这种格式也很流行。

(7) PCD 格式

这是由 KODAK 公司开发的 Photo CD 专用存储格式,由于其文件特别大,所以不得不存在 CD-ROM 上,但其应用领域特别广。

(8) PNG 格式

PNG 能存储 32 色的位图文件格式,其图像质量远胜过 GIF。与 GIF 一样,PNG 用无损压缩方式来减少文件的大小。目前,越来越多的软件开始支持这一格式,在不久的将来,它可能会在整个 Web 流行。PNG 可以是灰阶的(16 位)或彩色的(48 位),也可以是 8 位的索引色。PNG 用的是高速交替显示方案,显示速度很快,只需要下载 1/64 的图像信息就可以显示出低分辨率的预览图像。与 GIF 不同的是 PNG 格式不支持动画。

在这些图像格式中,我们使用最多的就是 BMP,JPEG 和 PNG 三种,它们的文件格式是必须要掌握的。

1.4 图像存储方式和图像文件格式的相互转换

MATLAB 中最基本的数据结构是数组(矩阵) 。在 MATLAB 中, 大多数图像用二维数组(矩阵) 存储, 矩阵中的一个元素对应于要显示图像的一个像素。例如, 一个由 100 行和 200 列的不同颜色的点组成的图像可以用一个 100× 200 的矩阵来存储。前面已经讨论了也有一些图像, 如 RGB 图像, 需要用三维数组来存储, 第一维表示红色像素的分量值, 第二维表示绿色像素的分量值, 第三维表示蓝色像素的分量值。

在默认的情况下, MATLAB 将图像中的数据存储为双精度型(double) , 即 64 位的浮点数。这种存储方法的优点在于使用中不需要数据类型转换, 因为几乎所有的 MATLAB 工具箱函数都可使用 double 作为参数类型。然而对于图像存储来说, 此种方式表示数据会导致巨大的存储量, 所以 MATLAB 还支持图像数据的另一种类型——无符号整数型(uint8) , 即图像矩阵中的每个数据占用一个字节。MATLAB 工具箱函数往往不支持 uint8 类型。uint8 的优势仅仅在于节省存储空间, 在涉及运算时要将其转换成 double 型。我们在今后的实验中一般都使用 double 型矩阵进行操作, 这既方便调用图像函数, 又与图像像素值的范围一致。

在处理图像时, MATLAB 通常使用 64 位的双精度类型, 即 64 位的浮点数。但是, 为了减少内存需求, 提高系统运行速率, MATLAB 还提供了两个重要的数字类型 uint8 和 uint16, 用以支持 8 位和 16 位的无符号整数类型。在 MATLAB 中, 数据矩阵中包含 uint8 数字类型的图像称为 8 位图; 同理, 数据矩阵中包含 uint16 数字类型的图像称为 16 位图。

利用 MATLAB 提供的 image 函数, 可以直接显示 8 位图像或 16 位图像, 而不必将其转换为双精度浮点类型。然而, 当图像是 uint8 或 uint16 类型时, MATLAB 对矩阵值的解释有所不同, 这主要依赖于图像的具体类型。

使用 MATLAB 一些基本函数可以对图像存储类型进行转换。例如, double 函数可以将 uint8 或 uint16 数据转换为双精度数据。存储类型间的转换将改变 MATLAB 理解图像数据的方式。如果用户希望转换后得到的数组能够被正确地理解为图像数据, 那么在转换时需要重新标度成偏移数据。例如, 如果将一个 uint16 类型的灰度图像转换为 uint8 类型的灰度图像, 那么函数 im2uint8 将对原始图像的灰度级进行量化, 换句话说, 所有 0 ～255 之间的原始图像数值都将变为 uint8 图像数据中的 0, 而 256 ～511 之间的数值都为 1, 以此类推。当使用一种位数较少的类型描述数字图像时, 通常有可能丢失用户图像的一些信息。例如, 一个 uint16 类型的灰度图像能够存储65 536种不同的灰度级, 但是一个 uint8 类型的灰度图像却只能存储 256 种不同的灰度级。一般这种信息的丢失不会产生严重的后果, 因为 256 种灰度级仍然超过人眼所能分辨的色彩数目。在使用 double 函数时应注意量化问题, 如将一个 uint8 矩阵转换为 double 型矩阵, 正确的操作是:

doublematrix = double(uint8matrix) /255;

1.4.1　8 位和 16 位索引图像

如果图像数据矩阵的类型是 uint8 或 uint16, 则其值在作用于颜色映射表的索引之前, 必须进行值为 1 的偏移, 即值 0 指向矩阵 map 的第一行, 值 1 指向第二行, 依此类推。并且因为 image 函数自动提供了这种偏移, 所以不管图像数据矩阵是双精度浮点类型, 还是 uint8 或 uint16 类型, 显示的方法都相同。

但是当用户在进行类型转换时, 由于偏移量的存在, 必须将 uint8 或 uint16 的数据加 1, 然后才能将其转换为双精度浮点类型。例如: X64 = double(X8) +1 或 X64 = double(X16) +1; 相反, 若将双精度浮点类型转换为 uint8 或 uint16 类型的数据时, 在转换之前, 必须将其减 1。例如: X8 = uint8(X64-1) 或 X16 = uint16(X64-1) 。

1.4.2　8 位和 16 位灰度图像

在 MATLAB 中, 双精度浮点类型的图像数组中的数值的取值范围通常为[0, 1] , 而 8 位和 16 位无符号整数类型的图像的取值范围分别为[0, 255] 和[0, 65535] 。

在 MATLAB 中, 将一个灰度图像从双精度转换为 uint16 的 16 位无符号整数类型, 必须首先将其乘以65 535。例如: I16 = uint16(round(I64* 65 535)) 。

与此相反, 将一个灰度图从 uint16 的 16 位无符号整数类型转换为双精度的浮点类型时, 必须首先除以65 535。例如: I64 = double(I16) /65 535。

1.4.3　8 位和 16 位 RGB 图像

8 位 RGB 图像的颜色数据是[0, 255] 之间的整数, 而不是[0, 1] 之间的浮点值。所以, 在 8 位 RGB 图像中, 颜色值为(255, 255, 255) 的像素显示为白色。不管 RGB 图像是何种类型, MATLAB 都通过以下代码来显示, 即: image(RGB) 。

将 RGB 图像从双精度的浮点类型转换为 uint8 无符号整数类型时, 首先要乘以 255, 即: RGB8 = uint8(round(RGB64* 255)) 。

相反, 如果将 uint8 无符号整数类型的 RGB 图像转换为双精度的浮点类型时, 首先要除以 255, 即: RGB64 = double(RGB8) /255。

另外, 如果将 RGB 图像从双精度浮点类型转换为 uint16 无符号整数类型时, 必须乘以 65 535, 即: RGB16 = uint16(round(RGB64* 65535)) 。

同样, 如果将 uint16 无符号整数类型的 RGB 图像转换为双精度浮点类型时, 首先要除以65 535, 即: RGB64 = double(RGB8) /65535。

1.4.4　其他相互转换的方法

MATLAB 图像处理工具箱还提供了图像存储类型间的转换函数, 这些函数包括 im2double, im2uint8 和 im2uint16, 这些函数可以自动进行原始数据的重新标度和偏

移。这三个函数的调用格式非常简单,输入参数为图像矩阵,输出为转换后的图像。例如,以下命令将一个描述双精度 RGB1 图像的矩阵(数据范围为[0,1])转换为一个 uint8 类型的 RGB2 图像矩阵([0,255]范围内):RGB2 = im2uint8(RGB1)。

1.4.5 图像文件格式的相互转换

图像格式间的转换可以间接利用图像读写函数来完成:首先使用 imread 函数按照原有图像格式进行图像读取,然后调用 imwrite 函数对图像进行保存,并指定图像的保存格式。例如,将一幅图像由 BMP 格式转换为 PNG 格式,则可以这样实现:首先使用 imread 读取 BMP 图像,然后调用 imwrite 函数来保存图像并指定为 PNG 格式:

```
bitmap = imread('mybitmap.bmp','bmp');
imwrite(bitmap,'mybitmap.png','png');
```

1.5 其他的颜色模型

1.5.1 颜色模型

在计算机图形学领域定义的颜色模型,就是在某种特定上下文中对于颜色的特性和行为的解释方法。我们前面对色彩的讨论都是基于通过红、绿、蓝三原色混合而产生其他颜色的成色机制上。RGB 颜色模型最便于在诸如视频监视器或打印机等硬件设备上表示颜色。但在具体的图形应用中,我们还会用到其他的一些颜色模型。

(1) HSV 模型

HSV 模型是面向用户的,是一种复合主观感觉的色彩模型。H,S,V 分别指的是色调(彩)(hue)、色饱和度(saturation)和明度(value)。所以在这个模型中,一种颜色的参数便是 H,S,V 三个分量构成的三元组。

HSV 模型不同于 RGB 模型的单位立方体,而是对应于一个圆柱坐标系中的一个立体锥形子集,如图 1.16(a)所示。在这个锥形中,边界表示不同的色彩。H 分量表示颜色的种类,取值范围为 0°~360°,相应的颜色从红、黄、绿、蓝绿、蓝、紫到黑变化,且它的值由绕 V 轴的旋转角决定,每一种颜色和它的补色之间相差 180°。S 分量的取值范围也是 0~1,表示所选色彩的纯度与该色彩的最大纯度的比例。相应的颜色从未饱和(灰度)向完全饱和(无白色元素)变化,当 S = 0.5 时表示所选色彩的纯度为二分之一。V 分量取值范围同样是 0~1,从锥形顶点 0 变化到顶部 1,相应的颜色逐渐变亮,顶点表示黑色,顶部表示色彩强度最大。

对于多数图像,128 种颜色、8 种色饱和度和 16 种明度就足够了。按这一参数范围,HSV 颜色模型可以提供 128× 8× 16 = 16 384种颜色,即要求用 14 位存放一个像素的色彩。

有必要澄清一下的是, HSV 模型中的明度(Value) 并不是我们通常说的亮度。一个像素的明暗可以认为是由 V 和 S 共同构成的: 在图 1. 16(a) 中任何一个由 V 和 S 构成的剖面三角形的斜边才是明暗亮度的体现。

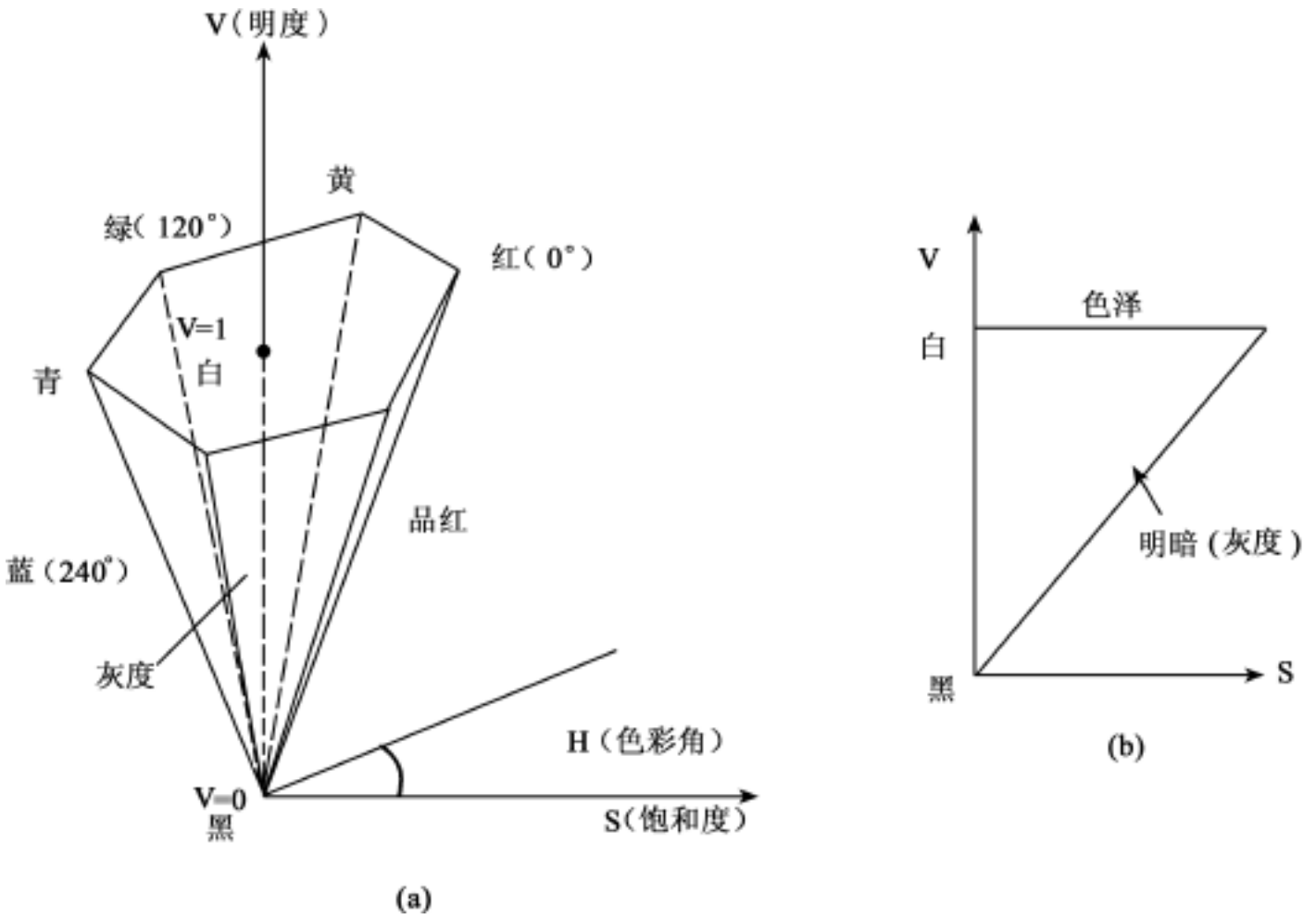

图 1. 16　(a) HSV 颜色模型　(b) 剖面三角形与明暗

(2) YCbCr 模型

YCbCr 模型又称为 YUV 模型, 是视频图像和数字图像中常用的色彩模型。在 YCbCr 模型中, Y 为亮度, Cb 和 Cr 共同描述图像的色调(色差) , 其中 Cb, Cr 分别为蓝色分量和红色分量相对于参考值的坐标。YCbCr 模型中的数据可以是双精度类型, 但存储空间为 8 位无符号整型数据空间, 且 Y 的取值范围为 16 ~235, Cb 和 Cr 的取值范围为 16 ~240。

在图像数字信号的处理中, 我们都要用到 YCbCr 模型。这是因为利用该模型可以达到数据压缩的目的。在目前通用的图像压缩算法中(如 JPEG 算法) , 首要的步骤就是将图像颜色空间转换为 YCbCr 空间。人类具有对色差的细微变化的感觉比对亮度变化的感觉迟钝的视觉特性。利用这一特性对色差信息进行间隔取样、编码就可以达到压缩数据的目的。也就是说, 对亮度信息的每个像素进行编码, 而对色差信息沿扫描方向取 2 个或 4 个像素的均值进行编码。通过这种方式, 两个色差信息的数据量将降为原来的 1/2 或 1/4, 总体的图像数据量就可以降低为原来的 2/3 或

1/2。其相应的采样处理方法被称为 YCbCr422 或 YCbCr411。详细的图像压缩方法请参见第三章的相关内容。

(3) NTSC 模型

NTSC 模型是一种用于电视图像的颜色模型。NTSC 模型使用的是 Y. I. Q 色彩坐标系, 其中, Y 为光亮度, 表示灰度信息; I 为色调, Q 为饱和度, 均表示颜色信息。因此, 该模型的主要优点就是将灰度信息和信息区分开来。

1.5.2 颜色模型之间的转换

1.5.1 节我们简单地介绍了各种颜色模型, 本节我们将介绍它们之间的相互转换关系。与 1.2 节一样, 我们的侧重点也是在 MATLAB 的操作上。表 1.2 列出了 MATLAB 中的用于颜色模型转换的函数。我们选择用得较多的两个转换加以介绍。

表 1.2　　颜色模型转换函数及其功能

函　数　名	函　数　功　能
hsv2rgb	HSV 模型转换为 RGB 模型
ntsc2rgb	NTSC 模型转换为 RGB 模型
rgb2hsv	RGB 模型转换为 HSV 模型
rgb2ntsc	RGB 模型转换为 NTSC 模型
rgb2ycbcr	RGB 模型转换为 YCbCr 模型
ycbcr2rgb	YCbCr 模型转换为 RGB 模型

(1) RGB 颜色模型与 HSV 颜色模型的转换

RGB 立方体从黑色(原点)到白色的立方体对角线与 HSV 锥的 V 轴相对应。同时, RGB 立方体的每一个子立方体与 HSV 锥的六边形剖面区域相对应。在 HSV 锥的任意剖面中, 六边形的各边和从 V 轴到任意顶点的射线都具有相同的明度值 V。对于任何一组 RGB 值, 参数 H 通过计算该点在六边形的六等分中的相对位置来确定。参数 S 按照该点到 V 轴的相等距离而确定。V 与该组 RGB 值的最大值相等, 与一组 RGB 值对应的 HSV 点位于明度值为 V 的六边形剖面上。

以 rgb2hsv 为例, 该函数用来将 RGB 模型转换为 HSV 模型, 其调用格式如下:

```
hsvmap = rgb2hsv( rgbmap) ;
HSV = rgb2hsv( RGB) ;
```

其中: hsvmap = rgb2hsv(rgbmap) 表示将 RGB 空间中 m× 3 的色彩表 rgbmap 转换成 HSV 色彩空间的颜色映射表 hsvmap。HSV = rgb2hsv(RGB) 表示将真彩图像 RGB 转换为 HSV 色彩空间中的图像 HSV。

我们先来执行以下代码, 得到图 1.17 的结果。

```
>> RGB = imread( 'c: \lenna. jpg', 'jpg');
>> HSV = rgb2hsv( RGB) ;
>> subplot( 1,2,1) ; imshow( RGB) ; title( '原图像');
>> subplot( 1,2,2) ; imshow( HSV) ; title( '变换后的图像');
```

图 1.17　RGB 模型转换为 HSV 颜色模型

有很多关于 MATLAB 中颜色模型的书都是用上述方法来体现 HSV 模型下的图像效果的(后面的图 1.18 是一样的)。事实上,我们认为这是一种非常无意义的做法。因为 imshow 函数本身是在 RGB 颜色空间下的图像显示函数。用该函数去查看一个非 RGB 颜色模型的图像就好比在时域中去看待频域函数一样,所看到的结果当然是错误的。所以我们不能因为在图 1.17 中看到 HSV 模型的图像效果很奇怪而去否定该模型。事实上,无论是 RGB 模型还是 HSV 模型以及 YCbCr 模型都同样能清晰地刻画一幅图像,且各有其优势和特点。图 1.17 只是能帮助我们感性地理解一下罢了。

(2) RGB 颜色模型与 YCbCr 颜色模型的转换

RGB 颜色模型与 YCbCr 模型的转换是通过如下线性变化完成的:

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.3316 & -0.50 \\ 0.50 & -0.4186 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

以 rgb2ycbcr 函数为例,该函数是用来将 RGB 模型转换为 YCbCr 模型的,其调用格式如下:

```
ycbcrmap = rgb2ycbcr( rgbmap) ;
YCbCr = rgb2ycbcr( RGB) ;
```

其中: ycbcrmap = rgb2ycbcr(rgbmap) 表示将 RGB 空间中的色彩表 rgbmap 转换为 YCbCr 空间中的颜色映射表 ycbcrmap。YCbCr = rgb2ycbcr(RGB) 表示将真彩图像 RGB 转换为 YCbCr 空间中的图像 YCbCr。

图 1.18 是将 lenna 图像转换为 YCbCr 模型下看到的效果。具体操作如下:

```
>> RGB = imread( 'c: \lenna. jpg', 'jpg');
>> YUV = rgb2ycbcr( RGB) ;
>> subplot( 1,2,1) ; imshow( RGB) ; title( '原图像');
>> subplot( 1,2,2) ; imshow( YUV) ; title( '变换后的图像');
```

图 1.18　RGB 模型转换为 YUV 颜色模型

接下来,我们来看一下 YCbCr 模型与灰度图像的关系。前面我们不止一次地提到,灰度图像就是 YCbCr 的 Y 分量,灰度图就是亮度图等,其实那只是一个简单的等价。事实上,灰度图像与图像亮度是两个不同的概念。就好比我们说一个对象物是亮或者暗与说一个对象物是黑还是白是不同的一样,前者是亮度的概念,后者讨论的是颜色的范畴。但由于“亮度”这个东西不太容易直观地感受到,我们往往需要借助于颜色去表现。于是,将 YCbCr 的 Y 分量的亮度数值人为地与 RGB 颜色模型(见图 1.1)的灰度对角线相对应,将亮度值体现为 RGB 模型下的颜色值,就构成了灰度图像。这种对应关系也就构成了我们在 1.1.4 节中提到的“默认的系统调色板”。当然,我们在以后的章节中并不严格区别灰度与亮度。

其他转换函数的使用方法与上述两个函数相似,我们就不一一赘述了。最后,计算机图像所包含的知识是相当丰富的,大家如果想进一步深入了解,可以查阅有关计算机图形学和数字图像处理等方面的书籍和文章。