第 3 章

CHAPTER 3

图 像 变 换

本章的结构如图 3-1 所示，具体要求如下：

（1）了解图像的几何变换。

（2）了解图像的离散余弦变换原理。

（3）掌握图像的离散傅里叶变换及其性质。

（4）掌握图像的小波变换。

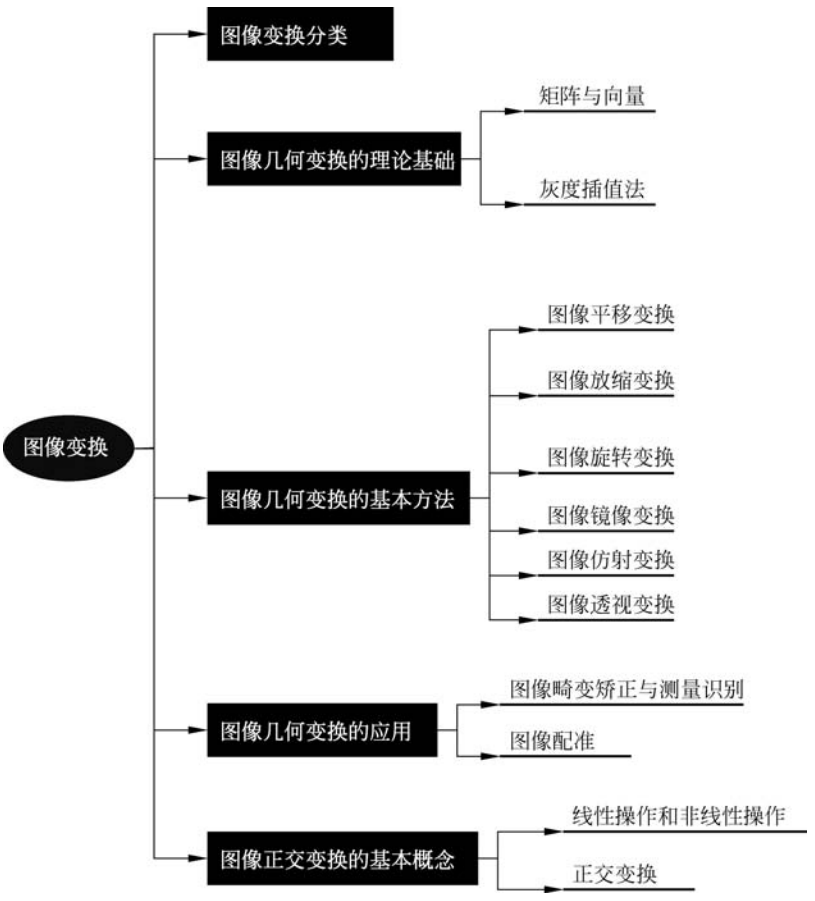


图 3-1　本章结构

图像变换

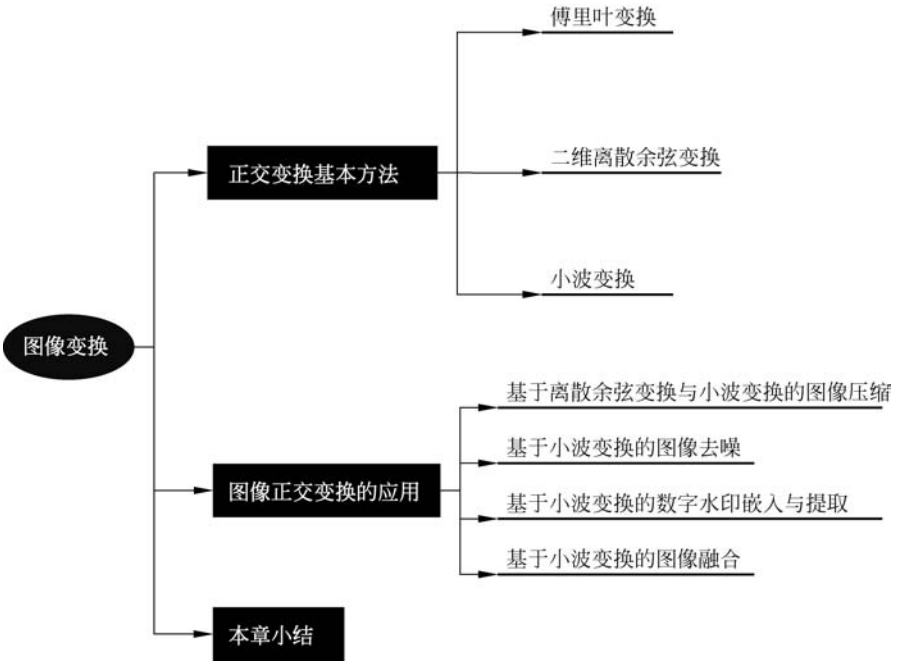


图 3-1 （续）

3.1 图像变换分类

图像变换在数字图像处理的领域中占有十分重要的地位。通常来说，图像变换的方法可以归为两种：空域法和频域法。图像变换的方法如图 3-2 所示。

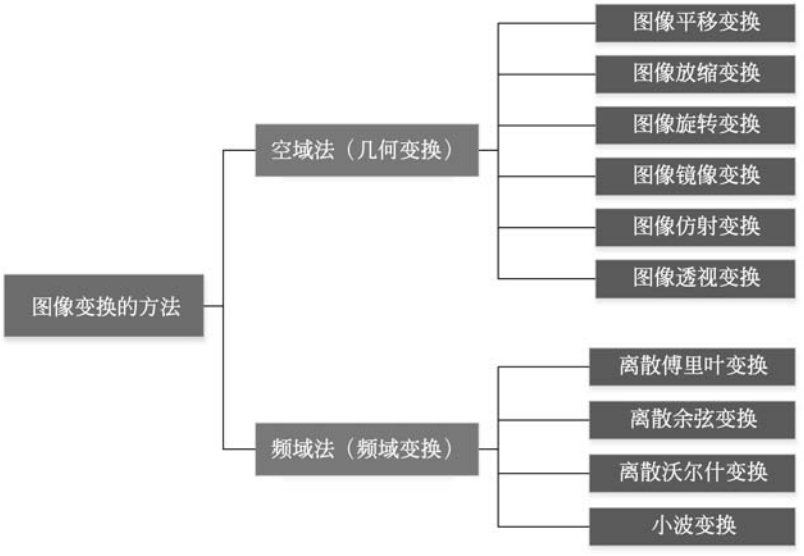


图 3-2 图像变换的方法

图像的空域就是通常所说的像素域，空域法就是在图像的空域进行像素的处理，比如图像叠加。本章的几何变换就是在图像的空域进行的。频域是描述信号频率特性的一种坐标系，它的横坐标轴是信号的频率，纵坐标轴是信号的幅度。图像变换借鉴了频域这一特性，发展出频域变换。频域法首先将图像从原空域以某种形式转换到频域，然后利用该空间的性质进行处理，最后再转换回图像原空域，实现所需效果。本章介绍的离散傅里叶变换、离散余弦变换、小波变换等正交变换就是在频域进行的图像变换。

3.2　图像几何变换的理论基础

3.2.1　矩阵与向量

在数学中，矩阵(Matrix)是一个按照矩形阵列排列的复数或实数集合，是高等代数中的常见工具，也常被用于统计分析等应用数学。一般来说，图像是一个标准的矩形，有长和宽。与之相应，矩阵有着行和列。因此，图像可以用矩阵来表示，其元素为像素。图像和二值矩阵的转换如图 3-3 所示。图 3-3(a)为一棵树。在进行过放大处理后，这棵树变得模糊不清，如图 3-3(b)所示。从图 3-3(b)中可以观察到，经过放大处理后的图片由一个个像素构成。图 3-3(c)为经过灰度处理的图像，灰度值的取值范围为 0～255，其中颜色越深，灰度值越小；颜色越浅，灰度值越大。设一个合适的阈值，当灰度值小于这个阈值时，将其置为 1；反之，当灰度值大于或等于这个阈值时，将其置为 0，如此得到如图 3-3(d)所示的二值矩阵。

(a) 原图像　(b) 放大后的图片　(c) 灰度化后的图片　(d) 二值矩阵

图 3-3　图像与二值矩阵的转换

在很多情况下，图像的处理都是使用矩阵理论来进行的。矩阵理论的应用在数学和计算机中都很常见且成熟。因而，图像很自然地被当作一个矩阵，图像变换也转换成矩阵理论的应用。实际上，所有的图像处理工具也是这么做的。

3.2.2　灰度插值法

在学习几何变换之前，需要先学习灰度插值法。在数字图像处理中，几何变换是将一幅图像映射到另一幅图像的操作，可以分为放缩、翻转、仿射，即平移加旋转、透视等。

图像在进行几何变换时，并不是所有的操作可以通过对像素进行赋值来实现。例如，将图像放大两倍，这必然会多出一些无法被直接映射的像素。对于这些像素，可以通过插值决定它们的灰度值。但是，不同的插值法得到的结果是不同的。常见的灰度插值法有最近邻插值法(Nearest Neighbor Interpolation，NNI)、双线性插值法(Bilinear Interpolation)和双立方插值法(Bicubic Interpolation)。

1. 最近邻插值法

顾名思义，最近邻插值法，就是用离待估像素最近的像素的灰度值来作为待估像素的灰度值。最近邻插值法示意如图 3-4 所示。图 3-4(a)为由 3×3 像素组成的原图像，其中，黑色方块为待估像素。如图 3-4(b)所示，待估像素即要计算像素 O 的灰度值，其中，A、B、C、D 分别表示原图像的像素。通过分别计算像素 A、B、C、D 到 O 的距离 a，b，c，d，可以得出，A 为距待估像素最近的像素，因而用像素 A 来代替我们要计算的像素的值，即像素 O 的灰度值。

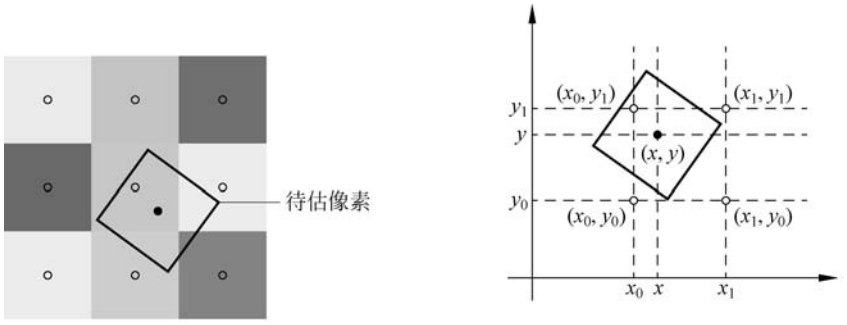
(a) 原图像　　(b) 各像素距离对比

图 3-4　最近邻插值法示意

最近邻插值法是一种最基本、最简单快捷的灰度插值法，该方法的计算量较小。但是，采用最邻近插值法放大后的图像有严重的马赛克现象，且在灰度变化的地方可能出现明显的锯齿状。并且，这种方法会使缩小后的图像产生严重失真。

2. 双线性插值法

双线性插值法对待估像素最邻近的 4 个像素的灰度值进行计算，得出待估像素的灰度值，如图 3-5 所示。图 3-5(a)为一幅 3×3 像素的原图像，黑色方块为待估像素。如图 3-5(b)所示，要得到像素(x,y)的灰度值，需要对其最邻近的 4 个像素(x_0,y_0)，(x_0,y_1)，(x_1,y_0)，(x_1,y_1)进行计算，具体操作为首先在 x 轴方向进行两次线性插值计算，然后在 y 轴方向上进行一次插值计算。



(a) 原图像　　(b) 最邻近4个像素与待估像素的距离

图 3-5　双线性插值法示意

假设已知(x_0,y_0)、(x_0,y_1)、(x_1,y_0)、(x_1,y_1)这 4 个像素及其灰度值，那么通过这 4 个像素可以确定一个矩形。在这个矩形内，任意像素的灰度值可以通过插值得到。首先，在

x 轴方向上进行两次线性插值，得到式(3-1)和式(3-2)。

$$f(x,y_0)=\frac{(x_1-x)}{(x_1-x_0)}f(x_0,y_0)+\frac{(x-x_0)}{(x_1-x_0)}f(x_1,y_0) \tag{3-1}$$

$$f(x,y_1)=\frac{(x_1-x)}{(x_1-x_0)}f(x_0,y_1)+\frac{(x-x_0)}{(x_1-x_0)}f(x_1,y_1) \tag{3-2}$$

然后，在 y 轴方向进行一次线性插值，得到式(3-3)。

$$f(x,y)=\frac{(y_1-y)}{(y_1-y_0)}f(x,y_0)+\frac{(y-y_0)}{(y_1-y_0)}f(x,y_1) \tag{3-3}$$

将式(3-1)和式(3-2)代入式(3-3)，可得式(3-4)，即双线性插值结果。

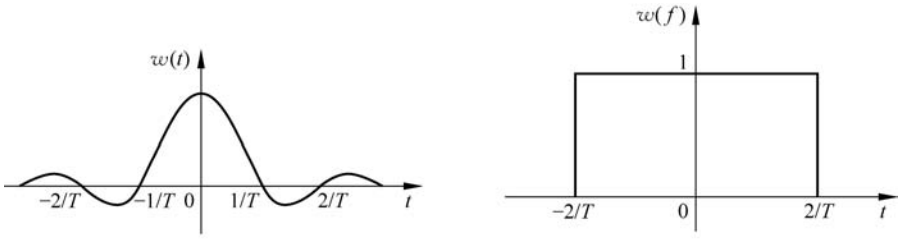
$$\begin{aligned}f(x,y)=&\frac{(y_1-y)(x_1-x)}{(y_1-y_0)(x_1-x_0)}f(x_0,y_0)+\frac{(y_1-y)(x-x_0)}{(y_1-y_0)(x_1-x_0)}f(x_1,y_0)+\\&\frac{(y-y_0)(x_1-x)}{(y_1-y_0)(x_1-x_0)}f(x_0,y_1)+\frac{(y-y_0)(x-x_0)}{(y_1-y_0)(x_1-x_0)}f(x_1,y_1)\end{aligned} \tag{3-4}$$

虽然，双线性插值法的计算比最邻近插值法复杂，且计算量较大，但没有灰度不连续的缺点，其结果基本令人满意。双线性插值法具有低通滤波性质，使高频分量受损，因而使图像的轮廓有一点模糊。

3. 双立方插值法

双立方插值法是一种更复杂的灰度插值法。与最近邻插值法和双线性插值法相比，通过双立方插值法得到的图像具有更平滑的图像边缘。双立方插值法通常被应用于图像处理软件、打印机驱动及数码相机，对原图像或原图像的某些区域进行放大。由 3.2.2.2 节可知，双线性插值法是根据待估像素最邻近的 4 个像素进行的计算，而双立方插值法则是通过计算待估像素周围的 16 个像素进行插值的。这种方法需要以插值基函数为基础进行计算，常用的插值基函数 BiCubic 函数如式(3-5)所示，其波形和频谱如图 3-6 所示。

$$w(t)=\begin{cases}1-2(a+3)\mid t\mid^2+(a+2)\mid t\mid^3, & \mid t\mid<1\\-4a+8a\mid t\mid-5a\mid t\mid^2+a\mid t\mid^3, & 1\leqslant\mid t\mid<2\\0, & \mid t\mid\geqslant 2\end{cases} \tag{3-5}$$



(a) BiCubic函数的波形　　(b) BiCubic函数的频谱

图 3-6　BiCubic 函数波形及其频谱

对于待插值的像素(x,y)，基于 BiCubic 函数的双立方插值法取其附近 4×4 邻域的像素(x_i,y_j)，$i,j=0,1,2,3$；然后按式(3-6)进行插值计算。

$$f(x,y)=\sum_{i=0}^{3}\sum_{j=0}^{3}f(x_i,y_j)w(x-x_i)w(y-y_j) \tag{3-6}$$

3 种灰度插值法的优缺点对比如表 3-1 所示。

表 3-1　3 种灰度插值法的优缺点对比

灰度插值法	优　　点	缺　　点
最近邻插值法	计算量小，实现较容易	容易产生马赛克现象，造成图像模糊
双线性插值法	图像放大效果得到了明显改善，在图像质量要求不高时，可以广泛应用	图像存在瑕疵
双立方插值法	图像放大时可以保留更多的细节	算法复杂，计算量大

3.3　图像几何变换的基本方法

图像几何变换不改变图像像素的灰度值，只是进行像素的重新安排。几何变换改变了像素之间的空间关系，使图像可以放大和缩小，也可以旋转、移动，或者使用其他方法进行其他形式的扩展。适当的几何变换可以最大程度地消除因成像角度、透视关系，甚至镜头自身造成的几何失真所产生的负面影响。这有利于数字图像处理技术将注意力集中于图像内容本身，更确切地说是图像中的对象，而不是该对象的角度和位置等失真。几何变换常常作为数字图像处理的预处理，是图像归一化的核心工作之一。几何变换基本方法如图 3-7 所示。

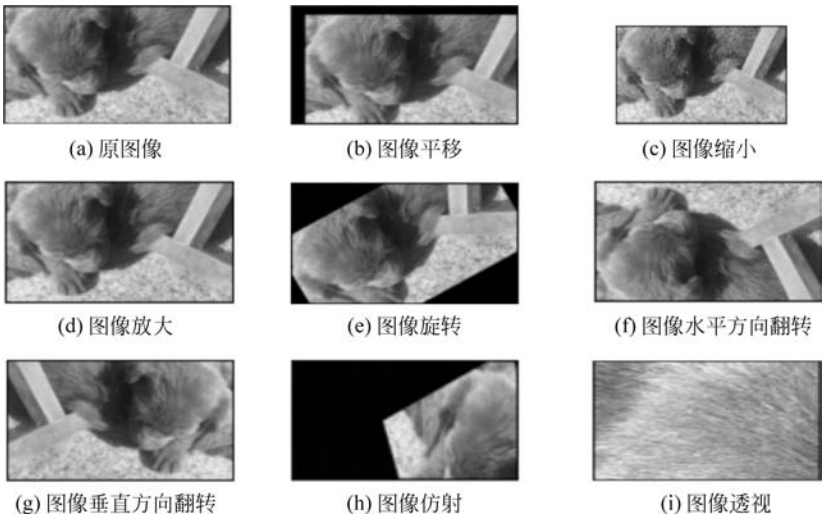


图 3-7　几何变换基本方法

3.3.1　图像平移变换

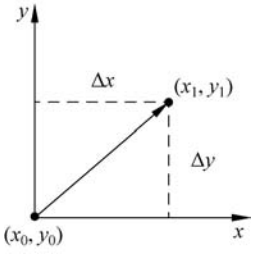


图 3-8　图像平移变换的几何原理

如前文所述，图像可以等价地与矩阵相对应。图像的几何变换就是在矩阵运算的基础上进行的。矩阵运算可以很快地找到像素之间的对应关系。在变换过程中，几何变换会在原图像像素和变换后图像像素之间建立一种映射关系。通过这种关系，几何变换能够计算出变换后像素的坐标。图像平移变换的几何原理如图 3-8 所示。

图像平移变换是将图像中的所有像素按照给定的平移量进行

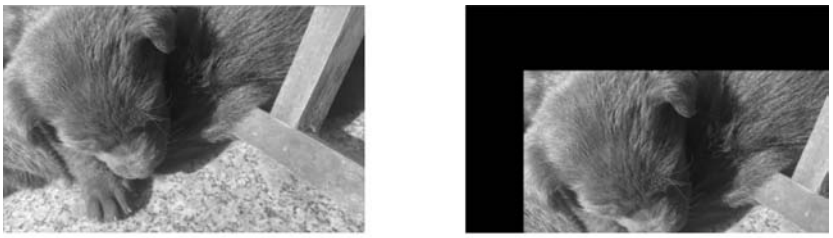
水平或垂直方向的位移。假设原图像像素的坐标为(x_0, y_0)，经过平移量$(\Delta x, \Delta y)$后，像素的坐标为(x_1, y_1)，如式(3-7)所示。

$$\begin{cases} x_1 = x_0 + \Delta x \\ y_1 = y_0 + \Delta y \end{cases} \tag{3-7}$$

式(3-7)的矩阵形式如式(3-8)所示。

$$\begin{bmatrix} x_1 & y_1 & 1 \end{bmatrix} = \begin{bmatrix} x_0 & y_0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \Delta x & \Delta y & 1 \end{bmatrix} \tag{3-8}$$

其中，矩阵$\begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ \Delta x & \Delta y & 1 \end{bmatrix}$为平移变换矩阵，记$\boldsymbol{M}$；$\Delta x$和$\Delta y$为平移量。MATLAB实现图像平移变换，首先要定义平移变换矩阵$\boldsymbol{M}$，然后调用warpAffine()函数。图3-9为图像平移变换的一个简单例子，图3-9(a)经过在垂直和水平方向分别平移了100像素，得到图3-9(b)。



(a) 原图像　　(b) 经过图像平移变换后的图像

图3-9　图像平移变换示例

3.3.2　图像放缩变换

图像放缩变换指的是对数字图像的大小进行调整的过程。经过图像缩小变换后，图像像素的个数减少，承载的信息量减少。图像缩小的方法有两种：基于间隔采样的图像缩小法，基于局部均值的图像缩小法。图像放大，从字面上来看是图像缩小的逆操作。但是，从信息处理的角度来看，图像缩小是对信息的简化和取舍，图像放大是对未知信息的估计，需要为增加的像素选择合适的灰度值。图像放大的方法也有两种：基于像素放大原理的图像放大法，基于双线性插值法的图像放大法。

1. 基于等间隔采样的图像缩小法

基于等间隔采样的图像缩小法通过对原图像进行均匀采样来等间隔地选取一部分像素，从而获得小尺寸图像，并且尽量保持原图像的特征。在图3-10中，对图3-10(a)所示的4×4的像素矩阵按2×2进行等间隔划分，并选取每个区块的左上角的深色像素，然后进行拼接，就得到了一个缩小的深色2×2像素矩阵。简单来说，基于等间隔采样的图像缩小法就是等间隔地选择一些像素，舍弃一些像素，并用选择的像素组成一幅新的图像，使它和原图像的内容差不多，其图像实例如图3-10(b)所示。

基于等间隔采样的图像缩小法由于舍弃了很多像素，无法完全反映被采样的像素信息。通过图3-10(b)可以看出，基于等间隔采样的图像缩小法得到的图像比较生硬，图像中的锯

(a) 等间隔采样原理　　(b) 图像实例

图 3-10　基于等间隔采样的图像缩小法

齿较多。

2. 基于局部均值的图像缩小法

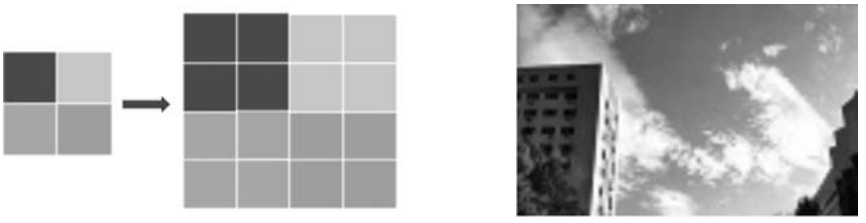
由于等间隔采样存在无法完全反映被采样像素信息的缺点，一种局部均值的缩小方法——基于局部均值的图像缩小法被提了出来。以图 3-11 为例，将原图像分成如图 3-11(a)所示的几个子块，然后分别对子块求均值，并进行拼接从而获得缩小图像。基于局部均值的图像缩小法所得到的图像较为平滑，会有轻微模糊的效果，其图像实例如图 3-11(b)所示。

(a) 局部均值采样原理　　(b) 图像实例

图 3-11　基于局部均值采样的图像缩小法

3. 基于像素放大原理的图像放大法

基于像素放大原理的图像放大法指的是如果需要将原图像放大 K 倍，那么需要将原图像中每个像素的灰度值填在新的图像对应的大小为 $K \times K$ 的子块中。如图 3-12(a)所示，原图像被放大两倍，就是将每个像素的灰度值填在一个大小为 2×2 的子块中。基于像素放大原理的图像放大法实例如图 3-12(b)所示。



(a) 像素放大原理　　(b) 图像实例

图 3-12　基于像素放大原理的图像放大法

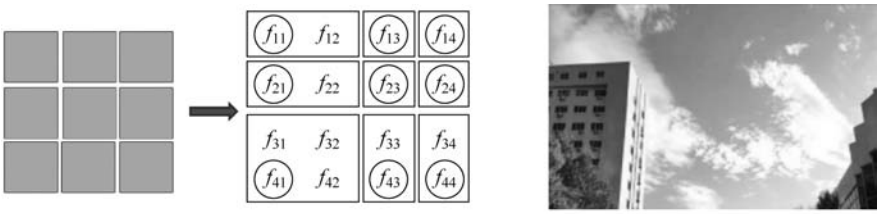
由于图像像素的扩大，基于像素放大原理的图像放大法会导致放大后的图像出现马赛克现象，使图像较为模糊，质量不高。因此，一种改进的图像放大法被提出，即基于双线性插值的图像放大法。

4. 基于双线性插值的图像放大法

基于双线性插值的图像放大法可以消除基于像素放大原理的图像放大法因图像像素扩大而产生的马赛克现象。基于双线性插值的图像放大法使图像的放大效果更加自然。在图 3-13 中，图 3-13(a)为双线性插值放大原理，图 3-13(b)为基于双线性插值的图像放大法实例。基于双线性插值的图像放大法具体步骤如下。

(1) 首先，按照基于像素放大原理的图像放大法，确定原图像的每一个像素在新图像中对应的子块。

(2) 然后，对于新图像中的每一个子块，选取该子块中的一个像素进行填充，如图 3-13(a)所示，其中，$f_{11}\sim f_{44}$ 字符分别表示各像素的灰度值。具体填充方法是：对于最右下角的子块，选取右下角的像素，如图 3-13(a)中的 f_{44}；对于末列非末行的子块，选取右上角的像素，如图 3-13(a)中的 f_{14} 和 f_{24}；对于末行非末列的子块，选取左下角的像素，如图 3-13(a)中的 f_{41} 和 f_{43}；剩下的子块，选取左上角的像素，如图 3-13(a)中的 f_{13}，f_{23}，f_{11} 和 f_{21}。



(a) 双线性插值放大原理　　(b) 图像实例

图 3-13　基于双线性插值的图像放大法

(3) 最后，通过双线性插值法计算图像中剩余像素的灰度值，得到扩大后的图像。

3.3.3　图像旋转变换

图像旋转是指图像以某一点为中心旋转一定的角度，形成一幅新图像的过程。图像旋转变换有一个旋转中心，这个旋转中心一般为图像的中心。如图 3-14 所示，图像像素原坐标为(x_0,y_0)，顺时针旋转 θ 后得到坐标(x_1,y_1)。

图 3-15 为图像旋转变换的实例，其中，图 3-15(a)为原图像，图 3-15(b)为原图像经过逆时针旋转 45°后得到的结果图像。

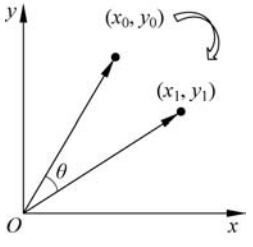
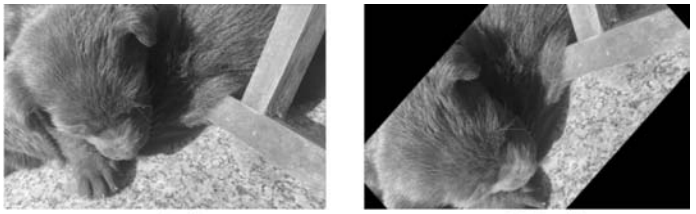


图 3-14　图像旋转变换原理



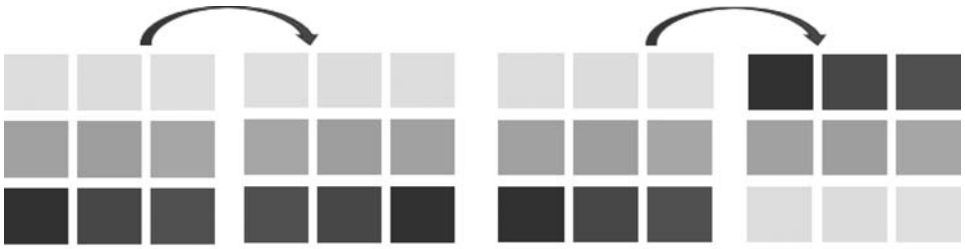
(a) 原图像　　(b) 结果图像

图 3-15　图像旋转变换实例

3.3.4　图像镜像变换

图像镜像变换是图像旋转变换的一种特殊情况，通常包括垂直方向和水平方向的镜像

变换。水平镜像通常是以原图像的垂直中轴线为中心，交换图像的左右部分。同理，垂直镜像是以图像水平中轴线为中心，交换图像的上下部分。图 3-16 为图像镜像变换原理，其中，图 3-16(a)为水平镜像原理，图 3-16(b)为垂直镜像原理。



(a) 水平镜像原理　　(b) 垂直镜像原理

图 3-16　图像镜像变换原理

3.3.5　图像仿射变换

图像仿射变换是一种从一个二维坐标系到另一个二维坐标系的线性变换，是指在几何空间中，由一个向量空间进行一次线性变换和一个平移变换，变换为另一个向量空间。图像仿射变换实例如图 3-17 所示，其中，图 3-17(a)为原图像，图 3-17(b)为结果图像。简单来说，仿射变换即线性变换和平移变换的叠加，是二维平面中一种重要的变换，在图像图形领域有广泛的应用。在二维图像变换中，图像仿射变换的表达式为

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} R_{00} & R_{01} & T_x \\ R_{10} & R_{11} & T_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3-9}$$

其中，T_x 和 T_y 分别表示平移变换的偏移量，R_{00}、R_{01}、R_{10}、R_{11} 分别表示图像线性变换的推移量。

(a) 原图像　　(b) 结果图像

图 3-17　图像仿射变换实例

此外，图像仿射变换保持了二维图像的平直性和平行性。平直性指直线经仿射变换后还是直线，圆弧经仿射变换后还是圆弧。平行性指直线之间的相对位置关系保持不变，平行线经仿射变换后依然为平行线，直线上点的位置顺序不会发生变化。

3.3.6　图像透视变换

如果说图像仿射变换是从一个二维坐标系变换到另一个二维坐标系，是线性变换和平

移变换的叠加，归根结底还是属于线性变换，那么透视变换就是从二维坐标系变换到三维坐标系的过程，属于非线性变换。图像透视变换通过投影的方式，把当前图像映射到另外一个平面。以现实中的例子来说，就像电影院里的胶片放映机，如果幕布或者胶片中任意一个与放映机发出的光线不是呈90°垂直，那么投射到幕布的图像就会发生畸变。这种畸变就是透视畸变的一种。对于畸变图像的校正，图像透视变换需要获得畸变图像4个点的坐标并作为一组，以及目标图像的4个点的坐标并作为一组，通过两组坐标计算出透视变换矩阵，然后对原图像执行透视变换矩阵的变换，实现图像校正。

以图3-18所示的图像透视变换为例，图3-18(a)所示电子书由于拍摄时摄像头角度的原因，导致产生一定程度的畸变；距离摄像头越近的点，看起来越大；越远的点，看起来越小。图像透视变换的目的就是要纠正这种畸变，得到电子书的正视角矩形图像，其校正效果相当于从电子书的正上方视角拍摄的图像，即图3-18(b)所示的图像。

(a) 原图像　　(b) 结果图像

图3-18　图像透视变换实例

图像透视变换的矩阵表达式为

$$[x' \quad y' \quad w'] = [u \quad v \quad w] \begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix} \tag{3-10}$$

其中，$\begin{bmatrix} a_{11} & a_{12} & a_{13} \\ a_{21} & a_{22} & a_{23} \\ a_{31} & a_{32} & a_{33} \end{bmatrix}$为透视变换矩阵，$[u \quad v \quad w]$为原图像矩阵，$[x' \quad y' \quad w']$为变换后的图像矩阵，$\begin{bmatrix} a_{11} & a_{12} \\ a_{21} & a_{22} \end{bmatrix}$实现线性变换，$\begin{bmatrix} a_{13} \\ a_{23} \end{bmatrix}$实现平移变换，$[a_{31} \quad a_{32}]$实现透视变换，$[a_{33}]$实现全比例变换。

图像透视变换的数学表达式如式(3-11)所示。

$$\begin{cases} x = \dfrac{x'}{w'} = \dfrac{a_{11}u + a_{21}v + a_{31}}{a_{13}u + a_{23}v + a_{33}} \\ y = \dfrac{y'}{w'} = \dfrac{a_{12}u + a_{22}v + a_{32}}{a_{13}u + a_{23}v + a_{33}} \end{cases} \tag{3-11}$$

由此可知，给定图像透视变换对应的4个点坐标，即原图像像素的坐标和变换后图像像素的坐标，联立可求得透视变换矩阵。

3.4 图像几何变换的应用

3.4.1 图像畸变校正与测量识别

一般提到图像畸变，大家都会想到镜头畸变，但本节的图像畸变是几何畸变，例如，仿射畸变、投影畸变。简单来说，几何畸变就是拍摄角度未对正，使拍摄的图像发生了形变。图像畸变校正利用消失点原理，使图像中原有的平行/垂直关系得以保持，得到正视图的效果。

几何透视如图 3-19 所示，相交于无穷远的平行线，经过投影变换相交于一点，即消失点。不同的平行线相交于不同的消失点，而一个平面上各组平行线的消失点共线。而消失线方程与投影变换矩阵有关。因此，在图像上描出几组平行关系，便可以计算消失点和消失线，进而反解出投影变换矩阵。图像畸变校正利用此矩阵进行图像变换，来去除几何畸变。

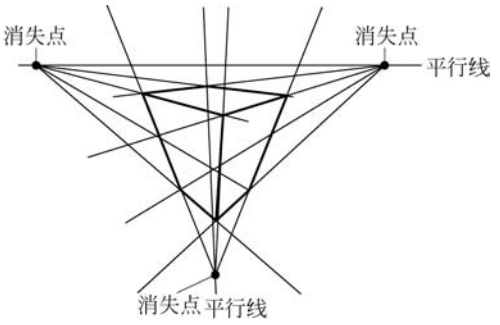


图 3-19 几何透视

随着经济和技术的迅猛发展，工业水平显著提高，道路上的机动车数量也呈爆炸式增长。道路资源分配的不合理导致道路堵塞，同时，现有的道路资源也没有得到充分利用。为了解决这一问题，透视变换被用于识别和过滤行人、自行车和摩托车等非机动车，只针对机动车进行检测，以便于合理安排交通资源，减少交通拥堵。

由于摄像机的成像特性及光学原理所产生的透视效果会导致图像不同位置的物体产生透视畸变。最明显的透视畸变就是远处的物体明显缩小，近处的物体明显增大，从而影响了基于面积和像素数量的目标识别过程，导致摄像头远端方向的有效目标可能被误判或者消除，对整体识别的效果产生很大干扰。要消除这方面的影响，就需要对图像进行校正。

真实世界坐标在摄像机平面的成像过程可以用式(3-12)来表示。

$$\boldsymbol{A}(X,Y)=\boldsymbol{H}\boldsymbol{B}(X,Y) \tag{3-12}$$

其中，$\boldsymbol{A}(X,Y)$表示真实世界坐标向量，$\boldsymbol{B}(X,Y)$表示平面坐标向量，$\boldsymbol{H}$ 表示变换矩阵。人们可以通过人工方式选择图像中的 4 个交点作为矫正参考点。这 4 个交点组成的区域的变换即为已知的矩形变换的过程，如式(3-13)所示。

$$\begin{bmatrix} XW \\ YW \\ W \end{bmatrix}=\begin{bmatrix} a & b & c \\ d & e & f \\ g & h & 1 \end{bmatrix}\begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \tag{3-13}$$

其中，a、b、c、d、e、f、g、h 为变换矩阵的参数。由式(3-13)可知

$$W=gx+hy+1 \tag{3-14}$$

将式(3-14)代入式(3-13)可得

$$\begin{cases} X = ax + by + c - 0 \times d + 0 \times e + 0 \times f - Xxg - Xyh \\ Y = 0 \times a + 0 \times b + 0 \times c + xd + ye + f - Yxg - Yyh \end{cases} \tag{3-15}$$

畸形变换前后的一对交点坐标都能够满足式(3-15)。图像畸形变换前后标定的 4 对交点的坐标可得出 4 对方程组，共 8 个方程。由这 8 个方程解出变换矩阵 $\boldsymbol{H}$ 的 8 个参数，即 a、b、c、d、e、f、g、h。根据这些参数即可求得变换后这 4 个交叉点的坐标。

视频画面也有透视变换现象，即较远物体的画面会产生较大的畸变。

3.4.2　图像配准

图像几何变换重要的应用之一是图像配准。图像配准是将两幅相同场景的图像加以对准的过程。图像配准具有广泛的应用场景，例如，同一个场景多幅图像的匹配或叠加。尤其是在医学图像、卫星图像分析和光流等领域，图像配准的应用非常普遍。在这些领域中，图像配准对同一场景的多幅图像进行叠加，以对因摄像机角度、距离、传感器分辨率和其他因素所导致的几何失真进行校正。目前，图像配准还没有找到一种普适的方法，能够应对所有的几乎失真，任何一种图像配准算法都必须考虑图像的成像原理、几何变形、噪声影响、配准精度等。图像配准分类如图 3-20 所示。

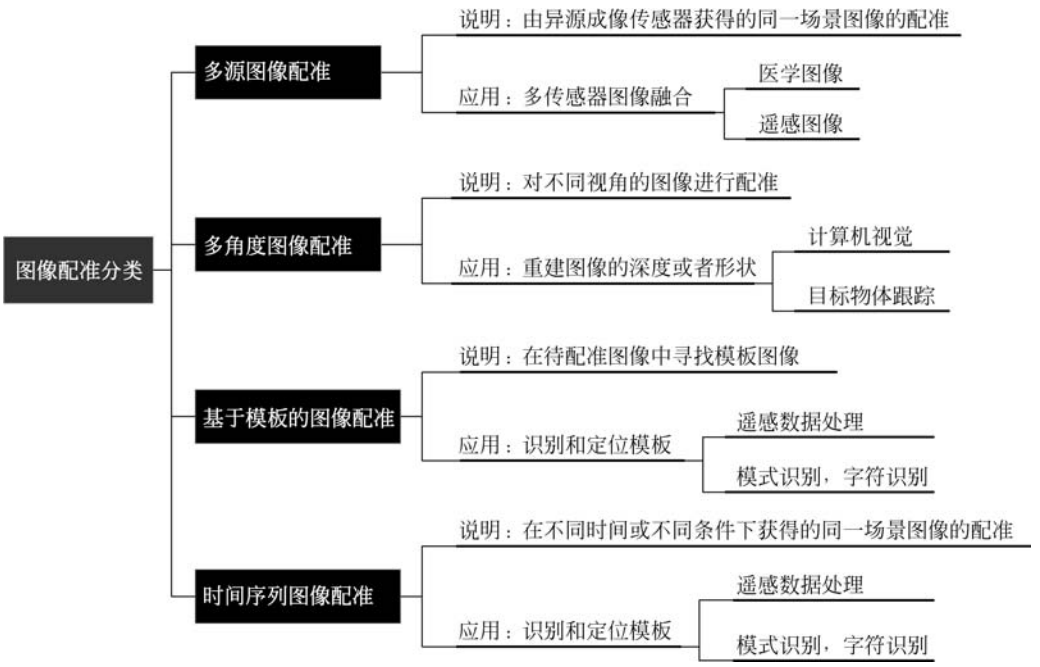


图 3-20　图像配准分类

根据图像配准的原理，图像配准分为特征检测、特征匹配、图像变换模型评估、图像变换这 4 个基本步骤，具体如下。

(1) 特征检测：根据图像畸变的程度，分别采用人工或者自动的方式检测图像的不变特征，例如，边界、轮廓、线交点、交点等均可以作为特征。

(2) 特征匹配：通过特征描述算子及相似性度量来建立所提取的图像特征之间的对应关系。特征匹配常用的信息有区域灰度、特征向量空间分布、特征符号描述等。

(3) 图像变换模型评估：根据待配准图像与参考图像之间几何畸变的情况，选择能最佳拟合两幅图像之间变化的图像变换模型。常见的图像变换模型包括仿射变换、透视变换、多项式变换等，其中，最常用的是仿射变换和多项式变换。

(4) 图像变换：将待配准图像进行相应的参数变换，使它与参考图像处于同一个坐标系。在这个过程中，由于图像变换后的坐标点不一定是整数，因此，需要考虑插值处理。

在这 4 个步骤中，特征匹配是一种通过提取两个或多个图像的特征，对特征进行参数描述，然后运用所描述的参数来进行匹配的算法。图像特征一般有颜色特征、纹理特征、形状特征、空间位置特征等。特征检测算法需要满足 3 个条件：显著性，抗噪性，一致性。显著性是指所提取的特征应该是比较明显的，分布广泛的、易于提取的特征。抗噪性是指所选特征具有较强的噪声抑制能力且对成像条件的变化不敏感。一致性是指该特征能够准确地检测出两幅图像的共有特征。

3.5 图像正交变换的基本概念

任何一种颜色都可以通过 RGB 三原色表示。调整这 3 种颜色的比例，就能混合出各种颜色。但是，如果只调整三原色中的一种来改变颜色并不容易，这就需要将色彩从 RGB 空间转换到图 3-21 所示的 HSV 空间。在 HSV 空间中，颜色变化可以很容易地表示出来。

RGB 空间和时域一样，都有着自身的局限性。虽然 RGB 三原色是最容易理解的色彩表现形式，但不一定是最便于计算的色彩形式。因此，需要进行一种变换，将在原空间难以解决的问题变换到方便计算的空间。以音频为例，对这个问题进行说明。

图 3-22 所示为一段音频的时域波形，其中，横轴是时间；纵轴是振幅，对应的是声音的响度。假设播放器读入这段音频进行音频播放，如果想让音量大一些，只需要将整段音频的振幅同比例扩大。但如果想要让音频的低音部分变得更加厚重，那么要如何处理呢？这个问题在时域上往往难以解决。但是，将音频转换到频域，对人声的低音部分保留或者增强，对人声的高音部分进行衰减，那么，这个问题迎刃而解。

图 3-21 HSV 空间示意

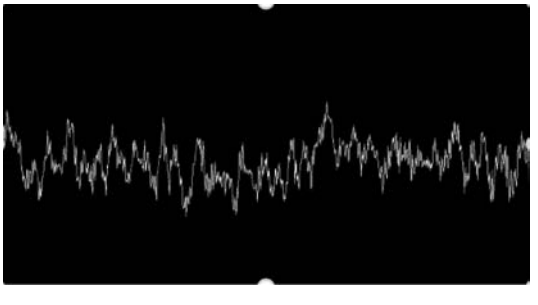


图 3-22 音频的时域波形

通过音频的例子不难发现，不管是时域，还是频域或者是空域，都有其局限性，都存在着自身无法看出的问题，这就好比苏轼的《题西林壁》中“横看成岭侧成峰，远近高低各不同”一般。时域、频域、空域这 3 个看似完全不相关的系统其实只是从不同的角度来看待同一个事

物。由于角度的不同，使原域中难以解决的问题竟然迎刃而解，这也是进行图像正交变换的原因。

人眼所看到的图像是在空域上的，其信息具有很强的相关性，因此，在进行数字图像处理时，人们经常将图像通过某种数学方法变换到其他正交矢量空间。在变换中，原图像一般称为空域图像，变换后的图像称为频域图像。频域图像可反变换为空域图像。

3.5.1　线性操作和非线性操作

在学习正交变换之前，本章首先介绍线性操作和非线性操作。如图 3-23 所示，对于输入 $X(t)$ 和输出 $Y(t)$ 而言，若系统满足齐次性及叠加性，则称系统为线性系统；否则为非线性系统。

图 3-23　系统模型

考虑算子 H，对于给定的输入图像 $f(x,y)$，产生输出图像 $g(x,y)$，如式(3-16)所示。

$$H[f(x,y)]=g(x,y) \tag{3-16}$$

如果 H 满足式(3-17)，则称 H 是线性算子。

$$H[a_i f_i(x,y)+a_j f_j(x,y)]=a_i g_i(x,y)+a_j g_j(x,y) \tag{3-17}$$

其中，a_i、a_j 表示任意常数，$f_i(x,y)$、$f_j(x,y)$ 表示大小相同的图像。

式(3-17)的输出是线性操作，这是因为两个输入相加得到的结果和分别输入得到的结果相同。这体现了线性操作的叠加性。另外，输入乘以常数的线性操作得到的输出与乘以该常数的原输入得到的输出结果相同，这种特性称为同质性。

例 3-1　设 H 为求和算子 Σ，检验其是否为线性的，具体过程如式(3-18)所示。

$$\begin{aligned}\Sigma[a_i f_i(x,y)+a_j f_j(x,y)]&=\Sigma a_i f_i(x,y)+\Sigma a_j f_j(x,y)\\&=a_i\Sigma f_i(x,y)+a_j\Sigma f_j(x,y)\\&=a_i g_i(x,y)+a_j g_j(x,y)\end{aligned} \tag{3-18}$$

由式(3-18)的推导过程可以看出：等号左边的展开式与右边的相等，因此，可以得出求和算子是线性的这一结论。此外，需要注意的是求和算子为阵列求和，并不是图像全部元素的求和。因此，单幅图像的求和是该图像的本身。

例 3-2　H 为求最大值算子 max，检验其是否为线性。假设 $\boldsymbol{f}_1=\begin{bmatrix}0&2\\2&3\end{bmatrix}$，$\boldsymbol{f}_2=\begin{bmatrix}6&5\\4&7\end{bmatrix}$，$a_1=1$，$a_2=-1$，代入式(3-17)，可以得出式(3-19)和式(3-20)。

$$\max\left\{1\times\begin{bmatrix}0&2\\2&3\end{bmatrix}+(-1)\times\begin{bmatrix}6&5\\4&7\end{bmatrix}\right\}=\max\left\{\begin{bmatrix}-6&-3\\-2&-4\end{bmatrix}\right\}=-2 \tag{3-19}$$

$$1\times\max\left\{\begin{bmatrix}0&2\\2&3\end{bmatrix}\right\}+(-1)\times\left\{\max\left\{\begin{bmatrix}6&5\\4&7\end{bmatrix}\right\}\right\}=-4 \tag{3-20}$$

可以看出，式(3-19)和式(3-20)的结果是不相等的，因而，求最大值算子是非线性的。

3.5.2　正交变换

图像由很多个像素构成。这一个个的像素可以看成点冲激函数，那么一幅图像可以看成由很多点冲激函数构成。对图像进行处理，就是对这些像素也就是点冲激函数进行处理。图像除了可以被分为一个个的像素外，还可以分解为基图像之和。基图像之间相互是正交

的，图像变换的本质就是寻找合适的基图像来表达图像。假设 $\boldsymbol{x}$ 是 $N\times 1$ 的列向量，$\boldsymbol{T}$ 是 $N\times N$ 的矩阵，那么，有

$$\boldsymbol{T}(u)=\sum_{x=0}^{N-1}f(x)\boldsymbol{g}(x,u),\quad 0\leqslant u\leqslant N-1 \tag{3-21}$$

其中，$\boldsymbol{T}$ 为 x 的正交变换，$\boldsymbol{g}(x,u)$为正向变换核。有这样一个正变换，与之对应就有一个逆变换，如式(3-22)所示。

$$\boldsymbol{f}(x)=\sum_{u=0}^{N-1}\boldsymbol{T}(u)\boldsymbol{h}(x,u),\quad 0\leqslant x\leqslant N-1 \tag{3-22}$$

其中，$\boldsymbol{h}(x,u)$为反向变换核。

由一维离散正交变换可以推导出二维离散正交变换。二维离散正交变换如式(3-23)所示，即分别在 x 轴方向和 y 轴方向进行求和。二维离散正交变换的逆变换如式(3-24)所示。

$$\boldsymbol{T}(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\boldsymbol{f}(x,y)\boldsymbol{g}(x,y,u,v),\quad 0\leqslant u,v\leqslant N-1 \tag{3-23}$$

$$\boldsymbol{f}(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\boldsymbol{T}(u,v)\boldsymbol{h}(x,y,u,v),\quad 0\leqslant u,v\leqslant N-1 \tag{3-24}$$

其中，$\boldsymbol{f}(x,y)$表示输入图像，$\boldsymbol{T}(u,v)$表示输出图像，$\boldsymbol{g}(x,y,u,v)$表示正向变换核，$\boldsymbol{h}(x,y,u,v)$表示反向变换核。若 $\boldsymbol{f}(x,y)$和 $\boldsymbol{T}(u,v)$大小均为 $N\times N$ 的图像，则正向变换核和反向变换核的大小均为 $N^2\times N^2$。

二维离散正交变换还有如下的性质。

(1) 可分离性。如果二维离散正交变换在 x 轴方向和 y 轴方向的变换是无关的，那么则称正向变换核和反向变换核是可分离的，如式(3-25)和式(3-26)所示。

$$\boldsymbol{g}(x,y,u,v)=\boldsymbol{g}_1(x,u)\boldsymbol{g}_2(y,v) \tag{3-25}$$

$$\boldsymbol{h}(x,y,u,v)=\boldsymbol{h}_1(x,u)\boldsymbol{h}_2(y,v) \tag{3-26}$$

(2) 对称性。若 $\boldsymbol{g}_1$ 与 $\boldsymbol{g}_2$ 的函数形式相同，则称正向变换核是对称的。同样地，若 $\boldsymbol{h}_1$ 与 $\boldsymbol{h}_2$ 的函数形式相同，则称反向变换核是对称的。

(3) 可分离变换。若变换核是可分离的，则称正交变换为可分离变换。具有可分离变换核的二维离散正交变换的计算分为两个步骤，每个步骤用一个一维离散正交变换完成，具体如下。

步骤 1　对 $\boldsymbol{f}(x,y)$的每一行进行一维离散正交变换，得到 $\boldsymbol{T}(x,v)$，如式(3-27)所示。

$$\boldsymbol{T}(x,v)=\sum_{y=0}^{N-1}\boldsymbol{f}(x,y)\boldsymbol{g}_2(y,v),\quad 0\leqslant x,v\leqslant N-1 \tag{3-27}$$

步骤 2　对 $\boldsymbol{T}(x,v)$的每一列进行一维离散正交变换得到 $\boldsymbol{T}(u,v)$，如式(3-28)所示。

$$\boldsymbol{T}(u,v)=\sum_{x=0}^{N-1}\boldsymbol{T}(x,v)\boldsymbol{g}_2(x,u),\quad 0\leqslant u,v\leqslant N-1 \tag{3-28}$$

3.6　正交变换基本方法

满足正交和完备这两个条件的函数集合或矩阵才能用于图像矩阵的分析。图像变换常用的可分离正交变换有：二维离散傅里叶变换，二维离散余弦变换，二维沃尔什变换，以及

小波变换。图像变换常用于图像特征提取、图像压缩、图像增强等。

3.6.1 傅里叶变换

傅里叶变换是线性系统一个有力的分析工具，能够定量地分析如数字图像等的数字化系统。傅里叶变换理论与物理解释的相结合，将有利于解决大多数图像处理问题，其物理意义是将图像的灰度分布函数变换为图像的频率分布函数。实际上，图像进行二维傅里叶变换所得到的频谱图就是图像梯度的分布图。

1. 傅里叶变换概述

从纯粹的数学意义上来看，傅里叶变换将一个图像函数转换为一系列周期函数，将图像的频率分布函数变换为灰度分布函数。傅里叶频谱图上的明暗不一的亮点实际上是表示图像中某一像素与邻域像素差异的强弱，即梯度的大小，也即该点频率的大小。如果频谱图中暗点数量多，那么该图像是比较柔和的。反之，如果频谱图中亮点数量多，那么该图像一定是尖锐的，边界分明且边界两边像素差异较大。

傅里叶变换是在以时间为自变量的信号和以频率为自变量的频谱函数之间的一种变换关系。傅里叶变换提供了一种在全新的频率空间认识信号的方式：一方面可能使在时域中较为复杂的问题在频域中变得简单起来，从而简化分析过程；另一方面使信号与系统的物理本质在频域中能更好地被揭示出来。

当自变量时间或频率的形式(连续、离散)不同时，可以形成多种不同的傅里叶变换对，即信号与频谱的对应关系。傅里叶变换包含连续傅里叶变换、离散傅里叶变换、快速傅里叶变换、短时傅里叶变换等。数字图像处理使用的是二维离散傅里叶变换。

在介绍傅里叶变换之前，首先要介绍狄里赫利条件。狄里赫利条件是指如果信号有有限个间断点、有限个极值点，且绝对可积，那么它的傅里叶变换是存在的。我们所接触的信号，一般来说都满足狄里赫利条件，因此，一般情况下，本章不再强调狄里赫利条件。

2. 傅里叶变换的定义

一维傅里叶变换的函数如式(3-29)所示。

$$F(u)=\int_{-\infty}^{\infty} f(x)\mathrm{e}^{-\mathrm{j}2\pi ux}\,\mathrm{d}x \tag{3-29}$$

式(3-29)的逆变换为

$$f(x)=\int_{-\infty}^{\infty} F(u)\mathrm{e}^{\mathrm{j}2\pi ux}\,\mathrm{d}u \tag{3-30}$$

对于一维傅里叶变换来说，它的变换函数是 $\mathrm{e}^{-\mathrm{j}2\pi ux}$。根据式(3-31)所示的欧拉公式可知，变换函数 $\mathrm{e}^{-\mathrm{j}2\pi ux}=\cos(2\pi ux)-\mathrm{j}\sin 2\pi ux$ 为一个复数，所以，傅里叶变换 $F(u)$在一般情况下是一个复数量，可以写成式(3-32)所示的实部加虚部的形式。

$$\mathrm{e}^{\mathrm{j}\omega}=\cos\omega+\mathrm{j}\sin\omega \tag{3-31}$$

$$F(u)=R(u)+\mathrm{j}I(u)=|F(u)|\,\mathrm{e}^{\mathrm{j}\phi(u)} \tag{3-32}$$

其中，$R(u)$表示实部；$I(u)$表示虚部；$|F(u)|$为 $F(u)$的实部和虚部平方和开根号，如式(3-33)所示，称为 $f(x)$的傅里叶幅度谱；$\phi(u)$如式(3-34)所示，称为 $f(x)$的傅里叶相位谱。

$$|F(u)|=\sqrt{R^2(u)+I^2(u)} \tag{3-33}$$

$$\phi(u)=\arctan\frac{I(u)}{R(u)} \tag{3-34}$$

傅里叶变换同样可以推广到二维函数。如果二维函数满足狄里赫利条件，那么存在式(3-35)所示的二维傅里叶变换对，这个变换对是由一维变换推广得到。

$$\begin{cases} F(u,v)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}f(x,y)\mathrm{e}^{-\mathrm{j}2\pi(ux+vy)}\,\mathrm{d}x\,\mathrm{d}y \\ f(x,y)=\int_{-\infty}^{\infty}\int_{-\infty}^{\infty}F(u,v)\mathrm{e}^{\mathrm{j}2\pi(ux+vy)}\,\mathrm{d}u\,\mathrm{d}v \end{cases} \tag{3-35}$$

二维傅里叶变换的变换函数变成了 $\mathrm{e}^{-\mathrm{j}2\pi(ux+vy)}$，$f(x,y)$在 x 轴方向进行变换的同时，也在 y 轴方向进行相应变换。二维傅里叶变换的逆变换函数相应地变成了 $\mathrm{e}^{\mathrm{j}2\pi(ux+vy)}$。类似于一维傅里叶变换，二维傅里叶变换也存在幅度谱和相位谱，如式(3-36)和式(3-37)所示。

$$|F(u,v)|=\sqrt{R^2(u,v)+I^2(u,v)} \tag{3-36}$$

$$\phi(u,v)=\arctan\frac{I(u,v)}{R(u,v)} \tag{3-37}$$

离散傅里叶变换是一种经典的正弦/余弦型正交变换。它建立了空域与频域之间的联系，具有明确的物理意义，能够更直观、方便地解决许多图像处理问题，被广泛应用于数字图像处理领域。

一维离散傅里叶变换的定义如下：设$\{f(n)|n=0,1,\cdots,N-1\}$为一维信号的 N 个抽样，其离散傅里叶变换及其逆变换如式(3-38)所示。

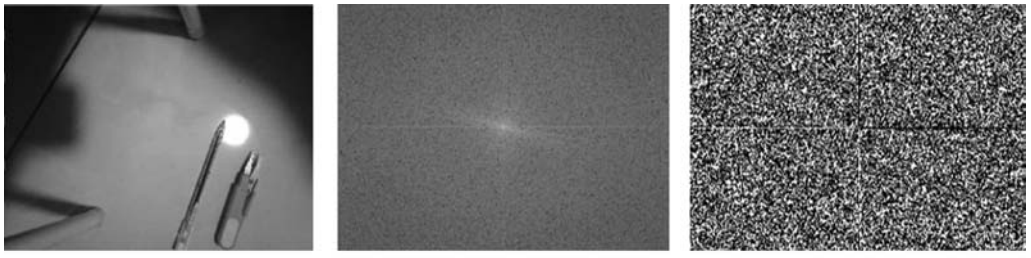
$$\begin{cases} F(u)=\dfrac{1}{N}\sum\limits_{x=0}^{N-1}f(x)\mathrm{e}^{-\mathrm{j}2\pi ux/N}, & u=0,1,2,\cdots,N-1 \\ f(x)=\sum\limits_{u=0}^{N-1}F(u)\mathrm{e}^{\mathrm{j}2\pi ux/N}, & x=0,1,2,\cdots,N-1 \end{cases} \tag{3-38}$$

二维离散傅里叶变换是由图像矩阵向频域矩阵的转换，如式(3-39)所示，其中，$1/N$ 是二维离散傅里叶变换进行归一化处理所添加的系数。对二维离散傅里叶变换来说，它的变换核是 $\mathrm{e}^{-\mathrm{j}2\pi\left(\frac{ux}{N}+\frac{vy}{N}\right)}$。

$$\begin{cases} F(u,v)=\dfrac{1}{N}\sum\limits_{x=0}^{N-1}\sum\limits_{y=0}^{N-1}f(x,y)\mathrm{e}^{-\mathrm{j}2\pi\left(\frac{ux}{N}+\frac{vy}{N}\right)} \\ f(x,y)=\dfrac{1}{N}\sum\limits_{u=0}^{N-1}\sum\limits_{v=0}^{N-1}F(u,v)\mathrm{e}^{\mathrm{j}2\pi\left(\frac{ux}{N}+\frac{vy}{N}\right)} \end{cases} \tag{3-39}$$

其中，$f(x,y)$的频谱为 $F(u,v)$。$|F(u,v)|=\sqrt{R^2(u,v)+I^2(u,v)}$，称为幅度谱。$\phi(u,v)=\arctan\frac{I(u,v)}{R(u,v)}$，称为相位谱。图像经过二维离散傅里叶变换的幅度谱和相位谱如图 3-24 所示。

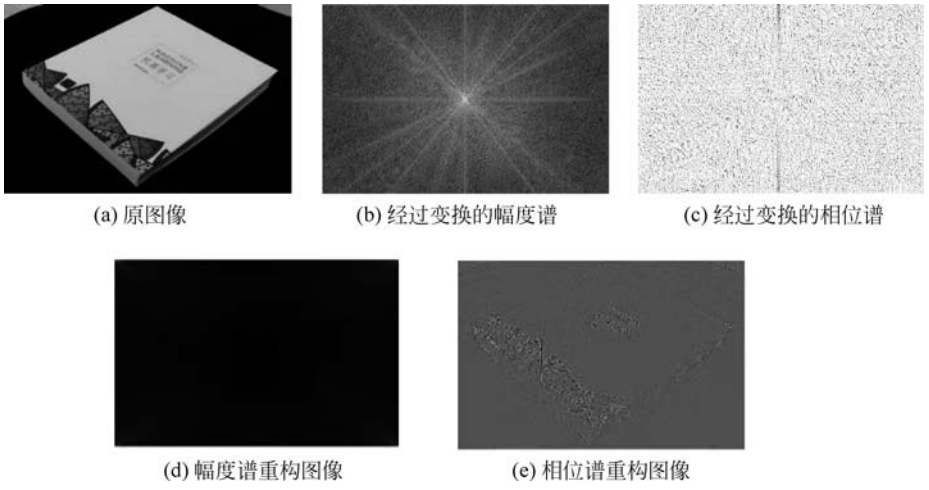
图像的重构如图 3-25 所示。图 3-25(a)所示为原图像，进行二维离散傅里叶变换后得到它的幅度谱和相位谱，如图 3-25(b)和图 3-25(c)所示。如果仅用频域的幅度谱进行傅里叶反变换，也就是图像重构，那么得到图 3-25(d)所示的幅度谱重构图像。可以看出，



(a) 原图像　　(b) 幅度谱　　(c) 相位谱

图 3-24　图像经过二维离散傅里叶变换的幅度谱和相位谱

图 3-25(a)与原图像差异非常大，基本无法看出任何信息。这也意味着幅度谱所包含的信息不足以完全表示原图像的空间信息。同样地，如果仅用相位谱来进行傅里叶反变换，也就是把幅度都置为 1，那么会得到图 3-25(e)所示的相位谱重构图像。



(a) 原图像　　(b) 经过变换的幅度谱　　(c) 经过变换的相位谱

(d) 幅度谱重构图像　　(e) 相位谱重构图像

图 3-25　图像的重构

可以看出，相位谱重构图像比幅度谱重构图像更清晰、更直观，也和原图像相似得多。因此，可以得出结论：相对于幅度谱而言，相位谱具有更重要的应用，或者说它所携带的信息更为重要。

3. 离散傅里叶变换的性质

本节将介绍二维离散傅里叶变换的一些性质，并设 $f(x,y)$ 表示大小为 $M\times N$ 的原图像。

(1) 可分离性。

由可分离性可知，二维傅里叶变换可分解为两步，每一步都是一维离散傅里叶变换。具体步骤如下。

步骤 1　$f(x,y)$ 按行进行一维离散傅里叶变换，得到 $F(x,v)$。

步骤 2　$F(x,v)$ 按列进行一维离散傅里叶变换，便可得到 $f(x,y)$ 的二维离散傅里叶变换结果 $F(u,v)$。

显然，$f(x,y)$ 先按列进行一维离散傅里叶变换，再按行进行一维离散傅里叶变换也是

可行的。具体步骤示意如图 3-26 所示。

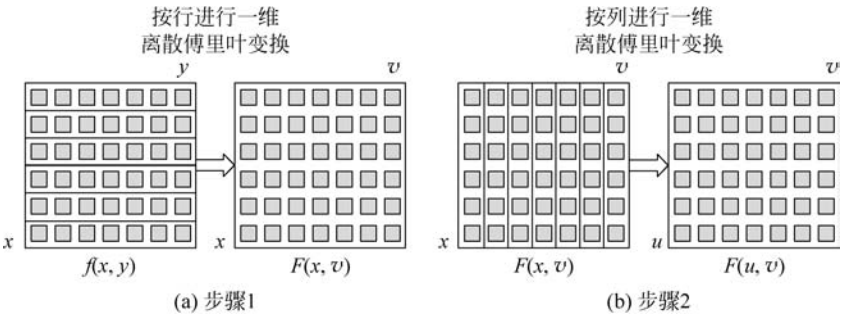


图 3-26　图像二维离散傅里叶变换步骤示意

(2) 周期性。

二维离散傅里叶变换及其反变换在 u 方向和 v 方向是无限周期的，如图 3-26 所示，其周期性如式(3-40)和式(3-41)所示。

$$F(u,v)=F(u+k_1M,v)=F(u,v+k_2N)=F(u+k_1M,v+k_2N) \tag{3-40}$$

$$f(x,y)=f(x+k_1M,y)=f(x,y+k_2N)=f(x+k_1M,y+k_2N) \tag{3-41}$$

其中，k_1 和 k_2 为整数。

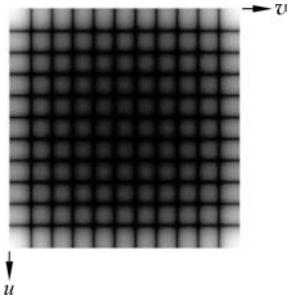


图 3-27　二维离散傅里叶变换的周期性

由图 3-27 可知，尽管对于无穷多个 u 和 v，$F(u,v)$重复出现，但只需要任意周期内的 N 个值，就可以从 $F(u,v)$得到 $f(x,y)$。也就是说，仅一个周期的 u 和 v，就可以完全确定 $F(u,v)$。这一性质对于 $f(x,y)$也同样成立。

(3) 对称性。

任意实函数或虚函数 $w(x,y)$可表示为奇函数和偶函数之和，如式(3-42)所示，其中每一个都可以是实部或虚部的和。

$$w(x,y)=w_e(x,y)+w_o(x,y) \tag{3-42}$$

其中，$w_e(x,y)$为偶数部分，$w_o(x,y)$为奇数部分，它们的定义如式(3-43)和式(3-44)所示。

$$w_e(x,y)\triangleq\frac{w(x,y)+w(-x,-y)}{2} \tag{3-43}$$

$$w_o(x,y)\triangleq\frac{w(x,y)-w(-x,-y)}{2} \tag{3-44}$$

将式(3-43)和式(3-44)代入式(3-42)，可以得出恒等式 $w(x,y)\equiv w(x,y)$，证明了式(3-42)的正确性。由奇偶性可得式(3-45)和式(3-46)。

$$w_e(x,y)=w_e(-x,-y) \tag{3-45}$$

$$w_o(x,y)=-w_o(-x,-y) \tag{3-46}$$

也就是说，偶函数 $w_e(x,y)$是对称函数，奇函数 $w_o(x,y)$是反对称函数。

由高等数学的基本知识可知，两个偶函数或两个奇函数的积是偶函数，一个奇函数和一个偶函数的积是奇函数。除此之外，使得离散函数为奇函数的唯一方法是令所有样本和为

0。由此可以推导出一个重要的结论：对于任意一个离散偶函数 w_e 和一个离散奇函数 w_o，有

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} w_e(x,y)w_o(x,y)=0 \tag{3-47}$$

(4) 叠加性。

傅里叶变换是线性变换，满足线性变换的叠加性，如式(3-48)所示。

$$F[a_1f_1(x,y)+a_2f_2(x,y)]=a_1F[f_1(x,y)]+a_2F[f_2(x,y)] \tag{3-48}$$

(5) 平移性。

图像经过二维离散傅里叶变换得到 $F(u,v)$，如图 3-28 所示。$F(u,v)$和 $f(x,y)$图像大小一致。在图 3-28 所示的幅度谱中，包含原图像的低频分量和直流分量，其中，顶点位置对应直流分量，越靠近中心位置表示原图像的频率越高，在正中心处对应的就是高频分量。

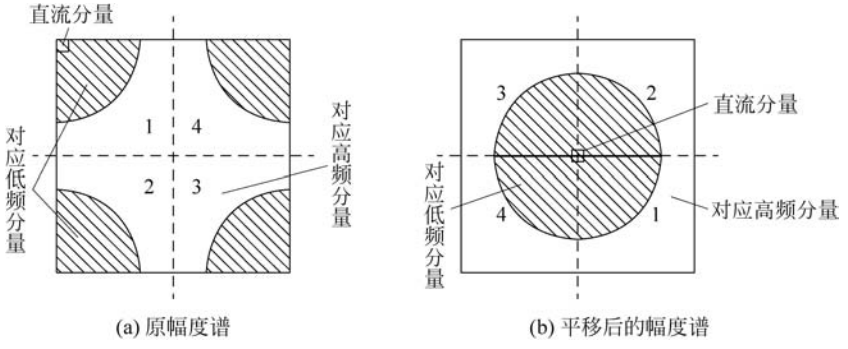


图 3-28　二维离散傅里叶变换后频域的幅度谱

由于图 3-28(a)有一定的对称性，为了显示方便，往往会进行移位，对 1 和 3、2 和 4 所在部分分别进行交叉移位，如图 3-28(b)所示。此时，在经过移位后，图 3-28(b)中心对应的是直流分量。这种平移称为对称平移，也就是对角线交换。

图 3-29(a)为原图像。经过傅里叶变换以后，它的幅度谱如图 3-29(b)所示。可以看出，图 3-29(b)的顶点是直流分量，中间部分是高频分量。图 3-29(b)经过对称平移后，得到图 3-29(c)，其中心位置对应的是直流分量，离直流分量越近的距离上，对应的是低频的分量，即图中亮区域；越远离直流分量的地方，也就是四周对应的是高频的分量，即图中暗区域。

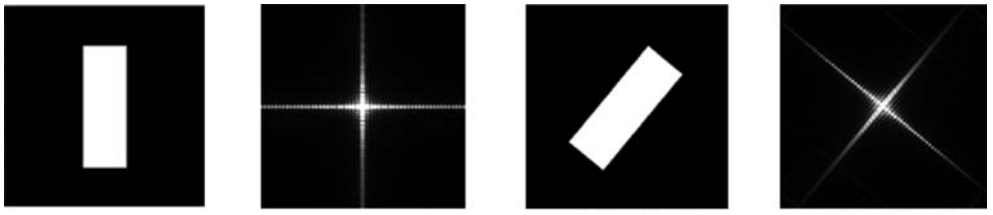
图 3-29　二维离散傅里叶变换平移性示例

(6) 旋转不变性。

引入极坐标，使 $\begin{cases} x=r\cos\theta \\ y=r\sin\theta \end{cases}$，$\begin{cases} u=\omega\cos\varphi \\ v=\omega\sin\varphi \end{cases}$。$f(x,y)$和 $F(u,v)$分别表示为 $f(r,\theta)$和 $F(\omega,\varphi)$，则可以得出(式 3-49)。

$$f(r,\theta+\theta_0)\Leftrightarrow F(\omega,\varphi+\theta_0) \tag{3-49}$$

从平面波的角度来理解二维离散傅里叶变换相对容易些。旋转并没有改变平面波的幅度和相位，只是将平面波旋转了一个角度。原图像、经过旋转后的图像，以及它们的二维离散傅里叶变换频谱图如图 3-30 所示。可以看出，图像在空域进行旋转，其二维离散傅里叶变换在频域也旋转了相同的角度。



(a) 原图像　(b) 频谱图　(c) 旋转后的图像　(d) 旋转后的频谱图

图 3-30　二维离散傅里叶变换旋转不变性

(7) 比例性。

在傅里叶变换中，设标量 a 和 b，有

$$af(x,y)\Leftrightarrow aF(u,v) \tag{3-50}$$

$$f(ax,by)\Leftrightarrow \frac{1}{|ab|}F\left(\frac{u}{a},\frac{v}{b}\right) \tag{3-51}$$

图 3-31(a)为原频谱图，图 3-31(b)为按比例调整后的频谱图，可以看出，图像在空域进行放大，对应于频域则为压缩，其幅度减少为原来幅度的$\frac{1}{|ab|}$。

(a) 原频谱图　(b) 按比例调整后的频谱图

图 3-31　二维离散傅里叶变换比例性示例

(8) 卷积。

一维卷积的定义如式(3-52)所示。

$$f(x)\otimes g(x)=\sum_{\alpha}^{N-1}f(\alpha)g(x-\alpha) \tag{3-52}$$

根据式(3-52)的一维卷积定义可以推断出二维卷积定义，如式(3-53)所示。

$$f(x,y)\otimes g(x,y)=\sum_{\alpha}^{N-1}\sum_{\beta}^{N-1}f(\alpha,\beta)g(x-\alpha,y-\beta) \tag{3-53}$$

其中，$\alpha,\beta=0,1,2,\cdots,N-1$。

二维卷积定理如式(3-54)和式(3-55)所示。

$$f(x,y)\otimes g(x,y)\Leftrightarrow F(u,v)G(u,v) \tag{3-54}$$

$$f(x,y)g(x,y)\Leftrightarrow F(u,v)\otimes G(u,v) \tag{3-55}$$

许多图像变换都是基于卷积运算，这是因为频域的乘积运算比空域的卷积运算快。快速傅里叶变换使这种特点更加明显。

4. 快速傅里叶变换

离散傅里叶变换的计算量大，运算时间长，其运算次数正比于 N^2。当 N 较大时，其运算时间将迅速增长，以致无法容忍。因此，对离散傅里叶变换的快速算法，即快速傅里叶变换(Fast Fourier Transform，FFT)的研究非常有必要。

一种称为逐次加倍法的快速傅里叶变换于 1965 年由 J. W. 库利和 T. W. 图基提出，其运算次数正比于 $N\mathrm{lb}N$。当 N 很大时，这种快速傅里叶变换的计算量可以大大降低。例如，当 $N=1\ 024$ 时，快速傅里叶变换的运算次数和离散傅里叶的运算次数之比为 1/102.4；当 $N=4\ 096$ 时，两者的比值可达 1/341.3。表 3-2 为离散傅里叶变换和快速傅里叶变换运算次数的比较。

表 3-2　离散傅里叶变换和快速傅里叶变换运算次数的比较

N	离散傅里叶变换 运算次数/次	快速傅里叶变换 运算次数/次	两者运算次数 的比值/倍
8	64	24	1/2.7
16	256	64	1/4.0
32	1 024	160	1/6.4
64	4 096	384	1/10.7
128	16 384	896	1/18.3
256	65 536	2 048	1/32.0
512	262 144	4 608	1/56.9
1024	1 048 576	10 240	1/102.4
2048	4 194 304	22 528	1/186.2

一个 n 次多项式可以被 n 个点唯一确定，那么可以把单位根的 0～($n-1$)次幂代入，并以此来确定多项式。设多项式 $A(x)$的系数为($a_0,a_1,a_2,\cdots,a_{n-1}$)，有

$$A(x)=a_0+a_1x+a_2x^2+a_3x^3+\cdots+a_{n-1}x^{n-1} \tag{3-56}$$

将式(3-56)系数的下标按照奇偶进行分类，得到式(3-57)。

$$A(x)=(a_0+a_2x^2+\cdots+a_{n-2}x^{n-2})+(a_1x+a_3x^3+\cdots+a_{n-1}x^{n-1}) \tag{3-57}$$

设有

$$A_1(x)=a_0+a_2x+\cdots+a_{n-2}x^{\frac{n-2}{2}} \tag{3-58}$$

$$A_2(x)=a_1+a_3x+\cdots+a_{n-1}x^{\frac{n-2}{2}} \tag{3-59}$$

将式(3-58)和式(3-59)代入式(3-57)，则可以得到式(3-60)。

$$A(x)=A_1(x^2)+xA_2(x^2) \tag{3-60}$$

将 $\omega_n^k\left(k<\frac{n}{2}\right)$ 代入式(3-60)，得到式(3-61)。

$$A(\omega_n^k)=A_1(\omega_n^{2k})+\omega_n^k A_2(\omega_n^{2k}) \tag{3-61}$$

同理，将 $\omega_n^{k+\frac{n}{2}}$ 代入式(3-60)，可得式(3-62)。

$$A(\omega_n^{k+\frac{n}{2}})=A_1(\omega_n^{2k+n})+\omega_n^{k+\frac{n}{2}}A_2(\omega_n^{2k+n}) \tag{3-62}$$

由式(3-61)和式(3-62)可知，两式中只有一个常数项不同，因此在枚举式(3-61)时，通过 $O(1)$ 可以得到式(3-62)的值，因此，式(3-62)的计算量减少了一半。

3.6.2 二维离散余弦变换

1. 二维离散余弦变换概述

离散余弦变换(Discrete Cosine Transform，DCT)是通过一组频率和幅度不同的余弦函数之和来近似地表示图像的方式，实际上是傅里叶变换的实数部分。离散余弦变换有一个重要的性质，即对于一幅图像而言，其大部分可视化信息都集中在少数的变换系数，因此，离散余弦变换经常用于图像压缩，例如国际压缩标准 JPEG 就采用了离散余弦变换。

离散余弦变换是一种可分离的正交变换，并且是对称的。它和傅里叶变换有着密切的联系，近年来得到了广泛应用，特别是在图像压缩领域。傅里叶变换有一个最大的问题：它的参数为复数。正因为如此，傅里叶变换在数据描述的数据量相当于实数的两倍，导致计算量大。因此，研究者们推出了改进的离散余弦变换。在傅里叶变换过程中，如果被展开的函数是实偶函数，那么其傅里叶变换中只包含余弦项。基于这一特点，人们提出了离散余弦变换。离散余弦变换首先将图像函数变换成偶函数形式，然后对其进行二维离散傅里叶变换，因此离散余弦变换可以看成是一种简化的傅里叶变换。

2. 二维离散余弦变换定义

函数 $f(x)$ 的一维离散余弦变换及其反变换为

$$\begin{cases}C(u)=a(u)\sum\limits_{x=0}^{N-1}f(x)\cos\dfrac{(2x+1)u\pi}{2N}\\ f(x)=\sum\limits_{x=0}^{N-1}a(u)C(u)\cos\dfrac{(2x+1)u\pi}{2N}\end{cases} \tag{3-63}$$

其中，$a(u)$ 为归一化加权系数，其定义如式(3-64)所示。

$$a(u)=\begin{cases}\sqrt{\dfrac{1}{N}}, & u=0\\ \sqrt{\dfrac{2}{N}}, & u=1,2,\cdots,N-1\end{cases} \tag{3-64}$$

一维离散余弦变换扩展到二维离散余弦变换，其变换及反变换为

$$\begin{cases}C(u,v)=a(u)a(v)\sum\limits_{x=0}^{N-1}\sum\limits_{y=0}^{N-1}f(x,y)\cos\left[\dfrac{(2x+1)u\pi}{2N}\right]\cos\left[\dfrac{(2y+1)v\pi}{2N}\right]\\ f(x,y)=\sum\limits_{x=0}^{N-1}\sum\limits_{y=0}^{N-1}a(u)a(v)C(u,v)\cos\left[\dfrac{(2x+1)u\pi}{2N}\right]\cos\left[\dfrac{(2y+1)v\pi}{2N}\right]\end{cases} \tag{3-65}$$

比较式(3-65)的两个式子可以看出，二维离散傅里叶变换是可分离的，如式(3-66)所

示。因此，二维离散余弦变换能够逐次应用一维离散余弦变换进行计算。

$$C(u,v)=a(u)\sum_{x=0}^{N-1}\left\{a(v)\sum_{y=0}^{N-1}f(x,y)\cos\left[\frac{(2y+1)v\pi}{2N}\right]\right\}\cos\left[\frac{(2x+1)u\pi}{2N}\right] \tag{3-66}$$

3. 二维离散余弦变换实现方式

首先，本章介绍一维离散余弦变换实现步骤，如式(3-67)所示。

$$C(u)=a(u)\sum_{x=0}^{N-1}f(x)\cos\frac{(2x+1)u\pi}{2N}=a(u)\mathrm{Re}\left\{\sum_{x=0}^{N-1}f(x)\mathrm{e}^{-\mathrm{j}\frac{(2x+1)u\pi}{2N}}\right\} \tag{3-67}$$

式(3-67)中的 $f(x)$ 可以延拓为

$$f_{\mathrm{e}}(x)=\begin{cases}f(x), & x=0,1,\cdots,N-1\\ 0, & x=N,\cdots,2N-1\end{cases} \tag{3-68}$$

则有

$$C(0)=\sum_{x=0}^{2N-1}f_{\mathrm{e}}(x) \tag{3-69}$$

$$C(u)=a(u)\mathrm{Re}\left\{\mathrm{e}^{-\mathrm{j}\frac{\pi u}{2N}}\sum_{x=0}^{2N-1}f_{\mathrm{e}}(x)\mathrm{e}^{-\mathrm{j}\frac{\pi ux}{N}}\right\} \tag{3-70}$$

二维图像离散余弦变换可以借助傅里叶变换来实现，具体步骤如下。

步骤 1　计算进行二维图像离散余弦变换的宽度和高度，如果不是 2 的整数次幂，则需要先对图像进行调整。

步骤 2　计算在水平和垂直方向上变换时的迭代次数。

步骤 3　用一维离散余弦变换进行水平方向变换，再用一维离散余弦变换进行垂直方向变换。

步骤 4　得到二维离散余弦变换。

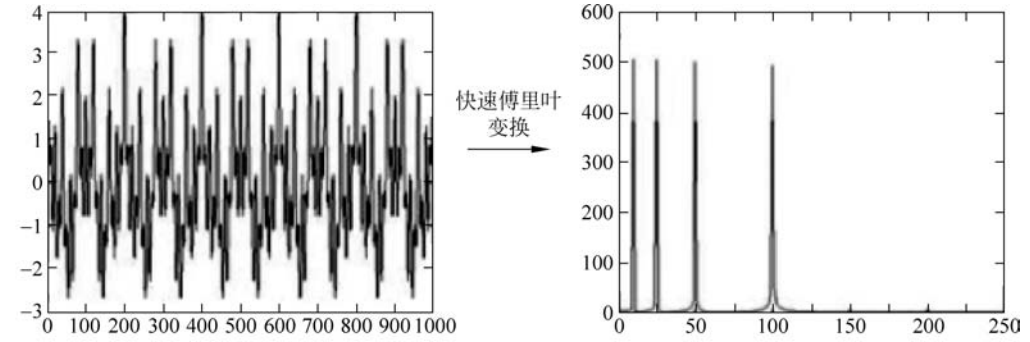
3.6.3　小波变换

在学习小波变换之前，请思考一个问题：既然傅里叶变化可以分析信号的频谱，那么为什么还要提出小波变换？这是因为对于非平稳过程而言，傅里叶变换有它的局限性。

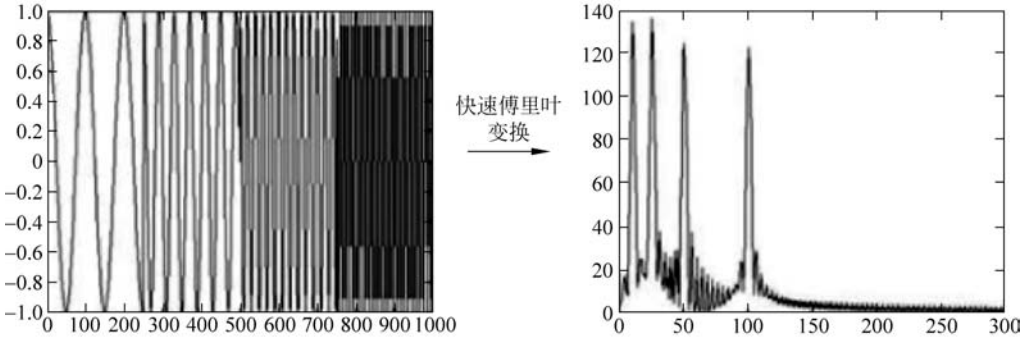
3 组信号及其频谱如图 3-32 所示。图 3-32(a)所示为平稳信号，进行 FFT 后，其频谱有 4 条清晰的线，说明该信号包含 4 个频率成分。

图 3-32(b)和图 3-32(c)是频率随着时间而变化的非平稳信号，它们包含的频率成分和图 3-32(a)相同。进行快速傅里叶变换后，可以发现，这 3 组在时域上有巨大差异的信号，它们的频谱却非常一致。尤其是图 3-32(b)和图 3-32(c)所示的两个非平稳信号，无法从频谱上进行区分，这是因为它们包含的 4 个信号的频率成分是一样的，只是出现的顺序不同。可见，傅里叶变换并不能完美地处理非平稳信号，只能获得一段信号总体包含哪些频率的成分，但是对各成分出现的时刻并不清楚，因而出现时域相差很大的两个信号的频谱一样的情况。然而，平稳信号大多是人为制造出来的，自然界的大量信号几乎是非平稳的。在生物医学信号分析等领域的论文中，基本看不到只采用傅里叶变换进行信号处理的方法。

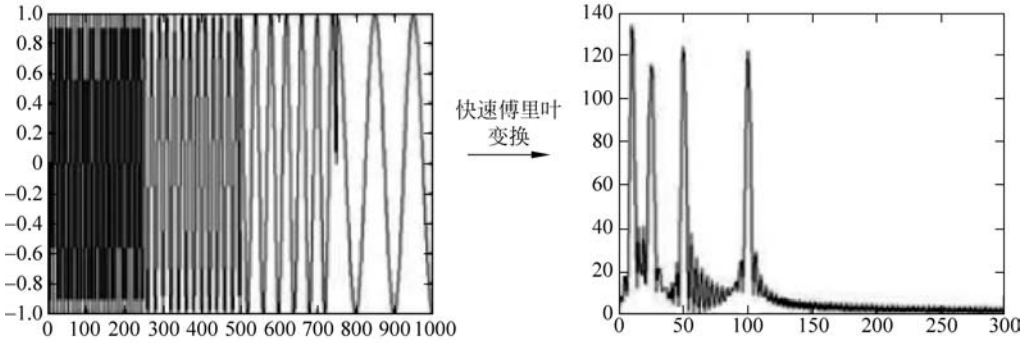
这时候，考虑加窗处理，就是把整个时域过程分解成无数个等长的小过程，每个小过程近似平稳每个小过程进行傅里叶变换，那么就知道在哪个时间点上出现了什么频率。这种方法被称为短时傅里叶变换。但是，短时傅里叶变换依然有不足。当窗太窄时，窗内的信号



(a) 平稳信号及其频谱



(b) 正序非平稳信号及其频谱



(c) 逆序非平稳信号及其频谱

图 3-32　3 组信号及其频谱图对比

太短会导致频率分析不够精准，频率分辨率差。当窗太宽时，时域上又不够精细，时间分辨率低。对于时变的非平稳信号，高频部分适合小窗口，低频部分适合大窗口。然而，短时傅里叶变换的窗口是固定的，在一次短时傅里叶变换中其宽度并不会变化。因此，非平稳信号变化的频率需求还是无法被满足。

与傅里叶变换相比，小波变换是时域信号的局部分析，通过伸缩平移运算对信号逐步进行多尺度细化，最终达到高频部分细分、低频部分细分，且能自动适应地分析时频信号。小波变换可聚焦到信号的任意细节，解决了傅里叶变换存在的问题，成为傅里叶变换以来信号处理方法的重大突破。小波变换的不同之处在于将无限长的三角函数基换成了有限长的会衰减的小波基。

1. 小波变换概述

傅里叶变换可以准确地知道信号中含有哪些频率成分，但是不能知道这些成分发生的时间及位置。小波变换的提出很好地解决了这些问题。小波分析的发展非常迅速，最早可以追溯到1900年希尔伯特的论述，以及1910年哈尔提出的规范正交基。小波变换在图像处理领域被十分重视。面向图像的压缩、特征检测、纹理分析等许多方法，如多分辨率分析、时频域分析、金字塔算法等，都最终归于小波变换的范畴。线性系统的傅里叶变换以在两个方向上都无限伸展的正弦曲线波作为正交基函数。对于如边缘这类图像瞬态信号或高度局部化信号而言，它们的频谱是相当混乱的。

为了克服上述缺陷，使用有限宽度基函数的变换方法逐步发展起来了。这些基函数在频率和位置上是变化的，是有限宽度的波，并被称为小波。基于它们的变换就是小波变换。

2. 小波变换定义

傅里叶变换基函数指的是复指数正弦波，这个波在正负无穷处无限延拓。而小波指的是哈尔小波变换的基函数正负脉冲对，宽度有限。形象地说，宽度有限的波可以称为小波，这种说法并不是十分严谨，只是直观地强调了可以用有限宽度或者两头快速衰减的基函数来进行变换。在分析瞬态信号的时候，小波变换比傅里叶变换更加方便。

$\psi(x)$定义了某个形状，又称为基本小波、母小波或者基小波。设$\psi(x)=L^2(R)$，如果$\psi(x)$的傅里叶变换$\psi(\omega)$满足式(3-71)所示条件，则称$\psi(x)$为基本小波，或者允许小波。其中，$C_\psi<+\infty$被称为允许条件。

$$C_\psi=\int_{-\infty}^{+\infty}\frac{|\psi(\omega)|^2}{|\omega|}\mathrm{d}\omega<+\infty \tag{3-71}$$

基本小波具有波动性、衰减性、带通性这3种性质。波动性指基本小波有正有负，且积分值为0。衰减性指当$|x|$趋于∞，$\psi(x)$趋于0的衰减足够快，即平方可积。带通性指当$\omega=0$和ω趋于∞时，$\psi(\omega)$趋于0。

基本小波经过一系列伸缩及平移变换后，形成基函数族$\{\psi_{a,b}(\omega)\}$。根据基函数族可以得出式(3-72)。

$$\psi_{a,b}(x)=\frac{1}{\sqrt{a}}\psi\left(\frac{x-b}{a}\right) \tag{3-72}$$

其中，$a\in\mathrm{R}^+$，称为尺度因子；$b\in\mathrm{R}$，称为平移因子；a，b均可以连续变化。当$|a|>1$时小波波形变宽，当$|a|<1$时小波波形变窄。因此，$\psi_{a,b}(x)$称为连续小波或者小波函数。因为基本小波$\psi(x)$的均值为0，所以各连续小波函数$\psi_{a,b}(x)$的均值也为0。由于各个小波基函数之间有很强的相关性，所以不适合用于图像压缩。离散小波变换适用于图像压缩，因此，接下来将$\psi_{a,b}(x)$的连续变量a和b进行离散化处理。

令$a=a_0^{-m}$，$b=nb_0a_0^{-m}$，则可以得到离散小波函数，如式(3-73)所示。

$$\psi_{m,n}(x)=a_0^{\frac{m}{2}}\psi(a_0^m x-nb_0) \tag{3-73}$$

其中，a和b为离散值，x为连续值。若x也离散化为i，则式(3-73)将成为离散小波变换，其正变换与反变换如式(3-74)和式(3-75)所示。

$$W(m,n)=\sum_i f(i)\overline{\psi_{m,n}}(i) \tag{3-74}$$

$$f(i)=\sum_{m=-\infty}^{\infty}\sum_{n=-\infty}^{\infty}W(m,n)\psi_{m,n}(i) \tag{3-75}$$

3. 小波变换的实现方式

图像的二维离散小波分解和重构过程如图 3-33 所示。

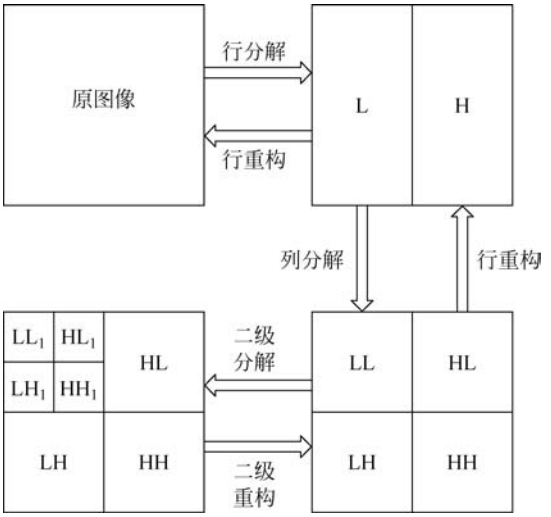


图 3-33　图像的分解与重构

二维离散小波变换的分解过程如下。首先图像逐行进行一维离散小波变换，获得原图像在水平方向的低频分量 L 及高频分量 H；然后，再逐列进行一维离散小波变换，获得原图像在水平和垂直方向都是低频的分量 LL，在水平方向是低频和垂直方向是高频的分量 LH，在水平方向是高频和垂直方向是低频的分量 HL，以及在水平和垂直方向都是高频的分量 HH。数字图像的二级分解是对原图像分量 LL 重复上述操作。

图像被变换成低频区域和高频区域两部分。低频区域表示为 LL，高频区域又分为 HL、LH 和 HH。

图像重构的过程如下。首先变换结果逐列进行一维离散小波逆变换，然后逐行进行一维离散小波逆变换，即可获得重构图像。

由上述过程可以看出，数字图像的小波分解是一个将信号按低频和有向高频进行分离的过程，在分解的过程中，还可以根据需要对分量 LL 进行进一步的小波分解（如图 3-33 所示的分量 LL_1，LH_1，HL_1，HH_1），直至达到要求。

3.7　图像正交变换的应用

3.7.1　基于离散余弦变换与小波变换的图像压缩

众所周知，图像的数据量非常大。为了更有效地传输和存储图像，有必要进行图像压缩。随着现代通信技术的发展，图像的信息种类和数据量越来越大。若不对图像进行压缩，相关技术便难以推广和应用。近几十年来，在视频会议、高清电视、远程医疗等商业应用的推动下，图像和视频的压缩编码技术发展极为迅速。原图像的数据是高度相关的，但存在很

大冗余。大多数图像内相邻像素之间有较大的相关性，这种相关性称为空间冗余。而图像压缩的目的就是消除这些冗余。

图像压缩的方法有许多种。按照原图像是否可以完全恢复的标准，可以分为有损压缩和无损压缩。有损压缩是一种以牺牲部分信息为代价的压缩方法，主要有离散余弦变换编码、差分脉冲预测编码等。

在目前常用的正交变换中，离散余弦变换性能接近最佳。离散余弦变换矩阵与图像内容无关，且由于其构造是对称的数据序列，避免了出现图像边界处的跳跃和不连续现象。又因离弦变换矩阵的基向量是固定的，计算相对简单。加之有快速算法，这使离弦余弦变换在图像压缩方面被广泛应用。

细节较少的图像及其频谱图如图 3-34 所示。由图 3-34(b)和图 3-34(c)可以看出，离散傅里叶变换的数据集中在频谱图的低频区域，离散余弦变换的数据集中在频谱图的左上角。

(a) 原图像　(b) 离散傅里叶变换频谱图　(c) 离散余弦变换频谱图

图 3-34　细节较少的图像及其频谱图

图 3-35 是一幅细节丰富的图像及其频谱图。通过观察不难发现，经过离散傅里叶变换后的数据很发散，离散余弦变换后的数据仍然比较集中。因此可以得出结论：如果从频谱恢复原图像，那么离散余弦变换更合理，这是因为离散余弦变换只需要存储更少的数据点。

(a) 原图像　(b) 离散傅里叶变换频谱图　(c) 离散余弦变换频谱图

图 3-35　细节丰富的图像及其频谱图

JPEG 专家组开发了两种基本的压缩算法，一种是以离散余弦变换为基础的有损压缩算法，另一种是以预测技术为基础的无损压缩算法。当采用有损压缩算法进行图像压缩时，在压缩比为 25∶1 的情况下，压缩被还原得的图像与原图像相比较，非图像处理领域的专家很难找出它们之间的区别。

基于离散余弦变换的图像有损压缩算法的主要过程如图 3-36 所示，具体如下。

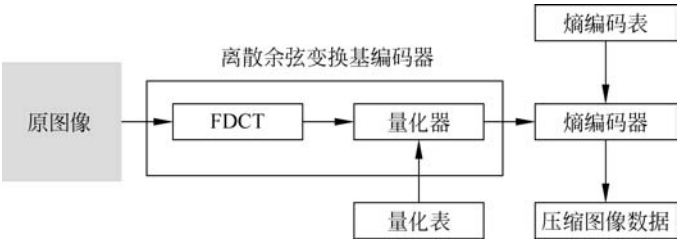


图 3-36　基于离散余弦变换的图像有损压缩算法的主要步骤

(1) 二维离散余弦变换：通常根据可分离性，二维离散余弦变换可通过两次一维离散余弦变换来完成，其算法流程与离散傅里叶变换类似。

(2) 量化：指对经过二维离散余弦变换的频率系数进行量化。量化的目的是减少非零系数的幅度及增加零值系数的数目。

(3) Z 字形编码：Z 字形编码指对量化后的系数进行重新编排。例如，把 8×8 的矩阵变成一个 1×64 的矢量，其中频率较低的系数放在矢量的顶部。

(4) 调制：对直流系数进行编码：使用 DPCM 技术，对相邻图像块之间量化直流系数的差值进行编码。

(5) 编码：使用游程长度编码(Run-Length Encoding，RLE)对交流系数进行编码。量化后交流系数的特点是 1×64 的矢量中包含有许多零系数，并且这些零系数是连续的。因此，可以使用非常简单且直观的 RLC 进行编码。

(6) 熵编码(Entropy Coding，EC)：可以对差分脉冲调制编码后的直流系数以及 RLC 后的交流系数做进一步的压缩。

除了离散余弦变换外，小波变换也被广泛地应用于图像压缩。小波变换将强相关的空间像素矩阵映射成完全不相关且能量分布紧凑的小波系数矩阵，其中，占少数且值较大的小波系数表示图像中主要的能量成分，占多数且值较小的小波系数表示一些不重要的细节分量。通过量化去除小系数所代表的细节分量，用很少的码字来描述大系数所代表的主要能量成分，从而达到高压缩比。

基于小波变换的图像压缩基本步骤如下：首先，用小波变换对图像层进行分解，并提取分解结构中的低频和高频系数；其次，对各频率成分进行重构；再次，对第一层低频信息进行压缩；最后，对第二层低频信息压缩。如图 3-37 为基于小波变换的图像压缩实例。

(a) 原图像　(b) 分解后的低频和高频信息　(c) 第一次压缩后的图像　(d) 第二次压缩后的图像

图 3-37　基于小波变换的图像压缩实例

3.7.2 基于小波变换的图像去噪

随着信息技术的不断发展，图像在人们的生活和工作中显得越来越重要。许多应用都会使用数字图像处理技术，例如，图像去噪。一般来说，现实中的图像都是含有噪声的，这是因为图像在采集、转换和传输过程中，常常受到成像设备和外部环境等影响而产生噪声，使图像的质量有所下降。图像质量下降不仅会影响视觉效果，而且会掩盖图像的部分细节，这对图像后面进行处理非常不利。有时候，噪声甚至会对图像融合、图像分割、特征提取、边缘检测等图像处理造成影响。因此，在尽可能有效去除噪声的同时，又不消除图像的有用信息，这已经成为数字图像处理研究的重要课题。

图像去噪又称之为图像滤波，是属于数字图像处理中的图像复原。图像去噪希望通过采取某种算法，尽可能有效地降低图像中所携带的噪声所产生的影响，使图像的质量得到提升，解决噪声对图像的污染问题，让图像得到最大程度的恢复。

小波去噪在数字图像处理领域已经逐渐发展成为一种经典算法。它是采用小波变换进行图像去噪，也就是说，在知道要处理图像的先验知识的情况下，利用不同变换尺度上信号的小波系数与噪声的小波系数所含有不同特性的原理，当图像信号的能量集中于少数小波系数时，这些系数的值必然大于能量分散的大量的噪声小波系数的值。只要选取适当的阈值，过滤掉绝对值小于阈值的小波系数，那么就可实现图像的去噪。小波去噪的基本流程如图 3-38 所示。

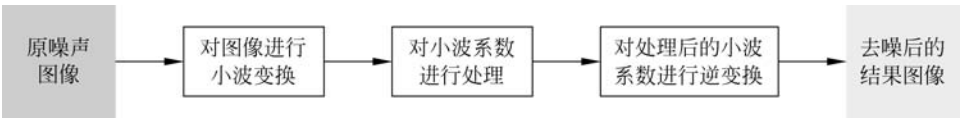


图 3-38 小波去噪的基本流程

小波去噪的步骤如下。首先，对图像进行小波变换：选择一个小波，计算原噪声图像的小波系数；然后，对细节系数通过阈值进行过滤，选择一个细节系数阈值，并对所有细节系数进行阈值化操作；最后，基于阈值化过滤后的细节系数及原始近似系数，使用小波变换进行重建，得到去噪后的结果图像。小波去噪效果实例如图 3-39 所示。

(a) 原图像 (b) 噪声图像 (c) 第一次去噪后的图像 (d) 第二次去噪后的图像

图 3-39 小波去噪效果实例

3.7.3 基于小波变换的数字水印嵌入与提取

随着互联网的普及和数字技术的广泛应用，数字产品的种类变得愈加丰富，且数字产品

的传播方式也更为便捷，随之而来的是抄袭事件频发。在我国相关法律法规不断完善的同时，人们的版权意识也在日渐加强。数字水印技术将数字水印隐藏于数字媒体中，以便发生版权纠纷时为版权所有者提供版权证明。

图 3-40 为数字水印技术的框架。首先，水印生成算法及密钥生成水印信号，然后，原图像与水印信号通过数字水印嵌入算法相结合，从而获得含水印的图像。在进行数字水印提取时，根据相同的密钥及数字水印生成算法将水印信号提取出来。

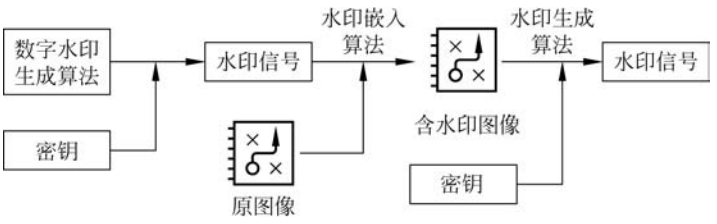


图 3-40　数字水印技术的框架

基于小波变换的数字水印嵌入实例如图 3-41 所示，其中，图 3-41(a)表示原图像，图 3-41(b)表示水印图像，图 3-41(c)表示原图像经过基于小波变换的数字水印嵌入技术的结果图。

(a) 原图像　(b) 水印图像　(c) 结果图

图 3-41　基于小波变换的数字水印嵌入实例

3.7.4　基于小波变换的图像融合

图像融合是数字图像处理技术的一个重要部分，可以利用同一场景的多种传感器协同输出一幅适合人类视觉，或者便于计算机进一步处理与分析的图像。该技术将多源信道所采集的关于同一目标的图像数据经过图像处理和计算机技术等，最大限度地提取各自信道数据的有利信息，合成高质量的图像。图像融合明显改善了单一传感器的不足，提高了结果图像的清晰度及所包含的信息量，有利于更为准确、可靠、全面地获取目标场景的信息。目前，图像融合主要应用于军事国防、遥感、医学图像处理、生物检测等领域。

图像融合可分为 3 个层次：像素级融合、特征级融合、决策级融合。像素级融合是后两者的基础，将原图像中相对应的部分进行融合处理，尽可能多地保留图像信息。像素级融合按照分类方法的不同又可分为 3 种：简单的图像融合法、基于塔形分解的图像融合法、基于小波变换的图像融合法。

与基于塔形分解的图像融合法相比，基于小波变换的图像融合法具有以下优势。

(1) 具有方向性。基于小波变换的图像融合法在提取图像低频信息的同时，还能获得水平、垂直和对角这 3 个方向的高频信息。

(2) 通过合理选择小波，可使离散小波变换在去除噪声的同时能更有效地提取纹理、边

缘等显著信息。

(3) 离散小波变换在不同尺度具有更高的独立性。

(4) 具有快速算法。在小波变换中,快速算法相当于快速傅里叶变换在傅里叶变换中的作用,这为小波变换应用提供了必要的手段。

以两幅图像为例,若对图像进行 N 层小波分解,那么将有$(3N+1)$个不同频带,其中,包含 $3N$ 个高频子图像和 1 个低频子图像。其融合处理的基本步骤如下。

步骤 1　每幅原图像分别进行小波变换,建立图像的小波塔形分解。

步骤 2　对各分解层分别进行融合处理。各分解层上的不同频率分量可采用不同的融合算子进行融合处理,最终得到融合后的小波金字塔。

步骤 3　对融合后的小波金字塔进行小波重构,所得到的重构图像即为融合图像。

在图像融合过程中,小波基的种类和小波分解的层数对融合效果有很大的影响。对特定的图像来说,哪一种小波基的融合效果最好,分解到哪一层最合适,都是需要考虑的问题,因此,可以引入融合效果的评价来构成一个闭环系统,如图 3-42 所示。

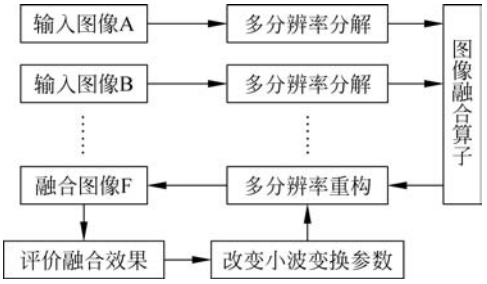


图 3-42　基于效果评价的图像融合原理

目前,基于小波变换的图像融合规则主要分为两种:基于像素的图像融合规则和基于区域特征的图像融合规则。基于像素的图像融合规则主要包括:小波系数的直接替换或追加,最大值选取及平均加权。基于区域特征融合的图像规则主要包括:基于梯度的方法、基于局域方差的方法、基于局域能量的方法。

基于像素的图像融合规则在进行融合处理时,表现出对边缘的高度敏感性,因而,在预处理时要求图像是严格对准的,否则处理结果将不尽人意。这加大了预处理的难度。基于区域的图像融合规则由于考虑了与相邻像素之间的相关性,降低了对边缘的敏感性,因而具有更加广泛的适用性。

3.8　本章小结

本章介绍了两类图像变换的方法,分别是几何变换和频域变换。关于几何变换,首先介绍了包括图像平移变换、图像放缩变换、图像旋转变换、图像镜像变换、图像仿射变换、图像透视变换等基本方法,并将其与实际相结合,解决了包括图像畸变校正及车辆识别等多个问题。关于图像的变换域处理,本章首先介绍了图像的傅里叶变换、离散余弦变换和小波变换的基本原理,并且介绍了相关实例。同时,本章还介绍了傅里叶变换、离散余弦变换、小波变换的定义、性质及其应用。本章通过对图像变换进行全面介绍,以方便读者了

解和掌握相关基础知识。

课后作业

(1) 简述图像的基本位置变换及其 MATLAB 实现。

(2) 简述基本的图像形状变换及其 MATLAB 实现。

(3) 简述直角坐标系中图像旋转的过程,并思考如何解决这个过程中所产生的图像空穴问题。

(4) 分别举例说明使用最近邻插值法和均值插值法进行空穴填充的过程。

(5) 令 $F(221,396)=18$,$F(221,397)=45$,$F(222,396)=27$,$F(222,397)=36$。对 $F(221.3,396.7)$①用最近邻插值法进行插值;②用双线性插值法写出双线性方程及各系数的值,并画出插值示意图。

(6) 假设有一幅从飞机窗后以某一角度拍摄的棉田照片,请对它进行校正,使其看起来像在视线的正下方。一块正方形棉田的四角位于(62,85)、(22,128)、(125,134)、(140,106)。求出校正图像所用的空间变换,画出校正后的棉田。

(7) 简述离散傅里叶变换的定义及其性质。

(8) 证明 $f(x)$的自相关函数的傅里叶变换就是 $f(x)$的功率谱(谱密度)$|f(u)|^2$。

(9) 已知 $N\times N$ 的数字图像为 $f(m,n)$,其离散傅里叶变换为 $F(u,v)$,求 $(-1)^{m+n}f(m+n)$的离散傅里叶变换。

(10) 求下列数字图像块的二维离散小波变换。

① $\boldsymbol{f}_1(m,n)=\begin{bmatrix}1&4&4&1\\1&4&4&1\\1&4&4&1\\1&4&4&1\end{bmatrix}$。

② $\boldsymbol{f}_2(m,n)=\begin{bmatrix}4&4&1&1\\4&4&1&1\\4&4&1&1\\4&4&1&1\end{bmatrix}$。

③ $\boldsymbol{f}_3(m,n)=\begin{bmatrix}4&4&4&4\\4&4&4&4\\4&4&4&4\\4&4&4&4\end{bmatrix}$。

(11) 利用 MATLAB 选择下面一种图像变换方法进行实现:

① 基于图像畸变矫正的测量识别。

② 图像配准处理。

③ 基于离散傅里叶变换的边缘检测。

④ 基于离散余弦变换的盲水印嵌入与提取。

⑤ 基于小波变换的图像压缩。

附录　MATLAB 实现代码

（1）最近邻插值法的 MATLAB 实现代码。

```
%最近邻插值
%输入图像文件及放大率
%输出根据放大率变化后的新图像
function nearest_neighbor = nearest_neighbor(filename,R)
%初始化,读入图像,图像数据为 m*n*color
img = imread(filename);
%变化后图像
[row,col,color] = size(img);                    %获得原图像的行列数及色板数
row = round(row*R);                              %新图像行
col = round(col*R);                              %新图像列

%新图像初始化
%使用 class 获得原图像的数据类型,使新图像数据类型与原图像保持一致
img_new = zeros(row,col,color,class(img));
%对新图像的行、列、色板赋值
for i = 1:row
    for j = 1:col
        for n = 1:color
            x = round(i/R);
            y = round(j/R);
            %为了避免 x 和 y 等于 0 而报错,采取 +1 处理即可
            if x == 0
                x = x+1;
            end
            if y == 0
                y = y+1;
            end
            img_new(i,j,n) = img(x,y,n);
        end
    end
end
```

（2）双线性插值法的 MATLAB 实现代码。

```
%双线性插值
%输入图像文件及放大率
%输出根据放大率变化后的新图像
function bilinear_interpolation = bilinear_interpolation(filename,R)

%初始化,读入图像,图像数据为 m*n*color
img = imread(filename);

%变化后图像
[row,col,color] = size(img);                    %获得原图像的行列数及色板数
```

```
row = round(row * R);                                    % 新图像行
col = round(col * R);                                    % 新图像列
% 新图像初始化
% 使用 class 获得原图像的数据类型,使新图像的数据类型与原图像保持一致
img_new = zeros(row,col,color,class(img));
% 对新图像的行、列、色板赋值
for i = 1:row
    for j = 1:col
        for n = 1:color
            x = round(i/R);
            y = round(j/R);
            if x == 0
                x = x + 1;
            end
            if y == 0
                y = y + 1;
            end
            u = i/R - floor(i/R);                        % 求取水平方向的权重
            v = j/R - floor(j/R);                        % 求取垂直方向的权重
            % 此处需要对图像边缘进行例外处理
            % 本例对图像右边缘及下边缘用最近邻插值计算
            if i >= row - R || j >= col - R
                img_new(i,j,n) = img(x,y,n);
            else
                img_new(i,j,n) = u * v * img(x,y,n) + (1 - u) * v * img(x + 1,y,n) + u * (1 - v)
* img(x,y + 1,n) + (1 - u) * (1 - v) * img(x + 1,y + 1,n);
            end
        end
    end
end
```

(3) 双立方插值法的 MATLAB 实现代码。

```
I = imread('/Users/apple/Downloads/IMG_3331.JPG');
I = double(I);
[oh,ow,od] = size(I);
zmf = 2;                                    % 放缩因子

% initial target image TI
th = round(oh * zmf);
tw = round(ow * zmf);
TI = zeros(th,tw,od);                       % 预分配内存提高计算速度

% add original image with 2 rows and 2 cols
% expand the border to prevent calculation overflow
a = I(1,:,:); b = I(oh,:,:);
temp_I = [a;a;I;b;b];
c = temp_I(:,1,:); d = temp_I(:,ow,:);
FI = [c,c,temp_I,d,d];
```

```
% fill target image with new pixels
for w = 1:tw
    j = floor(w/zmf) + 2; v = rem(w,zmf)/zmf;
    for h = 1:th
        i = floor(h/zmf) + 2; u = rem(h,zmf)/zmf;
        A = [s(u + 1),s(u),s(u - 1),s(u - 2)];
        C = [s(v + 1);s(v);s(v - 1);s(v - 2)];
        for d = 1:od                          % 图像的 3 个通道
           B = FI(i - 1:i + 2,j - 1:j + 2,d);
           TI(h,w,d) = A * B * C;
        end
    end
end

figure;
imshow(uint8(TI));
toc;
```

(4) 图像平移变换的 MATLAB 实现代码。

```
clc;                                          % 清空命令窗口
I = rgb2gray(imread('DORMITORY.JPG'));        % 读入图片并转换为灰度图
figure,imshow(I);                             % 建立窗口,显示灰度图 I
[r,c] = size(I);                              % 计算灰度图的大小 dst = zeros(r,c);
dx = 50;                                      % 平移的 x 轴方向的距离,这里是竖直方向
dy = 80;                                      % 平移的 y 轴方向的距离,这里是水平方向
tras = [1 0 dx;0 1 dy;0 0 1];                 % 平移变换矩阵
for i = 1:r
    for j = 1:c
        temp = [i;j;1];                       % 灰度图 I 要平移变换的点,这里用矩阵表示
        temp = tras * temp;                   % 矩阵相乘
        x = temp(1,1);                        % 把矩阵 temp 的第一行第一列的元素给 x
        y = temp(2,1);                        % 把矩阵 temp 的第二行第一列的元素给 y
        if(x > = 1&&x < = r)&&(y > = 1&&y < = c)   % 判断所变换后得到的点是否越界
            dst(x,y) = I(i,j);                % 得到平移结果矩阵
        end
    end
end
figure,imshow(uint8(dst));                    % 建立窗口,显示平移后的图像
```

(5) 基于等间隔采样的图像缩小法的 MATLAB 实现代码。

```
clc;clear;
f = rgb2gray(imread('D:/Code/Image/classic.jpg'));
figure,imshow(f);                    % 计算采样间隔
k1 = 0.7;
k2 = 0.6;                            % 缩小的倍数
 % 求出缩小后的图像
[row,col] = size(f);
```

```
for i = 1:row * k1                          % 遍历新画布,将旧画布像素有选择性地填充到新画布
    for j = 1:col * k2
        g(i,j) = f(round(i * (1/k1)),round(j * (1/k2)));
    end
end
figure,imshow(uint8(g));
```

(6) 基于像素放大原理的图像放大法的 MATLAB 实现代码。

```
% function [im] = resize(I,kr,kc)
% I = imread('img\han.jpg');
% [im] = dip(I,0.3,0.5);
function [im] = dip(I,kr,kc)
[m,n,d] = size(I);                          % 得到原图像尺寸
m2 = round(kr * m);
n2 = round(kc * n);                         % 得到新图像尺寸: m2 n2
J = zeros(m2,n2,d);                         % 初始化新图像矩阵 J
a = 1/kr; b = 1/kc;                         % 采样间隔
for i = 1:m2
    for j = 1:n2
        % 计算新图像对应原图像的 x-y 坐标
        x1 = round(a * (i - 1) + 1);        % 起始行
        x2 = round(a * i);                  % 结束行
        % 检查对应坐标边界条件
        if x2 > m
            x2 = m;
        end
        y1 = round(b * (j - 1) + 1);        % 起始行
        y2 = round(b * j);                  % 结束行
        if y2 > n
            y2 = n;
        end
        F = I(x1:x2,y1:y2,:);
        % 彩色图像(对每个颜色通道分别求均值)
        if(d > 1)
            J(i,j,1) = mean(mean(F(:,:,1)));   % mean 函数: 求数组的均值
            J(i,j,2) = mean(mean(F(:,:,2)));
            J(i,j,3) = mean(mean(F(:,:,3)));
        else
            J(i,j,1) = mean(mean(F(:,:,1)));   % 灰度图
        end
    end
end
im = uint8(J);
figure,imshow(I);title('原图像');
figure,imshow(im);title('基于像素放大原理放大后的图像');
end
```

（7）基于双线性插值的图像放大法的 MATLAB 实现代码。

```
%像素放大计算函数 extenRGB()
function Output = extenRGB(A,w,l)

%矩阵 A 分别表示矩阵 R、G、B
[m,n] = size(A);                          %读取 A 的行和列
A = [A;zeros(1,n)];                       %在 A 的最后一行加入两行 0
A = [A zeros(m+1,1)];                     %在 A 的最后一列加入两列 0
ini_u = (m-1)/(w*m-1);
ini_v = (n-1)/(l*n-1);

Output = zeros(w*m,l*n);                  %初始化输出矩阵
for j = 1:l*n;
    z_v = floor((j-1)*ini_v+1);
        v = (j-1)*ini_v+1 - z_v;
    for i = 1:w*m;
        z_u = floor((i-1)*ini_u+1);
            u = (i-1)*ini_u+1 - z_u;
    end
end
```

（8）图像旋转变换的 MATLAB 实现代码。

```
clc;
I = imread('potted-plantsk.jpg');
figure,imshow(I);
title('srcImage');
I1 = imrotate(I,30);                      %旋转 30°
I2 = imrotate(I,30,'crop');               %旋转 30°,并剪切图像,使得到的图像和原图像大小一致
I3 = imrotate(I,30,'bilinear','crop'); %旋转 30°,双线性插值法,并剪切图像,使得到的图像和原
                                          %图像大小一致
figure,imshow(I1);
title('I1');
figure,imshow(I2);
title('I2');
figure,imshow(I3);
title('I3');
```

（9）图像镜像变换的 MATLAB 实现代码。

```
clc;
I = rgb2gray(imread('DORMITORY.JPG'));
figure,imshow(I);
title('原图像');
[r,c] = size(I);
dst = zeros(r,c);                         %建立 r×c 的 0 矩阵,用来存储镜像变换后的矩阵
for i = 1:r
    for j = 1:c
        x = i;                            %镜像变换后 x 值
```

```
        y = c - j + 1;                    % 镜像变换后 y 值
        dst(x,y) = I(i,j);
    end
end
figure,imshow(uint8(dst));
title('镜像变换后的图片');
```

(10) 图像仿射变换的 MATLAB 实现代码。

```
img_x = "./1.png";
img_y = "./2.png";
x = imread(img_x);                                  % 读取图像 x
y = imread(img_y);                                  % 读取图像 y
figure;
subplot(1,2,1); imshow(x);                          % 显示图像 x
subplot(1,2,2); imshow(y);                          % 显示图像 y
set(gcf,"outerposition",get(0,"screensize"));       % 将图像全屏幕显示
[x0,y0] = ginput(3);                                % 通过鼠标单击 3 次,取得 3 个坐标点
[x1,y1] = ginput(3);                                % 通过鼠标单击 3 次,取得目标图中对应
                                                    % 的 3 个坐标点
close all;                                          % 关闭图像显示
in_points = [x0,y0];
out_points = [x1,y1];
tform2 = maketform('affine',in_points,out_points);  % 计算变换矩阵
T = affine2d(tform2.tdata.T);                       % 将变换矩阵转化为仿射变换矩阵
z = imwarp(x,T,'OutputView',imref2d(size(y)));      % 进行仿射变换
img_result = "./3.png";
imwrite(z,img_result);                              % 存储结果
```

(11) 离散傅里叶变换的 MATLAB 实现代码。

```
close all;
clear all;
clc;
I = imread('pepper.bmp');
J = rgb2gray(I);
K_1 = fft2(J);
L_1 = abs(K_1/256);
K_2 = fftshift(K_1);
L_2 = abs(K_2/256);
figure;
subplot(2,2,1);
imshow(I); title('原图像');
subplot(2,2,2);
imshow(J); title('灰度图像');
subplot(2,2,3);
imshow(uint8(L_1)); title('灰度图像的傅里叶频谱');
subplot(2,2,4);
imshow(uint8(L_2)); title('平移后的频谱');
```

（12）离散余弦变换的 MATLAB 实现代码。

```
close all;
clear all;
clc;
I = imread('boats.bmp');
J = dct2(I);
figure('Name','离散余弦变换');
subplot(1,2,1);
imshow(I); title('原图像');
subplot(1,2,2);
imshow(log(abs(J))); title('离散余弦变换系数图像');
```

（13）小波变换的 MATLAB 实现代码。

```
close all;
clear all;
clc;
I = imread('lena.jpg');
I = rgb2gray(I);
I = im2double(I);
[ca1,ch1,cv1,cd1] = dwt2(I,'haar');
[ca2,ch2,cv2,cd2] = dwt2(ca1,'haar');
figure(1);
imshow(I);

figure('Name','载体小波分解')
subplot(1,2,1);
imshow([ca1,ch1;cv1,cd1]);
title('一级小波分解');
subplot(1,2,2);
imshow([ca2,ch2;cv2,cd2]);
title('二级小波分解');
% 或者
subplot(1,2,1);
imagesc([wcodemat(ca1),wcodemat(ch1);wcodemat(cv1),wcodemat(cd1)]);
title('一级小波分解');
subplot(1,2,2);
imagesc([wcodemat(ca2),wcodemat(ch2);wcodemat(cv2),wcodemat(cd2)]);
```

（14）基于小波变换的图像分解与重构 MATLAB 实现代码。

```
clc;clear all;close all;
fs = 180;
N = 2000;
t = (1:N - 1)/fs;
s = 1.2 * sin(2 * pi * t * 20) + 0.5 * cos(2 * pi * t * 60);      % 滤掉 60Hz 的信号
% level = 8; wavename = 'bior2.6';
figure;
subplot(2,1,1); plot(t,s); title('原信号') ; grid on;
```

```
[f,spectrum ] = gan_fft(s,fs,N);
subplot(2,1,2);plot(f,spectrum); title('原信号频谱'); grid on;
[C,L] = wavedec(s,2,'db6');
X = waverec(C,L,'db6');
figure;
subplot(2,1,1);plot(t,X); title('原信号分解后又重构的信号') ; grid on;
[f,spectrum ] = gan_fft(X,fs,N);
subplot(2,1,2);plot(f,spectrum); title('原信号分解后又重构频谱'); grid on;
% 均方误差
MSE = sum((X - s).^2)/length(s);
```

(15) 基于离散余弦变换的图像压缩 MATLAB 实现代码。

```
clear all;close all;clc;
f = imread('F:\matlab\MATLAB 上机操作\图形\13100210.jpg');
f = rgb2gray(f);
f = im2double(f);
T = dctmtx(8);                                  % 产生 8 * 8 离散余弦变换矩阵
subplot(1,2,1),imshow(f,[]);
title('原图像');
B = blkproc(f,[8,8],'P1 * x * P2',T,T');        % T 和 T'是离散余弦变换函数 P1 * x * P2 的参数
mask = [1 1 1 1 0 0 0 0
        1 1 1 0 0 0 0 0
        1 1 0 0 0 0 0 0
        1 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0
        0 0 0 0 0 0 0 0];
B2 = blkproc(B,[8 8],'P1. * x',mask);
f1 = blkproc(B2,[8 8],'P1 * x * P2',T',T);
subplot(1,2,2);
imshow(f1,[]);
title('用离散余弦变换压缩过的图像');
```

(16) 小波去噪的 MATLAB 实现代码。

```
clear all;
load facets;
subplot(2,2,1);image(X);
colormap(map);
xlabel('(a)原图像');
axis square;
% 产生含噪声图像
init = 2055615866;
randn('seed',init);
x = X + 50 * randn(size(X));
subplot(2,2,2);image(x);
colormap(map);
xlabel('(b)含噪声图像');
```

```
axis square;
%下面进行图像的去噪处理
%用小波函数 coif3 对 x 进行 2 层小波分解
[c,s] = wavedec2(x,2,'coif3');
%提取小波分解中第一层的低频图像,即实现了低通滤波去噪
%设置尺度向量
n = [1,2];
%设置阈值向量 p
p = [10.12,23.28];
%对 3 个方向高频系数进行阈值处理
nc = wthcoef2('h',c,s,n,p,'s');
nc = wthcoef2('v',nc,s,n,p,'s');
nc = wthcoef2('d',nc,s,n,p,'s');
%对新的小波分解结构[c,s]进行重构
x1 = waverec2(nc,s,'coif3');
subplot(2,2,3);image(x1);
colormap(map);
xlabel('(c)第一次去噪图像');
axis square;
%对 nc 再次进行滤波去噪
xx = wthcoef2('v',nc,s,n,p,'s');
x2 = waverec2(xx,s,'coif3');
subplot(2,2,4);image(x2);
colormap(map);
xlabel('(d)第二次去噪图像');
```

(17) 基于小波变换的数字水印嵌入与提取的 MATLAB 实现代码。

```
clc;
clear;
close all
%主函数
I = imread('xuxian.jpg');                                    %读取载体图像
I = rgb2gray(I);                                             %转换为灰度图
W = imread('logo.tif');                                      %读取水印图像
W = W(12:91,17:96);                                          %剪裁为长宽相等
figure('Name','载体图像')
imshow(I);
title('载体图像')
figure('Name','水印图像')
imshow(W);
title('水印图像')
ntimes = 23;                                                 %密钥 1, Arnold 置乱次数
rngseed = 59433;                                             %密钥 2,随机数种子
flag = 1;                                                    %是否显示中间图像
[Iw,psnr] = setdwtwatermark(I,W,ntimes,rngseed,flag);        %水印嵌入
[Wg,nc] = getdwtwatermark(Iw,W,ntimes,rngseed,flag);         %水印提取
%嵌入函数
function [Iw,psnr] = setdwtwatermark(I,W,ntimes,rngseed,flag)     %小波水印嵌入
type = class(I);                                             %数据类型
```

```
I = double(I);                                  % 强制类型转换为双精度浮点型数据
W = logical(W);                                 % 强制类型转换为逻辑数据
[mI,nI] = size(I);
[mW,nW] = size(W);
if mW~ = nW                                     % 由于 Arnold 置乱只能对方正图像进行处理
    error('SETDWTWATERMARK:ARNOLD','ARNOLD 置乱要求水印图像长宽必须相等!')
end
% 对载体图像进行小波分解
% 一级哈尔小波分解
% 低频,水平,垂直,对角线
[ca1,ch1,cv1,cd1] = dwt2(I,'haar');
% 二级小波分解
[ca2,ch2,cv2,cd2] = dwt2(ca1,'haar');
if flag
    figure('Name','载体小波分解');
    subplot(1,2,1);
    imagesc([wcodemat(ca1),wcodemat(ch1);wcodemat(cv1),wcodemat(cd1)]);
    title('一级小波分解');
    subplot(1,2,2);
    imagesc([wcodemat(ca2),wcodemat(ch2);wcodemat(cv2),wcodemat(cd2)]);
    title('二级小波分解');
end
% 对水印图像进行预处理
% 初始化置乱数组
Wa = W;
% 对水印进行 Arnold 置乱
H = [1,1;1,2]^ntimes;
for i = 1:nW
    for j = 1:nW
        idx = mod(H * [i - 1;j - 1],nW) + 1;
        Wa(idx(1),idx(2)) = W(i,j);
    end
end

if flag
    figure('Name','水印置乱效果');
    subplot(1,2,1)
    imshow(W);
    title('原水印');
    subplot(1,2,2);
    imshow(Wa);
    title(['置乱水印,变换次数 = ',num2str(ntimes)]);
end
% 小波数字水印的嵌入
% 初始化嵌入水印的 ca2 系数
ca2w = ca2;
% 从 ca2 中随机选择 mW * nW 个系数
rng(rngseed);
% 在执行此程序之前,确保没有生成其他随机数,如果有,可使用 rng('default')或者重新启动 MATLAB
idx = randperm(numel(ca2),numel(Wa));
% 将水印信息嵌入到 ca2 中
```

```
for i = 1:numel(Wa)
    % 二级小波系数
    c = ca2(idx(i));
    z = mod(c,nW);
    % 添加水印信息
    if Wa(i)                                    % 水印对应二进制位 1
        if z < nW/4
            f = c - nW/4 - z;
        else
            f = c + nW * 3/4 - z;
        end
    else                                        % 水印对应二进制位 0
        if z < nW * 3/4
            f = c + nW/4 - z;
        else
            f = c + nW * 5/4 - z;
        end
    end
    % 嵌入水印后的小波系数
    ca2w(idx(i)) = f;
end
% 根据小波系数重构图像
ca1w = idwt2(ca2w,ch2,cv2,cd2,'haar');
Iw = idwt2(ca1w,ch1,cv1,cd1,'haar');
Iw = Iw(1:mI,1:nI);
% 计算水印图像峰值信噪比
mn = numel(I);
Imax = max(I(:));
psnr = 10 * log10(mn * Imax^2/sum((I(:) - Iw(:)).^2));

% 输出嵌入水印图像最后结果
I = cast(I,type);
Iw = cast(Iw,type);
if flag
    figure('Name','嵌入水印的图像');
    subplot(1,2,1);
    imshow(I);
    title('原图像');
    subplot(1,2,2);
    imshow(Iw);
    title(['添加水印,PSNR = ',num2str(psnr)]);
end
% 提取函数
function [Wg,nc] = getdwtwatermark(Iw,W,ntimes,rngseed,flag);          % 小波水印提取
[mW,nW] = size(W);
if mW~ = nW
    error('GETDWTWATERMARK:ARNOLD','ARNOLD 要求水印长宽相等!');
end
Iw = double(Iw);
W = logical(W);
ca1w = dwt2(Iw,'haar');
```

```
ca2w = dwt2(ca1w,'haar');
Wa = W;
rng(rngseed);
idx = randperm(numel(ca2w),numel(Wa));
for i = 1:numel(Wa)
    c = ca2w(idx(i));
    z = mod(c,nW);
    if z < nW/2
        Wa(i) = 0;
    else
        Wa(i) = 1;
    end
end
Wg = Wa;
H = [2 -1; -1 1]^ntimes;
for i = 1:nW
    for j = 1:nW
        idx = mod(H * [i - 1;j - 1],nW) + 1;
        Wg(idx(1),idx(2)) = Wa(i,j);
    end
end
% 提取和原水印相关系数的计算
nc = sum(Wg(:). * W(:))/sqrt(sum(Wg(:).^2))/sqrt(sum(W(:).^2));
if flag
    figure('Name','数字水印提取结果');
    subplot(1,2,1);
    imshow(W);
    title('原水印');
    subplot(1,2,2);
    imshow(Wg);
    title(['提取水印,NC = ',num2str(nc)]);
end
```