

本章主要介绍利用 MATLAB 实现数字图像处理的基本操作,为后续更好地学习并仿真各种图像处理算法奠定基础。本章主要包括以下内容:图像文件的读取与显示、图像类型的转换、色彩空间转换、视频文件的读写等。

3.1 图像文件的读取与显示

图像文件的读取与显示是进行图像处理的第一步。本节介绍 MATLAB 提供的相应函数及其实现。

3.1.1 图像文件信息读取

MATLAB 提供了函数 `imfinfo` 和 `imageinfo` 来获取图像文件的信息。

1. 函数 `imfinfo`

函数 `imfinfo` 用于返回一个结构体数组,以存储图像文件的相关信息。其调用格式如下。

`INFO=imfinfo(FILENAME,FMT)`: `FILENAME` 是当前路径下,或 MATLAB 路径下,或指定了路径的图像文件的文件名;`FMT` 是文件的扩展名;`INFO` 是一个结构体,包含了文件中的图像信息,不同格式的文件最终得到的 `INFO` 所包含的字段不同,但其前 9 个字段一致,如表 3-1 所示。如果 `FILENAME` 是包含不止一幅图像的文件,如 `TIFF`、`HDF`、`ICO`、`GIF` 或 `CUR` 等,则 `INFO` 是一个结构体数组,数组中每个元素是一个包含一幅图像信息的结构体,如 `INFO(2)` 是文件中第 2 幅图像的信息。

表 3-1 `imfinfo` 函数返回的结构体数组基本内容

INFO 结构体字段名	含 义
Filename	文件名称,包含文件所在路径
FileModDate	文件最近修改或者下载的日期和时间(日-月-年 时:分:秒)
FileSize	文件大小,整数,单位:字节
Format	文件格式或扩展名,由 <code>FMT</code> 指定
FormatVersion	文件格式版本号

续表

INFO 结构体字段名	含 义
Width	图像的宽度,整数,单位: 像素
Height	图像的高度,整数,单位: 像素
BitDepth	图像文件中每个像素存储所占位数,整数
ColorType	图像类型,包含但不限于'truecolor'-RGB 图像、'grayscale'-灰度图像、'indexed'-索引图像

2. 函数 imageinfo

函数 imageinfo 用于创建一幅图像信息工具,用于显示当前 figure(图形图像窗口)中图像的信息,包括宽、高、图像类型等。其调用格式如下。

(1) imageinfo(H): 基于 H 创建图像信息工具,H 为 figure、坐标系或图像对象的句柄。

(2) imageinfo(FILENAME): 根据文件名创建图像信息工具,图像不一定要在 figure 窗口显示。

(3) imageinfo(INFO): 使用 INFO 结构体创建图像信息工具。

(4) imageinfo(HIMAGE,FILENAME): 创建图像信息工具,显示被 HIMAGE 句柄指定的图像基本属性和 FILENAME 指定的图像文件的元数据。

(5) imageinfo(HIMAGE,INFO): 创建图像信息工具,显示被 HIMAGE 句柄指定的图像基本属性和结构体 INFO 指定的图像文件的元数据。

(6) HFIGURE=imageinfo(...): 创建图像信息工具,并返回图像信息工具窗口句柄。

【例 3-1】 读取图像文件信息并显示查看。

程序如下:

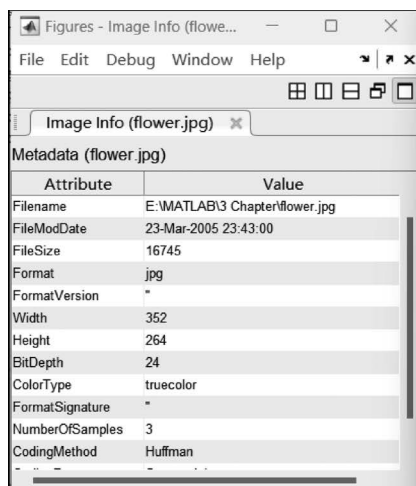
```
clear,clc,close all;
imageinfo('flower.jpg');           % 创建图像信息工具,显示 flower.jpg 图像元数据
h = imshow('flower.jpg');          % 显示图像,并返回 figure 窗口句柄
info1 = imfinfo('cameraman.jpg');  % 获取 cameraman.jpg 图像信息,返回结构体数据 info1
hfigure = imageinfo(h, info1);      % 创建图像信息工具,显示 flower.jpg 的基本属性和 cameraman.jpg 的元数据

info1                               % 在命令行窗口输出结构体 info1 的数据
info2 = imfinfo('water.gif');       % 获取 water.gif 图像信息,返回结构体数据 info2
info2(2)                            % 在命令行窗口输出 water.gif 文件中第 2 幅图像的信息
```

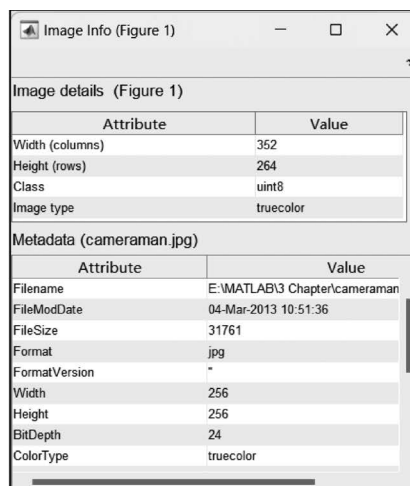
程序运行,创建的图像信息工具如图 3-1 所示。

在命令行窗口显示 INFO 结构体数据,ans 为 water.gif 文件中第 2 幅图像的信息。

```
Info =
  struct with fields:
    Filename: 'E:\MATLAB\3 Chapter\cameraman.jpg'
    FileModDate: '04 - Mar - 2013 10:51:36'
    FileSize: 31761
    Format: 'jpg'
    FormatVersion: ''
    Width: 256
    Height: 256
    BitDepth: 24
    ColorType: 'truecolor'
    FormatSignature: ''
```



(a) flower.jpg图像元数据



(b) flower.jpg的基本属性和cameraman.jpg的元数据

图 3-1 创建的图像信息工具

```

NumberOfSamples: 3
CodingMethod: 'Huffman'
CodingProcess: 'Sequential'
Comment: {}
Orientation: 1
Software: 'ACD Systems Digital Imaging'
DateTime: '2013:03:04 10:51:32'
YCbCrPositioning: 'Centered'
DigitalCamera: [1 x 1 struct]
ans =
struct with fields:
    Filename: 'E:\MATLAB\3 Chapter\water.gif'
    FileModDate: '05 - Nov - 2008 08:57:36'
    FileSize: 95308
    Format: 'GIF'
    FormatVersion: '89a'
    Left: 1
    Top: 57
    Width: 240
    Height: 264
    BitDepth: 8
    ColorType: 'indexed'
    FormatSignature: 'GIF89a'
    BackgroundColor: 215
    AspectRatio: 0
    ColorTable: [256 x 3 double]
    Interlaced: 'no'
    DelayTime: 13
    TransparentColor: 256
    DisposalMethod: 'LeaveInPlace'

```

3.1.2 图像文件数据读取

MATLAB 主要利用 `imread` 函数实现图像文件数据的读取,为适应不同的文件格式,有不同的

调用格式。

(1) `A=imread(FILENAME,FMT)`：从 `FILENAME` 指定的文件中读取图像数据。`FILENAME` 是当前路径下或指定了路径的图像文件的文件名；`FMT` 是文件的扩展名,可以使用 `imformats` 函数查看当前扩展名支持的格式；返回值 `A` 是包含图像数据的矩阵,对于灰度图像,`A` 为 $M \times N$ 的矩阵；对于真彩色图像,`A` 为 $M \times N \times 3$ 的矩阵；对于包含使用 CMYK 颜色模型图像的 TIFF 文件, `A` 为 $M \times N \times 4$ 的矩阵。

(2) `[X,MAP]=imread(FILENAME,FMT)`：读取索引图像数据,图像数据存放于 `X` 中,颜色映射表数据自动归一化到 $[0,1]$,存放于 `MAP` 中。

(3) `[...]=imread(FILENAME)`：根据文件内容推断图像类型,并根据待读取图像数据的类型选择格式 1 或格式 2。

(4) `[...]=imread(URL,...)`：读取来自网络的图像文件。

在 MATLAB 中,图像数据类型有 `uint8`、`uint16`、`double`、`logical`、`single` 等,在灰度级别的表示方面,`uint8` 型数据用 $0 \sim 255$ 表示,`uint16` 型数据用 $0 \sim 65535$ 表示,`double` 型数据用 $0 \sim 1$ 表示,`logical` 型数据用 0、1 表示。

【例 3-2】 读取不同类型图像,并查看各返回值。

程序如下：

```
clear,clc,close all;
Image1 = imread('flower.bmp');
Image2 = imread('bird.bmp');
[Image3,MAP3] = imread('pig.bmp');
subplot(131),imshow(Image1),title('彩色图像');
subplot(132),imshow(Image2),title('二值图像');
subplot(133),imshow(Image3,MAP3),title('索引图像');
```

程序运行结果如图 3-2 所示。在 MATLAB 工作窗口工作区,查看各变量,取值情况如表 3-2 所示,可以看出,彩色图像数据为 $M \times N \times 3$ 的 `uint8` 矩阵,二值图像数据为 $M \times N$ 的 `logical` 矩阵,索引图像的颜色映射表为取值为 $0 \sim 1$ 的 $P \times 3$ 矩阵, P 为颜色数目。

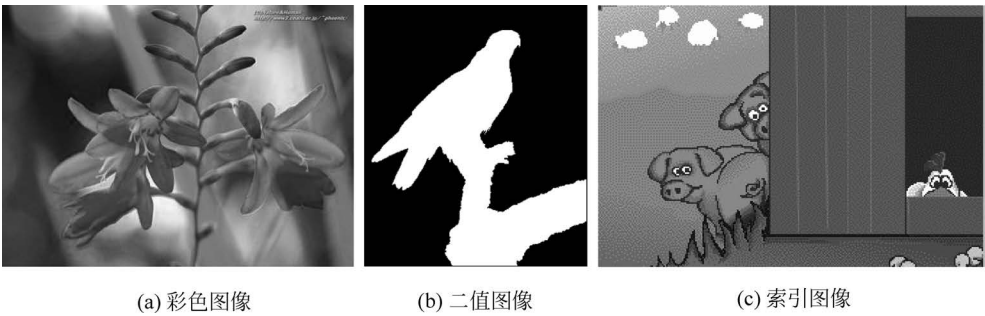


图 3-2 采用 `imread` 函数读取不同类型图像

表 3-2 例 3-2 各变量取值

名 称	尺 寸	数 据 类 型	最 小 值	最 大 值
Image1	264×352×3	uint8	0	255
Image2	359×304	logical	—	—
Image3	182×268	uint8	1	88
MAP3	256×3	double	0	1

(5) `[...] = imread(...,IDX)`: 读取包含多幅图像的 ICO、CUR 文件中的某一幅,IDX 是整型数据,指定读取图像中的第几幅图像,默认情况下,读取文件中第 1 幅图像。

(6) `[A,MAP,ALPHA] = imread(...)`: 返回图标文件的 AND 模板,用于处理透明像素信息。

(7) `[...] = imread(...,IDX)`: 读取动图 GIF 文件中的一幅或多幅图像,IDX 取整数或整数向量,如 3 或者 1:5,分别指读取第 3 幅或前 5 幅图像;IDX 默认情况下,读取第 1 帧数据。

(8) `[...] = imread(...,'Frames',IDX)`: 读取动图 GIF 文件中的一幅或多幅图像,IDX 可以取 'all',则所有帧全部被读取,并且按照在文件中的顺序返回。由于 GIF 的文件结构,特定帧的读取也需要读取所有帧数据,因此,IDX 使用指定向量或'all'读取所有帧数据,比采用循环读取多帧运算速度快。

(9) `[...] = imread(...,REF)`: 读取 HDF 文件中的一幅图像,REF 指定所读取图像的参考编号,但在 HDF 文件中,参考编号顺序和图像顺序未必一致,可以使用 `iminfo` 函数获取某一幅图像的参考编号;REF 默认情况下读取第 1 幅图像。

(10) `[...] = imread(...,'BackgroundColor',BG)`: 读取 PNG 图像文件,将透明像素与 BG 中的指定颜色合成。BG 取 'none',则不合成;如果输入的图像是索引图像,BG 取 `[1,P]` 范围的整数,其中 P 为颜色数目;如果输入的图像是灰度图像,BG 取 `[0,1]` 范围的值;如果输入图像是彩色图像,BG 是三维向量,其元素取值范围为 `[0,1]`。

(11) `[A,MAP,ALPHA] = imread(...)`: 假如存在透明信息,则返回 ALPHA 通道数据,否则 ALPHA 为 `[]`。这种格式下,BG 默认值为 'none';如果 PNG 文件包含了背景色,则 BG 默认值为背景色;如果不使用 ALPHA 通道并且文件不包含背景色,对于索引图像,BG 默认值为 1;灰度图像,BG 默认值为 0;RGB 图像默认值为 `[0 0 0]`。如果 'BackgroundColor' 被指定,则 ALPHA 数据为 `[]`。灰度图像或真彩色图像 MAP 数据为 `[]`。

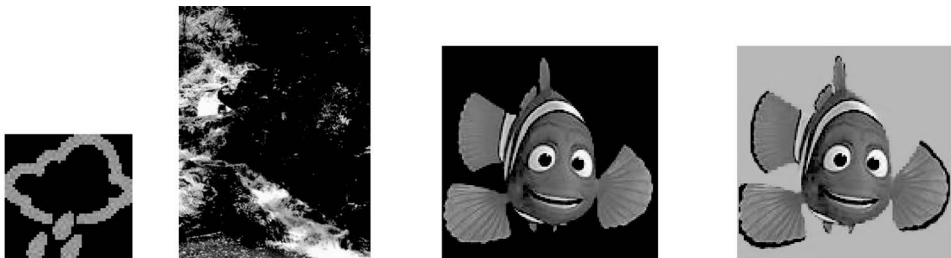
【例 3-3】 采用 `imread` 函数的不同调用格式读取 ICO、GIF、PNG 图像,并查看各返回值。

程序如下:

```
clear,clc,close all;
Image1 = imread('weather.ico',8);           % 读取 ICO 文件的第 8 幅图像
[Image2,MAP2] = imread('water.gif',2);      % 读取 GIF 文件的第 2 幅图像
[Image3,MAP3] = imread('water.gif');         % 读取 GIF 文件的第 1 帧
[Image4,MAP4] = imread('water.gif','frames','all'); % 读取 GIF 文件的全部帧
Image5 = imread('fish.png');                 % 读取 PNG 图像,透明像素与默认值合成
[A,MAP5,ALPHA] = imread('fish.png','BackgroundColor',[0 1 0.3]); % 读取 PNG 图像,透明像素与指定值合成

subplot(221),imshow(Image1),title('weather.ico 第 8 幅图像');
subplot(222),imshow(Image2,MAP2),title('water.gif 第 2 幅图像');
subplot(223),imshow(Image5),title('透明像素与默认的黑色合成');
subplot(224),imshow(A),title('透明像素与指定颜色合成');
```

程序运行结果如图 3-3 所示。



(a) ICO 第 8 幅图像 (b) GIF 第 2 幅图像 (c) PNG 透明像素与黑色合成 (d) PNG 透明像素与指定颜色合成

图 3-3 采用 `imread` 函数读取 ICO、GIF、PNG 图像

在 MATLAB 工作窗口工作区,查看各变量,取值情况如表 3-3 所示。

表 3-3 例 3-3 各变量取值

名 称	尺 寸	数 据 类 型	最 小 值	最 大 值
Image1	48×48×3	uint8	0	233
Image2	320×240	uint8	0	254
Image3	320×240	uint8	0	254
Image4	4-D	uint8	0	254
Image5	128×128×3	uint8	0	255
A	128×128×3	uint8	0	255
ALPHA	[]	double	—	—
MAP2	256×3	double	0	0.9843
MAP3	256×3	double	0	0.9843
MAP4	256×3	double	0	0.9843
MAP5	[]	double	—	—

由于 GIF 图像最多为 256 色,读取图像时,需要同时读取颜色映射表信息,全部帧共用同一颜色映射表,因此 MAP2、MAP3 和 MAP4 是一样的。

(12) `[...] = imread(..., 'Param1', value1, 'Param2', value2, ...)`: 设定参数读取图像, JPEG 2000 图像读取时参数如表 3-4 所示, TIFF 图像读取时参数如表 3-5 所示。

表 3-4 JPEG 2000 图像读取时参数表

参 数	取值及含义
ReductionLevel	一个非负整数,指定图像分辨率的降低。若为 L ,则图像分辨率降低一个因子 2^L 。默认值为 0,代表分辨率不减少。imfinfo 函数返回的结构体中, WaveletDecompositionLevels 字段指定分解级别,限制 ReductionLevel 的取值
PixelRegion	{ROWS,COLS}。imread 函数返回由 ROWS 和 COLS 中的值作为边界所指定的子图像。行列数都是二维向量,表示从 1 开始的索引[START STOP]。如果 ReductionLevel 大于 0,则 ROWS 和 COLS 是在尺寸减小的图像上的坐标
V79Compatible	逻辑值: 为真,返回的图像转换为和 imread 早期版本一致的灰度或 RGB 图像,采用本参数转换 YCC 图像为 RGB 图像。默认值为假

表 3-5 TIFF 图像读取时参数表

参 数	取值及含义
Index	正整数,指定 TIFF 图像文件中哪一幅图像被读取
Info	函数 imfinfo 输出的结构体数组。当读取包含多幅图像的 TIFF 文件时,采用 Info 作为参数将提高 imread 函数在文件中定位要读取图像的速度
PixelRegion	{ROWS,COLS}。imread 函数返回由 ROWS 和 COLS 中的值作为边界所指定的子图像。行列数为二维或三维向量,二维向量表示从 1 开始的索引[START STOP]; 三维向量表示从 1 开始的索引[START INCREMENT STOP],允许图像下采样

各种不同格式的文件在用 imread 函数读取时,有像素位数、对应文件类型的细致区别,如有需要可以查阅 MATLAB 帮助文件。

【例 3-4】 采用 imread 函数读取 JPEG、TIFF 图像,并查看各返回值。

程序如下:

```
clear,clc,close all;
Image1 = imread('football.jpg');
Image2 = imread('autumn.tif');
Image3 = imread('autumn.tif','PixelRegion',[100 200],[10 200]);
% 读取 TIFF 图像中的子图像

subplot(131),imshow(Image1),title('JPEG 图像');
subplot(132),imshow(Image2),title('TIFF 原图像');
subplot(133),imshow(Image3),title('TIFF 子图像');
```

程序运行结果如图 3-4 所示,Image3 仅为 Image2 中的一部分。在 MATLAB 工作窗口工作区,查看各变量,取值情况如表 3-6 所示。8 位的 JPEG 文件,无论采用有损还是无损压缩方式,读出来的数据类型都为 uint8 型数据,如 Image1。Image3 的宽高正如程序中设定的一样。



图 3-4 采用 imread 函数读取 JPEG、TIFF 图像

表 3-6 例 3-4 各变量取值

名 称	尺 寸	数 据 类 型	最 小 值	最 大 值
Image1	256×320×3	uint8	6	255
Image2	206×345×3	uint8	1	248
Image3	101×191×3	uint8	16	176

3.1.3 图像的显示

MATLAB 主要利用 imshow 函数实现图像的显示,此外,也提供了适应一些特殊需求的显示函数,如 image、imagesc、montage 和 imshowpair。

1. imshow 函数

imshow 函数用于在通用的图形图像窗口显示图像,自动设置图像窗口、坐标轴和图像属性。根据图像源文件的不同,有如下多种调用格式。

- (1) imshow(I): 显示灰度图像 I。
- (2) imshow(I,[LOW HIGH]): 指定灰度级范围[LOW HIGH]来显示灰度图像 I,小于或等于 LOW 值的显示为黑,大于或等于 HIGH 值的显示为白,默认按 256 个灰度级显示。若未指定 LOW 和 HIGH 值,则将图像中最低灰度显示为黑色,最高灰度显示为白色。
- (3) imshow(RGB): 显示真彩色图像 RGB。
- (4) imshow(BW): 显示二值图像 BW,像素值为 0 显示黑色,像素值为 1 显示白色。
- (5) imshow(X,MAP): 显示索引图像,X 为索引图像的数据矩阵,MAP 为其颜色映射表。

(6) `imshow(FILENAME)`: 显示 `FILENAME` 指定的图像。此格式下, `imshow` 通过调用 `imread` 或 `dicomread` 从文件 `FILENAME` 中读取图像数据。因此, 要求图像能够被 `imread` 或 `dicomread` 读取。若文件包括多帧图像, 则显示第 1 帧, 且文件必须在当前目录或 MATLAB 路径下。

(7) `HIMAGE=imshow(...)`: 返回创建的图像对象句柄。

(8) `imshow(...,PARAM1,VAL1,PARAM2,VAL2,...)`: 显示图像时指定相关参数及其取值, 参数如表 3-7 所示。

表 3-7 `imshow` 函数参数表

参 数	取值及含义
Border	一个字符串, 指明图像在 figure 窗口显示时是否显示边界, 可取 'tight' 和 'loose', 默认情况下取 'loose'
Colormap	$M \times 3$ 实数矩阵, 设置要显示图像的颜色映射表, 也可用来将灰度图像进行伪彩色显示
DisplayRange	二维向量 [LOW HIGH], 指定灰度范围显示灰度图像。在图像数据已经读取后显示, 可以省略参数名, 如 <code>imshow(I,[LOW HIGH])</code> ; 若 <code>imshow</code> 中采用 <code>FILENAME</code> 指定文件, 则不能省略。根据图像 I 的值取整数或浮点数
InitialMagnification	数值, 或字符串 'fit', 指定图像初始显示比例: 如 100, 则图像以 100% 比例显示; 若为 'fit', 则以适合窗口的比例显示整幅图像。若图像太大不能显示完全, 则将进行警告并以适合屏幕的最大比例显示图像。默认为 100。采用坐标定位显示时, 将忽略指定的数值, 取 'fit' 值; 使用 'Reduce' 参数时, 只能取 'fit' 值
Interpolation	指定缩放图像时采用的插值方法, 可选 'nearest' (最邻近插值) 和 'bilinear' (双线性插值)
Reduce	逻辑值, 指明是否对文件 <code>FILENAME</code> 中的图像进行下采样。仅适用 TIFF 图像, 用于显示大图像的概貌
Parent	指向图像对象的父对象的坐标系句柄
XData	二维向量, 用于建立非默认的空间坐标系
YData	二维向量, 用于建立非默认的空间坐标系

【例 3-5】 采用 `imshow` 函数的不同形式显示图像。

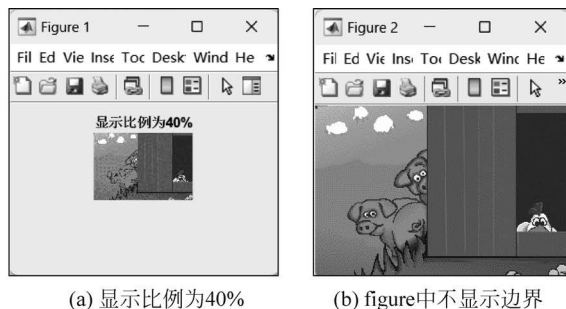
程序如下:

```
clear,clc,close all;
[Image1,MAP1] = imread('girl.bmp');
[Image2,MAP2] = imread('pig.bmp');
figure,imshow(Image2,MAP2,'InitialMagnification',40),title('显示比例为 40 %');
figure,imshow(Image2,MAP2,'Border','tight'),title('figure 中不显示边界');
figure,
subplot(221),imshow(Image1),title('默认显示');
subplot(222),imshow(Image1,[50 100]),title('指定灰度范围显示');
subplot(223),imshow(Image1,'Colormap',MAP2),title('用颜色映射表 MAP2 显示');
subplot(224),imshow(Image2,MAP2,'XData',[100 300], ...
'YData',[200 280]),title('建立新坐标系显示');
```

程序运行结果如图 3-5 所示。

2. `image` 和 `imagesc` 函数

`image` 函数将矩阵中的数据显示为图像, `imagesc` 函数采用拉伸过的色彩显示图像, 有多种调用格式。



(a) 显示比例为40%

(b) figure中不显示边界



(c) 子图显示图像

图 3-5 imshow 函数显示图像

(1) `image(C)`: 将矩阵 C 中的数据显示为图像, C 的每个元素指定图像中一个像素的颜色。当 C 为二维的 $M \times N$ 矩阵时, 每个元素的值作为像素颜色值在当前颜色映射表中的索引值, 由图像对象的 `CDataMapping` 属性决定: `CDataMapping` 取 'direct' (默认), C 中的元素值直接作为颜色索引; 取 'scaled', C 中的元素值先进行拉伸再作为索引值。当 C 为三维的 $M \times N \times 3$ 矩阵时, $C(:, :, 1)$ 、 $C(:, :, 2)$ 、 $C(:, :, 3)$ 依次作为颜色的 R、G、B 分量值。如果 C 中元素的数据类型为 `double`, 则颜色值变化范围为 $[0.0, 1.0]$; 如果 C 中元素为 `uint8` 或 `uint16` 数据类型, 则颜色值变化范围为 $[0, 255]$ 。

(2) `image(x, y, C)`: x 、 y 用于指定显示图像时, C 中元素的坐标位置。函数 `image` 显示图像时, 同时显示坐标轴, 不设定 x 、 y 的值, 则 $C(1, 1)$ 位于坐标 $(1, 1)$ 处, $C(M, N)$ 位于坐标 (M, N) 处; 设定 x 、 y 值, 则 $C(1, 1)$ 位于坐标 $(x(1), y(1))$ 处, $C(M, N)$ 位于坐标 $(x(end), y(end))$ 处; 如果 x 、 y 为单个值, 则 $C(1, 1)$ 位于坐标 (x, y) 处, $C(M, N)$ 根据图像宽高和其他设置自行确定。

(3) `imagesc(...)`: 数据经过拉伸作为颜色索引值, 显示图像。

(4) `imagesc(..., CLIM)`: 利用向量 $CLIM = [CLOW \ CHIGH]$ 设置拉伸范围, $CLOW$ 对应颜色映射表中的第 1 个颜色, $CHIGH$ 对应颜色映射表中的最后一个颜色, 中间的灰度线性对应颜色映射表中的其余颜色。

【例 3-6】 采用 image 和 imagesc 函数的不同形式显示图像。

程序如下：

```
clear,clc,close all;
Image1 = imread('football.jpg');
[Image2,MAP2] = imread('girl.bmp');
figure,image(Image1),title('彩色图像');
figure,colormap(hot), h1 = image(50,60,Image2),title('不拉伸映射');
% 设置窗口颜色映射表为 hot 型,指定数据矩阵(1,1)位于(50,60)处
Y1 = get(h1,'CDataMapping'); % 获取 CDataMapping 属性值
figure,colormap(hot),h2 = imagesc(Image2,[30,150]),title('拉伸映射');
% 灰度 30 对应颜色映射表第 1 个颜色,灰度 150 对应最后一个颜色
Y2 = get(h2,'CDataMapping'); % 获取 CDataMapping 属性值
```

程序运行结果如图 3-6 所示。

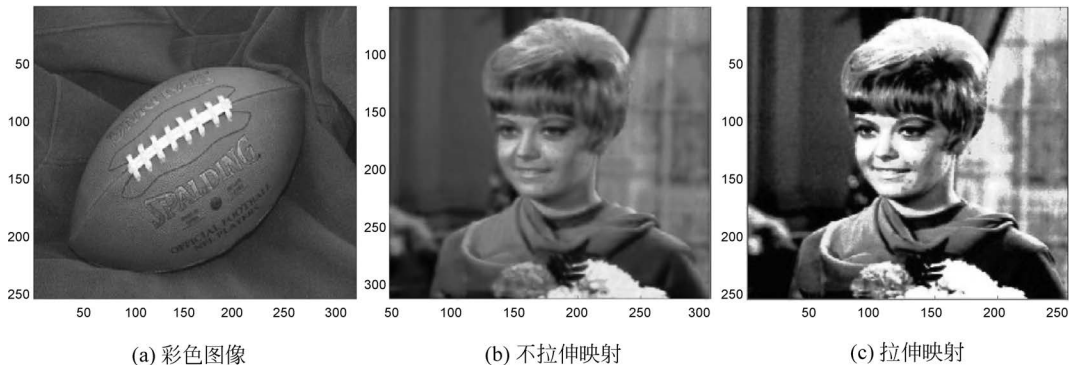


图 3-6 调用 image 和 imagesc 函数显示图像

从工作区可以看到, Y1 = 'direct', Y2 = 'scaled', 即采用 image 函数显示不拉伸数据, 而采用 imagesc 函数拉伸数据。

3. montage 函数

montage 函数用矩形蒙太奇方式显示多帧图像的每帧,其调用格式如下。

- (1) montage(FILENAMES): 显示 FILENAMES 指定的多帧图像。若 FILENAMES 不在当前目录或 MATLAB 路径下,则需要指明路径。
- (2) montage(I): 显示多帧图像 I,I 可以是二值、灰度、彩色图像序列。
- (3) montage(X,MAP): 将 X 中的所有灰度图像作为索引图像显示,共用颜色映射表 MAP。
- (4) montage(...,NAME1,VALUE1,NAME2,VALUE2,...): 定制显示,其参数如表 3-8 所示。
- (5) H=montage(...): 返回图像对象句柄。

表 3-8 montage 函数部分参数表

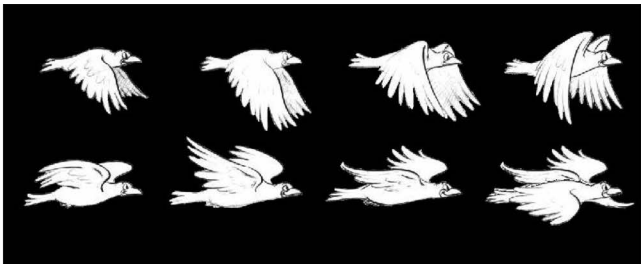
参 数	取值及含义
Size	二维向量[NROWS NCOLS],指定蒙太奇行列数。行列数之一可以设定为 NaN,在显示时,根据显示图像的总帧数和已知的行或列数目自动计算另一个
Indices	一个数字序列,指明哪些帧要显示,如 m:n,表示显示从第 m 帧到第 n 帧,默认为 1:K,K 为总帧数
DisplayRange	1×2 的向量[LOW HIGH],对显示的图像进行灰度拉伸,含义见函数 imshow

【例 3-7】 采用 montage 函数显示含有多帧的 GIF、TIF 图像。
程序如下：

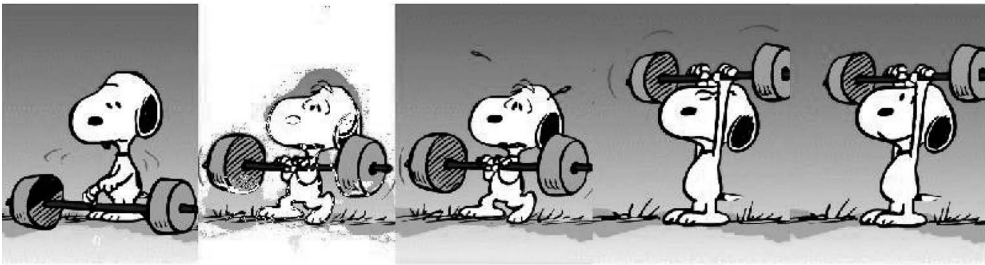
```
clear,clc,close all;
[Image1,MAP1] = imread('fly.gif','frames','all');
figure,montage(Image1,MAP1,'size',[2 NaN]),title('GIF 图像');
info = imfinfo('snoopy.tif');
len = length(info);
for i = 1:len
    [Image2(:, :, i),MAP2] = imread('snoopy.tif',i);
end
figure,montage(Image2,MAP2,'size',[1 NaN]),title('TIF 图像');
```

% 读取 GIF 图像
 % 两行排列显示 GIF 图像各帧
 % 获取 TIF 图像信息
 % TIF 图像含图像数
 % 依次读取 TIF 文件中各图像
 % 一行排列显示 TIF 图像各帧

程序运行结果如图 3-7 所示。



(a) GIF图像各帧



(b) TIF图像各帧

图 3-7 调用 montage 函数显示多帧图像

4. imshowpair 函数

imshowpair 函数将图像成对显示,以比较图像间的差异。其调用格式如下。

(1) H=imshowpair(A,B,METHOD)：将图像 A 和 B 间的差异以 METHOD 指定的方式实现可视化,并返回创建的图像句柄 H。如果 A 和 B 的大小不一致,则将较小的图像变为和较大图像同样大小,扩充的像素补 0。METHOD 的取值如表 3-9 所示。

表 3-9 imshowpair 函数参数表

参 数	取 值	含 义
METHOD	falsecolor	将 A 和 B 作为不同色彩通道合成 RGB 图像,默认值
	blend	采用 α 混合重叠 A 和 B
	checkerboard	从 A 和 B 创建具有交替矩形区域的图像
	diff	从 A 和 B 创建差异图像
	montage	将 A 和 B 在同一幅图像中相邻放置

续表

参 数	取 值	含 义
Scaling	independent	图像各自缩放,默认值
	joint	适用于在图像的动态范围之外具有大量填充值的单模态图像可视化
	none	不额外进行缩放
Parent		指向创建的图像父对象的坐标轴的句柄
ColorChannels		当 METHOD 取 'falsecolor' 时使用,将每个图像分配到输出图像中的特定颜色通道。设置为 [R G B], 指定哪一幅图像被指定到 RGB 对应通道, R、G、B 取 1 表示该通道指定第 1 幅图像,取 2 表示第 2 幅图像,取 0 表示没有图像被指定;可取 'red-cyan', 等同于 [R G B]=[1 2 2];可取 'green-magenta', 等同于 [R G B]=[2 1 2],默认值

(2) $H = \text{imshowpair}(A, RA, B, RB)$: 根据 RA 和 RB 提供的空间参考信息显示 A 和 B 图像的差异。RA 和 RB 由 `imref2d` 函数定义。

(3) `imshowpair(..., PARAM1, VAL1, PARAM2, VAL2, ...)`: 指定显示和混合方式显示图像。参数名称不区分大小写。各参数及取值含义见表 3-9。

【例 3-8】 采用 `imshowpair` 函数显示 GIF 图像的不同帧。

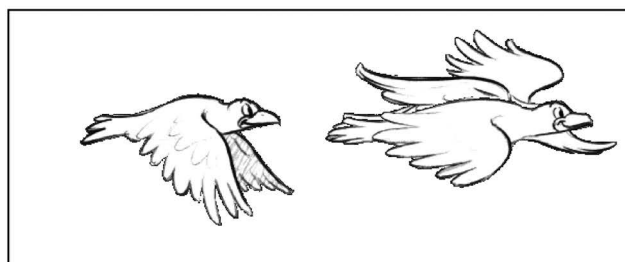
程序如下:

```
clear, clc, close all;
info = imfinfo('fly.gif');
len = length(info);
for i = 1:len
    [Image(:, :, :, i), MAP] = imread('fly.gif', i);
end
subplot(131), imshowpair(Image(:, :, :, 1), Image(:, :, :, len)); % 默认显示
subplot(132), imshowpair(Image(:, :, :, 1), Image(:, :, :, len), 'ColorChannels', [1 1 2]);
subplot(133), imshowpair(Image(:, :, :, 1), Image(:, :, :, len), 'diff');
figure, imshowpair(Image(:, :, :, 1), Image(:, :, :, len), 'montage');
```

程序运行结果如图 3-8 所示。



(a) 默认图像所在颜色通道显示 (b) 指定图像所在颜色通道显示 (c) 差异图像显示



(d) 并排放置显示

图 3-8 调用 `imshowpair` 函数显示图像差异

3.1.4 像素信息的获取与显示

MATLAB 利用 `imread` 和 `imreadinfo` 函数实现像素信息的获取。

1. 函数 `imread`

函数 `imread` 用于获取指定图像像素的 R、G、B 通道颜色值,其调用格式如下。

- (1) `P=imread(I)`: 鼠标指定灰度图像 `I` 中的像素,获取其颜色值。
- (2) `P=imread(X,MAP)`: 鼠标指定索引图像 `X` 中的像素,获取其颜色值。
- (3) `P=imread(_RGB)`: 鼠标指定真彩色图像 `RGB` 中的像素,获取其颜色值。

在以上 3 种调用中,`imread` 显示图像并等待用户用鼠标选择像素:单击选择像素,双击或右击表示选择最后一个像素,回车表示选择结束;可以使用退格键 `Backspace` 和删除键 `Delete` 来删除之前选择的像素,每次删除一个。若选择了 N 个点,则 `P` 为 $N \times 3$ 的 `double` 型数组,存放每个点 R、G、B 颜色值。

【例 3-9】 采用 `imread` 函数获取鼠标单击的像素的像素值。

程序如下:

```
clear,clc,close all;
[Image1,MAP1] = imread('kids.tif');           % 索引图像
P = imread(Image1,MAP1);                       % 获取像素颜色值,存放于 P 数组中
Image2 = imread('flower.jpg');                 % 真彩色图像
Q = imread(Image2);                             % 获取像素颜色值,存放于 Q 数组中
```

运行程序,首先显示 `kids.tif` 图像,在图像上用鼠标左键点两个点,回车结束选择;紧接着显示 `flower.jpg` 图像,在图像上用鼠标左键点一个点,双击结束选择;运行结束。`P`、`Q` 值如表 3-10 所示。`P` 是索引图像像素颜色值,取值是 `MAP1` 中的 0~1 的数据; `Q` 是彩色图像像素的红绿蓝色彩分量值,虽为 `double` 型数据,但在 0~255 范围内。

表 3-10 例 3-9 中 `P`、`Q` 取值

名 称	尺 寸	数 据 类 型	取 值
<code>P</code>	2×3	<code>double</code>	<code>[0.3137,0.1490,0.1020; 0.09800,0.0941,0.1020]</code>
<code>Q</code>	2×3	<code>double</code>	<code>[245,123,0; 165,15,0]</code>

- (4) `P=imread(I,C,R)`: `C`、`R` 指定灰度图像 `I` 中的像素,获取其颜色值。
- (5) `P=imread(X,MAP,C,R)`: `C`、`R` 指定索引图像 `X` 中的像素,获取其颜色值。
- (6) `P=imread(_RGB,C,R)`: `C`、`R` 指定真彩色图像 `RGB` 中的像素,获取其颜色值。
- (7) `[C,R,P]=imread(...)`: 返回指定像素坐标。

在以上 4 种调用中,通过 `C`、`R` 直接指定像素。`C` 和 `R` 为相同长度的向量,两向量中第 k 个对应元素构成像素的坐标 $(R(k),C(k))$ (矩阵坐标系),其颜色值为 `P` 的第 k 行数据。

- (8) `P=imread(x,y,I,xi,yi)`: 非默认坐标系统下,指定灰度图像 `I` 中的像素,获取其颜色值。
- (9) `P=imread(x,y,X,MAP,xi,yi)`: 非默认坐标系统下,获取索引图像 `X` 中指定像素的颜色值。

(10) `P=impixel(x,y,RGB,xi,yi)`: 非默认坐标系统下,获取真彩色图像 RGB 中指定像素的颜色值。

(11) `[xi,yi,P]=impixel(x,y,...)`: 返回指定像素坐标。

在以上 4 种调用中,`x,y` 为二维向量,指定图像坐标范围;通过 `xi,yi` 直接指定像素。`xi,yi` 为相同长度的向量,两向量中第 k 个对应元素构成像素的坐标($y_i(k)$, $x_i(k)$)(`x,y` 指定的坐标系统),其颜色值为 `P` 的第 k 行数据。

【例 3-10】 采用 `impixel` 函数获取指定的像素的像素值。

程序如下:

```
clear,clc,close all;
[Image1,MAP1] = imread('kids.tif');      % 读取索引图像
Image2 = imread('flower.jpg');           % 读取 RGB 真彩色图像
[N,M,color] = size(Image2);              % 获取真彩色图像尺寸
C = [20 40];      R = [50 100];          % 设定指定像素坐标
P1 = impixel(Image1,MAP1,C,R);            % 获取索引图像中指定像素的像素值,存于 P1 中
[C1,R1,P2] = impixel(Image2,C,R);        % 获取 RGB 图像中指定像素的像素值,并返回指定点坐标
x = [21 20 + M];   y = [51 50 + N];      % 设定图像坐标范围,尺寸未变,坐标值整体增加[20 50]
x1 = [40 60];     y1 = [100 150];        % 设定指定像素坐标
P3 = impixel(x,y,Image2,x1,y1);          % 获取 RGB 图像中指定像素的像素值,存于 P3 中
```

程序运行,各变量取值如表 3-11 所示。

表 3-11 例 3-10 中各变量取值

名 称	尺 寸	数 据 类 型	取 值
Image1	400×318	uint8	—
MAP1	256×3	double	—
Image2	264×352×3	uint8	—
P1	2×3	double	[0.4860,0.4275,0.3843; 0.4431,0.3686,0.3098]
P2	2×3	double	[19,86,68; 0,59,0]
P3	2×3	double	[19,86,68; 0,59,0]
x	1×2	double	[21,372]
y	1×2	double	[51,314]

P1 存放索引图像 Image1 中由 C、R 指定的两个像素的像素值;P2 存放 RGB 图像 Image2 中由 C、R 指定的两个像素的像素值;P3 存放改变图像坐标范围后的像素值。在设定的坐标系统中,图像尺寸未变,没有变形;指定的像素坐标分别为(100,40)和(150,60),对应原图中的(50,20)和(100,40),和 C、R 指定的像素一致,从表中可以看出,P3 和 P2 取值一样。

2. `impixelinfo` 函数

`impixelinfo` 函数用于在当前 figure 下创建像素信息,以显示光标所在位置处的图像像素信息。随着鼠标移动,可以显示 figure 中所有图像的像素的信息。

像素信息工具是一个 panel 控件,位于窗口的左下角,包含一个文本字符串"Pixel Info:",后面显示像素位置和像素值,显示内容与图像类型和光标位置有关,如表 3-12 所示。

表 3-12 impixelinfo 函数显示数据

图 像 类 型	显 示 字 串	示 例
光标在图像区域外	Pixel Info: (X,Y)Pixel Value	
灰度图像	Pixel Info: (X,Y) Intensity	Pixel Info: (13,30) 82
索引图像	Pixel Info: (X,Y) < index > [R G B]	Pixel Info: (2,6) < 4 > [0.29 0.05 0.32]
二值图像	Pixel Info: (X,Y) BW	Pixel Info: (12,1) 0
真彩色图像	Pixel Info: (X,Y) [R G B]	Pixel Info: (19,10) [15 255 10]

如果不显示"Pixel Info:"标签,可以使用 impixelinfoval 函数。

impixelinfo 函数的调用格式如下。

(1) impixelinfo: 默认情况下,创建像素信息工具。

(2) impixelinfo(H): 在句柄 H 指定的 figure 下创建像素信息工具,H 可以是图像、坐标系、panel 或者 figure 对象,后三者应包含至少一幅图像。

(3) impixelinfo(HPARENT,HIMAGE): HIMAGE 是图像句柄,在 HPARENT 指向的 figure 或 panel 上创建像素信息工具,用于显示 HIMAGE 中的像素信息。

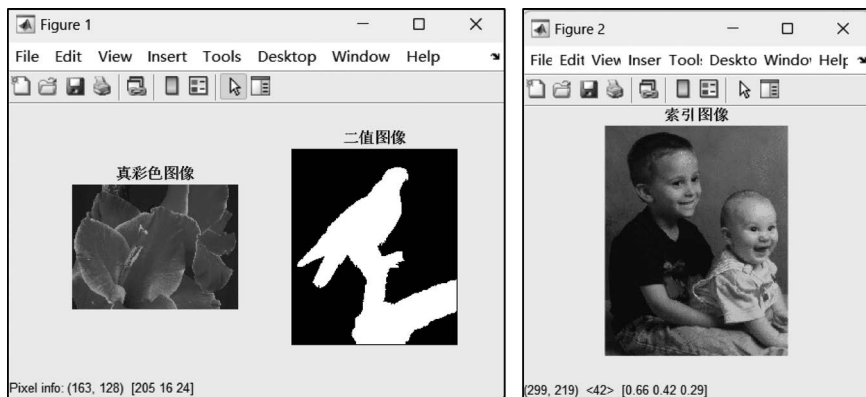
(4) HPANEL=impixelinfo(...): 创建一个像素信息工具,并返回像素信息工具的句柄。

【例 3-11】 采用 impixelinfo 函数获取像素信息。

程序如下:

```
clear,clc,close all;
Image1 = imread('pic4.bmp');
Image2 = imread('bird.bmp');
[Image3,MAP3] = imread('kids.tif');
figure;
subplot(121),imshow(Image1),title('真彩色图像');
subplot(122),imshow(Image2),title('二值图像');
impixelinfo
figure;
h = imshow(Image3,MAP3),title('索引图像');
H = impixelinfoval(gcf,h);
```

运行程序,在第 1 个 figure 中显示真彩色图像和二值图像,impixelinfo 创建像素信息工具,随着光标移动,显示数据同表 3-12 所示一致;在第 2 个 figure 中显示索引图像,impixelinfoval 创建像素信息工具,随着光标移动,显示数据同表 3-12 所示一致,但不显示"Pixel Info:"标签。程序运行结果如图 3-9 所示。



(a) impixelinfo 创建像素信息工具

(b) impixelinfoval 创建像素信息工具

图 3-9 像素信息工具显示效果

3.1.5 局部区域的获取与显示

在对图像进行处理的过程中,可能只需对图像的部分区域进行处理,MATLAB 提供了剪切函数 `imcrop` 用于实现图像局部区域的获取与显示,其调用格式如下。

(1) `I=imcrop`: 创建一个与当前 figure 中显示的图像相关联的交互式图像剪切工具。使用鼠标在图像上绘制区域,大小可调整、可移动,区域确定之后,双击或从右键菜单选择 `Crop Image` 命令实现该区域的剪切显示,并返回给 I。该工具可以通过按 `Backspace`、`Esc`、`Delete` 键,或者从右键菜单中选择 `Cancel` 命令取消,返回空值。

(2) `I2=imcrop(I)`: 在 figure 中显示图像 I 并创建一个与之关联的剪切工具。I 可以是灰度图像、RGB 图像或者逻辑矩阵; I2 为返回的剪切图像,类型与 I 一致。

(3) `X2=imcrop(X,MAP)`: 在 figure 中显示索引图像并创建一个与之关联的剪切工具。

(4) `I=imcrop(H)`: 创建一个与句柄 H 中的图像相关联的剪切工具。H 可以是图像、坐标系、`uipanel` 或 figure 的句柄,后三者的情况下,剪切工具作用在包含的第 1 幅图像上。

(5) `I2=imcrop(I,RECT)` 或者 `X2=imcrop(X,MAP,RECT)`: 指定剪切矩形实现剪切,非交互式。RECT 是一个 4 维向量 `[XMIN YMIN WIDTH HEIGHT]`,指定矩形的左上角位置及宽、高; RECT 也可以是一个 `Rectangle` 对象。若是非默认的空间坐标系,则需要指定 `XData` 和 `YData`。

(6) `[I2,RECT]=imcrop(...)`: 剪切的同时返回剪切矩形。

(7) `[X,Y,I2,RECT]=imcrop(...)`: 剪切的同时返回剪切矩形及目标图像的 `XData` 和 `YData` 值。

【例 3-12】 采用 `imcrop` 函数剪切区域。

程序如下:

```
clear,clc,close all;
Image = imread('toysflash.png');
figure;imshow(Image),title('原图');
[h,w,c] = size(Image);                                % 获取图像宽高信息
imcrop;                                                % 用鼠标选择剪切区域并显示
I1 = imcrop;                                           % 在当前 figure,即上一步剪切出来的图像中用鼠标选择区域剪切,返回给 I1
I2 = imcrop(Image,[w/2-60 h/2-120 w/3 h/3]);          % 在原图中指定矩形剪切
figure,imshow(I1),title('鼠标选定区域剪切');
figure,imshow(I2),title('原图中指定区域剪切');
```

运行程序,首先显示原图,如图 3-10(a)所示,等待用鼠标选择区域剪切;剪切后显示剪切区域,如图 3-10(b)所示;再等待用鼠标选择区域剪切,剪切后显示如图 3-10(c)所示,返回给 I1;紧接着显示在原图中指定矩形剪切的区域,如图 3-10(d)所示。

3.1.6 图像数据类型及转换

在处理图像的过程中,数据类型有可能发生变化,需要对其加以关注,避免出现错误。例如,在用函数 `imshow` 显示图像时,若数据为 `double` 型,则默认 0~1 为灰度范围;若 0~255 的 `uint8` 数据无意中转换为 0~255 的 `double` 型数据,显示时会把大于 1 的像素全部按 1(白色)显示。

图像数据类型可以根据需要转换,MATLAB 提供了数据类型转换的相关函数。

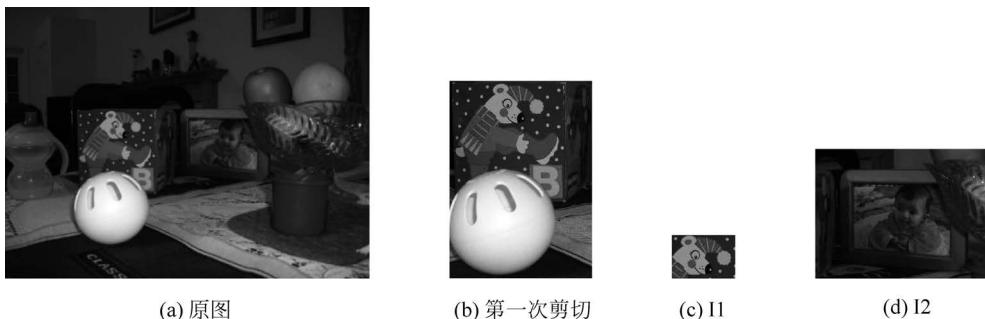


图 3-10 区域获取与显示

im2double、im2single、im2uint8、im2uint16、im2int16 函数分别实现原图像数据转换为 double、single、uint8、uint16、int16 型图像数据,取值范围调整为 $[0, 1]$ 、 $[0, 1]$ 、 $[0, 255]$ 、 $[0, 65535]$ 、 $[-32768, 32767]$ 。输入的图像可以是二值图像、灰度图像、真彩色图像或索引图像。

double、single、uint8、uint16、int16 函数将数据强制转换为对应数据类型,但数值取值范围不变。函数 mat2gray 用于将矩阵转换为灰度图像,其调用格式如下。

(1) $I = \text{mat2gray}(A, [AMIN \ AMAX])$: 将矩阵 A 转换为灰度图像 I 。 A 可以为逻辑型或数值型矩阵; I 的取值为 $0 \sim 1$ 的 double 型数据; $AMIN$ 和 $AMAX$ 是矩阵 A 中对应图像 I 内 0.0 和 1.0 的数据,小于 $AMIN$ 的数据变为 0 ,大于 $AMAX$ 的数据变为 1.0 。

(2) $I = \text{mat2gray}(A)$: 将矩阵 A 转换为灰度图像 I , A 内的最小值、最大值分别为 $AMIN$ 和 $AMAX$ 。

【例 3-13】 打开图像,转换图像类型,观察数据变化,并显示图像。

程序如下:

```
clear,clc,close all;
Image = imread('boy.bmp');           % 读取灰度图像,为 uint8 型数据
result1 = double(Image);              % 转换为 double 型数据,不改变取值范围
result2 = im2double(Image);           % 转换为 double 型数据,改变取值范围
result3 = im2uint16(Image);           % 转换为 uint16 型数据
[N,M] = size(Image);                 % 获取图像尺寸
A = rand(N,M);                       % 创建 0~1 随机数值矩阵
A(N/2-40:N/2+40,M/2-40:M/2+40) = 0; % 中心小正方形区域设为 0
result4 = mat2gray(A);                % 矩阵转换为灰度图像
subplot(221), imshow(result1), title('0~255double 型数据');
subplot(222), imshow(result2), title('0~1double 型数据');
subplot(223), imshow(result3), title('uint16 型数据');
subplot(224), imshow(result4), title('矩阵转图像');
```

运行程序,各变量取值如表 3-13 所示,运行结果如图 3-11 所示。将原 uint8 型数据用 double 函数强制转换为 double 型,取值范围不变,存放于 result1,图像显示为一片白色;用 im2double 函数将原 uint8 型数据转换为 $0 \sim 1$ 范围内的 double 型数据,存放于 result2,图像显示正常。

表 3-13 例 3-13 中各变量取值

名 称	尺 寸	数 据 类 型	最 小 值	最 大 值
Image	256×256	uint8	7	241
result1	256×256	double	7	241
result2	256×256	double	0.0275	0.9451
result3	256×256	uint16	1799	61937
result4	256×256	double	0	1
A	256×256	double	0	1.0

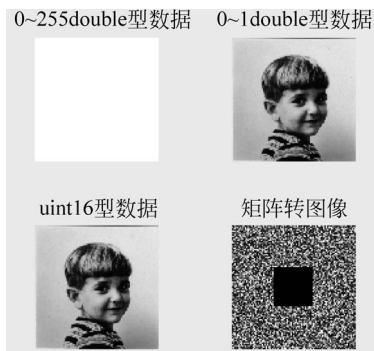


图 3-11 图像数据类型转换显示效果

3.1.7 图像文件的保存

MATLAB 主要利用 `imwrite` 函数实现图像文件的保存,其调用格式如下。

(1) `imwrite(A,FILENAME,FMT)`: 将图像 `A` 以 `FMT` 指定的格式写入 `FILENAME` 指定的文件,`A` 可以是 $M \times N$ 或者 $M \times N \times 3$ 的矩阵(即灰度图像或彩色图像)。若写为 TIFF 格式的文件,`A` 可以是 $M \times N \times 4$ 的 CMYK 数据。

(2) `imwrite(X,MAP,FILENAME,FMT)`: 将索引图像 `X` 及其关联的颜色映射表 `MAP` 以 `FMT` 指定的格式写入 `FILENAME` 指定的文件。若写为 GIF 图像,`X` 应当是 $M \times N \times 1 \times P$ 的矩阵,`P` 是图像中的帧数。

(3) `imwrite(...,FILENAME)`: 根据 `FILENAME` 指定的文件的后缀来推断格式,将图像写入。

(4) `imwrite(...,PARAM1,VAL1,PARAM2,VAL2,...)`: 指定参数控制输出文件的各种属性,保存图像。不同的文件格式参数不一致,支持 GIF、HDF、JPEG、TIFF、PNG、PBM、PGM 及 PPM 等格式。

【例 3-14】 打开一幅图像,将其保存为不同图像格式的文件。

程序如下:

```
clear,clc,close all;
info = imfinfo('snoopy.tif');           % 获取一幅 TIF 图像的信息
len = length(info);                     % 获取 TIF 图像中的图像数目
for i = 1:len
    [Image(:,:,i),MAP] = imread('snoopy.tif',i); % 逐帧读取图像数据
end
Image1 = Image(:,:,1);                  % 第一帧图像
imwrite(Image1,MAP,'snoopy1.bmp');       % 将第一帧索引图像存为 BMP 格式
imwrite(Image,MAP,'snoopy2.gif','DelayTime',0.2,'LoopCount',Inf);
                                           % 将 TIF 图像存为 GIF 图像,帧间播放间隔 0.2s
```

程序运行,在当前目录下存储新图像 `snoopy1.bmp` 和 `snoopy2.gif`。

3.2 图像类型的转换

如第 1 章所讲,图像有二值图像、灰度图像、彩色图像等不同类型,当其色彩数目小于或等于 256 时,又常存储为索引图像。在图像处理系统中,从输入图像到得到最终结果,图像的表现形式也在不

断地发生变化,即不同类型的图像可以通过图像处理算法来转换,以满足图像处理系统的需求。

3.2.1 彩色图像转换为灰度图像

将彩色图像转换为灰度图像,称为灰度化。彩色图像信息量大,数据量也大,在某些情况下,为了简化算法,需要进行灰度化。

灰度化一般是用像素的亮度值作为像素值,亮度值的计算可以通过变换颜色模型来计算,如式(3-1)、式(3-2)、式(3-3)所示。

$$Y = 0.299 \times R + 0.587 \times G + 0.114 \times B \quad (3-1)$$

$$I = (R + G + B) / 3 \quad (3-2)$$

$$V = \max(R, G, B) \quad (3-3)$$

记录每个像素的 Y 、 I 或 V 值,则把彩色图像转换为灰度图像。也可以采用保留彩色图像不同色彩通道的数据的方法。

MATLAB 中的 `rgb2gray`、`im2gray` 函数利用式(3-1)将色彩值转换为亮度值,但 `im2gray` 可以接受灰度图像输入。调用格式如下:

- (1) `I=rgb2gray(RGB)`: 将真彩色图像 RGB 转换为灰度图像。
- (2) `NEWMAP=rgb2gray(MAP)`: 将 MAP 颜色映射表转换为灰度颜色映射表 NEWMAP。
- (3) `I=im2gray(RGB)`: 如果输入的 RGB 实际是灰度图像,则输出 I 和输入一样。

另外,可以采用 `imsplit` 函数将彩色图像的 3 个色彩通道转换为单个通道,每个通道都可以看作一幅灰度图像的数据。其调用格式如下。

- (4) `[c1,c2,c3,...,ck]=imsplit(I)`: 将 $M \times N \times k$ 的图像 I 拆分为 k 个二维矩阵。

【例 3-15】 打开彩色图像,采用不同的方法将其灰度化,查看效果。

程序如下:

```
clear,clc,close all;
[Image1,MAP1] = imread('snoopy.gif',1);
Image2 = im2double(imread('house.jpg')); % 将读取的 uint8 型数据转换为 double 型
MAP2 = rgb2gray(MAP1); % 将颜色映射表灰度化
Y = rgb2gray(Image2);
r = Image2(:, :, 1); g = Image2(:, :, 2); b = Image2(:, :, 3); % 不同颜色通道,或[r,g,b] =
% imsplit(Image2);
I = (r + g + b) / 3; % 利用式(3-2)灰度化
subplot(231), imshow(Image1,MAP2), title('灰度化颜色映射表');
subplot(232), imshow(Y), title('rgb2gray 函数输出');
subplot(233), imshow(I), title('亮度 I 输出');
subplot(234), imshow(r), title('红色通道');
subplot(235), imshow(g), title('绿色通道');
subplot(236), imshow(b), title('蓝色通道');
```

运行程序,输出图像如图 3-12 所示,颜色映射表 MAP1 和 MAP2 的两项取值如表 3-14 所示。MAP1 中各颜色分量不同,呈现彩色;MAP2 中各颜色分量相同,呈现灰色。程序中,读取彩色图像时,需要将数据转换为 double 型,若不转换,在计算 $I = (r + g + b) / 3$ 时,数据会溢出,将导致亮度 I 的计算不正确。



(a) 灰度化颜色映射表



(b) rgb2gray函数输出



(c) 亮度I输出



(d) 红色通道



(e) 绿色通道



(f) 蓝色通道

图 3-12 彩色图像灰度化

表 3-14 例 3-15 中 MAP1 和 MAP2 取值对比

变 量	尺 寸	类 型	前两项取值		
MAP1	256×3	double	0	0.0039	0.0078
			0.0392	0.5059	0.7451
MAP2	256×3	double	0.0032	0.0032	0.0032
			0.3937	0.3937	0.3937

3.2.2 多值图像转换为二值图像

将彩色图像、灰度图像、索引图像转换为二值图像,也称二值化。通常采用图像分割的方法完成,即把图像分成两个区域,前景用 1、背景用 0 来表示,转换为二值图像,这是比较直接的转换方法;也可以根据具体需求转换,如检测到目标后,把目标区域用 1 来表示,背景部分用 0 来表示,转换为二值图像以便进行模板操作。

MATLAB 中的 `imbinarize` 函数实现灰度图像的二值化,通过设定阈值或采用 OTSU 方法(见第 9 章)自动计算合适的阈值,将亮度值大于阈值的像素变为 1,其余的变为 0,实现二值化。调用格式如下。

(1) `BW=imbinarize(I)`: 采用基于 OTSU 方法的全局阈值实现灰度图像 `I` 的二值化。

(2) `BW=imbinarize(I,METHOD)`: `METHOD` 可选 'global' 和 'adaptive',前者指定 OTSU 方法,后者采用局部自适应阈值方法,实现灰度图像 `I` 的二值化。

(3) `BW=imbinarize(I,T)`: 利用阈值 `T` 实现灰度图像 `I` 的二值化。

对于彩色图像,可以先将其转换为灰度图像再二值化,也可以利用图像的特点设定三维阈值,或者设定特殊条件,将满足条件的像素置为 0(或 1)实现二值化。

对于索引图像,可以将其数据看作灰度图像进行二值化,或者将其转换为彩色图像,进行二值化。

【例 3-16】 分别打开彩色图像、灰度图像、索引图像,实现二值化,查看效果。

程序如下:

```
clear,clc,close all;
Image1 = imread('coins.png');           % 打开灰度图像
result1 = imbinarize(Image1);            % 采用自动全局阈值实现二值化
figure,imshow(result1),title('灰度图像二值化');
Image2 = im2double(imread('rose.jpg')); % 打开彩色图像
[r,g,b] = imsplit(Image2);              % 将三维矩阵分为3个二维矩阵
result2 = zeros(size(b));               % 创建和图像宽高一致的二维零矩阵
result2(b>0.4 & r>0.4) = 1;            % 设定蓝色和红色通道大于阈值0.4的像素为1,实现二值化
figure,imshow(result2),title('彩色图像二值化');
[Image3,MAP3] = imread('kids.tif');     % 打开索引图像
result3 = zeros(size(Image3));
result3(Image3>max(Image3(:))/2) = 1;   % 将索引图像数据矩阵当作灰度图像实现二值化
figure,imshow(result3),title('索引图像二值化');
```

程序运行结果如图 3-13 所示。

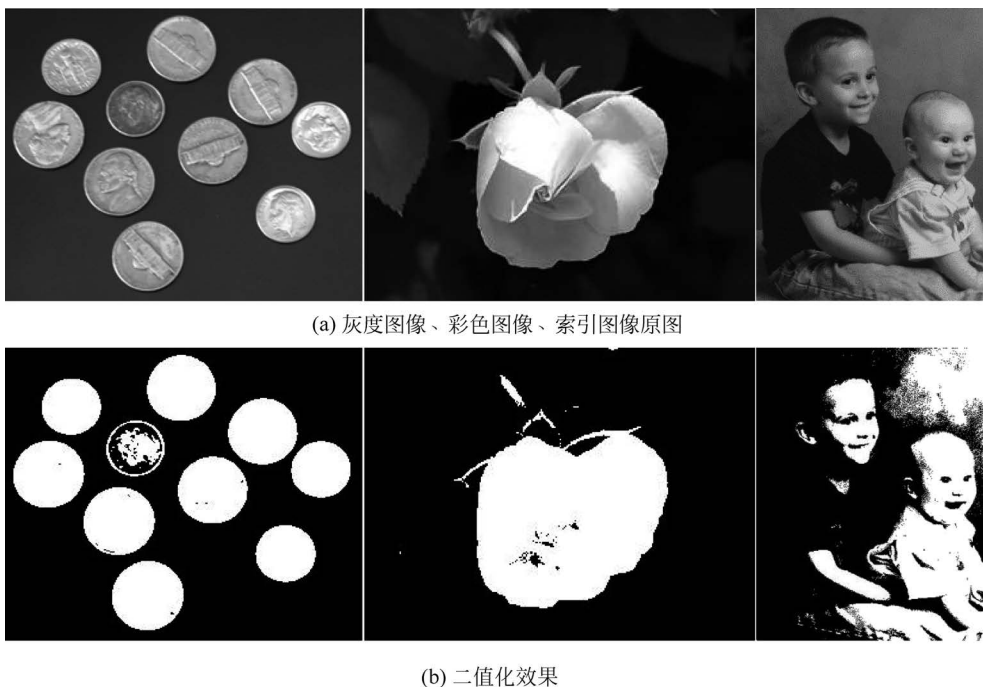


图 3-13 图像二值化

3.2.3 索引图像的转换

MATLAB 提供了相应的函数,用于实现索引图像和真彩色图像、灰度图像之间的互相转换,为便于后续程序设计,本节详细介绍这些转换函数。

1. RGB 图像和索引图像的相互转换

函数 `rgb2ind` 和 `ind2rgb` 用于实现 RGB 图像和索引图像之间的相互转换,调用格式如下。

(1) `[X,MAP]=rgb2ind(RGB,N)`: 采用最小方差量化的方法将 RGB 图像转换为索引图像 X,

MAP 颜色映射表至少包含 N 个颜色, $N \leq 65536$ 。

(2) $X = \text{rgb2ind}(\text{RGB}, \text{MAP})$: 将 RGB 图像转换为索引图像 X , MAP 是 X 的颜色映射表, 转换中将 RGB 中的颜色和 MAP 中最接近的颜色匹配。

(3) $[X, \text{MAP}] = \text{rgb2ind}(\text{RGB}, \text{TOL})$: 利用均匀量化的方法将 RGB 图像转换为索引图像 X , TOL 取值范围为 $0.0 \sim 1.0$, MAP 最多包含 $(\text{FLOOR}(1/\text{TOL}) + 1)^3$ 种颜色。

(4) $[\dots] = \text{rgb2ind}(\dots, \text{DITHER_OPTION})$: 转换时设置 DITHER_OPTION 参数, 选择是否采用颜色抖动, 可取为 'dither' (默认) 和 'nodither', 前者损失空间分辨率获得较好的色彩分辨率, 后者仅将原图中的色彩与新颜色映射表中最接近的颜色匹配。

(5) $\text{RGB} = \text{ind2rgb}(X, \text{MAP})$: 将矩阵 X 和对应的颜色映射表 MAP 转换为 RGB 图像。 X 可以为 uint8 、 uint16 或 double 型数据, RGB 为 $M \times N \times 3$ 的 double 型矩阵。

【例 3-17】 实现 RGB 图像和索引图像的相互转换。

程序如下:

```
clear, clc, close all;
RGB1 = imread('house.jpg');
figure, image(RGB1), title('原图');
[X, MAP] = rgb2ind(RGB1, 16);           % 转换为只有 16 种颜色的索引图像
figure, image(X), colormap(MAP), title('索引图像');
RGB2 = ind2rgb(X, MAP);                 % 逆变换回 RGB 图像
figure, image(RGB2), title('还原的 RGB 图像');
```

运行程序, 各变量如表 3-15 所示。原图 RGB1 各通道有 256 个灰度级别, 转换为索引图像 X 后为 16 种颜色, 对应 MAP 为 16×3 的矩阵。程序运行效果如图 3-14 所示。由于程序中真彩色图像转换为索引图像时减少了颜色, 因此索引图像及再变换回的 RGB2 图像色彩不够细腻, 图像显得比较粗糙。

表 3-15 例 3-17 中各变量取值情况表

变 量	尺 寸	类 型	最 小 值	最 大 值
RGB1	$234 \times 352 \times 3$	uint8	0	255
RGB2	$234 \times 352 \times 3$	double	0.0039	0.9686
X	234×352	uint8	0	15
MAP	16×3	double	0.0039	0.9686



图 3-14 RGB 图像和索引图像的相互转换

2. 灰度图像和索引图像的相互转换

gray2ind 和 ind2gray 函数用于实现灰度图像和索引图像之间的相互转换, 其调用格式如下。

(1) $[X, \text{MAP}] = \text{gray2ind}(I, N)$: 将灰度图像 I 转换为索引图像 X ; 颜色映射表 MAP 为 $N \times 3$

灰度映射表, $N \leq 256$ 时, X 为 uint8 型数据, 否则为 uint16 型数据。 N 应当是 1~65536 的整数, 默认情况下, N 为 64。

(2) $[X, \text{MAP}] = \text{gray2ind}(\text{BW}, N)$: 将二值图像 BW 转换为索引图像 X, 颜色映射表 MAP 为 $\text{gray}(N)$ 。默认情况下, N 为 2。

(3) $I = \text{ind2gray}(X, \text{MAP})$: 将索引图像 X 转换为灰度图像 I, 去掉了色调和饱和度信息, 仅保留亮度值。

【例 3-18】 实现灰度图像和索引图像的相互转换。

程序如下:

```
clear, clc, close all;
I = imread('cameraman.tif');           % 读取灰度图像
[X, MAP] = gray2ind(I, 16);            % 转换为有 16 种颜色的索引图像
figure, imshow(X, MAP), title('索引图像');
J = ind2gray(X, MAP);                 % 转换回灰度图像
figure, imshow(J), title('还原的灰度图像');
```

运行程序, 各变量如表 3-16 所示。原图 I 有 256 个灰度级别, 转换为索引图像 X 后为 16 个灰度级, 对应 MAP 为 16×3 的矩阵。运行结果如图 3-15 所示。由于程序中灰度图像转换为索引图像时减少了灰度级, 因此索引图像及再转换回的灰度图像显得比较粗糙。

表 3-16 例 3-18 中各变量取值情况表

变 量	尺 寸	类 型	最 小 值	最 大 值
I	256×256	uint8	7	253
J	256×256	uint8	0	255
X	256×256	uint8	0	15
MAP	16×3	double	0	1



(a) 原图



(b) 16级灰度的索引图像

图 3-15 灰度图像和索引图像的相互转换

3.3 色彩空间转换

颜色有多种表示方式, 可以根据需要将图像转换到不同的色彩空间, 便于处理。本节介绍图像在 RGB 空间和 HSV、YCbCr、YIQ、LAB 空间之间的转换。

3.3.1 RGB空间和HSV空间的转换

MATLAB 提供了函数 `rgb2hsv` 和 `hsv2rgb` 用于实现图像在 RGB 和 HSV 空间之间的转换,其调用格式如下。

(1) $H = \text{rgb2hsv}(M)$: 将 RGB 颜色映射表 M 转换为 HSV 颜色映射表 H 。 M 、 H 均为取值为 $0 \sim 1$ 的 $P \times 3$ 矩阵, M 的每一行为一种颜色的 RGB 分量; H 的每一行为一种颜色的 HSV 分量。

(2) $\text{HSV} = \text{rgb2hsv}(\text{RGB})$: 将 RGB 图像转换为 HSV 图像。RGB 是 `uint8`、`uint16`、`double` 型数据时, HSV 为 $0 \sim 1$ 的 `double` 型数据; RGB 为 `single` 数据时, HSV 也为 `single` 型数据。

(3) $M = \text{hsv2rgb}(H)$: 将 HSV 颜色映射表 H 转换为 RGB 颜色映射表 M 。

(4) $\text{RGB} = \text{hsv2rgb}(\text{HSV})$: 将 HSV 图像转换为 RGB 图像。当 HSV 为 `logical`、`double` 型数据时, 输出的 RGB 为 `double` 型数据; 当 HSV 为 `single` 型数据时, RGB 也为 `single` 型数据。

【例 3-19】 将图像在 RGB 和 HSV 空间之间转换, 尝试修改饱和度和亮度值, 查看效果。

程序如下:

```
clear, clc, close all;
Image1 = imread('montreal.jpg');           % 打开彩色图像
[Image2, MAP1] = rgb2ind(Image1, 256);     % 转换为 256 级的索引图像
H = rgb2hsv(MAP1);                         % 将颜色映射表转换到 HSV 空间
H(:, 2) = H(:, 2) * 2;   H(H > 1) = 1;     % 饱和度增强为原来的 2 倍, 超出范围的限幅为 1
MAP2 = hsv2rgb(H);                         % 将增强饱和度后的颜色映射表转换回 RGB 空间
H(:, 3) = H(:, 3) * 1.5;   H(H > 1) = 1;   % 亮度增强为原来的 1.5 倍, 超出范围的限幅为 1
MAP3 = hsv2rgb(H);                         % 将增强饱和度和亮度后的颜色映射表转换回 RGB 空间
HSV = rgb2hsv(Image1);                    % 将原真彩色图像转换到 HSV 空间
HSV(:, :, 3) = HSV(:, :, 3) * 1.5;        % 将亮度增强为原来的 1.5 倍
Image3 = hsv2rgb(HSV);                     % 将增强亮度后的 HSV 数据转换回 RGB 空间
subplot(221), imshow(Image1), title('原图');
subplot(222), imshow(Image2, MAP2), title('增强饱和度');
subplot(223), imshow(Image3), title('增强亮度');
subplot(224), imshow(Image2, MAP3), title('增强饱和度和亮度');
```

程序运行效果如图 3-16 所示。



图 3-16 RGB 空间和 HSV 空间的相互转换

程序将 RGB 图像 `Image1` 转换为索引图像, 并将颜色映射表 `MAP1` 转换为 HSV 空间的 H , H 为 256×3 的矩阵, 其 3 列分别为每一种颜色的色调、饱和度和亮度的值, 对饱和度进行 2 倍拉伸, 反转换回 RGB 空间的颜色映射表 `MAP2`, 转换后图像色彩相对原图鲜艳, 如图 3-16(b) 所示; 再对 H

的亮度进行 1.5 倍拉伸,反转换回 RGB 空间的颜色映射表 MAP3,图像整体变亮,颜色变得鲜艳,如图 3-16(d)所示。将图像矩阵 Image1 转换到 HSV 空间的三维矩阵 HSV,对其第三维(亮度)进行 1.5 倍拉伸,反转换回 RGB 空间,如图 3-16(c)所示,仅亮度增加。

3.3.2 RGB 空间和 YCbCr 空间的转换

MATLAB 提供了 rgb2ycbcr 和 ycbcr2rgb 函数用于实现图像在 RGB 和 YCbCr 空间之间的转换,其调用格式如下。

(1) YCBCRMAP=rgb2ycbcr(MAP): 将 RGB 颜色映射表 MAP 转换为 YCbCr 颜色映射表 YCBCRMAP。YCBCRMAP 为 $P \times 3$ 矩阵,每一行对应 MAP 中同一行的颜色,3 列元素分别为该颜色的 Y、Cb、Cr 分量。

(2) YBCR=rgb2ycbcr(RGB): 将 RGB 图像转换为 YCbCr 图像,RGB 是 $M \times N \times 3$ 的矩阵。若 RGB 为 uint8 型数据,则 $Y \in [16, 235]$, $Cb, Cr \in [16, 240]$; 若输入为 double 或 single 型数据,则 $Y \in [16/255, 235/255]$, $Cb, Cr \in [16/255, 240/255]$; 若输入为 uint16 型数据,则 $Y \in [4112, 60395]$, $Cb, Cr \in [4112, 61680]$ 。

(3) RGBMAP=ycbcr2rgb(YCBCRMAP): 将 YCbCr 空间的颜色映射表 YCBCRMAP 转换为 RGB 空间的 RGBMAP。

(4) RGB=ycbcr2rgb(YBCR): 将 YCbCr 图像转换为 RGB 图像。

【例 3-20】 将图像在 RGB 和 YCbCr 空间之间转换,尝试修改数值,并查看效果。

程序如下:

```
clear,clc,close all;
Image1 = imread('montreal.jpg');
YBCR = rgb2ycbcr(Image1); % RGB 图像转换到 YCbCr 空间
YBCR(:, :, 1) = YBCR(:, :, 1) * 1.6; % 增强亮度值
Image2 = ycbcr2rgb(YBCR); % 反转换回 RGB 空间
[Image3,MAP1] = imread('kids.tif'); % 打开索引图像
YCBCRMAP = rgb2ycbcr(MAP1); % 颜色映射表 MAP1 转换到 YCbCr 空间
YCBCRMAP(:, 1) = YCBCRMAP(:, 1) * 1.1; % 略增强亮度 Y
YCBCRMAP(:, 3) = YCBCRMAP(:, 3) * 0.95; % 弱化 Cr 值
YCBCRMAP(YCBCRMAP > 1) = 1; % 限幅
MAP2 = ycbcr2rgb(YCBCRMAP); % 颜色映射表反转换回 RGB 空间
subplot(221), imshow(Image1), title('真彩色图原图');
subplot(222), imshow(Image2), title('增强亮度');
subplot(223), imshow(Image3,MAP1), title('索引图原图');
subplot(224), imshow(Image3,MAP2), title('增强 Y, 弱化 Cr');
```

程序运行效果如图 3-17 所示。程序中将 RGB 图像 Image1 转换到 YCbCr 空间,将亮度 Y 增大为原来的 1.6 倍,反转换后图像的效果如图 3-17(b)所示,亮度增强。MAP1 为索引图像的颜色映射表,转换为 YCbCr 空间的 YCBCRMAP, YCBCRMAP 为 256×3 的矩阵,其 3 列分别为每一种颜色的 Y、Cb 和 Cr 值,将亮度 Y 增强为原来的 1.1 倍,Cr 值弱化为原来的 0.95 倍,反转换回 RGB 空间的颜色映射表 MAP2,转换后图像在一定程度上修正了原图偏红的色彩,如图 3-17(d)所示。

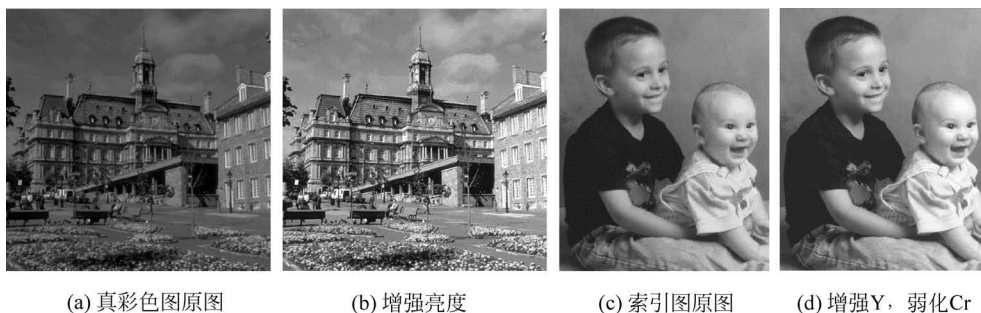


图 3-17 RGB 空间和 YCbCr 空间的相互转换

3.3.3 RGB 空间和 YIQ 空间的转换

MATLAB 提供了 `rgb2ntsc` 和 `ntsc2rgb` 函数用于实现图像在 RGB 和 YIQ 空间之间的转换,其调用格式如下。

(1) `YIQMAP=rgb2ntsc(RGBMAP)`: 将 RGB 颜色映射表 RGBMAP 转换为 YIQ 颜色映射表 YIQMAP。YIQMAP 为 $P \times 3$ 矩阵,每一行对应 RGBMAP 中同一行的颜色,3 列元素分别为该颜色的 Y、I、Q 分量。

(2) `YIQ=rgb2ntsc(RGB)`: 将 RGB 图像转换为 YIQ 图像。

(3) `RGBMAP=ntsc2rgb(YIQMAP)`: 将 YIQ 颜色映射表 YIQMAP 转换为 RGB 颜色映射表 RGBMAP。YIQMAP 和 RGBMAP 均为 $P \times 3$ 的 double 型矩阵。

(4) `RGB=ntsc2rgb(YIQ)`: 将 YIQ 图像转换为 RGB 图像。

【例 3-21】 将图像在 RGB 和 YIQ 空间之间转换,并查看数据。

程序如下:

```
clear,clc,close all;
RGB = imread('montreal.jpg');
YIQ = rgb2ntsc(RGB); % 真彩色图像转换为 YIQ 数据
[X,RGBMAP] = imread('kids.tif');
YIQMAP = rgb2ntsc(RGBMAP); % 索引图像颜色映射表转换为 YIQ 颜色映射表
```

运行程序,各变量取值情况如表 3-17 所示,RGB 数据为 `uint8` 型,YIQ 数据为 `double` 型,图像尺寸一致;RGBMAP 和 YIQMAP 均为 `double` 型 256×3 的矩阵。

表 3-17 例 3-21 中各变量取值

变 量	尺 寸	类 型	最 小 值	最 大 值
RGB	$512 \times 512 \times 3$	<code>uint8</code>	—	—
YIQ	$512 \times 512 \times 3$	<code>double</code>	—	—
X	400×318	<code>uint8</code>	0	63
RGBMAP	256×3	<code>double</code>	0	0.9961
YIQMAP	256×3	<code>double</code>	-0.0577	0.8500

3.3.4 RGB空间和LAB空间的转换

MATLAB 提供了函数 `rgb2lab` 和 `lab2rgb` 用于实现图像在 RGB 和 CIE 1976 $L^*a^*b^*$ 空间之间的转换,其调用格式如下。

(1) `lab=rgb2lab(rgb)`: 将 RGB 值转换为 CIE 1976 $L^*a^*b^*$ 值。

(2) `rgb=lab2rgb(lab)`: 将 CIE 1976 $L^*a^*b^*$ 值转换为 RGB 值。

【例 3-22】 将图像在 RGB 空间和 LAB 空间之间进行转换,计算并查看色差图像。

程序如下:

```
clear,clc,close all;
RGB = imread('flower.jpg');
[N,M,C] = size(RGB);
LAB = rgb2lab(RGB); % 转换到 LAB 空间
L = LAB(:, :, 1); A = LAB(:, :, 2); B = LAB(:, :, 3);
delta = zeros(N,M);
for i = 2:M-1
    for j = 2:N-1
        for m = -1:1 % 计算每个像素和周围 8 个点的色差和
            for n = -1:1
                delta(j,i) = delta(j,i) + sqrt((L(j,i) - L(j+n,i+m))^2 +
                    (A(j,i) - A(j+n,i+m))^2 + (B(j,i) - B(j+n,i+m))^2);
            end
        end
    end
end
delta = delta/max(delta(:));
LAB(:, :, 1) = L * 0.8; % 降低亮度
result = lab2rgb(LAB); % 转换回 RGB 色彩空间
subplot(131), imshow(RGB), title('真彩色图原图');
subplot(132), imshow(delta, []), title('色差图像');
subplot(133), imshow(result), title('降低亮度逆变换');
```

程序运行效果如图 3-18 所示。

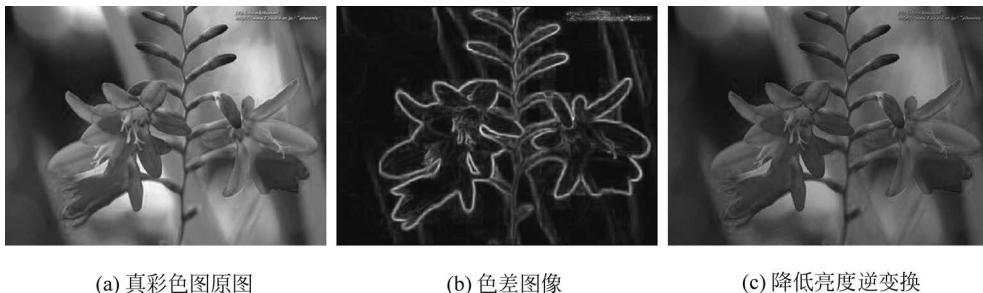


图 3-18 RGB 空间和 LAB 空间的相互转换

3.4 视频文件的读写

数字视频,又称动态图像,是多帧位图的有序组合,在很多情况下,需要处理的对象正是数字视

频。本节介绍 MATLAB 提供的对视频文件进行操作的相关函数。

3.4.1 视频文件信息读取

MATLAB 提供的 mmfileinfo 函数能够获取多媒体文件的信息,其调用格式如下。

INFO=mmfileinfo(FILENAME): 获取 FILENAME 指定的多媒体文件的音频、视频信息,存储于 INFO 结构体中。INFO 结构体各变量如表 3-18 所示。

表 3-18 mmfileinfo 函数返回结构体各变量

变 量	含 义	内 容	
Filename	文件名	一个字符串	
Path	文件绝对路径	一个字符串	
Duration	文件时长,单位为秒		
Audio	一个结构体,文件中的音频信息	Format	音频格式
		NumberOfChannels	音频通道数
Video	一个结构体,文件中的视频信息	Format	视频格式
		Height	帧高
		Width	帧宽

【例 3-23】 读取两种不同格式的视频文件,并查看返回结构体数据。

程序如下:

```
clear,clc,close all;
INFO1 = mmfileinfo('tilted_face.avi');
INFO2 = mmfileinfo('xylophone.mpg');
```

程序运行结果如表 3-19 所示。

表 3-19 例 3-23 中各变量取值

变量	分量	取 值		变量	分量	取 值	
INFO1	Filename	'tilted_face.avi'		INFO2	Filename	'xylophone.mpg'	
	Path	'E:\MATLAB\3 Chapter'			Path	'E:\MATLAB\3 Chapter'	
	Duration	13.7667			Duration	4.7020	
	Audio	Format	' '		Audio	Format	'MPEG'
		NumberOfChannels	[]			NumberOfChannels	2
	Video	Format	'MJPG'		Video	Format	'MPEG1'
		Height	480			Height	240
		Width	640			Width	320

3.4.2 视频文件数据读取

VideoReader 函数创建一个多媒体读取对象,进而能读取视频数据,其调用格式如下。

(1) OBJ=VideoReader(FILENAME): 创建一个多媒体读取对象 OBJ,可以通过 OBJ 读取多媒体文件的视频数据。

(2) OBJ=VideoReader(FILENAME,'P1',V1,'P2',V2,...): 设置相关属性创建多媒体读取对

象 OBJ, 相关属性见表 3-20。

表 3-20 函数 VideoReader 属性

属 性	含 义	属 性	含 义
Name	被读取的文件名	Height	帧高, 单位为像素
Path	被读取的文件路径	Width	帧宽, 单位为像素
Duration	文件时长, 单位为秒	BitsPerPixel	每像素所占位数
CurrentTime	当前读取帧在文件中的位置	VideoFormat	视频格式: 取值可以为 'RGB24'、'Grayscale'、'Indexed'
Tag	用户设置的一般字符串		
UserData	用户自定义数据	FrameRate	帧率

多媒体读取对象 OBJ 有 read、readFrame、hasFrame、VideoReader、getFileFormats 4 种方法, read 方法从视频文件中读取若干帧, readFrame 方法读取下一个可读帧, hasFrame 方法判断是否有可读取的帧, VideoReader.getFileFormats 方法用于查看系统中 VideoReader 支持的格式。各自的调用格式如下。

(1) FLAG=hasFrame(OBJ): 有可读帧, 则返回 true; 否则返回 false。

(2) VIDEO=readFrame(OBJ): 读取下一个可读帧, 返回 VIDEO。VIDEO 是 $H \times W \times B$ 的矩阵, H 、 W 、 B 依次为帧图像的高、宽、色彩通道数, OBJ 的 VideoFormat 指定格式返回数据。

(3) VIDEO=readFrame(OBJ, 'native'): 读取下一个可读帧。指定参数 'native' 时, 返回值与默认情况下有所区别, 如表 3-21 所示。

表 3-21 VideoFormat 属性与 VIDEO 返回值

VideoFormat	指定 'native'	VIDEO 数据类型	VIDEO 维数	描 述
'RGB24'	否	uint8	$M \times N \times 3$	真彩色图像
'Grayscale'		uint8	$M \times N \times 1$	灰度图像
'Indexed'		uint8	$M \times N \times 3$	真彩色图像
'RGB24'	是	uint8	$M \times N \times 3$	真彩色图像
'Grayscale'		struct	1×1	MATLAB movie *
'Indexed'		struct	1×1	MATLAB movie *

注: MATLAB movie 是一个帧结构体矩阵, 包含 "cdata" 和 "colormap" 两个字段, 分别为帧图像数据矩阵和颜色映射表。后文直接采用 MATLAB movie 来表示。

(4) VIDEO=read(OBJ): 读取所有帧, VIDEO 是 $H \times W \times B \times F$ 的矩阵, F 表示帧数。

(5) VIDEO=read(OBJ, INDEX): 读取指定的帧。INDEX 可以是单个数或者指定帧范围的二维数组, Inf 表示最后一帧。

(6) VIDEO=read(OBJ, 'native'): 参数 'native' 含义同前所述。

(7) FORMATS=VideoReader.getFileFormats(): 获取 VideoReader 支持的文件格式。

【例 3-24】 使用 VideoReader 函数创建多媒体读取对象, 读取并显示视频, 查看返回数据。

程序如下:

```
clear, clc, close all;
OBJ = VideoReader('atrium.mp4'); % 创建 OBJ 多媒体读取对象
OBJ.CurrentTime = 0.1;          % 设置开始读取帧的位置
currAxes = axes;                % 在当前 figure 下使用默认属性值创建一个坐标系图形对象
while hasFrame(OBJ)             % 用循环语句读取每一帧, 有可读帧则读取
```

```
Frame = readFrame(OBJ);           % 读取可读帧,返回给 Frame
image(Frame, 'Parent', currAxes); % 显示当前帧
currAxes.Visible = 'off';         % 不显示坐标
pause(1/OBJ.FrameRate);           % 帧和帧之间的时间间隔
end
```

运行程序,在 figure 中显示视频画面,如图 3-19 所示。程序中 OBJ 变量取值如表 3-22 所示;Frame 变量为 $360 \times 640 \times 3$ 的 uint8 型数据。



图 3-19 视频显示画面

表 3-22 例 3-24 中 OBJ 取值情况表

属 性	取 值	属 性	取 值
Name	'atrium. mp4'	Height	360
Path	'E:\MATLAB\3 Chapter'	Width	640
Duration	20. 0000	BitsPerPixel	24
CurrentTime	20. 0000	VideoFormat	'RGB24'
Tag	''	FrameRate	30
UserData	[]		

3.4.3 视频的播放

在例 3-24 中,采用循环语句依次读取并显示视频中的每一帧,帧和帧之间暂留时间小于视觉暂留时间,则看到连贯流畅的动态画面。除此之外,MATLAB 还提供了 movie 函数,用于实现视频的播放。本节介绍 movie 函数及与其相关的函数。

1. getframe 函数

getframe 函数通过捕捉当前坐标系下的快照来获取 MATLAB movie 中的帧,一般用于循环语句中,以获取多帧,其调用格式如下。

- (1) getframe(H): 从句柄 H 指定的 figure 或坐标系中获取一帧。
- (2) getframe(H,RECT): 从指定的矩形中获取位图数据,RECT 相对于句柄 H 指定对象的左下角定义,单位为像素。
- (3) F=getframe(...): F 表示包含两个字段"cdata"和"colormap"的 MATLAB movie 结构体。其中,"cdata"为 uint8 型 $H \times W \times 3$ 的图像数据矩阵,"colormap"是 double 型矩阵;若系统采用真彩色图形显示方式,则 F 的"colormap"为空。

2. movie 函数

movie 函数用来播放视频帧,其调用格式如下。

(1) movie(M): 在当前坐标系下播放 M 中的 MATLAB movie 一次,M 可以通过 getframe 函数获取。

(2) movie(M,N): 将 M 播放 N 次。若 N 为负数,则每次播放包含正向一次,逆向一次;若 N 是一个向量,则第一个元素是播放次数,其余元素为要播放的帧,例如,M 包含 4 帧,而 $N=[10\ 4\ 4\ 2\ 1]$,则播放 MATLAB movie 十次,每次按第 4 帧、第 4 帧、第 2 帧、第 1 帧播放。

(3) movie(M,N,FPS): 以 FPS 帧每秒的速度播放视频,FPS 默认时为 12 帧每秒,若达不到指定的速度,则以能达到的最快速度播放。

(4) movie(H,...): 在对象 H 中播放 MATLAB movie,H 可以是 figure 或坐标系的句柄。

(5) movie(H,M,N,FPS,LOC): 指定位置播放 MATLAB movie。LOC 是位置向量[X Y 未使用 未使用],X、Y 相对位于对象 H 的左下角,单位为像素;后 2 维虽未使用,但必须有。

【例 3-25】 使用 getframe 函数获取 MATLAB movie 数据并播放。

程序如下:

```
clear,clc,close all;
[Image1,MAP1] = imread('snoopy.gif','Frames','all'); % 打开 GIF 图像
info = imfinfo('snoopy.gif');
len = length(info); % 获取 GIF 图像帧数
for j = 1:len
    imshow(Image1(:,:,j),MAP1); % 依次显示 GIF 中各帧
    M(j) = getframe; % 获取 MATLAB movie 帧
    pause(1/25); % 画面显示时间间隔
end
figure,movie(M); % 新建 figure 下显示 MATLAB movie
figure,movie(M,[6 len 1 floor(len/2)]); % 将 M 中第 len、1、len/2 帧依次显示 6 遍
figure, currAxes = axes; % 不显示坐标系
movie(M,3,3,[50 50 300 248]); % 将 M 中各帧在位置[50 50]处以 3 帧每秒的速度显示 3 遍
```

运行程序,依次显示 4 段视频: figure 1 中是循环语句,依次显示 GIF 中各帧; figure 2 中在左下角显示视频; figure 3 中将 M 中第 5、1、2 帧依次显示 6 遍; figure 4 中将 M 中各帧在位置[50 50]处以 3 帧每秒的速度显示 3 遍。播放停止后的画面如图 3-20 所示。

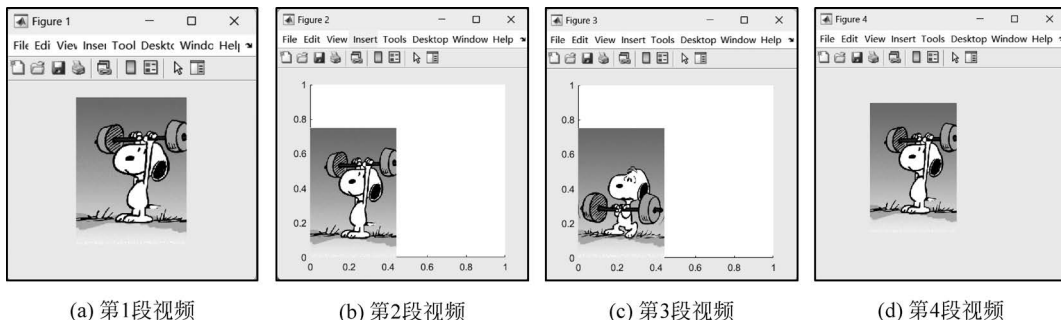


图 3-20 视频显示画面

3. im2frame 和 frame2im 函数

im2frame 函数用于将索引图像转换为 MATLAB movie 格式；frame2im 函数用于返回 MATLAB movie 帧的图像数据,其调用格式如下。

(1) $F = \text{im2frame}(X, \text{MAP})$: 将索引图像 X 及其颜色映射表 MAP 转换为 MATLAB movie 帧 F 。若 X 为真彩色图像,则 MAP 是可选项但无效。

(2) $F = \text{im2frame}(X)$: 将图像 X 转换为 MATLAB movie 帧 F 。若 X 为索引图像,则使用当前颜色映射表。

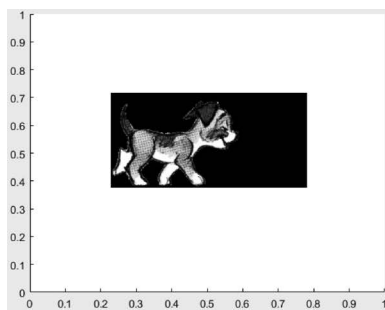
(3) $[X, \text{MAP}] = \text{frame2im}(F)$: 从 MATLAB movie 帧 F 返回索引图像 X 及其颜色映射表 MAP 。若 F 为真彩色,则 MAP 为空。

【例 3-26】 使用 im2frame 函数获取 MATLAB movie 数据,播放视频并查看数据。

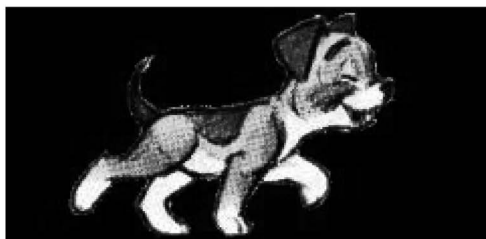
程序如下:

```
clear,clc,close all;
[Image1,MAP1] = imread('dog.gif','Frames','all');
info = imfinfo('dog.gif');
len = length(info);
for j = 1:len
    M(j) = im2frame(Image1(:,:,j),MAP1);    % 将 GIF 图像的每一帧转换为 MATLAB movie 帧
end
figure,movie(M,-2,4,[100 130 240 117]);
    % 将 M 中各帧在位置[100 130]处以 4 帧每秒的速度显示 2 遍,每遍正向一次,逆向一次
[image2,MAP2] = frame2im(M(floor(len/2)));    % 将中间帧转换为索引图像 image2
figure,imshow(image2,MAP2),title('中间帧');
```

运行程序,在 figure 1 中播放两遍视频,每遍正向播放一次,逆向播放一次;把中间帧转换为索引图像 image2 并显示,如图 3-21 所示。



(a) 视频播放画面



(b) 中间帧

图 3-21 视频显示画面

3.4.4 视频文件的保存

MATLAB 主要利用 VideoWriter 函数实现视频文件的存储。VideoWriter 函数用于创建一个视频写入对象,其调用格式如下。

(1) $\text{OBJ} = \text{VideoWriter}(\text{FILENAME})$: 创建一个视频写入对象,将视频数据写入 FILENAME

指定的 AVI 文件。若 FILENAME 未包含扩展名 '.avi', 则函数会自动附加。

(2) OBJ=VideoWriter(FILENAME,PROFILE): 设定属性创建一个视频写入对象, 如表 3-23 所示。PROFILE 可取 'Archival'、'Motion JPEG AVI'、'Motion JPEG 2000'、'MPEG-4'、'Uncompressed AVI'、'Indexed AVI'、'Grayscale AVI', 对应不同的视频压缩编码方法。

表 3-23 函数 VideoWriter 属性

属 性	含 义
Path	文件完全路径
Filename	文件名
Duration	文件时长, 单位为秒
FileFormat	字符串, 文件类型
Colormap	$P \times 3$ 颜色映射表
FrameCount	帧数
FrameRate	帧率
Height	帧高
Width	帧宽
LosslessCompression	布尔值, 为 true, 无损压缩, 则 CompressionRatio 参数无效, 用于 Motion JPEG 2000 文件中
ColorChannels	输出视频帧的色彩通道数
MJ2BitDepth	输入图像数据中最低有效位数, 取值为 1~16, 仅用于 Motion JPEG 2000 文件中
Quality	0~100 的整数, 仅用于 Motion JPEG AVI 和 MPEG-4 配置中; 取值越大, 视频质量越好但文件越大
CompressionRatio	压缩比, 仅用于 Motion JPEG 2000 文件中
VideoBitsPerPixel	输出视频帧每像素位数
VideoCompressionMethod	字符串, 指定视频压缩类型
VideoFormat	字符串, 视频格式

视频写入对象 OBJ 有 open、close、writeVideo、getProfiles 4 种方法。可以根据需要设置 OBJ 的 Colormap、FrameRate、LosslessCompression、Quality、CompressionRatio 等属性, 但必须在调用 open 函数前。4 种方法各自的调用格式如下。

(1) open(OBJ): 打开要写入视频数据的文件, 必须打开才能写入。

(2) close(OBJ): 在写入结束后关闭文件。

(3) writeVideo(OBJ,FRAME): 将 FRAME 写入与 OBJ 关联的视频文件。FRAME 是一个包括 cdata 和 colormap 字段的结构体, 可以通过 getframe 函数获取。文件中每一帧的宽、高应一致。

(4) writeVideo(OBJ,MOV): 将 MOV 指定的 MATLAB movie 写入视频文件。

(5) writeVideo(OBJ,IMAGE): 将 IMAGE 中的数据写入一个视频文件。IMAGE 可以为 single、double、uint8 型矩阵。

(6) writeVideo(OBJ,IMAGES): 将一系列彩色图像数据写入视频文件。IMAGES 是四维矩阵: 高×宽×1×帧数的灰度图像或者高×宽×3×帧数的彩色图像。

(7) PROFILES=VideoWriter.getProfiles(): 返回 VideoWriter 支持的配置。

【例 3-27】 读取 GIF 图像, 保存为 AVI 文件。

程序如下:

```
clear,clc,close all;
```

```
[Image1,MAP1] = imread('fly.gif','Frames','all');
info = imfinfo('fly.gif');
len = length(info);
vidObj = VideoWriter('fly.avi');           % 创建视频写入对象
vidObj.FrameRate = 10;                     % 修改帧率
open(vidObj);                             % 打开要写入的视频文件
for i = 1:len
    Image2(:,:,i) = ind2rgb(Image1(:,:,i),MAP1); % 将 GIF 帧转换为真彩色图像
end
for j = 1:len * 3
    for i = 1:len
        Images(:,:,len * j + i) = Image2(:,:,i);
    end
    % 重复 GIF 帧数据,存入 Images 图像序列中,增大帧数
end
writeVideo(vidObj,Images);                 % 将 Images 中的图像序列写入视频文件
close(vidObj);                             % 关闭视频文件
```

运行程序,创建的视频写入对象取值如表 3-24 所示,在当前路径下增加了 fly.avi 文件。

表 3-24 例 3-27 中 vidObj 取值情况表

属 性	取 值	属 性	取 值
Duration	20	Height	248
Filename	'fly.avi'	Width	300
Path	'E:\MATLAB\3 Chapter'	VideoFormat	'RGB24'
FileFormat	'avi'	Quality	75
ColorChannels	3	VideoBitsPerPixel	24
FrameCount	200	VideoCompressionMethod	'Motion JPEG'
FrameRate	10		

3.5 实例

【例 3-28】 实现电影中常见的去彩色效果。

设计思路:

读取一段视频,按照时间剪切其中的一段,将其中的每一帧灰度化,转换为灰度视频段;同时将该段视频的每一帧转换到 HSV 空间,提取其亮度通道,转换为亮度视频段;最后将灰度视频段和亮度视频段分别保存为 AVI 文件。

程序如下:

```
clear,clc,close all;
OBJ = VideoReader('vippedtracking.mp4'); % 创建视频读取对象
OBJ.CurrentTime = 1; % 读取视频从第 1 秒开始
j = 1;
while hasFrame(OBJ) && OBJ.CurrentTime < 9 % 判断是否已有可读下一帧,是否在 9s 前
    Frame = readFrame(OBJ); % 读取下一帧
    grayFrame = rgb2gray(Frame); % 灰度化
    hsvFrame = rgb2hsv(Frame); % 转换到 HSV 空间
    M1(:,:,j) = grayFrame; % 灰度视频段图像序列
    M2(:,:,j) = hsvFrame(:,:,3); % 亮度视频段图像序列
    j = j + 1;
end
```

```

end
vidObj1 = VideoWriter('grayvideo.avi');           % 创建视频写入对象
open(vidObj1);
writeVideo(vidObj1,M1);                           % 打开视频文件并写入灰度视频
close(vidObj1);                                   % 关闭视频文件
vidObj2 = VideoWriter('vvideo.avi');              % 创建视频写入对象
open(vidObj2);
writeVideo(vidObj2,M2);                           % 打开视频文件并写入亮度视频
close(vidObj2);                                   % 关闭视频文件
subplot(131), imshow(Frame), title('彩色视频段最后一帧');
subplot(132), imshow(grayFrame), title('灰度视频段最后一帧');
subplot(133), imshow(hsvFrame(:, :, 3)), title('亮度视频段最后一帧');

```

运行程序,将生成两段 AVI 视频 grayvideo.avi 和 vvideo.avi,各段视频最后一帧显示如图 3-22 所示。



(a) 彩色视频段最后一帧



(b) 灰度视频段最后一帧



(c) 亮度视频段最后一帧

图 3-22 各段视频最后一帧

【例 3-29】 实现图像编辑中的保留色彩。

设计思路:

显示一幅图像,用鼠标左键在图像上选择至少 3 个点,根据选择点确定要保留色彩的色调范围;扫描图像,确定色彩在选择范围内的像素,形成模板;同时将图像灰度化。模板与原图相乘,反色的模板和灰度化图像相乘,两者相加,使得模板内的像素保留色彩,模板外的像素呈现灰色。

程序如下:

```

clear,clc,close all;
Image = imread('peony.jpg');
[height,width,color] = size(Image);
figure,imshow(Image),title('选择至少 3 个像素用于确定要保留的色彩范围');
[C,R,P] = impixel(Image);
HSV = rgb2hsv(Image);

```

```
gray = double(rgb2gray(Image));  
len = length(C);      coH = zeros(len,1);  
for i = 1:len  
    coH(i) = HSV(R(i),C(i),1);  
end  
minH = min(coH);      minH = max(minH - 0.2 * minH, 0);  
maxH = max(coH);      maxH = min(maxH + 0.2 * maxH, 1);  
mask = zeros(height,width);  
mask(HSV(:, :, 1) >= minH & HSV(:, :, 1) <= maxH) = 1;  
maskRGB = double(cat(3,mask,mask,mask));  
result = uint8(mask. * double(Image) + (1 - mask). * gray);  
figure, imshow(result);
```

% 选择点的色调值

% 确定色彩范围

% 形成模板

% 生成保留色彩的图像

程序运行结果如图 3-23 所示。



图 3-23 保留色彩实例效果图

本章小结

本章主要介绍了在 MATLAB 中实现图像处理的基本操作,包括图像文件的读写、图像的显示、数据转换、类型转换、色彩空间转换、视频文件的读写、视频的播放等。本章内容是后续图像处理的前提,应熟悉各函数功能,掌握输入输出变量的含义及要求。