第3章

数字图像的基本运算

本章学习目标

(1) 掌握图像点运算的基本定义和常见方法,掌握在 MATLAB 中图像点运算的方法。

(2) 理解图像代数/逻辑运算的基本定义和常见方法,了解噪声模型及对图像添加噪声的基本方法,掌握在 MATLAB 中图像代数/逻辑运算的方法。

(3) 理解图像几何运算的基本定义和常见方法,掌握在 MATLAB 中图像插值的方法,实现图像缩放、图像旋转等几何运算。

本章旨在利用 MATLAB 软件,实现图像的点运算、代数运算、几何运算等基本操作及其应用。

3.1 图像点运算

在数字图像处理中,点运算是简单且具有代表性的重要算法之一,也是其他图像处理运算的基础。

设原图像 $f(x,y)$和变换后的图像 $g(x,y)$,则点运算可表示为

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

其中,T 为灰度变换函数。由此可见,点运算可以在不改变图像内的空间坐标关系基础上,利用灰度变换函数改变图像的灰度值。因此,点运算通常也可写成如下的简单形式

$$s = T(r)$$

其中,r 为原图像 $f(x,y)$在点(x,y)处的灰度值,s 为变换后图像 $g(x,y)$在点(x,y)处的灰度值[3]。

3.1.1 线性点运算

若灰度变换函数为线性时,此时的灰度变换称为线性点运算。例如,

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

显然，当 $\alpha=1,\beta=0$ 时，原图像灰度值不发生任何变化；当 $\alpha=1,\beta\neq0$ 时，图像灰度值增加或降低；当 $\alpha>1$ 时输出图像时比度增大；当 $0<\alpha<1$ 时，输出图像对比度减小；当 $\alpha<0$ 时，图像亮区变暗，暗区变亮。当图像的曝光不足或过度时，图像灰度值就会限制在一个较小的范围内，此时利用线性点运算对图像进行处理，就能增强图像的对比度，改善图像的视觉效果[23]。

通过分段线性变换实现图像的对比度拉伸。增强图像中一些感兴趣的细节，而抑制不感兴趣的部分。例如，设分段线性变换函数为

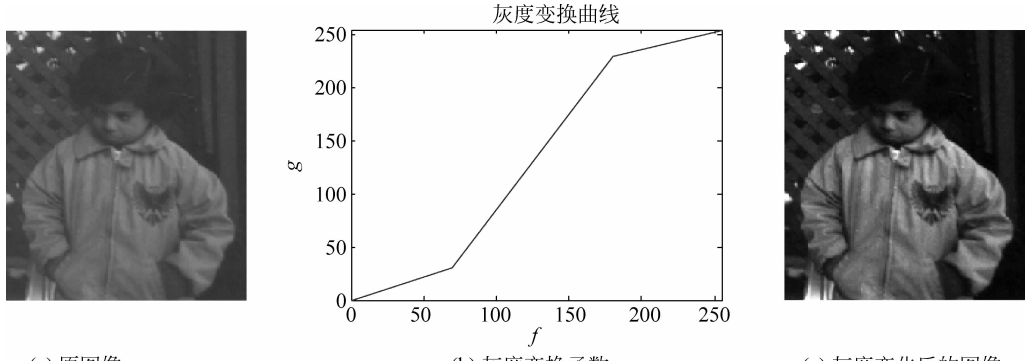
$$g(x,y)=\begin{cases}\dfrac{c'-a'}{c-a}(f(x,y)-a)+a', & a\leqslant f(x,y)<c\\ \dfrac{d'-c'}{d-c}(f(x,y)-c)+c', & c\leqslant f(x,y)<d\\ \dfrac{b'-d'}{b-d}(f(x,y)-d)+d', & d\leqslant f(x,y)<b\end{cases}$$

其中，a、b、c、d 为变换前图像的灰度值，a'、b'、c'、d' 为变换后图像的灰度值。

例 3.1 分段线性变换程序，其结果对比图如图 3.1 所示。

```
>> clf                                          % 清屏
X1 = imread('pout.tif');
figure, imshow(X1)
f0 = 0;g0 = 0;f1 = 70;g1 = 30;f2 = 180;g2 = 230; f3 = 255;g3 = 255;
figure, plot([f0,f1,f2,f3],[g0,g1,g2,g3])    % 绘制变换曲线
axis tight, xlabel('f'), ylabel('g')
title('intensity transformation')
r1 = (g1 - g0)/(f1 - f0);                    % 求 0～70 灰度值范围内的压缩比
b1 = g0 - r1 * f0;
r2 = (g2 - g1)/(f2 - f1);                    % 求 70～180 灰度值范围内的压缩比
b2 = g1 - r2 * f1;
r3 = (g3 - g2)/(f3 - f2);                    % 求 180～255 灰度值范围内的压缩比
b3 = g2 - r3 * f2;
[m,n] = size(X1);                            % 求矩阵的行数 m,列数 n
X2 = double(X1);                             % 将数据类型转换为双精度型
 % 变换矩阵中的每个元素
for i = 1:m
    for j = 1:n
      f = X2(i,j);
      g(i,j) = 0;
```

```
        if (f>=0)&(f<=f1);                    % 找出灰度值范围在 0～70 的元素
        g(i,j)=r1*f+b1;                       % 灰度值在 0～30 的元素进行灰度变换
        elseif (f>=f1)&(f<=f2)                % 找出灰度值范围在 70～180 的元素
        g(i,j)=r2*f+b2;                       % 灰度值在 70～180 元素的进行灰度变换
        elseif (f>=f2)&(f<=f3)                % 找出灰度值范围在 180～255 的元素
        g(i,j)=r3*f+b3;                       % 灰度值在 180～255 元素的进行灰度变换
      end
    end
end
figure,imshow(mat2gray(g))                    % 函数 mat2gray( )将数据矩阵转换成灰度图像
```



(a) 原图像　　(b) 灰度变换函数　　(c) 灰度变化后的图像

图 3.1　图像点运算分段线性变换对比图

由此可见，该例中，灰度变换公式为

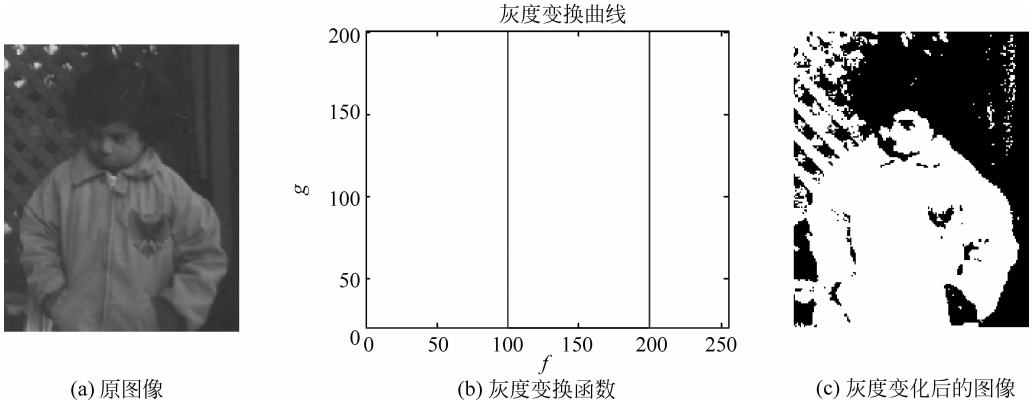
$$g(x,y)=\begin{cases} r_1(f(x,y)+b_1, 0\leqslant f(x,y)<f_1 \\ r_2(f(x,y)-c)+b_2, f_1\leqslant f(x,y)<f_2 \\ r_3(f(x,y)-d)+b_3, f_2\leqslant f(x,y)<f_3 \end{cases}$$

从上例可以看出，线性点运算虽然简单，但选择合适的变换因子可以有效地改变图像的对比度。

例 3.2　灰度切分的程序，其结果对比图如图 3.2 所示。

```
>> X1=imread('pout.tif');
subplot(131),imshow(X1);
s1=100;s2=200;g1=20;g2=200;
subplot(132),plot([0 s1 s1 s2 s2 255],[g1 g1 g2 g2 g1 g1]);
axis tight,xlabel('f'),ylabel('g'),title('灰度变换曲线')
[m n]=size(X1);
```

```
X2 = double(X1);
for I = 1:m
    for J = 1:n
        f = X2(I,J);
        g(I,J) = 0;
        if (f >= s1)&(f <= s2)
            g(I,J) = g2;
        else
            g(I,J) = g1;
        end
    end
end
subplot(133),imshow(mat2gray(g));
```



(a) 原图像　　(b) 灰度变换函数　　(c) 灰度变化后的图像

图 3.2　图像点运算灰度切分变换对比图

若设灰度变换公式为

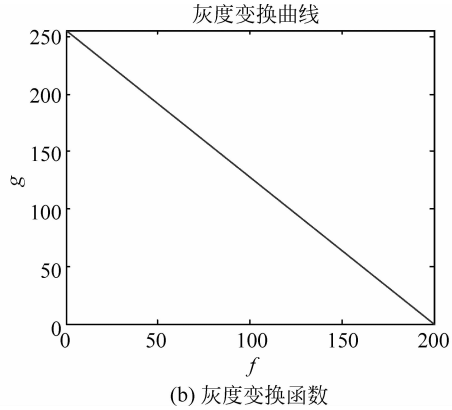
$$g(x,y) = L-1-f(x,y)$$

则可以得到原图像的反色图像。其中,图像灰度值的范围为[0　$L-1$]。

例 3.3　反色图像程序,其结果对比图如图 3.3 所示。

```
>> clf                                          % 清屏
X1 = imread('pout.tif');
subplot(131),imshow(X1);
f1 = 200;g1 = 256;
subplot(132),plot([0 f1],[g1 0]);
axis tight,xlabel('f'),ylabel('g'),title('灰度变换曲线')
k = g1/f1;
```

```
[m n] = size(X1);
X2 = double(X1);
for I = 1:m
    for J = 1:n
        f = X2(I,J);
        g(I,J) = 0;
        if (f >= 0)&(f <= f1)
            g(I,J) = g1 - k * f;
        else
            g(I,J) = 0;
        end
    end
end
subplot(131),imshow(mat2gray(g));
```



(a) 原图像　　(b) 灰度变换函数　　(c) 灰度变化后的图像

图 3.3　图像点运算反转变换对比图

3.1.2　非线性点运算

顾名思义，若灰度变换函数为非线性时，称为非线性点运算。引入非线性点运算主要是考虑在成像时由于成像设备的非线性失衡需要进行校正，或者主观要求需要强化部分灰度区域信息。非线性点运算时要进行扩展的灰度范围是有选择的。

对数变换常用来扩展低值灰度、压缩高值灰度，这样可以使低值灰度的图像细节更容易看清楚。例如，常用的对数变换

$$g(x,y) = \log[f(x,y)+1]$$

例 3.4 对数变换程序，结果对比图如图 3.4 所示。

```
>> I = imread('pout.tif');
subplot(121),imshow(I);
I = double(I);J = log(I + 1);
subplot(122),imshow(J,[]);
```

图 3.4 图像点运算对数变换前后对比图

此外，常用的幂次变换

$$s = c\,r^{\gamma}$$

其中，c 和 γ 为正的常数。r 为原图像的灰度值，s 为变换后图像的灰度值。

3.1.3 灰度直方图与点运算

1. 灰度直方图

灰度直方图是图像预处理中涉及最广泛的基本概念之一。在数字图像处理中，直方图是一种最简单和最有用的工具，它概括了一幅图像的灰度级分布概率。设一幅图像的灰度直方图可以表示为

$$h(r_k) = n_k$$

其中，r_k 表示的图像不同的灰度值；n_k 表示灰度值为 r_k 的像素的个数。可见，灰图像的直方图事实上就是图像的亮度分布的概率密度函数，是一幅图像的所有像素集合的最基本的统计规律。图像的概率密度函数可以表示为

$$p(r_k) = n_k/n$$

其中，n 为图像中的像素个数。

在 MATLAB 中使用 imhist 函数显示指定图像的灰度直方图。其一般的语法格式

```
imhist(I,n)
imhist(X,map)
[counsts,x] = imhist( … )
```

其中，I 为输入图像；n 为指定的灰度级数，默认值为 256。imhist(X,map)计算和显示索引色图像为 X 的直方图，map 为调色板。[counsts,x]=imhist(…)返回直方图数据向量 counts 或相应的色彩值向量 x。

例 3.5　图像的直方图程序，其结果图如图 3.5 所示。

```
>> clf                                          % 清屏
   I = imread('rice.tif');
   subplot(121);imshow(I);
   subplot(122);imhist(I);
```

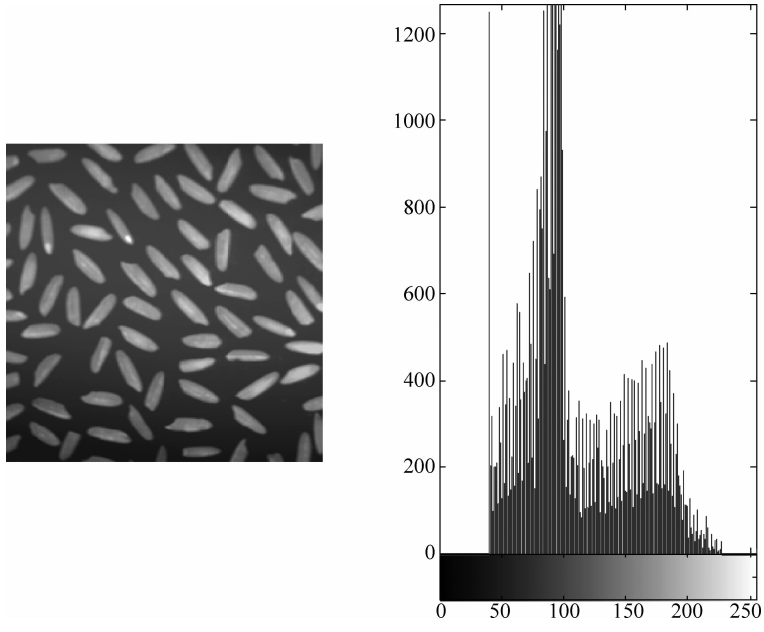


图 3.5　图像及其直方图

2. 点运算

点运算是一种在确定的函数关系下所进行的像素变换运算，因此，点运算之后输出图像和输入图像之间的直方图也具有与变换函数相关联的对应关系，详见第 4 章。

3.2 图像代数/逻辑运算

图像代数运算是指对两幅或多幅输入图像进行点对点的加、减、乘或除运算而得到输出图像的运算。在4种算术操作中,减法和加法在图像增强中最为有用。两幅图像相除,可看作用一幅的取反图像与另一幅相乘。图像平均处理可以减少噪声,除法处理的最主要作用就是增强两幅图像的差异。图像代数运算是点运算的典型应用,也不会改变图像的空间坐标。

使用MATLAB的基本算术符(+、-、*、/等)可以执行图像的算术操作,但是在此之前必须将图像转换为适合进行基本操作的双精度类型。为了更方便地对图像进行操作,MATLAB图像处理工具箱包含了一个能够实现所有非稀疏数值数据的算术操作的函数集合。表3.1列举了所有图像处理工具箱中的图像代数运算函数。

表3.1 图像处理工具箱中的代数运算函数

函 数 名	功 能 描 述
imadd	两幅图像的加法
imabsdiff	两幅图像的绝对差值
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
imsubtract	两幅图像的减法

特别指出,使用图像处理工具箱中的图像代数运算函数无须再进行数据类型间的转换,这些函数能够接受uint8和uint16数据,并返回相同格式的图像结果。虽然在函数执行过程中元素是以双精度进行计算的,但是MATLAB工作平台并不会将图像转换为双精度类型。代数运算的结果很容易超出数据类型允许的范围。例如,uint8数据能够存储的最大数值是255,各种代数运算尤其是乘法运算的结果很容易超过这个数值,有时代数操作(主要是除法运算)也会产生不能用整数描述的分数结果。图像的代数运算函数使用截取规则使运算结果符合数据范围的要求,即超出数据范围的整型数据将被截取为数据范围的极值,分数结果将被四舍五入。例如,如果数据类型是uint8,那么大于255的结果(包括无穷大inf)将被设置为255。注意,无论进行哪一种代数运算都要保证两幅输入图像的大小相等,且类型相同。

3.2.1　图像的加法运算

图像相加一般用于对同一场景的多幅图像求平均效果，以便有效地降低具有叠加性质的随机噪声。直接采集的图像品质一般都较好，不需要进行加法运算处理，但是对于那些经过长距离模拟通讯方式传送的图像（如卫星图像），这种处理是必不可少的。

例 3.6　图像叠加程序，叠加结果如图 3.6 所示。

```
>> clc;clear;                          % 清除 workspace 的变量
I = imread('flower.tif');              % 读入图像
imshow(I);
J = imread('1.tif');                   % 读入另一幅图像
figure, imshow(J);
K = imadd(I,J);                        % 两幅图像叠加
figure, imshow(K);
```

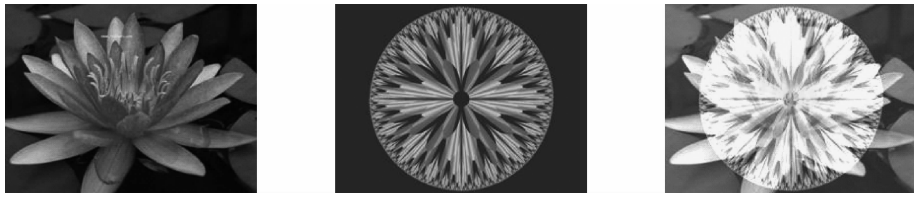


图 3.6　图像相加的效果

如果对图像的像素点进行相加操作，其结果很可能超过图像数据类型所支持的最大值。当数据值发生溢出时，imadd 函数将会截取为数据类型的最大值，这种截取效果称为饱和处理。

MATLAB 工具箱中还提供了一个噪声添加函数 imnoise，以便模拟噪声信息。Imnoise 函数的语法格式为

```
J = imnoise(I,type)
J = imnoise(I,type,parameters)
```

其中，参数 type 指定噪声的种类；parameters 是与噪声种类有关的集体参数；I 是输入图像；J 是对 I 增添噪声后的输出图像；imnoise 函数中噪声及参数的说明如表 3.2 所示。

表 3.2　imnoise 函数中噪声及参数的说明

种类(type)	参数(Parameters)	说　明
gaussian	m，v	均值为 m，方差为 v 的高斯噪声
localvar	v	均值为 0，方差为 v 的高斯白噪声

续表

种类(type)	参数(Parameters)	说　　明
passion	无	泊松噪声
salt pepper	d	噪声密度为 d 的椒盐噪声
speckle	d	噪声密度为 d 的乘性噪声

例 3.7　图像添加噪声程序,原图以及加入不同噪声后的图像如图 3.7 所示。

```
>> i = imread('cj.png');
j1 = imnoise(i,'salt & pepper',0.02);          % 加入椒盐噪声
j2 = imnoise(i,'gaussian',0,0.01);             % 加入高斯白噪声
j3 = imnoise(i,'poisson');                     % 加入泊松噪声
j4 = imnoise(i,'speckle',0.04);                % 加入乘法噪声
imshow(i);
figure;
subplot(221),imshow(j1);
subplot(222),imshow(j2);
subplot(223),imshow(j3);
subplot(224),imshow(j4);
```

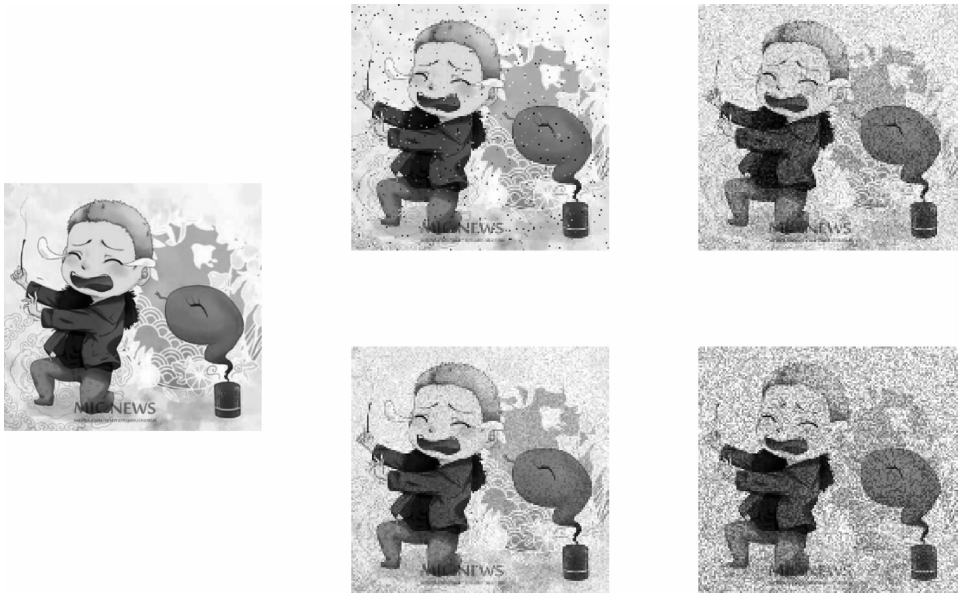


图 3.7　原图以及加入不同噪声后的图像

例 3.8　噪声图像求平均值的方法去除噪声。

```
>> clear;                                          % 清除 workspace 的变量
I = imread('eight.tif'); J = imnoise(I,'gaussian',0,0.02);
subplot(1,2,1),imshow(I),title('原图像');
subplot(1,2,2),imshow(J),title('加噪声后图像');
K = zeros(242,308);
for i = 1:100
J = imnoise(I,'gaussian',0,0.02);
J1 = im2double(J);
K = K + J1;
end
K = K/100;                                         % 求图像的平均
figure,imshow(K),title('相加求平均后的图像');
```

3.2.2　图像的减法运算

图像减法也称为差分方法，是一种常用于检测图像变化及运动物体的图像处理方法。图像减法可以作为许多图像处理工作的准备步骤。例如，可以使用图像减法来检测一系列相同场景图像的差异。图像减法与阈值化处理的综合使用往往是建立机器视觉系统最有效的方法之一。在利用图像减法处理图像时往往需要考虑背景的更新机制，尽量补偿由于天气、光照等因素对图像显示效果造成的影响。

例 3.9　图像相减程序，其结果如图 3.8。

```
>> cameraman = imread('cameraman.tif');
background = (rice, strel('disk',15));
cameraman2 = imsubtract(cameraman.tif, background);
imshow(cameraman);
figure,imshow(cameraman2);
```

图 3.8　原始图像及减去背景后的图像

3.2.3 图像的乘法运算

两幅图像进行乘法运算可以实现掩模操作，即屏蔽掉图像的某些部分。一幅图像乘以一个常数通常被称为缩放，这是一种常见的图像处理操作。如果使用的缩放因子大于1，那么将增强图像的亮度，如果缩放因子小于1，则会使图像变暗。缩放通常将产生比简单添加像素偏移量自然得多的明暗效果，这是因为这种操作能够更好地维持图像的相关对比度。此外，由于时域的卷积或相关运算与频域的乘积运算对应，因此乘法运算有时也被作为一种技巧来实现卷积或相关处理。

例 3.10 图像乘法运算，使用给定的缩放因子对图 3.9 左侧所示的图像进行缩放，从而得到如图 3.9 右侧所示的较为明亮的图像。

```
>> I = imread(room.tif');
J = immultiply(I,1.5);
imshow(I);
figure, imshow(J);
```

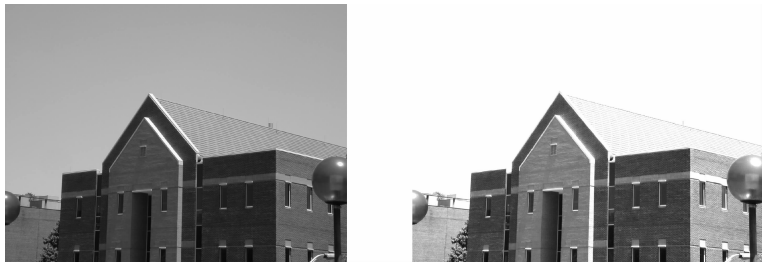


图 3.9 原图和乘以因子 1.5 后的较为明亮的图像

值得注意的是，uint8 图像的乘法操作一般都会发生溢出现象。immultiply 函数将溢出的数据截取为数据类型的最大值。为了避免产生溢出现象，可以在执行乘法操作之前将 uint8 图像转换为一种数据范围较大的图像类型，例如 uint16。

3.2.4 图像的除法运算

除法运算可用于校正成像设备的非线性影响，这在特殊形态的图像（如断层扫描等医学图像）处理中常常用到。图像除法也可以用来检测两幅图像间的区别，但是除法操作给出的是相应像素值的变化比率，而不是每个像素的绝对差异，因而图像除法也称为比率变换。

例 3.11　将图 3.8 所示的两幅图像进行除法运算，其结果如图 3.10 所示。

```
>> cameraman = imread('cameraman.tif');
I = double(cameraman);
J = I * 0.43 + 90;
cameraman2 = uint8(J);
Ip = imdivide cameraman, cameraman 2);
Imshow(Ip, []);
```

图 3.10　原图和减背景后的图像相除后的图像效果

3.2.5　图像的逻辑运算

图像间的逻辑运算主要有“与”、“或”、“非”等，把它们组合起来可以构成其他逻辑运算。

当对灰度级进行逻辑操作时，像素值作为一个二进制数来处理，逻辑操作按位进行。“与”和“或”运算通常用作模板，即通过这些操作可以从一幅图像中提取子图像，更加突出子图像的内容。下例中的“与”和“或”运算通过逐点进行。

例 3.12　图像的逻辑运算程序。

```
>> A = zeros(128);
A(40:67,60:100) = 1;
B = zeros(128);
B(50:80,40:70) = 1;
C = and(A,B);
```

```
D = or(A,B);
E = not(B);
F = xor(A,B);
subplot(231);imshow(A);title('A 图')
subplot(232);imshow(B);title('B 图')
subplot(233);imshow(C);title('A 和 B 相与图')
subplot(234);imshow(D);title('A 和 B 相或图')
subplot(235);imshow(E);title('B 取反图')
subplot(236);imshow(F);title('A 和 B 异或图')
```

3.3 图像几何运算

在数字图像处理的过程中，有时需要对图像的大小和几何关系进行调整，例如对图像进行缩放及旋转等变换，这时图像中像素的灰度值和坐标位置都要发生变化。由于数字图像的坐标是整数，而经过这些变换之后的坐标不一定是整数，因此要对变换之后的坐标位置进行估计。

3.3.1 图像插值法

在对数字图像进行缩放及旋转等变换时，像素的坐标将会发生变化。为确保变化之后的像素坐标正好落在整数坐标处，需要进行插值。

插值是常用的数学运算，通常是利用曲线拟合的方法，通过离散的采样点建立一个连续函数来逼近真实的曲线，用这个重建的函数便可以求出任意位置的函数值。

最近邻插值是最简便的插值，在这种算法中，每一个插值输出像素的值就是在输入图像中与其最临近的采样点的值。最近邻插值是工具箱函数默认使用的插值方法，而且这种插值方法的运算量非常小。当图像中包含像素之间灰度级变化的细微结构时，最近邻插值法会在图像中产生人工的痕迹。双线性插值法的输出像素值是它在输入图像中 2×2 邻域采样点的平均值，它根据某像素周围 4 个像素的灰度值在水平和垂直两个方向上对其插值。双三次插值的插值核为三次函数，其插值邻域的大小为 4×4。它的插值效果比较好，但相应的计算量也比较大。MATLAB 提供了一些函数来实现这些功能。

3.3.2 图像变换

图像的缩放是指在保持原有的图像形状的基础上对图像的大小进行扩大或缩小。例

如对于行数为 m 列数为 n 的图像，通过放大可以将图像变为行数为 $2\times m$，列数为 $2\times n$ 的矩阵，在此过程中用到图像的差值算法。

在 MATLAB 图像处理工具箱中，函数 imresize 可以实现图像的缩放运算。imresize 函数的语法格式为

```
imresize(I, scale,method)
```

其中，I 是要进行缩放的图像矩阵；scale 是进行缩放的倍数；如果 scale 小于 1，则进行缩小操作；如果 scale 大于 1，则进行放大操作；method 用于指定插值的方法，可选用的值为 'nearest'（最邻近法），'bilinear'（双线性插值），'bicubic'（双三次插值），默认为 'nearest'。

例 3.13　图像的缩放运算程序，其运行结果图如图 3.11 所示。

```
>> I = imread('cameraman.tif');
figure,imshow(I);
scale = 0.5;
J = imresize(I,scale);
figure,imshow(J);
```

也可以指定目标图像的大小，此时函数 imresize 的调用格式为

```
imresize(I,[320,480])
```

其中，[320，480]表示将图像的大小调整为 320×480，由此可见，imresize 函数可以改变图像的原有分辨率。

图 3.11　原图及其缩放后的图像

图像的旋转是指使图像做某一角度的转动。在 MATLAB 中，图像旋转的函数是 imrotate。与 imresize 函数一样，imrotate 函数需要调用图像插值法，对旋转后的图像进

行插值。imrotate 常的语法格式为

```
imrotate(I,angle,method)
```

其中，method 用于指定插值的方法，可选用的值为'nearest'（最邻近法）、'bilinear'（双线性插值）、'bicubic'（双三次插值），默认为'nearest'。

例 3.14 图像的旋转运算程序，其运行结果图如图 3.12 所示。

```
>> I = imread('cameraman.tif');
   figure,imshow(I);
   theta = 30;
   K = imrotate(I,theta);                    % 逆时针旋转 30 度
   figure, imshow(K)
```

图 3.12 原图及其旋转后的图像

图像的裁剪是指将图像不需要的部分切除，只保留感兴趣的部分。在 MATLAB 中，图像裁剪的函数是 imcrop，其常见语法格式为

```
I0 = imcrop(I,rect)
```

图像的裁剪第一种调用方法是交互式的操作，即首先显示一幅图像，然后执行这条命令，用鼠标在图像中选中感兴趣的区域，然后这个感兴趣的区域就会存储在矩阵 I0 中。第二种调用方法 rect 规定了裁剪后的图像的区域。

例 3.15 图像的交互式裁剪运算程序，其运行结果图如图 3.13 所示。

```
>> i = imread('xx.png');                     % 读取图像
figure,imshow(i)
i0 = imcrop;                                 % 交互式裁剪
figure,imshow(i0);
```

图 3.13　图像的交互式裁剪运算对比结果图

例 3.16　图像的参数式裁剪运算程序，其运行结果图如图 3.14 所示。

```
>> i = imread('xx.png');                    % 读取
i2 = imcrop(i,[75 68 130 112]);             % 规定裁剪区域的起始位置
imshow(i),figure,imshow(i2)
```

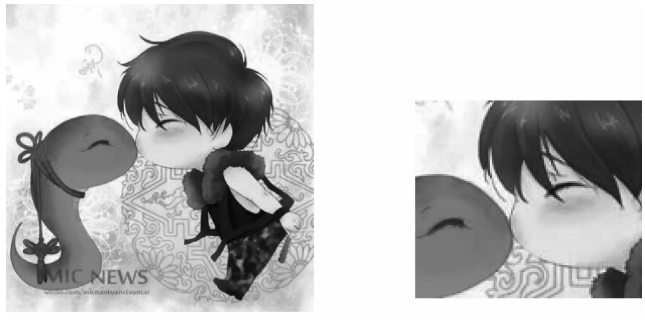


图 3.14　图像的参数式裁剪运算对比结果图

思考题 3

1. 为什么用 I = imread('lena.bmp')命令得到的图像矩阵 I 不可以进行算术运算？
2. 由图像算术运算的运算结果思考，图像减法运算在什么场合上更能发挥其优势？