数学建模

-Matlab图像处理

主要内容

- 图像格式和类型
- 图像数据的读写
- 图像运算与图像几何变换
- 图像线性变换
- 图像数据的直方图
- 图像锐化
- 练习

一、图像格式和类型

1.图像格式

图像格式是指图像文件的存储格式。

MATLAB中常用的图像格式有：

bmp,cur,gif,hdf,ico,jpg或jpeg,pcx,png,tif或tiff等。

2.MATLAB图像类型

MATLAB支持五种图像类型，即二值图像（黑白图像）、索引图像、灰度图像、RGB图像和多帧图像阵列。

（1）RGB图像：double型或uint8型或uint16型，又称真彩色图像，在MATLAB中存储为m*n*3的数据矩阵，它表示图像中每个像素的红，绿，蓝3个颜色分量的数值。

利用R、G、B三个分量表示一个像素的颜色。取值0~255.

红

25 42 63
64 71 64
64 71 71

绿

177 65 89
80 80 78
80 78 87

蓝

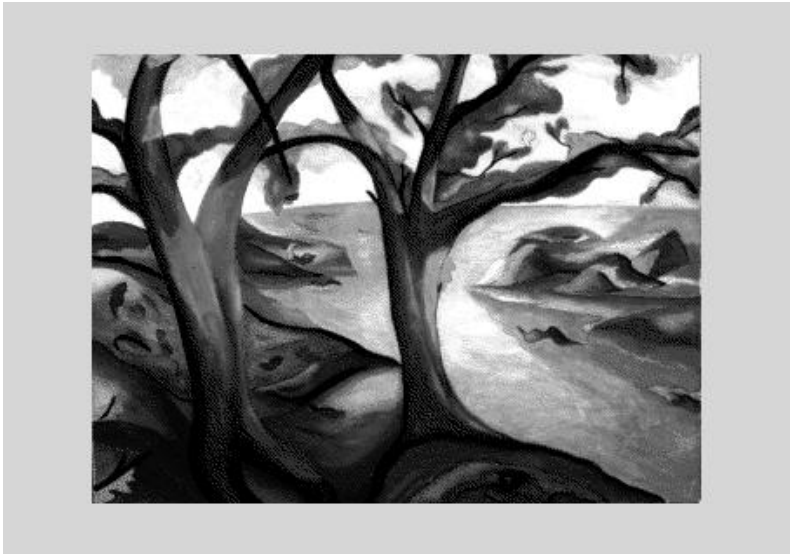
74 112 79
93 73 93
197 93 73

（2）索引图像:对不同颜色进行编号，组成一个调色板，图像数据记录像素对应的调色板颜色的序号。

Copy	1	2	3	4	5	6	7	8	9
1	109	117	99	109	88	91	91	87	91
2	113	113	113	106	113	106	113	106	106
3	113	113	106	113	113	106	106	106	106
4	106	106	106	106	106	113	106	106	94
5	113	106	113	106	106	106	106	106	106
6	106	106	106	113	106	113	106	106	106
7	106	113	106	106	113	106	113	106	106
8	106	106	113	113	113	106	106	106	106
9	106	106	113	115	118	113	113	106	106
10	118	121	124	126	124	113	113	113	106

（3）灰度图像

只有图像的强度信息，没有颜色信息。



（4） 二值图像：元素值为0或1的矩阵，只有黑白两种颜色的图像；

（5）多帧图像阵列：由多帧图像组成，每一帧图像可以为前4种图像中的一种，但组成一个多帧图像阵列的图像必须为同一种。

颜色映象

MATLAB 有一个叫颜色映象的数据结构来代表颜色值。颜色映象定义为一个有三列和若干行的矩阵。利用0到1之间的数，矩阵的每一行都代表了一种色彩。任一行的数字都指定了一个RGB值，即红、黄、蓝三种颜色的强度，形成一种特定的颜色。一些有代表性的RGB值在下表中给出。

Red(红)	Green(绿)	Blue(蓝)	颜色
0	0	0	黑
1	1	1	白
1	0	0	红
0	1	0	绿
0	0	1	蓝
1	1	0	黄
1	0	1	洋红
0	1	1	青蓝
0.667	0.667	1	天蓝
1	0.5	0	橘黄
0.5	0	0	深红
0.5	0.5	0.5	灰色

有十个 MATLAB 函数产生预定的颜色映象：

hsv	色彩饱和值
hot	从黑到红到黄道白
cool	青蓝和洋红的色度
pink	粉红的彩色度
bone	带一点蓝色的灰度
jet	hsv的一种变形(以蓝色开始和结束)
copper	线性铜色度
prim	三棱镜。交替为红,橘黄,黄,绿,天蓝
flag	交替为红,白,蓝,黑
gray	x线性灰度

按缺省，上面所列的各个颜色映象产生一个 64 × 3 的矩阵，指定了 64 种颜色 RGB 的描述。这些函数都接受一个参量来指定所产生矩阵的行数。比如 hot(m) 产生一个 m × 3 的矩阵，它包含的 RGB 颜色值的范围从黑经过红、橘红和黄，到白。

可以用多种途径来显示一个颜色映象。其中一个方法是观察颜色映象矩阵的元素。

```
hot(8)
ans=
0.3333      0           0
0.6667      0           0
1.0000      0           0
1.0000    0.3333        0
1.0000    0.6667        0
1.0000    1.0000        0
1.0000    1.0000      0.5000
1.0000    1.0000      1.0000
```

上面的数据显示出第一行是1/3红色，而最后一行是白色。

另外函数pcolor可以用来显示一个颜色映象。例如：

```
n=32;
colormap(hsv(n))
pcolor([1:n+1;1:n+1])

n=16;
colormap(jet(n))
pcolor([1:n+1;1:n+1])
```

二、图像的输入输出

- **imread( )**

  支持对**bmp、 jpg、 df、ico、pcx、png、tif和xwd**等格式图像的输入。

- **imwrite( )**

  支持对**bmp、df、ico、jpg、pcx、png、tif和xwd**等格式图像的输出。

1.imread函数

用途：读入图像。

格式：A=imread(‘FILENAME.FMT’)。

该函数把FILENAME 中的图像读到A中， FILENAME指明文件，FMT指明文件格式。

若文件包含一个灰度图,则为二维矩阵；若文件包含一个真彩图(RGB),则A为一三维矩阵。

格式：[X,MAP]=imread (‘FILENAME.FMT’).

把FILENAME中的索引图读入X,其相应的调色板读到MAP中.图像文件中的调色板会被自动在范围[0,1]内重新调节。

FMT的可能取值为jpg或jpeg,tif或tiff, bmp, png, hdf, pcx, xwd。

2.imshow函数

- 格式:imshow (I,N),用N级离散灰度级显示灰度图像I。若省略N,默认用256级灰度显示24位图像,64级灰度显示其他系统。
- 格式:imshow (I,[LOW HIGH])，把I作为灰度图显示。LOW值指定为黑色,HIGH指定为白色，中间为按比例分布的灰色。若[LOW,HIGH]为[],则函数把图像中的最小值显示为黑色,最大值显示为白色。

- **Matlab读取图像**

  示例图像 a.bmp (800X600)

  im=imread('d:\a.bmp');
- **图像读入后保存在三维矩阵变量im中**

  im的三维为 600X800X3
- **显示图像**

  Imshow(im,[0,255])

  Imshow(im,[])

  Imshow(im)

3.imwrite函数

用途：用于把图像写入图形文件中。
格式：imwrite(A,FILENAME,FMT)把图像A写入文件FILENAME中。FILENAME指明文件名, FMT指明文件格式。A既可以是一个灰度图,也可以是一个真彩图像。
格式：imwrite (X,MAP,FILENAME,FMT)把索引图及其调色板写入FILENAME中。MAP必须为合法的MATLAB调色板,大多数图像格式不支持多于256色的调色板。FMT的可能取值为tif或tiff,jpg或jpeg,bmp,png,hdf,pcx,xwd。

- **Matlab将数据写入图像**
  示例数据im为 600X800X3
  将im写入到a.bmp中
  imwrite(im, 'd:\test.bmp');

例：图像文件写入

```
I=imread('cameraman.tif');
imwrite(I,'E:\test1.tif')
```

执行程序后，可以看到在路径**E:**下写入的图像文件“**test1.tif**”

4. rgb2gray 函数。其功能为实现彩色图像灰度化。调用格式为：

I = rgb2gray(RGB)。

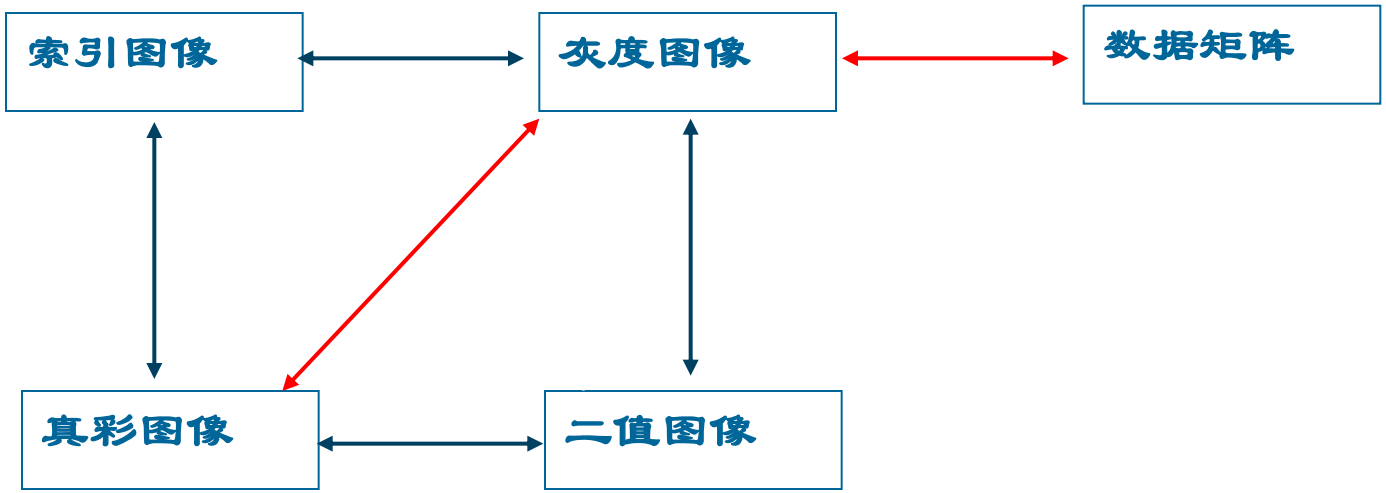
5. rgb2hsv 函数。其功能为实现RGB数据图像向HSV数据图像的转换。调用格式为：

HSV = rgb2hsv(RGB)。 RGB为RGB彩色图像，为三维矩阵；HSV为三维 HSV图像矩阵，三维依次为H、S、V,取值均在[0,1]范围内。

6. rgb2ycbcr 函数。功能为实现RGB数据图像向YCbCr数据图像的转换。调用格式为：

YCbCr = rgb2ycbcr(RGB)。

图像类型的转换

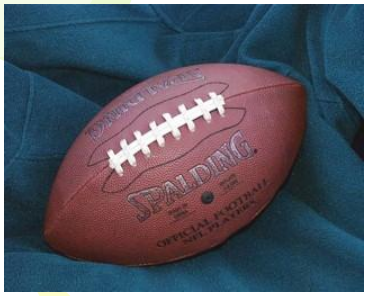


Matlab图像处理工具箱提供了许多图像类型转换的函数，来实现上面各种变换

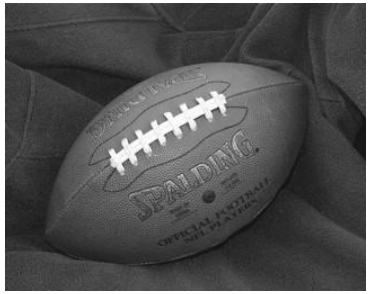
例 编写程序，实现彩色图像到灰度图像和二值图像的转换.

解：程序如下：

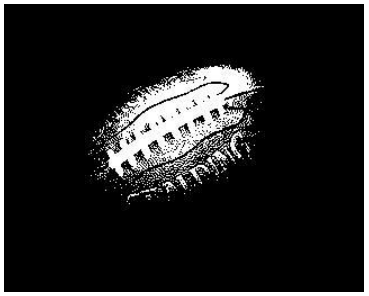
```
clear al;close all;clc;
Image = imread('football.jpg'); %打开图像并将像素值转化到[0,1]
gray=rgb2gray(Image);                    im=im2bw(gray);
subplot(231),imshow(Image),title('彩色图像');
subplot(232),imshow(gray),title('灰度图像');
subplot(233),imshow(im),title('二值图像');
%公式实现转换
r = Image( : , : ,1); %提取红色通道
g = Image( : , : ,2); %提取绿色通道
b = Image( : , : ,3); %提取蓝色通道
Y = 0.299*r + 0.587*g + 0.114*b; %计算亮度值Y实现灰度化
I = (r + g + b)/3; %计算亮度值I实现灰度化
BW = zeros(size(Y));
BW(Y>80) = 1; %阈值为80,实现灰度图二值化
subplot(234),imshow(Y),title('亮度图 Y');
subplot(235),imshow(I),title('灰度图像I ');
subplot(236),imshow(BW),title('二值图像');
imwrite(gray,'gray.jpg');              imwrite(im,'im.jpg');
imwrite(Y,'亮度图 Y.jpg');             imwrite(I,'灰度图I.jpg');
imwrite(BW,'自定阈值二值图.jpg');
```



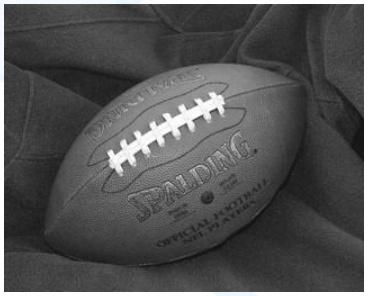
（a）彩色图像



（b）灰度图像

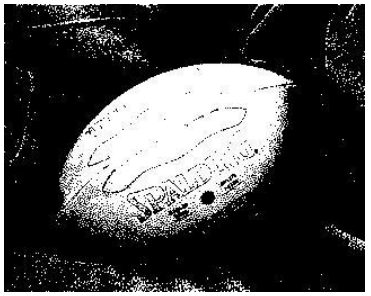


（c）二值图像



（c）亮度图Y

（d）灰度图I



（e）自定阈值二值图像

不同图像类型间的转换

三、 图像运算与图像几何变换

1. 图像的代数和逻辑运算

$$g(x,y)=\alpha f_1(x,y)+\beta f_2(x,y)$$

$$g(x,y)=\alpha f_1(x,y)-\beta f_2(x,y)$$

$$g(x,y)=\alpha f_1(x,y)\times f_2(x,y)$$

$$g(x,y)=\alpha f_1(x,y)\div f_2(x,y)$$

Matlab图像处理工具箱中的代数运算函数

函数名	功能描述
imabsdiff	两幅图像的绝对差值
imadd	两幅图像的加法
imcomplement	补足一幅图像
imdivide	两幅图像的除法
imlincomb	计算两幅图像的线性组合
immultiply	两幅图像的乘法
imsubstract	两幅图像的减法

（1）加法运算

- 和值处理 $g(x,y)=f_1(x,y)+f_2(x,y)$
- 截断处理

$$\begin{cases} g(x,y)=255 & g(x,y)>255 \\ g(x,y)=g(x,y) & \text{其他} \end{cases}$$

- 加权求和

$$g(x,y)=\alpha f_1(x,y)+(1-\alpha)f_2(x,y) \quad \alpha\in[0,1]$$

- 主要应用
  - 多幅图像相加求平均去除叠加性噪声。
  - 将一幅图像的内容经配准后叠加到另一幅图像上去，以改善图像的视觉效果。
  - 用于图像合成和图像拼接。

- 例
  - 函数 Z = imadd(X,Y)
  - 程序

```
Before=imread('clock2.gif');
Back=imread('clock1.gif');
ZJ1=imadd(Before,-80);
ZJ2=imadd(Back,-80);
result1=imadd(Back,Before);
result2=imadd(Back,ZJ1);
result3=imadd(ZJ2,Before);
subplot(231),imshow(Before),title('前聚焦图');
subplot(232),imshow(Back),title('后聚焦图');
subplot(233),imshow(ZJ2),title('暗前景图');
subplot(234),imshow(result1),title('前聚焦叠加后聚焦');
subplot(235),imshow(result3),title('暗背景叠加前聚焦');
subplot(236),imshow(result2),title('暗前景叠加后聚焦');
```

效果

（a）前聚焦图

（b）后聚焦图

（c）暗前聚焦图

（d）前聚焦叠加后聚焦

（e）暗背景叠加前聚焦

（f）暗前景叠加后聚焦

图像加法运算示例图

（2）减法运算

- 差值处理 $g(x,y)=f_1(x,y)-f_2(x,y)$
  - 截断处理
$$\begin{cases} g(x,y)=0 & g(x,y)<0 \\ g(x,y)=g(x,y) & \text{其他} \end{cases}$$
  - 取绝对值 $g(x,y)=\left|f_1(x,y)-f_2(x,y)\right|$
- 主要应用
  - 显示两幅图像的差异，检测同一场景两幅图像之间的变化。
  - 去除不需要的叠加性图案，或去除图像上每一个像素处均已知的附加污染等。
  - 图像分割：如分割运动的车辆，减法去掉静止部分，剩余的是运动元素和噪声。
  - 生成合成图像。

- 例
  - 函数

    Z = imsubtract(X,Y)

    Z = imabsdiff(X,Y)
  - 程序

```
Before=imread('clock2.gif');
Back=imread('clock1.gif');
result=imabsdiff(Before,Back);
result2=imsubtract(Before,Back);
subplot(221),imshow(Back),title('后聚焦图');
subplot(222),imshow(Before),title('前聚焦图');
subplot(223),imshow(result*5),title('相减取绝对值结果');
subplot(224),imshow(result2*5),title('直接相减结果');
```

效果

（a）前聚焦图

（b）后聚焦图

（c）相减取绝对值结果

（d）直接相减结果

图像减法运算示例图

（3）乘法运算

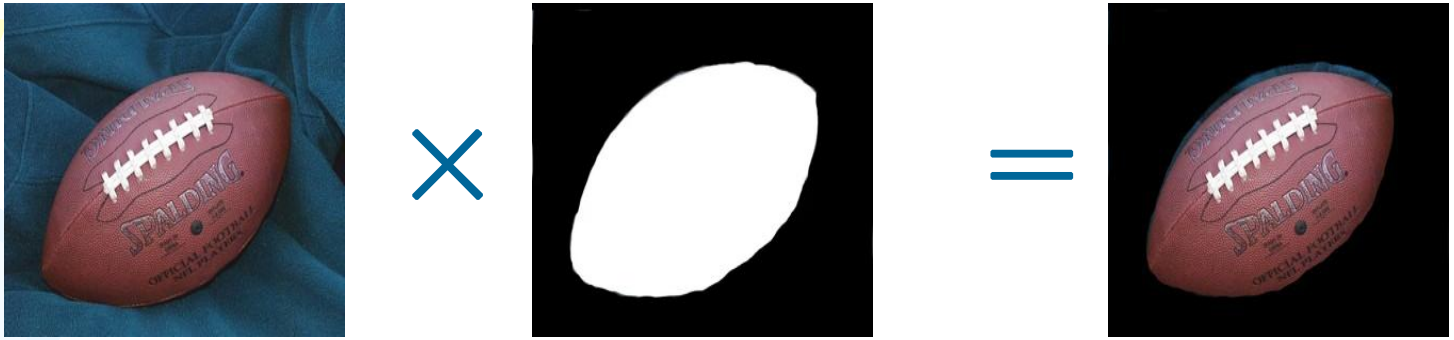
$$g(x,y)=f_1(x,y)\times f_2(x,y)$$

- 主要应用
  - 图像的局部显示和提取：用二值模板图像与原图像做乘法来实现。
  - 生成合成图像

- 函数　Z = immultiply(X,Y)
- 程序

```
Image=im2double(imread('football.jpg'));
Templet=im2double(imread('footballtemplet.jpg'));
result=immultiply(Templet,Image);
subplot(131),imshow(Image),title('背景');
subplot(132),imshow(Templet),title('模板');
subplot(133),imshow(result),title('相乘结果');
```

效果



（a）背景图　　（b）模板　　（c）相乘结果

图像乘法运算示例图

（4）除法运算

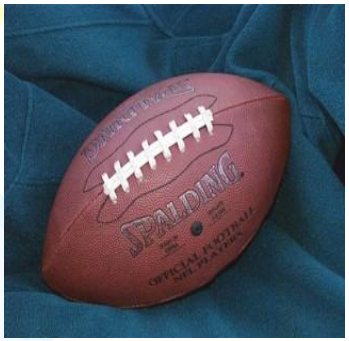
$$g(x,y)=f_1(x,y)\div f_2(x,y)$$

- 主要应用
  - 用于消除空间可变的量化敏感函数、归一化、产生比率图像等

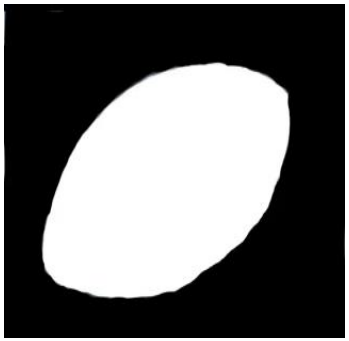
- 例
- 函数　Z = imdivide(X,Y)
  - 程序

```
clear all;close all;clc;
I=double(imread('football.jpg'));
J=double(imread('footballtemplet.jpg'));
Ip = imdivide(I, J);
subplot(131),imshow(uint8(I)),title('背景图
');
subplot(132),imshow(uint8(J)),title('模板');
subplot(133),imshow(Ip),title('相除结果');
imwrite(Ip,'相除结果.jpg');
```

效果



/



=

（a）背景图　　（b）模板　　（c）相除结果

图像除法运算示例图

图像逻辑运算

- 原理

非：$g(x,y)=255-f(x,y)$

与：$g(x,y)=f_1(x,y)\&f_2(x,y)$

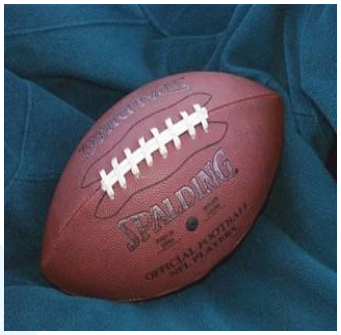
或：$g(x,y)=f_1(x,y)|f_2(x,y)$

- 函数
  - C = bitcmp(A)：按位求补
  - C = bitand(A,B)：按位求与
  - C = bitor(A,B)：按位求或
  - C = bitxor(A,B)：按位求异或
  - &、|、~：按运算数据的值，不按位

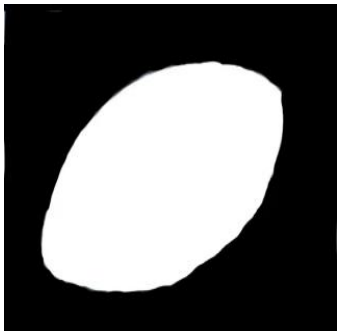
- 程序

```
Back=imread('bird.jpg');
Templet=imread('birdtemplet.bmp');
result1=bitcmp(Back);
result2=bitand(Templet,Back);
result3=bitor(Templet,Back);
result4=bitxor(Templet,Back);
subplot(221),imshow(result1),title('求反');
subplot(222),imshow(result2),title('相与');
subplot(223),imshow(result3),title('相或');
subplot(224),imshow(result4),title('异或');
```

效果

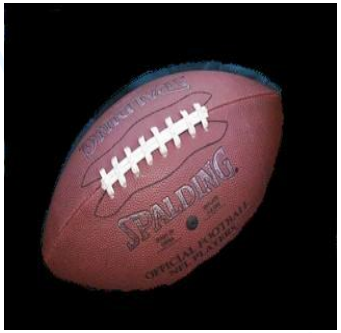


（a）原图



（b）模板

（c）求反



（d）相与

（e）相或

（f）异或

逻辑运算结果图

2.基本的图像几何变换

- 图像的几何变换主要包含位置变换和形状变换。
- 图像的位置变换是指图像的大小和形状不发生变化，只是图像像素点的位置发生变化，含平移、镜像、旋转。
- 图像的位置变换主要是用于目标识别中的目标配准。
- 图像的形状变换主要包括图像的放大、缩小等。

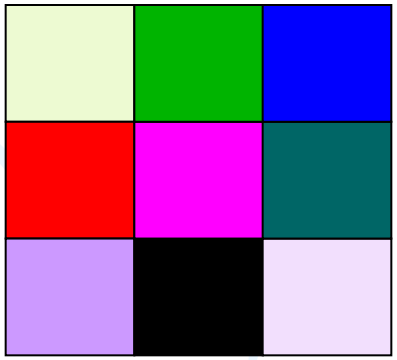
（1）平移变换

- 原理

$$\begin{cases} x' = x + \Delta x \\ y' = y + \Delta y \end{cases}$$

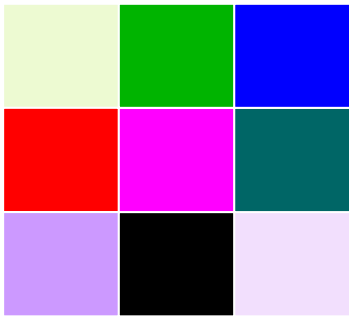
矩阵表示

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$



$\Delta x = 1$

$\Delta y = 1$



若平移后不丢失信息，需扩大“画布”

- 函数

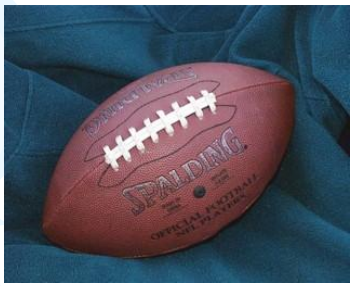
T = maketform(TRANSFORMTYPE,...)

B = imtransform(A,TFORM,INTERP,
PARAM1,VAL1,PARAM2,VAL2,...)

- 程序

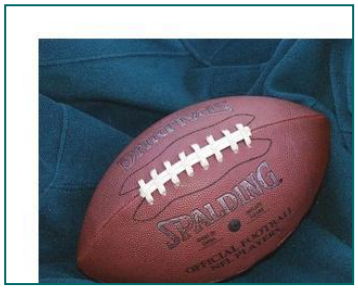
```
Image=imread('football.jpg'); deltax=20; deltay=20;
T=maketform('affine',[1 0 0;0 1 0;deltax deltay 1]);
NewImage1=imtransform(Image,T,
                      'XData',[1 size(Image,2)],
                      'YData',[1,size(Image,1)],
                      'FillValue',255);
NewImage2=imtransform(Image,T,
                      'XData',[1 size(Image,2)+deltax],
                      'YData',[1,size(Image,1)+deltay],
                      'FillValue',255);
subplot(131),imshow(Image),title('原图');
subplot(132),imshow(NewImage1),title('画布尺寸不变平移');
subplot(133),imshow(NewImage2),title('画布尺寸扩大平移');
```

- 效果

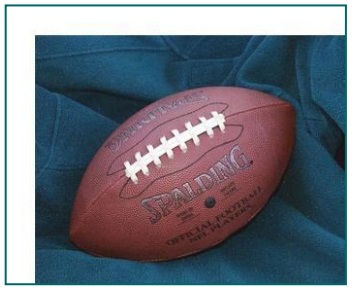


画布不变

$\triangle x=20$



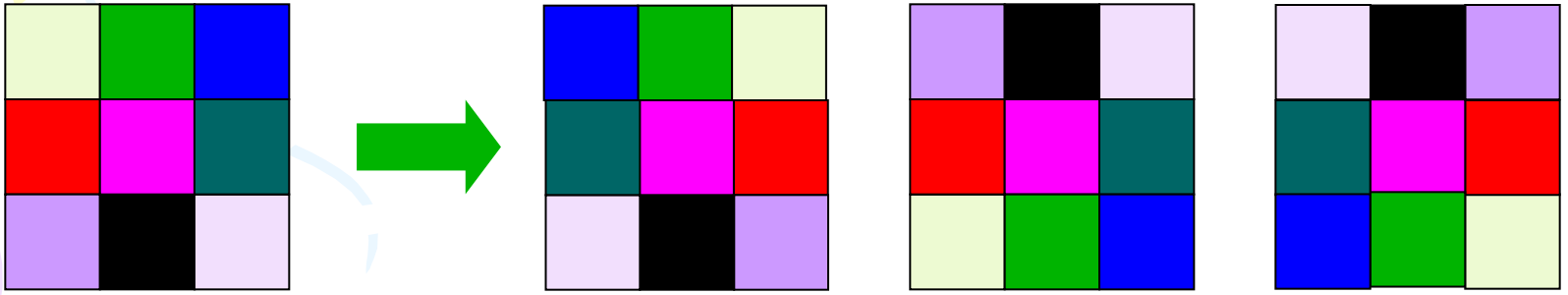
画布增大



（2）镜像

- 原理
  - 水平镜像
  - 垂直镜像
  - 对角镜像

$$\begin{cases} x' = M - 1 - x \\ y' = y \end{cases} \qquad \begin{cases} x' = x \\ y' = N - 1 - y \end{cases} \qquad \begin{cases} x' = M - 1 - x \\ y' = N - 1 - y \end{cases}$$



水平镜像　垂直镜像　对角镜像

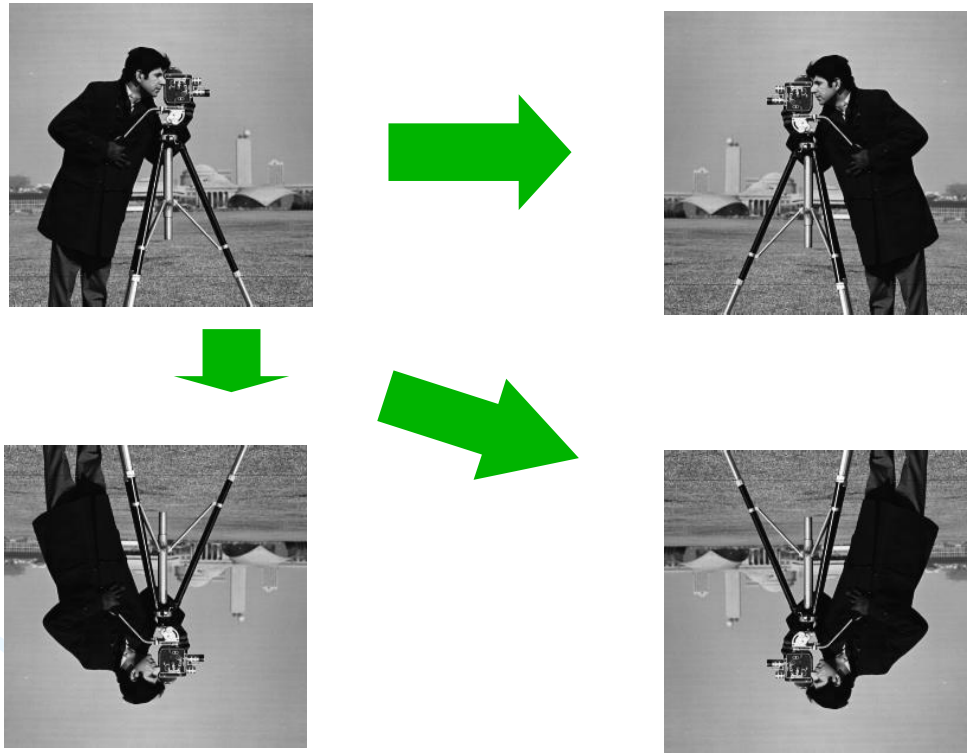
- 函数

  fliplr(X) flipud(X) flipdim(X,DIM)

- 程序

```
Im=imread('cameraman.tif');
HImage=flipdim(Im,2);
VImage=flipdim(Im,1);
CImage=flipdim(HImage,1);
subplot(221),imshow(Im),title('原图');
subplot(222),imshow(HImage),title('水平镜像');
subplot(223),imshow(VImage),title('垂直镜像');
subplot(224),imshow(CImage),title('对角镜像');
```

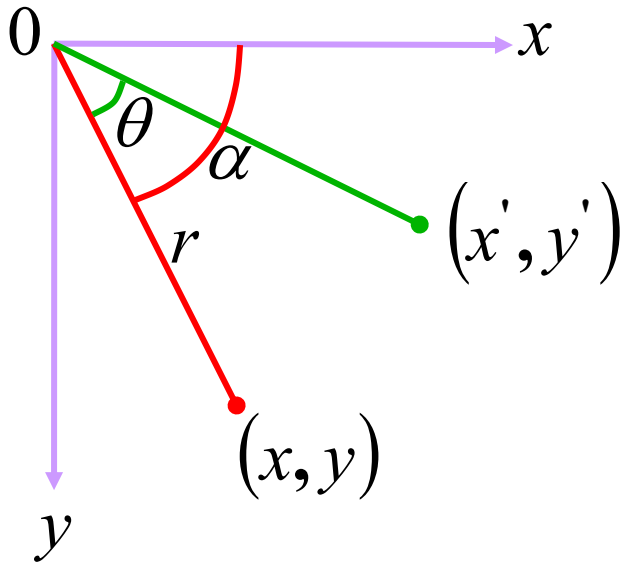
效果



（3）旋转

- 原理

绕图像原点逆时针旋转



$$\begin{cases} x = r\cos\alpha \\ y = r\sin\alpha \end{cases}$$

$$\begin{cases} x' = r\cos(\alpha-\theta) \\ y' = r\sin(\alpha-\theta) \end{cases} \Rightarrow$$

$$\begin{cases} x' = r\cos\alpha\cdot\cos\theta + r\sin\alpha\cdot\sin\theta = x\cos\theta + y\sin\theta \\ y' = r\sin\alpha\cdot\cos\theta - r\cos\alpha\cdot\sin\theta = -x\sin\theta + y\cos\theta \end{cases}$$

绕原点旋转计算公式

$$\begin{cases} x' = x\cos\theta + y\sin\theta \\ y' = -x\sin\theta + y\cos\theta \end{cases}$$

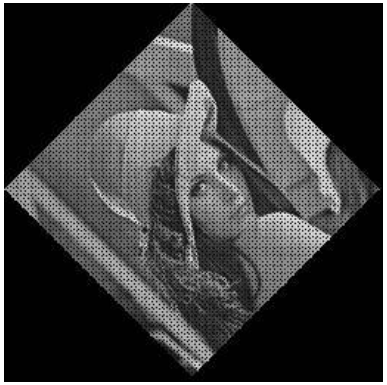
旋转逆变换公式

$$\begin{cases} x = x'\cos\theta - y'\sin\theta \\ y = x'\sin\theta + y'\cos\theta \end{cases}$$

（3）旋转

图像旋转变换的矩阵表达为

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$



（a）原图　（b）逆时针旋转45，未插值　（c）逆时针旋转45，双线性插值

图像的旋转变换

（3）旋转

B=imrotate(A，ANGLE，METHOD，BBOX)， A为要进行旋转的图像；ANGLE为要旋转的角度(°)，逆时针为正，顺时针为负； METHOD为图像旋转插值方法，可取“'nearest', 'bilinear’,’bicubic’”，默认为nearest；BBOX指定返回图像大小：可取"crop"，输出图像B与输入图像A具有相同的大小，对旋转图像进行剪切以满足要求；可取“loose”，默认时，B包含整个旋转后的图像。

```
Image=im2double(imread('cameraman.tif '));
NewImage1=imrotate(Image,20);
NewImage2=imrotate(Image,20,'bilinear');
imwrite(NewImage1,'rotate11.jpg');
imwrite(NewImage2,'rotate12.jpg');
subplot(1,3,1),imshow(Image),title('原图');
subplot(1,3,2),imshow(NewImage1),title('最近邻插值旋转结果');
subplot(1,3,3),imshow(NewImage2),title('双线性插值旋转结果');
```

（a）原图

（b）最近邻插值旋转结果

（c）双线性插值旋转结果

4. 缩放变换

图像缩放变换的矩阵表达为

$$\begin{pmatrix} x' \\ y' \\ 1 \end{pmatrix} = \begin{pmatrix} k_x & 0 & 0 \\ 0 & k_y & 0 \\ 0 & 0 & 1 \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix}$$

MATLAB中实现图像缩放功能的函数是imresize（），具体调用方式是：
B=imresize(A，SCALE，METHOD)：返回原图A的SCALE大小图像B。

B=imresize（A，[NUMROWS NUMCOLS]，METHOD）：对原图A进行比例缩放，返回图像B的行数和列数由NUMROWS、NUMCOLS指定，如果二者为NaN，则表明MATLAB自动调整了图像的缩放比例，保留图像原有的宽高比。

[Y, NEWMAP]=imresize（X，MAP，SCALE，METHOD）：对索引图像进行成比例缩放。

4. 缩放变换

```
Image = im2double(imread('lena256.bmp'));
NewImage1 = imresize(Image,1.2,'nearest');
NewImage2 = imresize(Image,1.2,'bilinear');
imwrite(NewImage1,'最近邻1.2倍插值.jpg');
imwrite(NewImage2,'双线性1.2倍插值.jpg');
subplot(1,3,1),imshow(Image),title('原图');
subplot(1,3,2),imshow(NewImage1),title('最近邻1.2倍插值');
subplot(1,3,3),imshow(NewImage2),title('双线性1.2倍插值');
```

（a）原图

（b）最近邻插值1.2倍放大结果

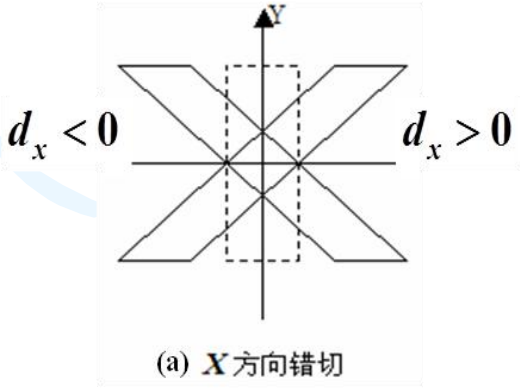
（c）双线性插值1.2倍放大结果

5. 错切变换

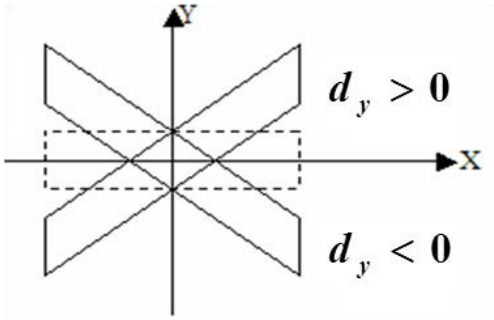
水平和垂直方向同时错切的数学表达式为

$$\begin{cases} x' = x + d_x y \\ y' = y + d_y x \end{cases}$$

图像的错切变换是平面景物在投影平面上的非垂直投影。错切使图像产生扭曲变形。这种扭变只在水平或垂直方向上产生时，分别称为水平方向上的错切和垂直方向上的错切



(a) X方向错切



(b) Y方向错切

图像的错切变换示意图

```
Image=im2double(imread('lena256.bmp'));
tform1=maketform('affine',[1 0 0;0.5 1 0; 0 0 1]);
tform2=maketform('affine',[1 0.5 0;0 1 0; 0 0 1]);
NewImage1=imtransform(Image,tform1);
NewImage2=imtransform(Image,tform2);
subplot(1,2,1),imshow(NewImage1),title('水平错切');
subplot(1,2,2),imshow(NewImage2),title('垂直错切');
```

（a）原图

（b）水平方向错切

（c）垂直方向错切

图像错切效果图

四、图像线性变换变换

就是借助于变换函数将输入图像像素灰度值映射成一个新的输出值，通过改变像素的亮度值来增强图像。

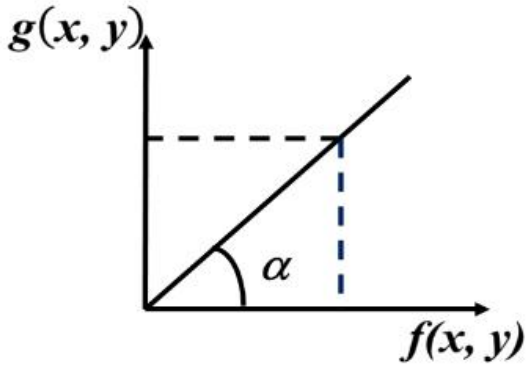
☆灰度变换是一种点处理方法，它将输入图像中每个像素(x,y)的灰度值f(x,y)，通过映射函数T(·)，变换成输出图像中的灰度g(x, y)，即：

$$g(x,y)=T[f(x,y)]$$

根据变换函数的不同，灰度级变换可以分为线性灰度级变换和非线性灰度级变换。

正比例线性变换

$$g(x,y)=f(x,y)\cdot\tan\alpha$$



当k=1，灰度值范围没有发生变化，图像无变化；
当 k<1，变换后灰度值范围压缩，图像均匀变暗；
当 K>1，变换后灰度值范围拉伸，图像均匀变亮。

（a）原图

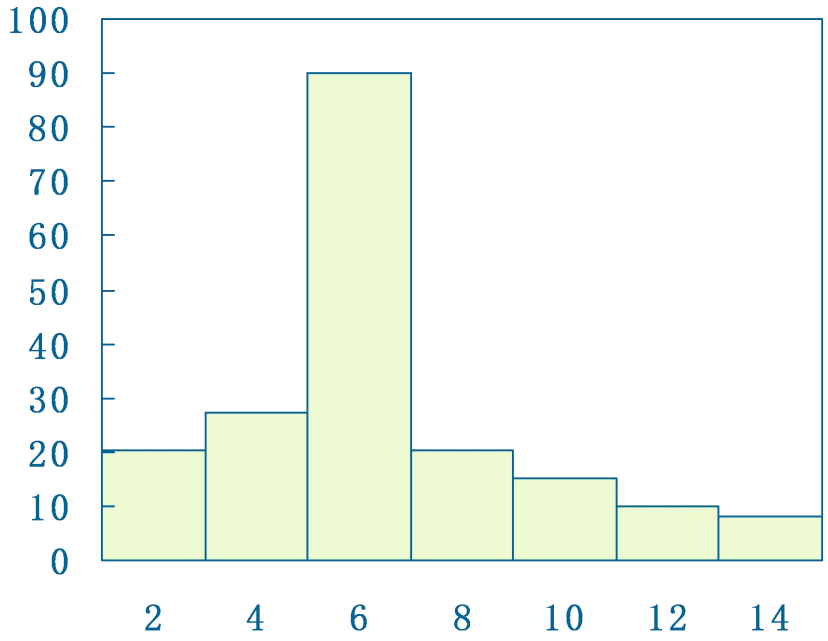
（b）$k=0.5$ 变暗

（c）$k=2$ 变亮

正比例线性变换结果

五、图像数据的直方图

- 直方图又称质量分布图，柱状图，是一种几何形图表，它是根据数据分布情况，画成以组距为底边、以频数为高度的一系列连接起来的直方型矩形图。



灰度直方图反映了数字图像中每一灰度级与其出现频率间的关系,它能描述该图像的概貌。

直方图（Histogram）定义：指*灰度统计*直方图，即数字图像中像素灰度值的分布情况；

具体地说，就是数字图像中的*每一灰度级*与其出现的*频数*间的统计关系；

设一幅数字图像有L个灰度级，灰度级范围为[0, L-1]，则其直方图定义为离散函数：

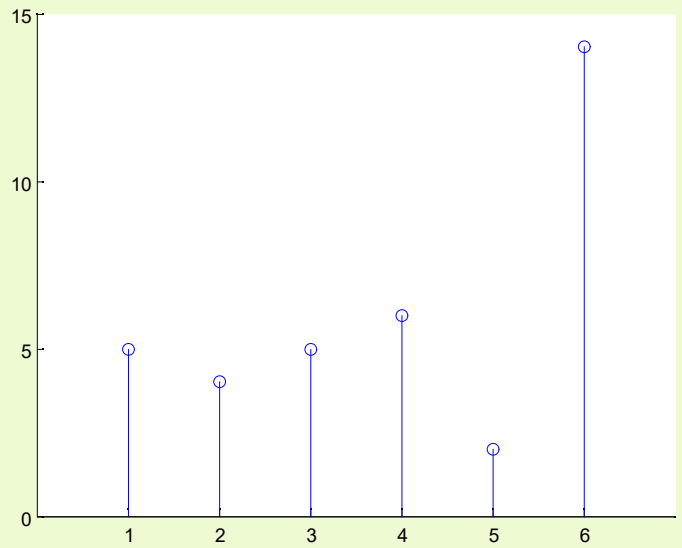
$$h(r_k)=n_k$$

其中，r_k是[0, L-1]内的第k级灰度，n_k是图像中灰度级为r_k的像素个数(即出现的频率)，其中k=0,1,2,…,L-1。

- 把离散函数$h(r_k)$用图形表示出来即得到直方图。横坐标为图像的灰度级，纵坐标为灰度级出现的像素个数。

1	2	3	4	5	6
6	4	3	2	2	1
1	6	6	4	6	6
3	4	5	6	6	6
1	4	6	6	2	3
1	3	6	4	6	6

r_k	1	2	3	4	5	6
n_k	5	4	5	6	2	14

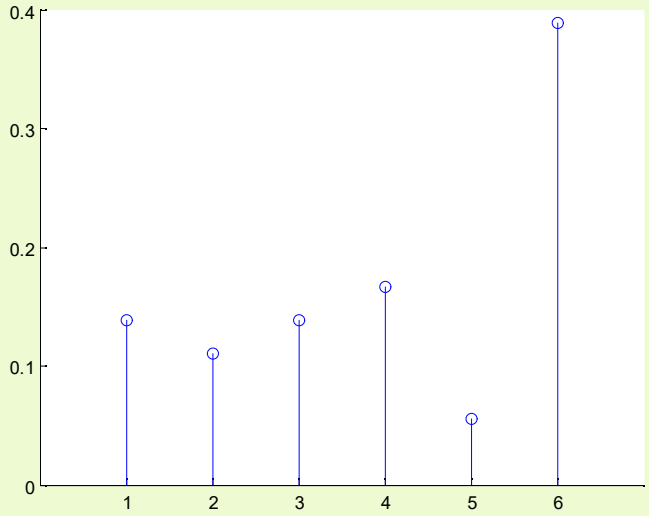


- 定义：用$h(r_k)$除以图像中的像素总数MN所得到的图形，记为：

$$p(r_k)=h(r_k)/MN=n_k/MN$$

1	2	3	4	5	6
6	4	3	2	2	1
1	6	6	4	6	6
3	4	5	6	6	6
1	4	6	6	2	3
1	3	6	4	6	6

r_k	1	2	3	4	5	6
n_k	5	4	5	6	2	14
$p(r_k)$	0.14	0.11	0.14	0.17	0.05	0.39



可知：

$$p(r_k)=\frac{\text{灰度值为}r_k\text{的像素个数}}{\text{图像上总的像素个数}}$$

且

$$\sum_{k=0}^{L-1} p(r_k) = 1$$

故可认为$p(r_k)$是灰度级取值为r_k的概率分布。

- 直方图的绘制思路：

  图像的取值为[0，255]

- 针对每一个值作为一组进行统计，共有256组
- 可设置数组h(256)用以保存对应每一个值的频数。
- 最后用bar把h画出来。

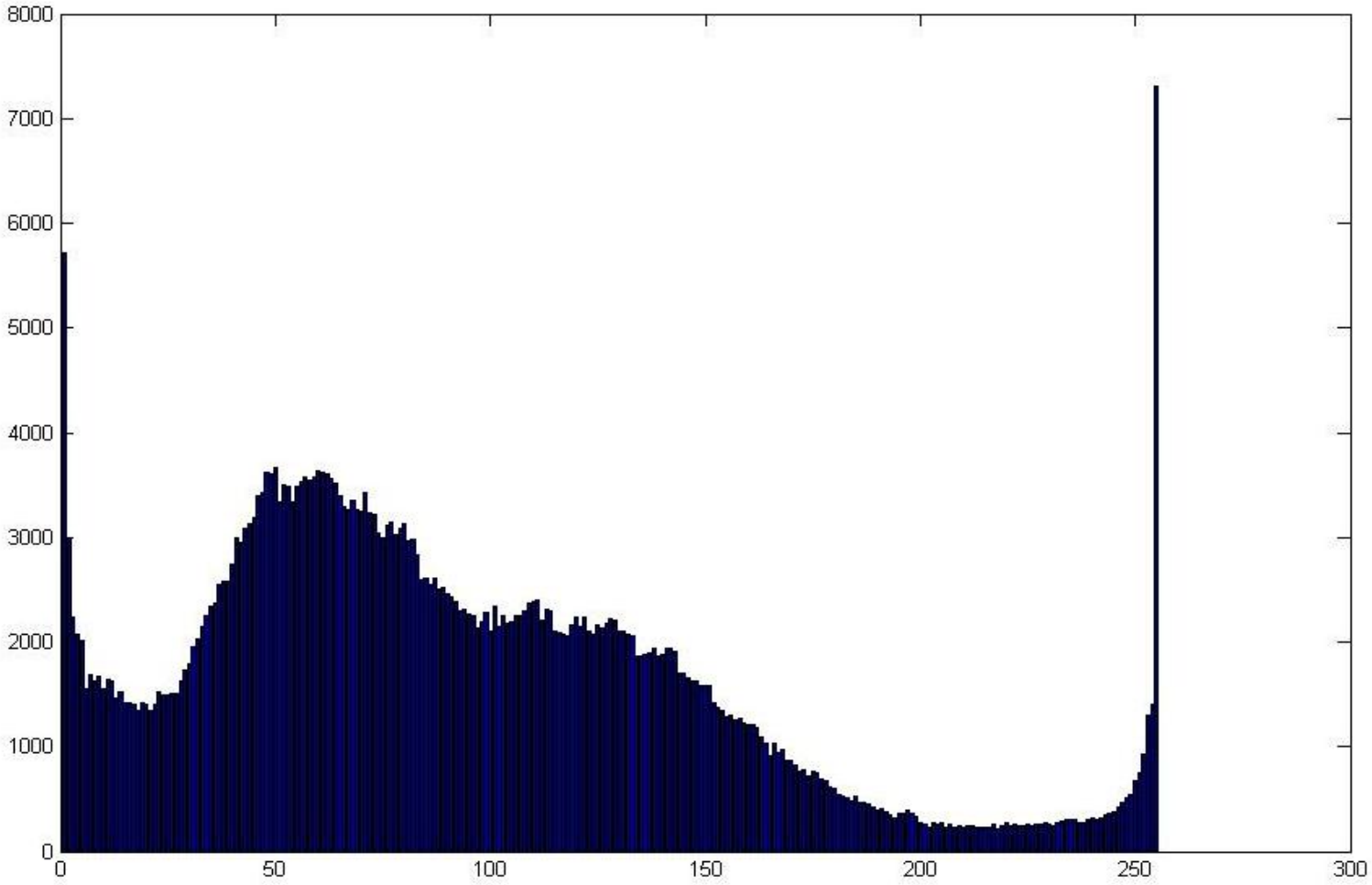
图像数据的直方图

- 直方图的绘制代码：

```
im=imread('d:\b.jpg');
[m,n,l]=size(im);
h=zeros(1,256);
for i=1 : m
    for j=1 : n
        h(im(i,j,1)+1)=h(im(i,j,1)+1)+1;
    end
end
bar(h);
```

图像数据的直方图

- 效果显示



直方图函数imhist()：

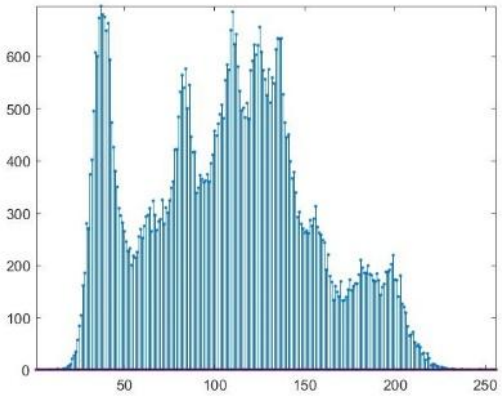
语法：h=imhist(f,b)

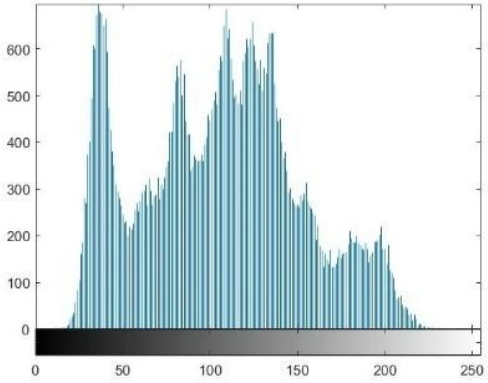
说明：f为输入图像，h为其直方图（1维向量），b为划分的灰度级个数，其默认值为256。

举例：p=imhist(f)/numel(f) %获得归一化直方图

例 基于Matlab编程统计图像的灰度直方图

```
Image=imread('lena256.bmp');
histgram=zeros(256);
[h w]=size(Image);
for x=1:w
   for y=1:h                                    %循环扫描
      histgram(Image(y,x)+1)=histgram(Image(y,x)+1)+1;   %统计并累加
   end
end
figure,imshow(Image);title('灰度图像');
figure,stem(histgram(),'.'); %title('定义统计直方图');
axis tight;
colormap(gray)
figure,imhist(Image); %title('系统函数统计直方图');
axis tight;
colormap(gray);
```





（a）原图　　b）定义统计灰度直方图　　（c）imhist函数统计灰度直方图

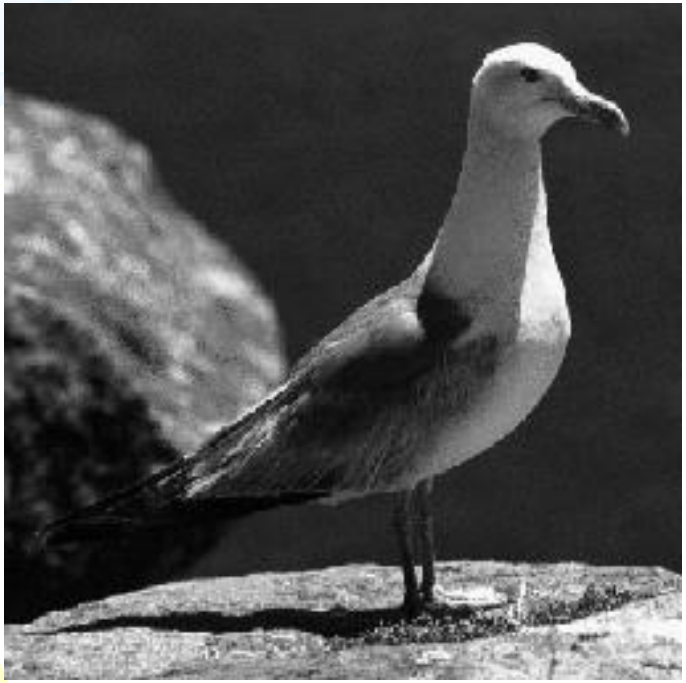
图像及其灰度直方图

六、图像锐化(Sharpening)

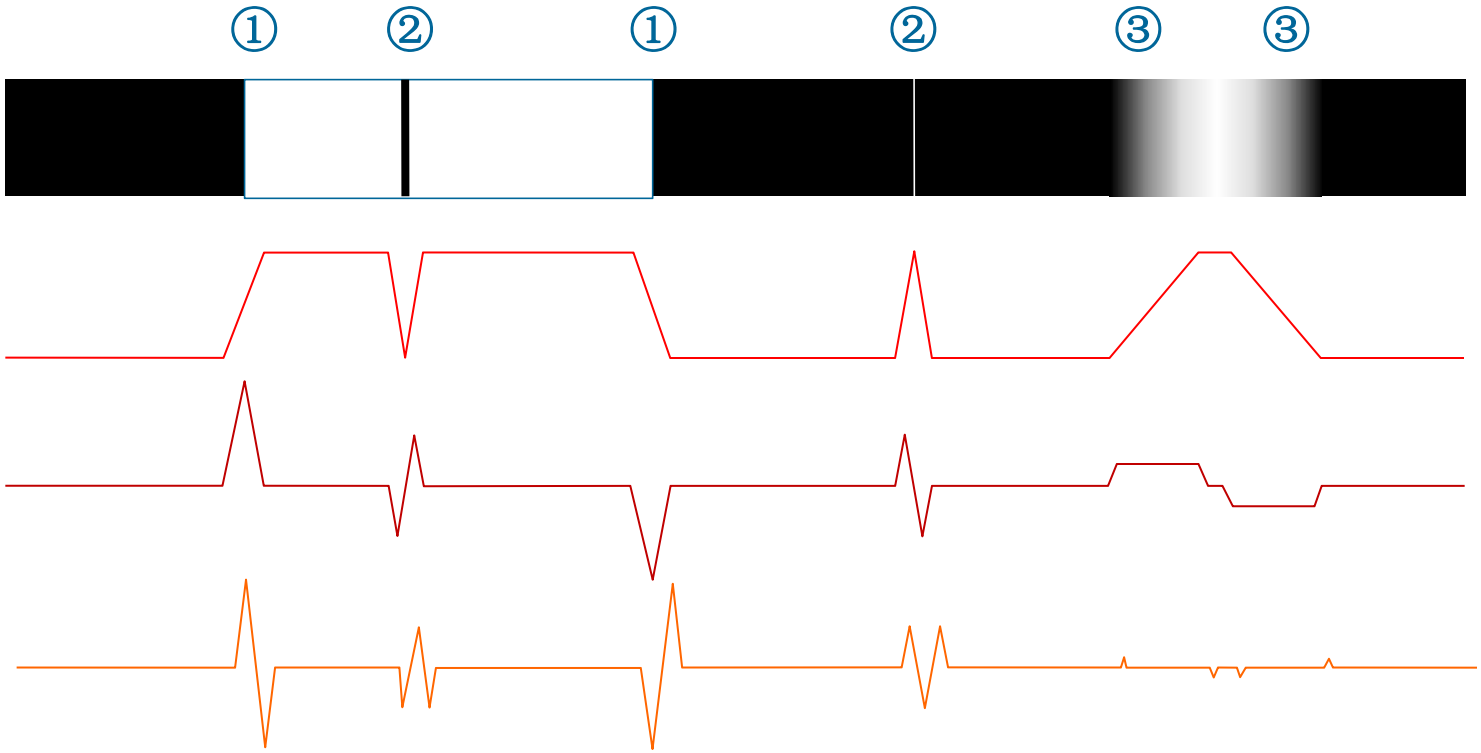
作用:增强图像边缘或轮廓，增强灰度反差，从而加强图像中的轮廓边缘和细 节，因为边缘和轮廓都位于灰度突变的地方。

实现：基于微分实现

（图像平滑是通过积分使图像边缘模糊）



图像的边缘



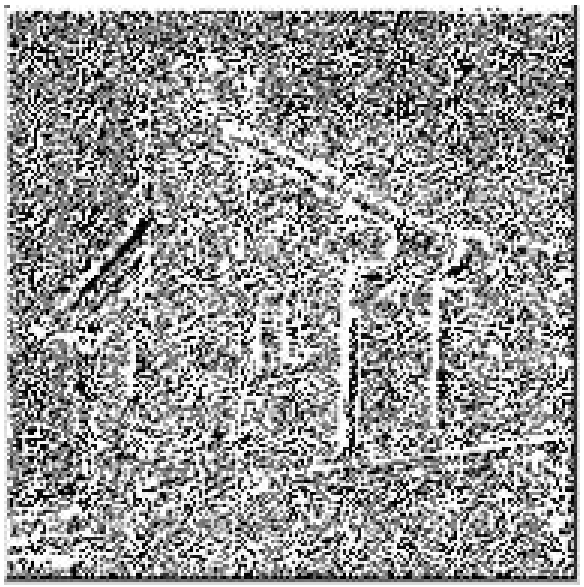
- 突变型细节：检测一阶微分极值点，二阶微分过0点
- 细线型细节：检测一阶微分过0点，二阶微分极值点
- 渐变型细节：难检测，二阶微分信息略多于一阶微分

例　编程实现梯度锐化图像。

解：程序如下：

```
Image=imread('house.png');
Image=rgb2gray(Image);
subplot(131),imshow(Image),title('原图像');
[h,w]=size(Image);
edgeImage=zeros(h,w);
for x=1:w-1
   for y=1:h-1
      edgeImage(y,x)=abs(Image(y,x+1)-
Image(y,x))+abs(Image(y+1,x)-Image(y,x));
   end
end
sharpImage=Image+uint8(edgeImage);
subplot(132),imshow(edgeImage),title('梯度图像');
subplot(133),imshow(sharpImage),title('锐化图像');
```

(a)原始图像



(b)梯度图像
梯度锐化图像

(c)锐化增强图像

例 Matlab编程实现Roberts算子的边缘检测和图像锐化。

解：程序如下：

```
Image=imread('house.png');
Image=rgb2gray(Image);
BW= edge(Image,'roberts');
H1=[1 0; 0 -1];
H2=[0 1;-1 0];
R1=imfilter(Image,H1);
R2=imfilter(Image,H2);
edgeImage=abs(R1)+abs(R2);
sharpImage=Image+edgeImage;
subplot(221),imshow(Image),title('原始图像');
subplot(222),imshow(BW),title('边缘检测');
subplot(223),imshow(edgeImage),title('Robert梯度图像');
subplot(224),imshow(sharpImage),title('Robert锐化图像');
```

(a)原始图像

(b)边缘检测

(c)Robert梯度图像

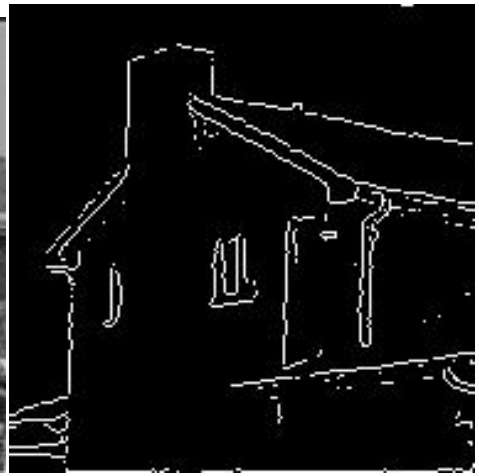
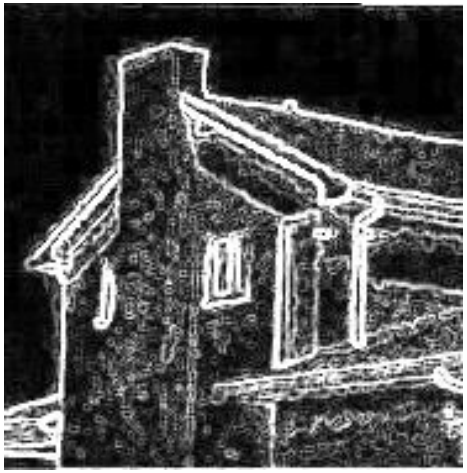
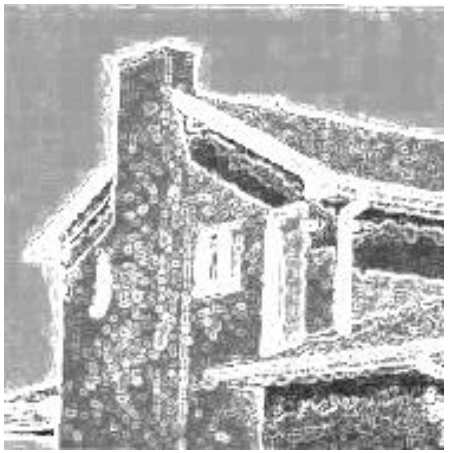
(d)Robert锐化图像

Roberts锐化图像

例 Matlab编程Sobel实现模板边缘锐化

解：程序如下：

```
clear all;close all;clc;
Image=im2double(imread('house.png'));
Image=rgb2gray(Image);
figure,imshow(Image),title('原图像');
BW= edge(Image,'sobel');
figure,imshow(BW),title('边缘检测');
H1=[-1 -2 -1;0 0 0;1 2 1];
H2=[-1 0 1;-2 0 2;-1 0 1];
R1=imfilter(Image,H1);
R2=imfilter(Image,H2);
edgeImage=abs(R1)+abs(R2);
figure,imshow(edgeImage),title('Sobel梯度图像');
sharpImage=Image+edgeImage;
figure,imshow(sharpImage),title('Sobel锐化图像');
```

(a)原图　　(b)边缘检测　　(c)Sobel梯度图像　　(d)Sobel锐化图像

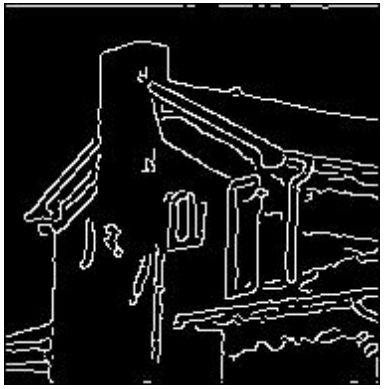
Sobel算子锐化增强

例 Matlab编程实现Canny边缘检测

解：程序如下：

```
clear all;close all;clc;
Image=im2double(imread('house.png'));
Image=rgb2gray(Image);
figure,imshow(Image),title('原图像');
BW= edge(Image,'canny');
figure,imshow(BW),title('Canny边缘检测');
```

原图



Canny边缘检测

练习

1.绘制下面图像的直方图。

练习

2.实现如下效果

原图

反色(y=-x+255)

亮度增加(y=x+64)