# 第 3 章　数字图像的基本运算

图像处理的实质是通过对图像的某种运算处理，使处理后的图像满足人的视觉或某种应用需求。从严格的意义上来说，各种图像处理方法都是一种图像运算方法。但从一般意义上来说，图像运算仅指对图像中的所有像素实施了相同处理的那些运算，比如对图像的点运算、直方图运算、代数运算、几何运算、灰度插值运算等。

图像的点运算是指按照某种灰度变换关系，逐像素点地对图像中的每个像素的灰度值进行变换的方法，本章介绍图像的灰度反转和对数变换，图像点运算的其他内容详见 4.1 节。

图像直方图本身就是逐个地对图像中各像素的灰度值出现的频数进行统计的结果，在此基础上形成的图像直方图又引出了一系列的运算和处理方法，最具有代表性的直方图处理方法有直方图均衡和直方图规定化（详见 4.2 节）。从表面上看，直方图均衡和直方图规定化是用于改变图像的直方图分布，但实质上也是一种对图像中的所有像素实施了相同处理的运算，因此也可以称为直方图运算。但由于直方图均衡和直方图规定化的直接目的是进行图像增强，所以一般把该部分内容都和图像增强的相关内容一起介绍。

图像的代数运算是指对两幅（分辨率大小相同的）输入图像进行的点对点的灰度值运算，包括两幅图像的相加运算和两幅图像的相减运算。图像处理中的那些属于模板运算（详见 4.3.4 节）类的处理方法中的相乘运算，比如 4.3 节的图像锐化和 4.4 节的图像噪声消除等，由于其不属于“两幅输入图像进行的点对点的灰度值运算”，所以本书未把这类运算中涉及的相乘运算列入图像的代数运算中来。同理，本书的图像代数运算也不包括图像的相除运算。

图像的几何运算也称为图像的几何变换，主要包括图像的平移变换、图像的旋转变换、图像镜像、图像的转置、图像的缩小与放大等。

本章首先介绍图像的灰度反转和对数变换，然后介绍灰度图像直方图，接着介绍图像的代数运算和几何运算，最后介绍图像的插值运算。

## 3.1　灰度反转

黑白图像的反转就是使灰度值为 1 的像素值变成 0，使灰度值为 0 的像素值变成 1。

对于 256 灰度级图像来说，图像的灰度反转值就是用 255 分别减去原图像 $f(x,y)$ 的各个像素的灰度值。一般地，设图像的灰度级为 $L$，则图像的灰度反转可表示为

$$g(x,y) = L - 1 - f(x,y) \tag{3.1}$$

256 灰度级图像的灰度反转如图 3.1 所示。一般的黑白照片（也即灰度图像照片）的底板和照片之间的关系即为灰度反转关系。

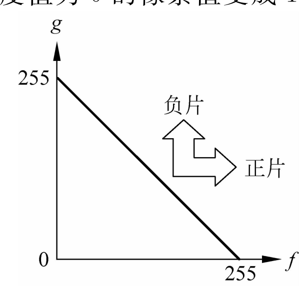



图 3.1　灰度图像的反转关系曲线

## 3.2 对数变换

对原图像 $f(x,y)$进行对数变换的解析式可表示为

$$g(x,y)=c\cdot\log(1+f(x,y)) \tag{3.2}$$

其中，$c$ 是一个常数。

对数变换的作用是对原图像的灰度值动态范围进行压缩，主要用于调高输入图像的低灰度值。比如对于傅里叶频谱来说，要显示的值的范围往往比较大，而傅里叶频谱要显示的重点是突出最亮的像素（对应于低频成分），而这在一个 8 比特的显示系统中将会把频谱图像中的低灰度值部分（对应于高频成分）损失掉。在这种情况下，就可依据式（3.2）对频谱进行变换（这时 $c=1$），调高频谱图像的低灰度值而对高灰度值又尽可能地影响最小。傅里叶频谱的显示一般都是通过这种方式的调整来实现显示的。输入图像和输出图像的对数变换关系如图 3.2 所示。

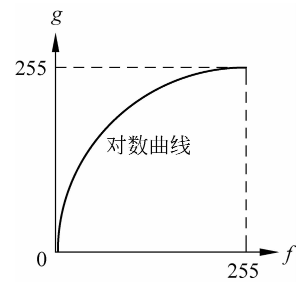



图 3.2　图像的对数变换曲线

## 3.3 灰度直方图

在数字图像处理中，灰度图像的直方图（简称为灰度直方图）是一种描述一幅灰度图像中灰度级内容的最简单且最有用的工具，也是对灰度图像进行多种处理的基础。

### 3.3.1 灰度直方图的概念及分布特征

灰度图像的直方图是一种表示数字图像中各级灰度值及其出现频数的关系的函数。描述灰度图像直方图的二维坐标的横坐标用于表示像素的灰度级别，纵坐标用于表示该灰度出现的频数（像素的个数）。

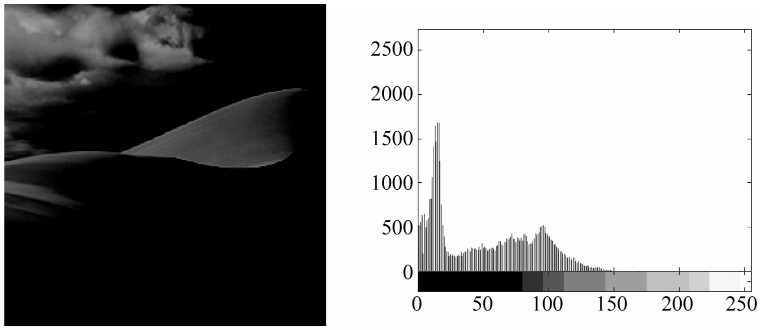
设一幅数字图像的灰度级范围为[0,$L-1$]，则该图像的灰度直方图可定义为

$$h(r_k)=n_k,\quad r_k=0,1,2,\cdots,L-1 \tag{3.3}$$

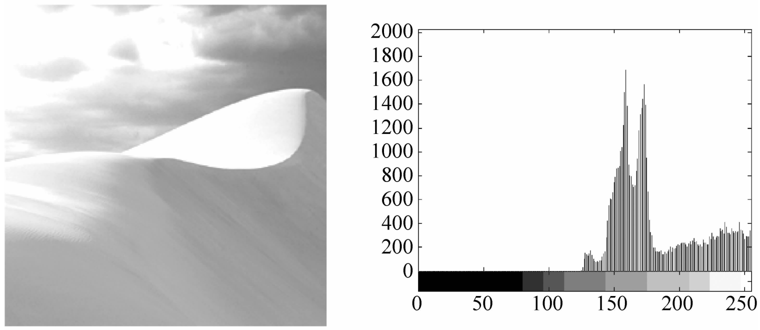
其中，$r_k$ 表示第 $k$ 级灰度值；$n_k$ 表示图像中灰度值为 $r_k$ 的像素的个数；$h(r_k)$是灰度图像的直方图函数。在有些文献中所提及的一维直方图即是本节所讲的灰度图像的直方图。

如图 3.3 所示的是具有四种基本图像类型（暗、亮、低对比度、高对比度）的图像及其灰度直方图。这里，图像的对比度是指图像中一个目标之内或目标与周围背景之间光强的差别。如果成像系统在物体成像过程中的对比度选取的比较低，那么该物体所成的像（目标）看起来就比实际物体要暗一些；如果对比度降至 0，则物体将会从图像中消失。

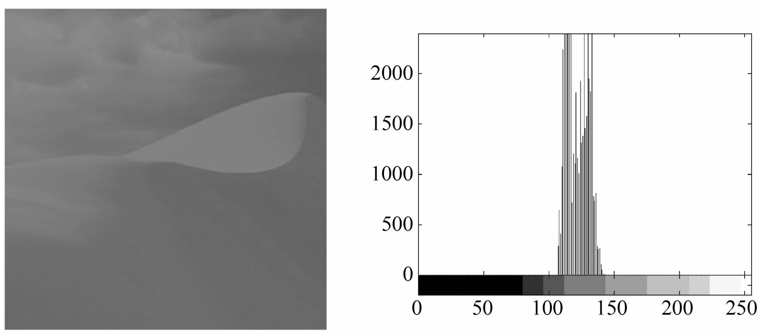
图 3.3 说明了四种基本图像类型图像的直方图分布特征：当图像比较暗时，图像中的各像素的灰度值就比较小，所以直方图中的灰度分布主要集中在低像素级一端（左端）；当图像比较亮时，图像中的各像素的灰度值就比较大，所以直方图中的灰度分布主要集中在高像素级一端（右端）；当图像的对比度比较差（低）时，说明图像中多数较亮的那些像素的灰度值与图像中多数较暗的那些像素的灰度值的差别比较小，所以图像的直方图中的灰度分布就比较集中地分布在某些灰度值范围内，也即图像直方图就会聚集在某些灰度值范围内；当图像的对比度比较好（高）时，图像的直方图中的灰度就会相对比较均匀地分布在整个灰度级范围内。



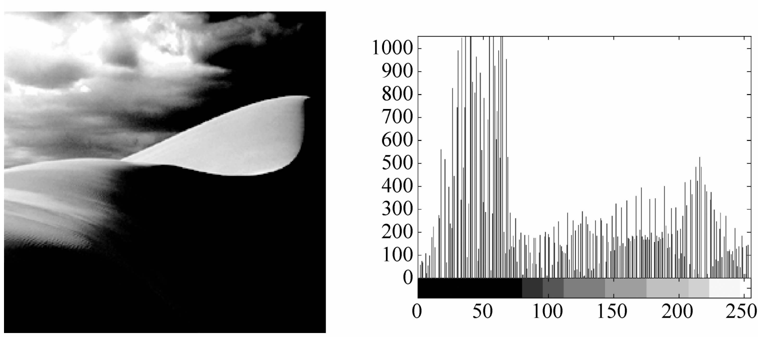

(a) 图像比较暗时的直方图分布特征




(b) 图像比较亮时的直方图分布特征




(c) 图像对比度差时的直方图分布特征




(d) 图像对比度好时的直方图分布特征

图 3.3　四种基本图像类型(暗、亮、低对比度、高对比度)图像及其灰度直方图

### 3.3.2 归一化灰度图像直方图

由于式(3.3)所定义的灰度直方图反映的是图像中各灰度的实际出现频数，这样当某个灰度值的频数(计数值)远远大于其他灰度值的频数时，根据图像的某个或某些像素出现的最大频数来确定直方图的纵坐标的最大尺度既不方便也不太现实，所以就引入了归一化直方图的概念，也即人们通常所说的直方图就是指归一化的直方图。

设 $r_k$ 为图像 $f(x,y)$ 的第 $k$ 级灰度值，$n_k$ 是图像 $f(x,y)$ 中具有灰度值 $r_k$ 的像素的个数，$n$ 是图像 $f(x,y)$ 中的像素总数，则图像 $f(x,y)$ 的灰度直方图定义为

$$P(r_k) = \frac{n_k}{n}, \quad 0 \leqslant r_k \leqslant 1;\ k = 0,1,\cdots,L-1 \tag{3.4}$$

显然，$P(r_k)$ 给出的是 $r_k$ 出现概率的估计，提供的是图像的灰度值分布情况。

### 3.3.3 灰度直方图的特征

灰度图像直方图具有如下一些特征：

(1) 直方图仅能描述图像中每个灰度值具有的像素个数，不能表示图像中每个像素的位置(空间)信息。

(2) 任一特定的图像都有唯一的直方图，不同的图像可以具有相同的直方图。

(3) 对于空间分辨率为 $M\times N$，且灰度级范围为 $[0,L-1]$ 的图像，有关系

$$\sum_{j=0}^{L-1} h(j) = M \times N \tag{3.5}$$

(4) 如果一幅图像由两个不连接的区域组成，则整幅图像的直方图等于两个不连接的区域的直方图之和。

有关直方图的应用，将会在后续相关的章节中介绍。

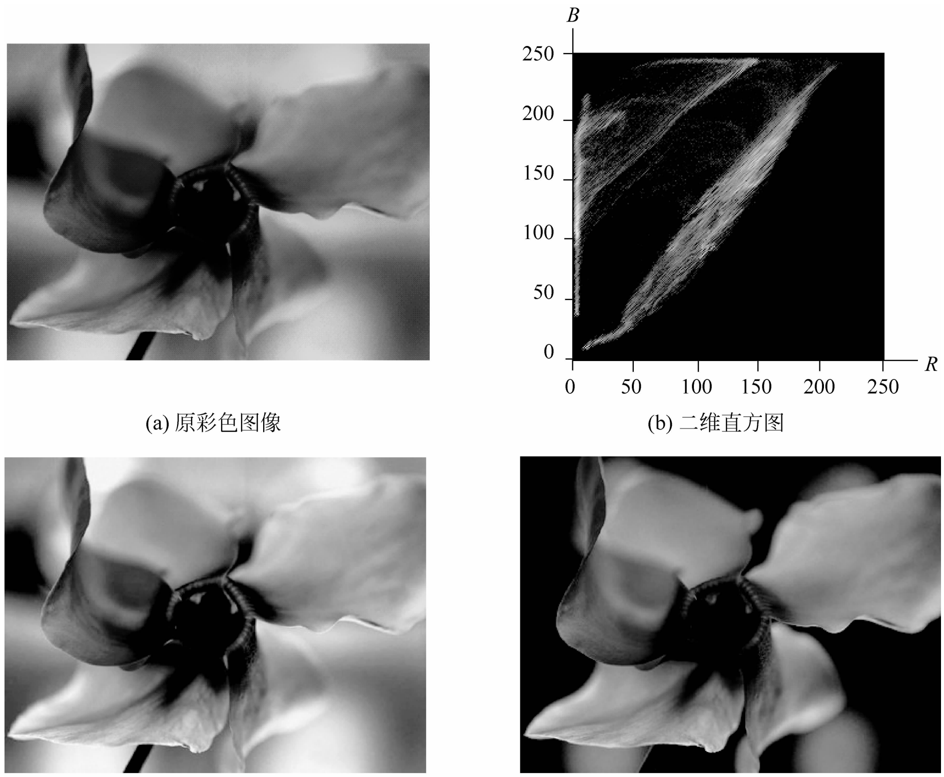
## 3.4 二维直方图

由于灰度直方图在图像描述和图像处理方面的优势和应用，人们在彩色图像研究中引入了二维直方图。

在彩色图像处理中，有一种称为真彩色(true color)的图像，其颜色多达 $2^8\times2^8\times2^8=256\times256\times256=16\ 777\ 216$ 种。为了处理和分析上的方便，这种图像的每个像素直接用 $R$、$G$、$B$ 三种纯色分量表示(每个分量的亮度值用一个字节表示)，也即用一个三分量的值组 $(B,G,R)$ 表示真彩色图像中的一个像素的实际颜色值。

二维直方图是一种描述彩色图像中红色色分量图像的亮度(灰度)值与蓝色色分量图像的灰度值的关系的函数，用一个二维坐标表示，横坐标表示红色分量图像的灰度级别，纵坐标表示蓝色分量图像的灰度级别。也即坐标 $(D_R,D_B)$ 处的值是指在红色分量图像中具有灰度值 $D_R$，且在蓝色分量图像中具有灰度值 $D_B$ 的像素(对)的个数。图 3.4(a)是一幅彩色

图像(本图和第 11 章中的所有彩色图像详见附录 B),该图的纯蓝色分量图像如图 3.4(c)所示,纯红色分量图像如图 3.4(d)所示,反映红色分量图像与蓝色分量图像灰度值关系的二维直方图如图 3.4(b)所示。




(a) 原彩色图像　(b) 二维直方图

(c) 纯蓝色分量图像　(d) 纯红色分量图像

图 3.4　彩色图像及其对应的蓝光图像、红光图像和二维直方图

显然,二维直方图的深度(从纸内穿出至纸外)反映了彩色图像中其红色亮度值为 $D_R$,且蓝色亮度值为 $D_B$ 的像素点的值对($D_R$,$D_B$)的分布情况。若图像的大小为 $M\times N$,则在 256×256 的平面中共有 $M\times N$ 个这样的像素点值对。比如,当 $M=1024$,$N=768$ 时,在 256×256 的平面中共有 786 432 个像素点值对。图 3.4(b)是对 256×256 的平面中的像素点值对($D_R$,$D_B$)的数目先进行对数运算(为了显示效果),然后将其值映射到[0,255]的范围而得到的反应彩色图像中其红色亮度值为 $D_R$,蓝色亮度值为 $D_B$ 的像素点值对($D_R$,$D_B$)的分布情况的(256 灰度级深度)图像。也即反映红色分量图像与蓝色分量图像灰度值关系的二维直方图。

同理可直接用二维直方图描述同一幅彩色图像中的红色分量与绿色分量的亮度(灰度)值的关系。

二维直方图表示了图像中两种颜色分量的灰度值的组合及分布情况。比如,如果二维直方图除 45°斜线上有值外,其余各处值为 0,则表示红色分量图像与蓝色分量图像相同;如果蓝色分量的图像比红色分量的图像更亮一些,则大多数像素值对($D_R$,$D_B$)中的蓝色分量 $D_B$ 就大于红色分量 $D_R$,直方图中的点或由点组成的线将主要分布在 45°斜线以上,图 2.16

就是这种情况。反之，如果红色分量的图像比蓝色分量的图像更亮一些，则大多数像素值对$(D_R,D_B)$中的红色分量$D_R$就大于蓝色分量$D_B$，直方图中的点或由点组成的线将主要分布在45°斜线以下。

## 3.5 图像的代数运算

图像的代数运算包括两幅图像的相加运算和相减运算。

### 3.5.1 图像的相加运算

图像相加是通过对两幅大小相同的图像对应位置像素的相加运算，以产生一幅新的含有两幅图像信息的图像的方法。图像相加也称为图像合成。设$f_1(x,y)$和$f_2(x,y)$分别表示大小为$M\times N$的两幅输入图像，图像$f_1(x,y)$和图像$f_2(x,y)$相加后得到的结果输出图像为$g(x,y)$，且$x\in[0,M-1]$，$y\in[0,N-1]$，则两幅图像的相加运算可表示为

$$g(x,y)=f_1(x,y)+f_2(x,y) \tag{3.6}$$

两幅256灰度级图像对应坐标位置像素值的相加，其结果必然会超过其最大的灰度表示范围256，显然对图像相加运算的结果都需要进行处理，基本方法有三种：一是将两像素灰度值相加后的平均值作为相加结果；二是根据两幅图像所有像素灰度值相加结果的最小值和最大值情况，作等比例缩小，使其结果灰度值符合0～255的灰度值范围；三是当两像素灰度值相加后的值超过255时，取255即可。

图3.5给出了一个两幅图像相加的示例。图3.5(a)和图3.5(b)是两幅不同的输入图像，图3.5(c)是由两幅输入图像进行相加运算后合成的图像。

(a) 输入图像1　　(b) 输入图像2　　(c) 相加结果图像

图3.5　图像的相加运算示例

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。对于那些经过长距离通信传送的图像（如卫星图像），有时就需要进行这种处理。

### 3.5.2 图像的相减运算

设$f_1(x,y)$和$f_2(x,y)$表示大小为$M\times N$的两幅输入图像，从图像$f_1(x,y)$中的各位置的像素值中减去图像$f_2(x,y)$的相应位置的像素值后，得到的结果输出图像为$g(x,y)$，且$x\in[0,M-1]$，$y\in[0,N-1]$，则两幅图像的相减运算可表示为

$$g(x,y) = f_1(x,y) + f_2(x,y) \tag{3.7}$$

当两幅256灰度级图像对应坐标位置像素值相减的结果大于或等于零时，则取其为结果图像中对应位置像素的灰度值；当相减结果小于零时，一般都是取零为结果值。当然，对于某些特殊的应用目的，也可以取其绝对值为结果值。

图像相减运算的典型应用是图像的变化检测。比如，在目前得到广泛应用的图像监控系统，通过定时地将图像监控系统拍摄的现场图像与该现场初始情况下的图像进行相减运算，就可以判定被监控场景是否有异样情况出现。如图3.6所示，图3.6(a)是监控系统某时刻拍摄的现场监控图像，图3.6(b)是被监控现场的初始图像，图3.6(c)是从图3.6(a)中减去图3.6(b)后的结果图像。


(a) 某时刻监控图像 (b) 现场初始图像 (c) 相减结果图像

图3.6 图像的相加运算示例

这里需要注意的问题是，图像的灰度级是一个已有约定的有限大小的整数。以256的灰度级图像为例，图像代数运算的结果理应要求像素值不能大于255，也不能为负数。所以在有关的应用中一般都有对运算结果中不符合要求的像素值的处理约定。比如，对运算结果中像素值为负的值取绝对值，对运算结果大于255的像素值取关于256的模运算(MOD)结果等。

## 3.6 图像的几何运算

图像的几何运算又称为图像的几何变换，用于使原图像产生大小、形状和位置等变化效果。图像的几何运算包括图像的平移变换、图像的旋转变换、图像镜像、图像转置、图像的缩放等。

### 3.6.1 图像平移变换

图像平移(image translation)变换是指将一幅图像或一幅图像中的子图像块(以下简称为图像块)中的所有像素点，都按指定的 $x$ 方向偏移量 $\Delta x$ 和 $y$ 方向偏移量 $\Delta x$ 进行移动。

设初始坐标为$(x_0, y_0)$的像素平移$(\Delta x, \Delta y)$后的坐标为$(x_1, y_1)$，如图3.7所示，则有

$$\begin{cases} x_1 = x_0 + \Delta x \\ y_1 = y_0 + \Delta y \end{cases} \tag{3.8}$$

图像平移变换的矩阵形式为

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & \Delta x \\ 0 & 1 & \Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{3.9}$$

同理，可以根据坐标$(x_1,y_1)$求解原始坐标$(x_0,y_0)$，即有图像平移变换的逆变换为

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & -\Delta x \\ 0 & 1 & -\Delta y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{3.10}$$

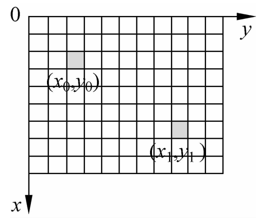



图 3.7　平移原理图

图像（块）的平移结果一般分两种情况：

（1）图像块平移。也即将一幅图像中的某个子图像块平移到另一处。例如在图 3.8 中，lena 图像中 lena 眼部的一个子图像块被平移到了该图像的右下角，该子图像块同时覆盖掉了新位置上原来的那些图像内容。

(a) 原图像　　(b) 平移图像子块后的图像

图 3.8　图像（图像子块）平移示例

（2）整幅图像平移。整幅图像平移后，相对于原来图像（位置）来说，如果完整保持被平移的原图像内容，形成的新结果图像的幅面就被放大了；如果平移后的结果图像仍保持原来图像的幅面大小，被移出的部分就要被截掉。图 3.9 给出了整幅图像平移后不放大而将移出的部分截断的例子。

(a) 移动前的图像　　(b) 平移后的图像

图 3.9　整幅图像平移截断移出部分图像的示例

### 3.6.2　图像旋转变换

图像旋转（image rotation）变换是指以图像的中心为原点，将图像中的所有像素（也即整幅图像）旋转一个相同的角度。

与图像平移变换类似，图像旋转变换的结果图像也分为两种情况：一是旋转后的图像幅面被放大，如图 3.10 所示；二是保持图像旋转前后的幅面大小，把旋转后图像被转出原幅面大小的那部分截断。下面介绍的图像旋转变换方法仅考虑第一种情况，也即不考虑如何截断转出部分的细节问题。

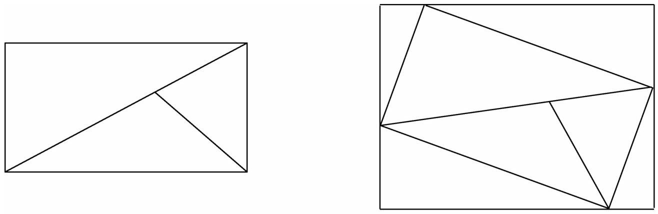

(a) 旋转前的图像　　(b) 旋转后的图像

图 3.10　整幅图像旋转后图像幅面放大示例

**1. 基于 *xoy* 平面坐标系的点旋转变换**

在图 3.11 的 $xoy$ 平面坐标系中，设位于$(x_0, y_0)$处的点到坐标原点的直线 $r$ 与 $x$ 轴的夹角为 $\alpha$，直线 $r$ 顺时针旋转 $\beta$ 角度后使位于$(x_0, y_0)$处的点被旋转至$(x_1, y_1)$处。

显然，在旋转前

$$\begin{cases} x_0 = r\cos\alpha \\ y_0 = r\sin\alpha \end{cases} \tag{3.11}$$

旋转后

$$\begin{cases} x_1 = r\cos(\alpha-\beta) = r\cos\alpha\cos\beta + r\sin\alpha\sin\beta \\ y_1 = r\sin(\alpha-\beta) = r\sin\alpha\cos\beta - r\cos\alpha\sin\beta \end{cases} \tag{3.12}$$

将式(3.11)代入式(3.12)得

$$\begin{cases} x_1 = x_0\cos\beta + y_0\sin\beta \\ y_1 = -x_0\sin\beta + y_0\cos\beta \end{cases} \tag{3.13}$$

由式(3.13)即可得到基于 $xoy$ 平面坐标系的点旋转变换的矩阵表示形式为

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{3.14}$$

同理，可以根据旋转后的点坐标$(x_1, y_1)$求解旋转前的点坐标$(x_0, y_0)$，也即有基于 $xoy$ 平面坐标系的点旋转变换的逆变换为

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{3.15}$$

**2. 图像像素点的旋转变换**

前面介绍的是基于 $xoy$ 平面坐标系的点旋转变换。由于图像像素点的旋转变换是基于图像的显示坐标的，所以还需将式(3.14)的变换结果映射到图像的显示坐标中。图 3.12 给出了图像的像素点与图像的显示坐标和 $xoy$ 平面坐标的关系。

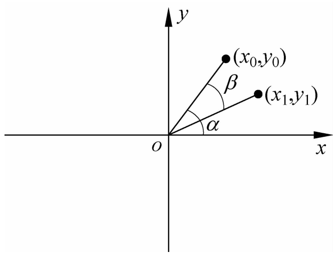



图 3.11　基于 $xoy$ 平面坐标系的点旋转原理示意图

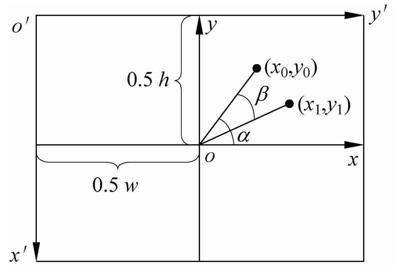



图 3.12　图像的像素点与图像的显示坐标和 $xoy$ 平面坐标的关系示例

假设图像的宽度 width 用 $w$ 表示，图像的高度 high 用 $h$ 表示，则由图 3.12 可知：原点 $o'$在左上角，$x'$轴方向朝下，$y'$轴方向朝右的图像显示坐标 $x'o'y'$中，如果用$(x'_0, y'_0)$表示$(x_0, y_0)$在图像显示坐标 $x'o'y'$中位置，用$(x'_1, y'_1)$表示$(x_1, y_1)$在图像显示坐标 $x'o'y'$中位置，则有

$$\begin{cases} x'_0 = 0.5h - y_0 \\ y'_0 = 0.5w + x_0 \end{cases} \tag{3.16}$$

$$\begin{cases} x'_1 = 0.5h - y_1 \\ y'_1 = 0.5w + x_1 \end{cases} \tag{3.17}$$

式(3.16)和式(3.17)的矩阵表示形式分别为

$$\begin{bmatrix} x'_0 \\ y'_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0.5h \\ 1 & 0 & 0.5w \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{3.18}$$

$$\begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0.5h \\ 1 & 0 & 0.5w \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{3.19}$$

将式(3.14)代入式(3.19)，可得图像像素点的旋转变换公式为

$$\begin{bmatrix} x'_1 \\ y'_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0.5h \\ 1 & 0 & 0.5w \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & \sin\beta & 0 \\ -\sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{3.20}$$

同理，将式(3.15)代入式(3.18)，可得图像像素点的旋转变换的逆变换公式为

$$\begin{bmatrix} x'_0 \\ y'_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & -1 & 0.5h \\ 1 & 0 & 0.5w \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\beta & -\sin\beta & 0 \\ \sin\beta & \cos\beta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{3.21}$$

在利用式(3.20)进行图像(块)旋转变换时，需要注意以下问题：

(1) 图像的高度 $h$ 的值和宽度 $w$ 的值取自于图像矩阵的大小。比如，若图像的像素矩阵为 640×480，含义是说该图像的行数为 640，列数为 480。所以图像的高度 $h$ 的值为 640，图像的宽度 $w$ 的值为 480。

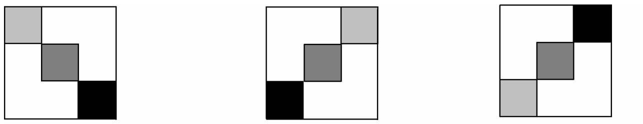
(2) 关于式(3.20)中$(x_0, y_0)$的取值问题。因为假设位于$(x_0, y_0)$处的点是相对于 $xoy$ 平面坐标系的，所以当位于图像显示坐标 $x'o'y'$中的图像要旋转时，可取图像的高度 $h$ 和宽

度 $w$ 的中值处为 $xoy$ 平面坐标系的原点，然后按照图像中不同位置的像素点在该 $xoy$ 平面坐标系中的四个不同象限的取值特点(为正或为负)，就可取得相应的 $x_0$ 的值和 $y_0$ 的值。

(3) 由式(3.20)可知，计算得到的坐标值 $x_1'$ 和 $y_1'$ 一般不会是整数，但数字图像旋转后的坐标值必须是整数，因此应尽可能地取与 $x_1'$ 和 $y_1'$ 最接近的整数值。也正是这种近似，所以图像旋转后会有一定的改变，但这种改变肯定不会明显。

### 3.6.3 图像镜像变换

图像镜像(image mirror)变换分为图像水平镜像变换和图像垂直镜像变换两种。图像水平镜像是指以原图像为参照，使原图像和水平镜像结果图像与虚拟的垂直轴成对称关系，如图 3.13(a)和图 3.13(b)所示；图像垂直镜像是指以原图像为参照，使原图像和垂直镜像结果图像与虚拟的水平轴成对称关系，如图 3.13(a)和图 3.13(c)所示。



(a) 原图像 (b) 水平镜像结果图像 (c) 垂直镜像结果图像

图 3.13 图像镜像结果示例

#### 1. 图像水平镜像

设图像的高度(high)为 $h$，宽度(width)为 $w$；原图像中位于$(x_0,y_0)$处的像素点，经水平镜像后在水平镜像结果图像上的对应像素点为$(x_1,y_1)$。则图像水平镜像变换可表示为

$$\begin{cases} x_1 = w - x_0 \\ y_1 = y_0 \end{cases} \tag{3.22}$$

式(3.22)的矩阵表示形式为

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} -1 & 0 & w \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{3.23}$$

图像水平镜像变换的逆变换可表示为

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & w \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{3.24}$$

#### 2. 图像垂直镜像

设图像的高度(high)为 $h$，宽度(width)为 $w$；原图像中位于$(x_0,y_0)$处的像素点，经垂直镜像后在垂直镜像结果图像上的对应像素点为$(x_1,y_1)$。则图像垂直镜像变换可表示为

$$\begin{cases} x_1 = x_0 \\ y_1 = h - y_0 \end{cases} \tag{3.25}$$

式(3.25)的矩阵表示形式为

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & h \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{3.26}$$

图像垂直镜像变换的逆变换可表示为

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & -1 & h \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{3.27}$$

### 3.6.4 图像转置变换

图像转置(image transpose)变换是指将图像显示坐标的 $x$ 轴与 $y$ 轴对换。

设原图像位于$(x_0, y_0)$处的像素点，经图像转置变换后在转置变换结果图像上的对应像素点为$(x_1, y_1)$。则图像转置变换可表示为

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{3.28}$$

同理，有图像转置变换的逆变换

$$\begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} \tag{3.29}$$

需要特别注意的是，图像的转置变换与图像的旋转变换是不一样的。原图像不论是顺时针旋转 90°还是逆时针旋转 90°，都不会得到图像转置后的结果。图像转置的示例如图 3.14所示。

(a) 转置前的图像

(b) 转置后的图像

图 3.14　图像转置变换示例

### 3.6.5 图像缩放

图像缩放(image scaling)是指对图像进行缩小或放大，也即对数字图像的大小进行调

整的过程。

设原图像中位于$(x_0,y_0)$处的像素点，经对原图像的行和列按相同比例 $r$ 缩放后，在缩放后的结果图像上的对应像素点为$(x_1,y_1)$。则图像缩放前后像素点的坐标可表示为

$$\begin{cases} x_1 = rx_0 \\ y_1 = ry_0 \end{cases} \tag{3.30}$$

图像缩放的矩阵表示形式为

$$\begin{bmatrix} x_1 \\ y_1 \\ 1 \end{bmatrix} = \begin{bmatrix} r & 0 & 0 \\ 0 & r & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_0 \\ y_0 \\ 1 \end{bmatrix} \tag{3.31}$$

其中，当 $0<r<1$ 时为缩小原图像；当 $r>1$ 时为放大原图像。

### 1. 缩小图像

缩小图像的目的一般有两个：一是为了使缩小后的图像符合显示区域的大小要求；二是为了生成被缩小图像(原图像)的缩略图。

最简便的图像缩小方法是将图像的行和列都缩小一半，从整体上看就是将原图像缩小到原来大小的 1/4。

对于一般的行数和列数都为偶数的图像来说，一种方法是只取原图像偶数行和偶数列交叉处的像素，如图 3.15 所示；另一种方法是只取原图像奇数行和奇数列交叉处的像素，如图 3.16 所示。

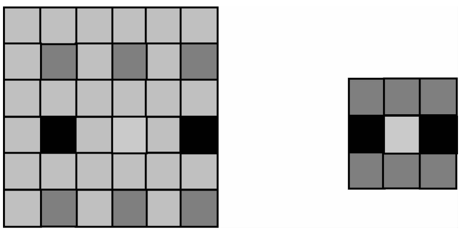


图 3.15 仅取偶数行和偶数列像素缩小原图像

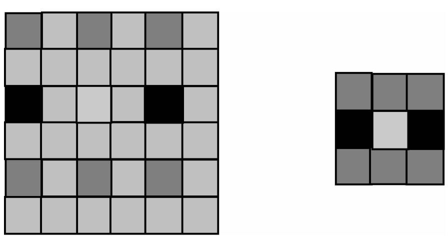


图 3.16 仅取奇数行和奇数列像素缩小原图像

如果要再将缩小后的图像缩小为原来的 1/4，只要在前述缩小的图像的基础上，采用原来方法进行即可。

当然还有其他一些不按整数比例缩小图像的方法，鉴于篇幅所限，不再赘述。

### 2. 放大图像

放大图像的目的一般是为了使放大后的图像更好地显示在更高分辨率的显示设备上。放大图像的方法相对比较多，也相对复杂一些，从了解图像放大原理的角度。下面介绍最基本也是最典型的最近邻域插值图像放大方法。

1）按整数倍放大图像的最近邻域插值法

按整数倍放大图像的最近邻域插值法放大图像的基本思想是：将原图像中的每一个原像素原封不动地复制映射到放大后的新图像中的 4 个像素中，其原理如图 3.17 所示。

(a) 原图像　　(b) 放大后的图像

图 3.17　按最近领域插值法将图像放大 4 倍示例

2）按非整数倍放大图像的最近邻域插值法

设放大前的图像称为原图像，其在显示坐标的 $x$ 方向和 $y$ 方向的坐标值分别用 $x_{\text{old}}$ 和 $y_{\text{old}}$ 表示，图像高度（$x$ 方向）用 $h_{\text{old}}$ 表示，图像宽度（$y$ 方向）用 $w_{\text{old}}$ 表示。放大后的图像称为目标（新）图像，其在显示坐标的 $x$ 方向和 $y$ 方向的坐标值分别用 $x_{\text{new}}$ 和 $y_{\text{new}}$ 表示，图像高度（$x$ 方向）用 $h_{\text{new}}$ 表示，图像宽度（$y$ 方向）用 $w_{\text{new}}$ 表示。则按非整数倍放大图像的最近邻域插值法的原图像和目标图像的坐标关系可表示为

$$\begin{cases} x_{\text{old}} = x_{\text{new}} \times (h_{\text{old}}/h_{\text{new}}) \\ y_{\text{old}} = y_{\text{new}} \times (w_{\text{old}}/w_{\text{new}}) \end{cases} \tag{3.32}$$

下面用一个例子来说明按非整数倍放大图像的最近邻域插值法的应用方法。

**【例 3.1】**　设已知有一个 3×3 的灰度图像，如图 3.18(a)所示。利用按非整数倍放大图像的最近邻域插值法将该图像放大为 4×4 的图像。

| | | |
|---|---|---|
| 234 | 38 | 22 |
| 67 | 44 | 12 |
| 89 | 65 | 63 |

(a) 原图像

| | | | |
|---|---|---|---|
| 234 | 38 | 22 | 22 |
| 67 | 44 | 12 | 12 |
| 89 | 65 | 63 | 63 |
| 89 | 65 | 63 | 63 |

(b) 放大后的图像

图 3.18　按非整数倍放大图像的最近邻域插值法的放大结果示例

**解**：根据式(3.32)逐个地计算插值放大后的目标图像中从(0,0)至(3,3)的每个像素的值。

对于目标图像中坐标为(0,0)处的像素，因为有

$$x_{\text{old}} = x_{\text{new}} \times (h_{\text{old}}/h_{\text{new}}) = 0 \times (3/4) = 0$$
$$y_{\text{old}} = y_{\text{new}} \times (w_{\text{old}}/w_{\text{new}}) = 0 \times (3/4) = 0$$

也即，目标图像中位于(0,0)处的像素值应是原图像中位于(0,0)处的像素值，也

即 234。

对于目标图像中坐标为(0,1)处的像素,因为有

$$x_{\text{old}} = x_{\text{new}} \times (h_{\text{old}}/h_{\text{new}}) = 0 \times (3/4) = 0$$

$$y_{\text{old}} = y_{\text{new}} \times (w_{\text{old}}/w_{\text{new}}) = 1 \times (3/4) = 0.75 \approx 1$$

也即,目标图像中位于(0,1)处的像素值应是原图像中位于(0,1)处的像素值,也即 38。

同理,可得目标图像中位于(0,2)处和(0,3)处的像素值都是 22;位于(1,0)处、(1,1)处、(1,2)和(1,3)处的像素值分别是 67、44、12 和 12;位于(2,0)、(2,1)、(2,2)和(2,3)处的像素值分别是 89、65、63 和 63;位于(3,0)、(3,1)、(3,2)和(3,3)处的像素值分别是 89、65、63 和 63。放大后的结果图像如图 3.18(b)所示。

最近领域插值法放大的图像可保留图像中所有的原始信息,但是会产生锯齿现象和马赛克现象。为了克服最近领域插值法的不足,人们进一步提出了双线性插值法、三次样条插值和基于边缘的图像插值方法等,感兴趣的读者可以参考有关文献。

## 习题 3

3.1 解释下列术语:

(1) 灰度反转　　(2) 图像对比度

(3) 归一化直方图　　(4) 图像合成

(5) 图像平移　　(6) 图像旋转

(7) 图像镜像　　(8) 图像转置

(9) 图像缩放

3.2 在有些图像处理系统中,为什么要对输入图像的像素亮度进行对数运算处理?

3.3 灰度直方图有哪些性质?

3.4 简述灰度直方图在图像处理中有哪些用途。

3.5 简述并举例说明灰度直方图是如何描述图像的暗、亮和对比度特征的。

3.6 简述图像变化检测的实现方法。

3.7 简述图像加法运算的实现方法。

3.8 简述按整数比例缩小图像的方法。

3.9 简述按整数比例放大图像的方法。

3.10 简述二维直方图描述了彩色图像的哪些特征。

3.11 简述图像的减法运算有哪些典型应用。