第三章 载体信号的时频分析

由于大量的信息隐藏算法是针对变换域和压缩域的,所以数字信号处理将在信息隐藏中反复大量地出现。不了解数字信号处理的基本原理和特点,我们就无法深入理解各个信息隐藏的算法,继而无法在已有算法的基础上有所创新。在计算机图像处理中,图像信号变换是一种为了达到某种目的(通常是从图像中获取某种重要信息)而对图像使用的一种数学技巧,经过变换后的图像将更为方便、容易地处理和操作。图像变换在图像处理中有着非常重要的地位,在图像增强、图像分析、图像复原、图像编码压缩以及特征抽取方面有着广泛的应用。特别是在信息隐藏方面有着非常重要的作用。

从实际操作来看,图像变换就是对原图像函数寻找一个合适的变换核的数学变换,由于这种变换方法是针对图像函数而言的,所以称其为图像变换。从本质上来说,图像变换有着深刻的数学和物理背景。例如,对函数的一次 Fourier 变换(Fourier Transform)反映了函数在系统频谱面上的频谱分布。在频谱上做某些处理(如图像增强或滤波处理),再对图像进行第二次 Fourier 变换(Fourier 反变换),就能改变原函数的某些特征,以达到我们需要的结果。另外,图像经过一定的变换(Fourier 变换、离散余弦变换)后,图像频谱函数的统计特征表明:图像的大部分能量都集中在低、中频段,高频分量很弱,仅仅体现了图像的某些细节。因此,可以通过图像变换来消除图像的高频段,从而达到图像压缩的目的,或者实现诸多隐藏和水印算法。

本章主要介绍在计算机图像处理中常用的也是信息隐藏中主要的几种变换(DFT, FFT, DCT, DWT)以及它们的一、二维变换公式和性质,并应用这些数学变换编程对图像进行具体的变换示例。

3.1 离散 Fourier 变换

3.1.1 DFT 原理

在图像变换中,最基础的变换就是 Fourier 变换,掌握了 Fourier 变换,人们就可以在空域和频域中同时处理问题。对信号进行 Fourier 变换就是求信号的频谱。它能够定量地分析诸如数字化系统、采样点、电子放大器、卷积滤波器、噪音和显示点等的作用。把 Fourier 变换的理论同其物理解释相结合,将有助于解决大多数图像处理

的问题。Fourier 变换分为连续形式的和离散形式的，在计算机上使用的 Fourier 变换通常都是离散形式的，即离散傅氏变换（Discrete Fourier Transform, DFT）。DFT 就是将离散信号的 Fourier 变换再离散化。使用离散 Fourier 变换类型的根本原因有两个：一是 DFT 的输入、输出均为离散形式的，这使得计算机非常容易操作；二是因为计算 DFT 存在快速算法（FFT），因而计算比较方便。结合图像实际，我们简单介绍一下二维 DFT 的定义。

在 M× N 的正方形网格上对函数 f(x, y) 进行采样，可以得到二维离散化后的函数 f(m, n)。定义二维 DFT 和 IDFT 的关系如下：

$$F(p,q)=\sum_{m=0}^{M-1}\sum_{n=0}^{N-1}f(m,n)e^{-j(2\pi/M)pm}e^{-j(2\pi/N)qn}\qquad p=0,1,2,\cdots,M-1$$

$$q=0,1,2,\cdots,N-1 \qquad (3.1)$$

$$f(m,n)=\frac{1}{MN}\sum_{p=0}^{M-1}\sum_{q=0}^{N-1}F(p,q)e^{j(2\pi/M)pm}e^{j(2\pi/N)qn}\qquad m,n=0,1,2,\cdots,M-1$$

$$n=0,1,2,\cdots,N-1 \qquad (3.2)$$

粗略地讲，式（3.1）表明 f(m, n) 可以表示为无穷多个不同频率的复指数幂（正弦的）之和，在频率（p, q）上贡献的幅值和相位由 F(p, q) 给出。F(p, q) 通常称为 f(m, n) 的频域表示。F(p, q) 是复值函数，在 p 和 q 上都是周期性的，且周期为 2π。因为其具有周期性通常只显示 $-\pi\leq p, q\leq\pi$ 的范围。注意 F(0, 0) 是 f(m, n) 的所有值之和，因此，F(0, 0) 通常称为 Fourier 变换的恒定分量或 DC 分量（直流分量）。如果 f(m, n) 是一幅图像，则 F(p, q) 是它的谱，变量 p 对应着 X 轴，q 对应着 Y 轴。有时，我们又将 F(p, q) 称为图像的空间频率。

通常计算一维 DFT 所需要的乘法和加法操作的次数是 N^2 次，因为它把所有的复指数值都存在一张表中，这样的计算量实在太大。而快速 Fourier 算法（FFT）将 DFT 计算式分解，可以将操作降到（$N\log_2(N)$）数量级，尤其当 N 是 2 的幂（即 $N=2^p$，其中 p 是整数）时，计算效率最高，实现起来也最简单。

3.1.2　DFT 应用示例

（1）快速卷积

Fourier 变换的卷积定理指出了 Fourier 变换的一个主要好处：与其在空域中作不直观的、难懂的卷积，不如在频域中作乘法，而且可以达到相同的效果。

根据卷积定理：

$$F[A\times B]=F[A]\cdot F[B]$$

则有

$$A\times B=F^{-1}\{F[A]\cdot F[B]\}$$

下面的程序就是按上述原理实现快速卷积。首先构造两个矩阵 A 和 B，代码如下：

```
>> A = magic( 3) ;  B = ones( 3) ;
>> A( 8, 8)  = 0;  B( 8 , 8)  = 0 ;
>> A2  = fft2( A) ;
>> B2  = fft2( B) ;
>> M =  A2 *  B2;
>> C = ifft2( M) ;
>> C = C( 1∶5, 1∶5) ;
>> C = real( C) ;
```

以上这段程序的过程如下: 先用 1 ~9 之间的数产生一个 3× 3 方阵 A, 再生成一个全 1 的 3× 3 方阵 B, 用 0 将 A 和 B 分别补成 8× 8 方阵, 对 A 和 B 进行 Fourier 变换, 将变换的结果在频域内进行点乘, 最后将点乘的结果变回空域, 并截取有效数据, 最后结果如图 3. 1 所示。

Array Editor: C

File Edit View Web Window Help

Numeric format: shortG Size: 5 by 5

	1	2	3	4	5
1	8	9	15	7	6
2	11	17	30	19	13
3	15	30	45	30	15
4	7	21	30	23	9
5	4	13	15	11	2

图 3. 1　卷积结果

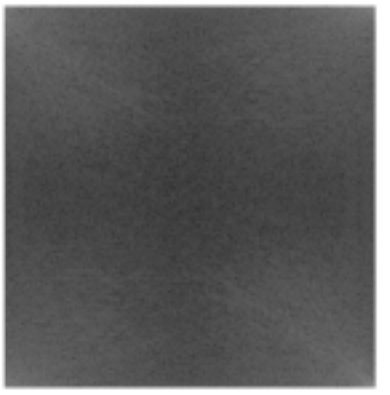
(2) 图像 Fourier 变换实例

一幅图像的 Fourier 变换的结果如图 3. 2 所示。图 3. 2(a) 是原始图像, 图3. 2(b) 是幅值分布图像, 图 3. 2(c) 是将低频直流成分移动到图像中心的幅值图像, 图 3. 2(d) 是相位图像。

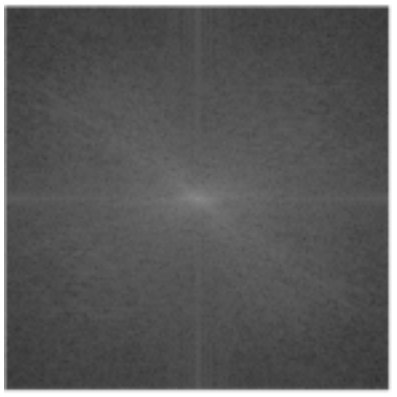
下面结合图 3. 2(b) , 图 3. 2(c) 说明图像 Fourier 变换的幅值分布。图 3. 2(b) 表明变换结果的左上、右上、左下、右下 4 个角的周围对应于低频成分, 中央部位对应于高频成分。为了分析方便, 一般采用如图 3. 3 所示的换位方法使低频直流成分出现在变换结果图像的中央。图 3. 2(c) 就是换位移动后的幅值图像。

但应注意到, 换位后的数组当再进行反变换时得不到原图像。也就是说, 在进行反变换时, 必须使用四角代表低频部分, 中央对应高频部分的变换结果, 这样的二维 Fourier 变换称为光学 Fourier 变换。下面介绍如何利用 MATLAB 实现图像的 Fourier 变换。MATLAB 函数 fft, fft2 和 fftn 分别可以实现一维、二维和 N 维 DFT 快速 Fourier

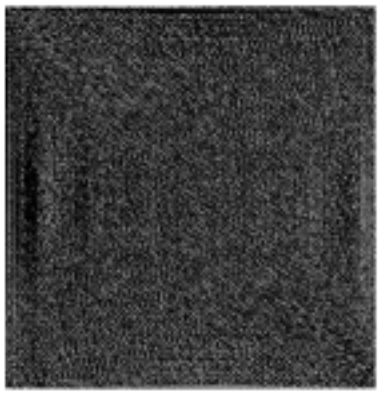
(a)原始图像



(b)幅值分布图像



(c)幅值图像



(d)相位图像

图 3.2　lenna 的 Fourier 变换

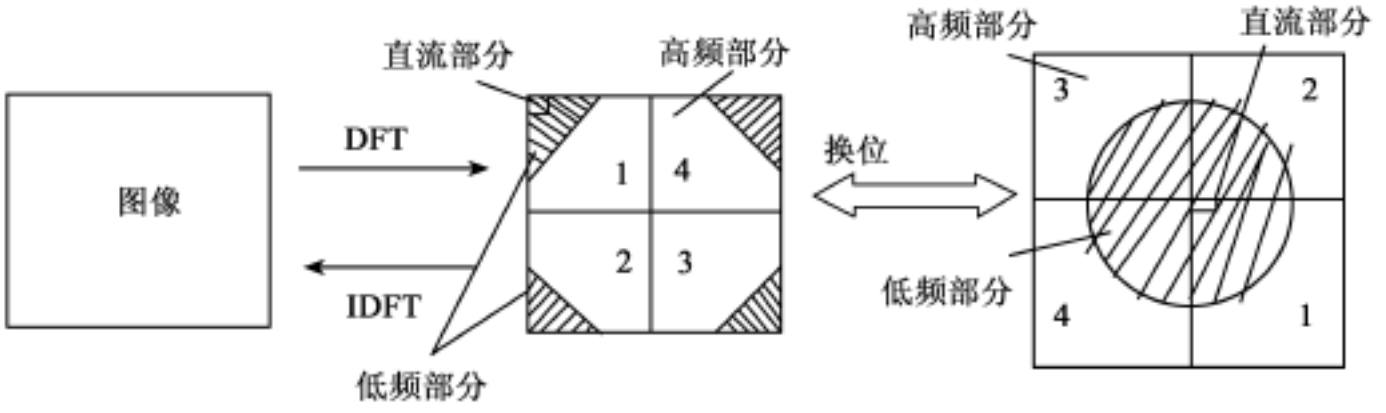


图 3.3　分块换位

变换算法, 而函数 ifft, 和 ifftn 则用来计算反 DFT, 它们是需要进行反变换的图像作为输入参数, 计算得到输出图像的。这些函数的调用格式如下:

A = fft(X, N, DIM)

其中, X 表示输入图像, N 表示采样间隔点。如果 X 小于该数值, 那么 MATLAB 将会对 X 进行填充, 否则将进行截取, 使之长度为 N。DIM 表示要进行离散 Fourier 变换的维数, A 为变换后的返回矩阵。

A = fft2(X, MROWS, NCOLS)

其中, MROWS 和 NCOLS 指定对 X 进行零填充后的 X 大小(可缺省) 。

A = fftn(X, SIZE)

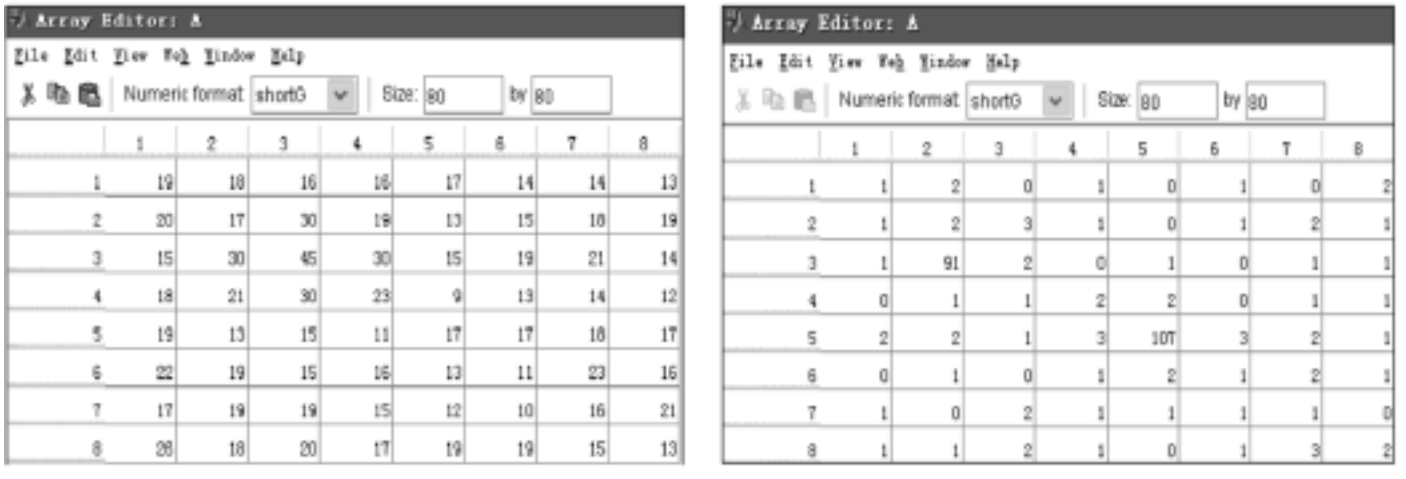
其中, SIZE 是一个向量, 它们每一个元素都将指定 X 相应维进行零填充后的长度。函数 ifft, iftt2 和 ifftn 的调用格式与对应的离散 Fourier 变换函数一致。

利用 MATLAB 实现图 3.2 这个实验的代码如下:

```
>> f = imread('lenna.jpg');
>> F = fft2(f);
>> F1 = log(abs(F));          % 求幅值
>> imshow(F1, [-1 25]);
>> F2 = fftshift(F);          % 如图 3.3 换位
>> F3 = log(abs(F2));         % 求幅值
>> imshow(F3, [-1 25]);
>> F4 = ANGLE(F);             % 求相位
>> imshow(F4, [-1 5]);
```

图 3.2(c), 图 3.2(d) 分别是图像 Fourier 变换的幅值图像和相位图像, 也许大家会认为幅值图像更重要, 因为它至少表现出一些可辨认的结构, 而相位图像看起来则是完全随机的, 但事实却恰恰相反。幅值谱表明了各正弦分量出现的多少, 而相位信息表明了各正弦分量在图像中出现的位置。只要各正弦分量保持原位, 则对于图像整体来说, 幅值就显得不那么重要了。

图 3.4 是一个二维矩阵及其 Fourier 变换得到的幅值矩阵。注意, 低频直流成分已经移到矩阵中心。需要指出的是, 幅值矩阵元素的值只截取了一小段。



(a) 输入矩阵　　(b) 幅值矩阵

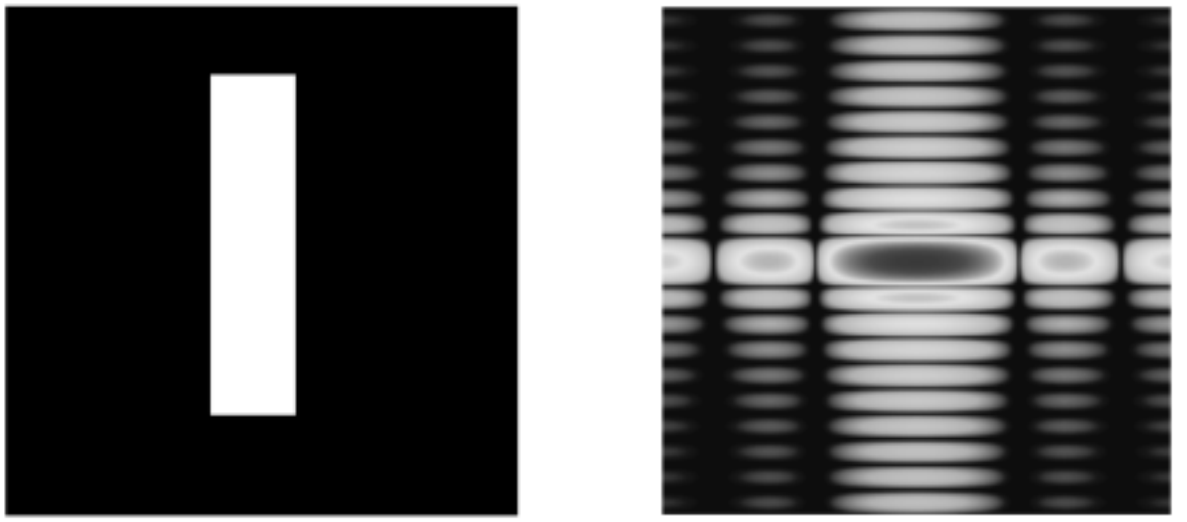
图 3.4

由图 3.4 可以看出, 图像的能量主要集中在低频部分, 高频部分的幅值很小。对大多数无明显颗粒噪声的图像来讲, 低频部分集中了 90% 的能量, 这一事实已成为图像变换压缩方法的理论基础。例如, 变换后仅保留低频分量, 而舍弃高频分量, 反

变换时再将高频分量恢复为零值与低频分量一起进行反变换即可还原图像。下面再举几个具体的例子。

[例 3-1]　在 MATLAB 执行区输入如下代码，得到如图3.5所示的结果。

```
>> f = zeros( 30, 30) ;
>> f( 5: 24, 13: 17)  = 1;
>> imshow( f, 'notruesize') ;
>> F = fft2( f, 256, 256) ; % Fourier 变化
>> F1 = fftshift( F) ;
>> imshow( log( abs( F1) ) , [ -1  5] ) ;
>> colormap( jet)
```



(a) 矩阵 f 的二进制图像　　(b) 矩阵 f 二进制图像的 Fourier 变换结果

图 3.5

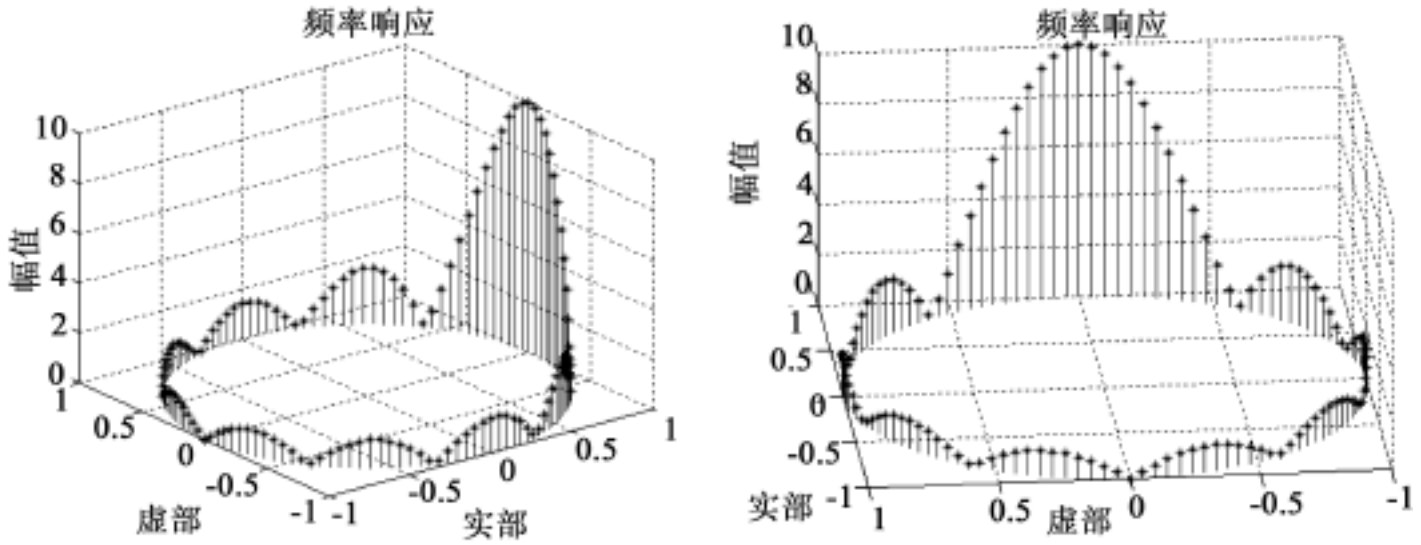
[例 3-2]　在 MATLAB 执行区输入如下代码，利用三维枝干图显示快速 Fourier 变换的计算过程，得到如图 3.6 所示的结果，代码如下：

```
>> th = ( 0: 127) /128*  2*  pi;
>> x = cos( th) ;
>> y = sin( th) ;                          % 计算复平面上单位圆
>> f = ( abs( fft( ones( 10, 1) , 128) ) ) ;    % 计算一步频率响应的幅值
>> stem3( x, y, f, '*') ;                  % 绘制三维枝干图
>> xlabel( '实部') ; ylabel( '虚部') ; zlabel( '幅值') ; title( '频率响应') ;
```

如果需要换个角度查看三维枝干图，则首先执行：rotate3d on。实验结果如图3.6所示。

[例 3-3]　在 MATLAB 执行区输入如下代码，创建一个视频动画，演示快速 Fourier 变换的过程。动画播放过程中的几个画面如图 3.7 所示。

```
>> axis equal;
```



(a)快速 Fourier变换的三维枝干图　　(b)旋转后的三维枝干图

图 3.6

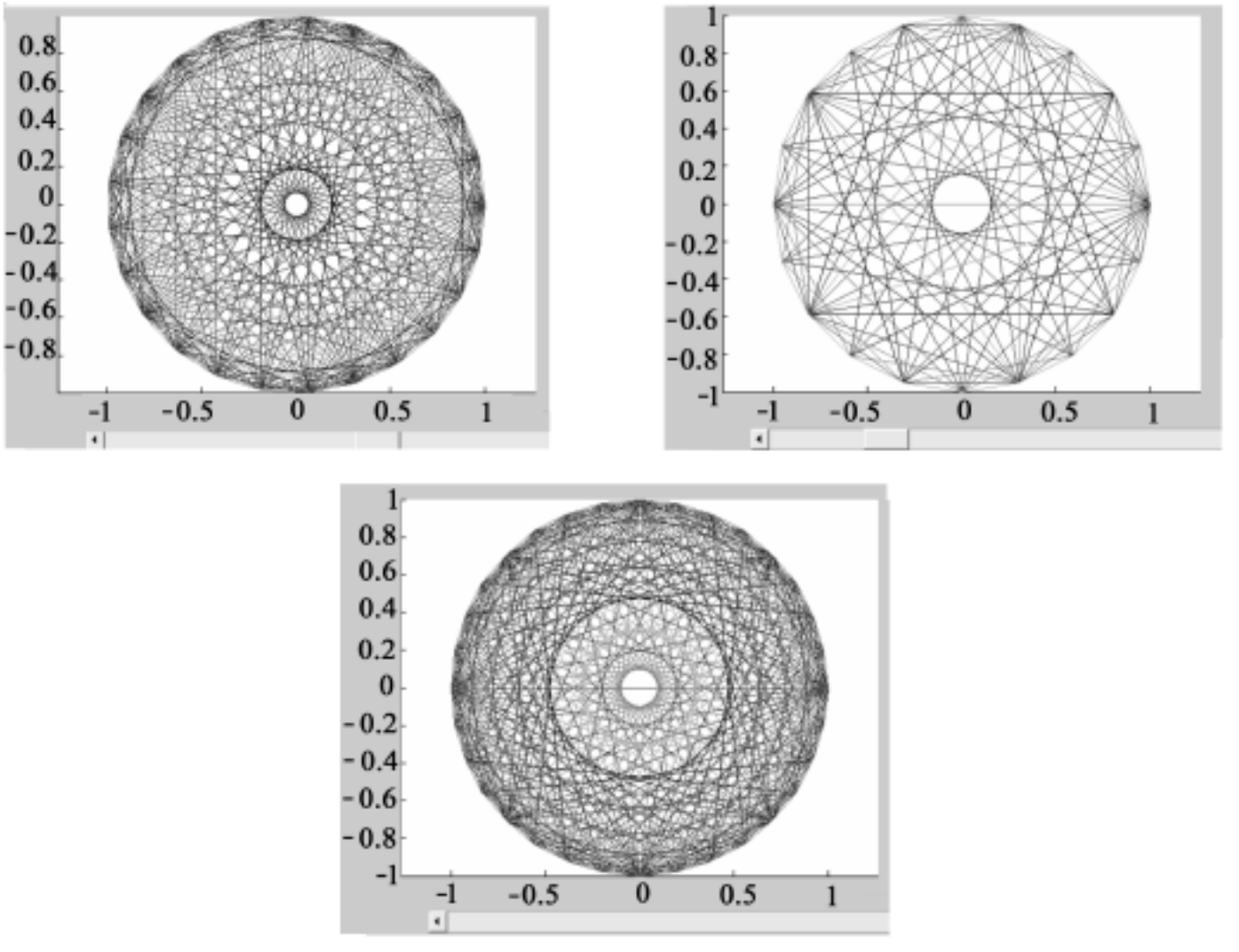


图 3.7　快速 Fourier 变换演示动画截取帧

```
>> M = moviein( 16, gcf) ;
>> set( gca, 'nextplot', 'replacechildren') ;
```

```
>> h = uicontrol( 'style', 'slider', 'position',...
    [100 10 500 20], 'min',1, 'max',16);
>> for j = 1:16
       plot( fft( eye( j +16) ) )
       set( h, 'value', j)
       M(:, j) = getframe( gcf);
>> end
>> clf
>> axes( 'position', [0 0 1 1] );
>> movie( M, 30);
```

3.2　离散余弦变换

3.2.1　DCT 原理

离散余弦变换(Discrete Cosine Transform, DCT) 是一种实数域变换, 其变换核为实数的余弦函数。利用 Fourier 变换的对称性, 采用图像边界褶翻操作将图像变换为偶函数形式, 然后对这样的图像进行二维离散 Fourier 变换, 变换后的结果将仅包含余弦项, 故称为离散余弦变换。对一幅图像进行离散余弦变换, 有这样的性质: 许多有关图像的重要可视信息都集中在 DCT 变换的一小部分系数中。因此, 离散余弦变换(DCT) 是有损图像压缩 JPEG 的核心, 同时也是所谓“ 变换域信息隐藏算法 ”的主要“ 变换域(DCT 域) ”之一。因为图像处理运用二维离散余弦变换, 所以在此我们直接介绍二维 DCT。

一个 M× N 矩阵 A的二维 DCT 定义如下:

$$B_{pq} = a_p a_q \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} A_{mn} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}$$

$$0 \leq p \leq M-1, 0 \leq q \leq N-1$$

其中:

$$a_p = \begin{cases} \frac{1}{\sqrt{M}}, p = 0 \\ \sqrt{\frac{2}{M}}, 1 \leq p \leq M-1 \end{cases}, a_q = \begin{cases} \frac{1}{\sqrt{N}}, q = 0 \\ \sqrt{\frac{2}{N}}, 1 \leq q \leq N-1 \end{cases} \tag{3.3}$$

数值 B_{pq}称为 A的 DCT 系数。

逆 DCT 变换定义如下:

$$A_{mn} = \sum_{p=0}^{M-1} \sum_{q=0}^{N-1} a_p a_q B_{pq} \cos \frac{\pi(2m+1)p}{2M} \cos \frac{\pi(2n+1)q}{2N}$$

$$0 \leq m \leq M-1, 0 \leq n \leq N-1 \tag{3.4}$$

DCT 逆变换方程可理解为: 任意 M× N 的矩阵 A 都可以写成 M× N 个式(3.5)所示函数的加权组合:

$$a_p a_q B_{pq} \cos\frac{\pi(2m+1)p}{2M} \cos\frac{\pi(2n+1)q}{2N}$$

$$0 \leq p \leq M-1, 0 \leq q \leq N-1 \tag{3.5}$$

这些函数被称为 DCT 基本函数。DCT 系数 B_{pq} 可以看成是应用于每一个函数的权值。二维 DCT 变换具有可分离性, 可以分解为双重的一维 DCT 变换, 实现较为方便。

为了方便 DCT 运算的程序实现以及适应将来分块 DCT 的需要, 我们引入一个 DCT 变换矩阵的概念。M× M 变换矩阵 T 由下式给出:

$$T_{pq} = \begin{cases} \dfrac{1}{\sqrt{M}}, & p = 0, 0 \leq q \leq M-1 \\ \sqrt{\dfrac{2}{M}} \cos\dfrac{\pi(2q+1)p}{2M}, & 1 \leq p \leq M-1, 0 \leq q \leq M-1 \end{cases} \tag{3.6}$$

对于一个 M× N 矩阵 A, T× A 是一个 M× N 矩阵, 该矩阵的列包含矩阵 A 列的一维 DCT。A 的二维 DCT 可以通过计算 B = T× A× T 获得。由于 T 是一个实标准正交矩阵, 所以其逆变换的形式与变换形式一致, 因此, B 的二维逆 DCT 由 T′× A× T 给出。这给我们后面的编程带来了极大的方便。正是因为 DCT 可以这样实现, 我们也将 DCT 看做是一个典型的图像正交变换。

最后我们来看一个简单的二维 DCT 的例子。

[例 3-4] 设原始信号为 2× 2 的矩阵 A, $A = \begin{bmatrix} 1 & 2 \\ 3 & 4 \end{bmatrix}$, 根据式(3.3) 对其做 DCT 变换, 得:

$$B(0,0) = a_{p_0} a_{q_0} \sum_{m=0}^{1} \sum_{n=0}^{1} A_{mn} = \frac{1}{2}(1+2+3+4) = 5$$

$$B(0,1) = a_{p_0} a_{q_1} \sum_{m=0}^{1} \sum_{n=0}^{1} A_{mn} \cos\frac{(2n+1)\pi}{4}$$

$$= \frac{1}{\sqrt{2}}\left(A_{00}\cos\frac{\pi}{4} + A_{01}\cos\frac{3\pi}{4} + A_{10}\cos\frac{\pi}{4} + A_{11}\cos\frac{3\pi}{4}\right)$$

$$= -1$$

$$B(1,0) = a_{p_1} a_{q_0} \sum_{m=0}^{1} \sum_{n=0}^{1} A_{mn} \cos\frac{(2m+1)\pi}{4}$$

$$= \frac{1}{\sqrt{2}}\left(A_{00}\cos\frac{\pi}{4} + A_{01}\cos\frac{\pi}{4} + A_{10}\cos\frac{3\pi}{4} + A_{11}\cos\frac{3\pi}{4}\right)$$

$$= -2$$

$$B(1,1) = a_{p_1} a_{q_1} \sum_{m=0}^{1} \sum_{n=0}^{1} A_{mn} \cos \frac{(2m+1)\pi}{4} \cos \frac{(2n+1)\pi}{4}$$

$$= \left(A_{00} \cos \frac{\pi}{4} \cos \frac{\pi}{4} + A_{01} \cos \frac{\pi}{4} \cos \frac{3\pi}{4} + A_{10} \cos \frac{3\pi}{4} \cos \frac{\pi}{4} + A_{11} \cos \frac{3\pi}{4} \cos \frac{3\pi}{4} \right)$$

$$= 0$$

3.2.2　DCT 的 MATLAB 实现

下面我们介绍如何用 MATLAB 实现图像的 DCT 变换。MATLAB 提供了两种实现离散余弦变换的方法，即两种函数。

第一种方法是使用函数 dct2，该函数使用一个基于 FFT 的快速算法来提高当输入较大的输入方阵时的计算速度。dct2 函数的调用格式如下：

```
B = dct2( A, [ M N] )  或
B = dct2( A, M, N)
```

其中，A 表示要变换的图像，M 和 N 是可选参数，表示填充后的图像矩阵大小。B 表示变换后得到的图像矩阵。

第二种方法使用由函数 dctmtx 返回的 DCT 变换矩阵，按照式(3.6)所表明的方法完成图像 DCT。这种方法较适合于较小的输入方阵(如 8× 8 或 16× 16 方阵)以及后面我们通常要使用到的分块 DCT(block DCT)。dctmtx 的调用格式如下：

```
D = dctmtx( N)
```

其中，N 表示 DCT 变换矩阵的维数，D 表示得到的 N 维 DCT 变换矩阵。

下面以 lenna 图像为例，说明各种 DCT 变换函数的使用方法以及变换得到的 DCT 系数的性质。先对图像整体做 DCT 变换，在 MATLAB 中输入如下代码：

```
>> RGB = imread( 'c: \lenna. jpg' ) ;
>> RGB = double( RGB) /255;
>> RGBr = RGB( : , : , 1) ; % 提取 R 层做 DCT
>> DCTmatrix = dct2( RGBr) ;
>> subplot( 1,2,1) , imshow( RGB) ; title( '原始图像' )
>> subplot( 1,2,2) , imshow( log( abs( DCTmatrix) ) , [ ] ) , colormap( jet( 64) ) ;
>> title( '图像 DCT 系数的光谱表示' ) ;
```

由于 dct2 是针对二维矩阵的处理函数，所以我们先将原始的 RGB 图像转换为二维图像，并且由于结果太大，我们只取 R 层的 DCT 系数矩阵进行分析。图 3.8 显示了变化的结果，其中 DCT 系数用光谱的形式给出，直观地表明了低频和高频系数的分布规律。图 3.9 是具体的图像 DCT 系数矩阵。

对照式(3.3)，当 p，q 不断增大时，相应的余弦函数的频率也不断增大，得到的系数可以认为就是原始图像信号在频率不断增大的余弦函数上的投影，所以也被称为低频系数、中频系数和高频系数。这里所谓的频率系数与上一节中的 DFT 频率系

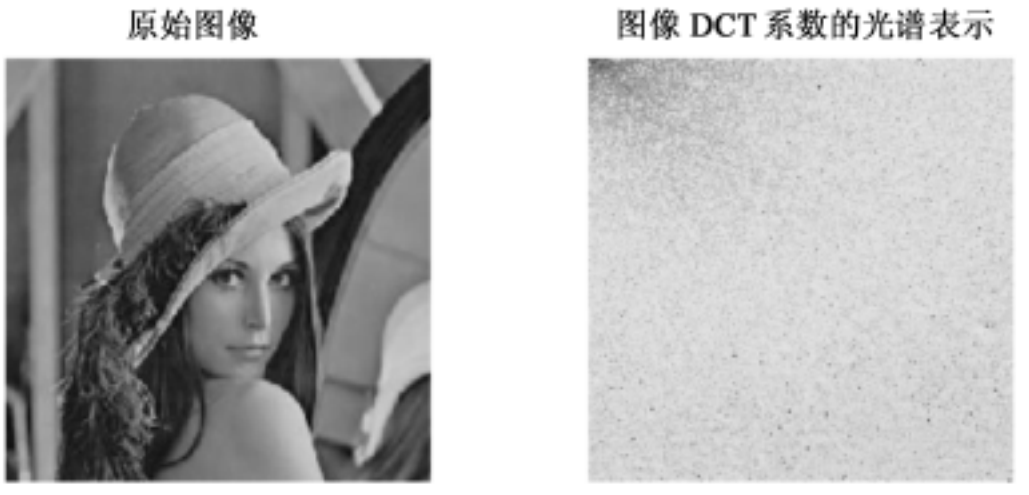


图 3.8　lenna 的 DCT 变换

数是基本一致的概念。观察图 3.9 中的 DCT 系数,可以明显地发现如下规律:大体上,沿左上到右下的方向 DCT 系数(绝对值)是依次递减的。所以,也就是说一个图像的 DCT 低频系数分布在 DCT 系数矩阵的左上角,高频系数分布在右下角,低频系数的绝对值大于高频系数的绝对值。图 3.8 也说明了这一点,在图 3.8 的图像 DCT 系数的光谱表示中,左上角是红色(低频)。

	1	2	3	4	5	6	7	8	9
1	193.86	12.66	13.862	-0.39403	-6.6805	-16.465	-5.2989	16.588	5.3057
2	10.699	9.3912	4.0858	3.0574	7.6258	8.2774	-9.9922	-2.2258	-0.7832
3	-0.99652	5.619	-0.92945	-0.22192	-1.5707	-3.406	-8.7976	2.4626	1.5661
4	-2.8892	1.2426	-0.41263	-0.063076	-4.6199	-1.0057	11.71	-3.9991	0.5996
5	-4.0926	3.4914	1.0029	0.058989	-1.5448	0.42427	3.2226	0.47513	0.60047
6	-0.65119	-0.43604	-0.16724	-0.1471	-0.1551	0.12144	7.6653	2.6967	-1.5574
7	4.3789	0.45406	0.14899	-0.17685	0.020306	-2.7689	3.9294	-1.526	-2.5636
8	5.3237	-0.59396	-1.3978	-0.94533	-0.12871	-3.89	-2.5032	3.1221	-0.72837
9	-0.60137	-0.79918	0.22371	0.09008	0.84826	0.73051	0.35509	-1.128	-0.42559
10	0.73895	-0.63292	-0.22941	-0.34624	0.00048986	-1.1141	0.40106	0.22671	-0.35113
11	-1.6218	-0.18245	-0.18491	0.23316	-0.41314	1.8954	-1.6154	-0.97743	0.43243
12	-2.7778	-0.25082	0.18286	0.014299	-0.58781	1.8701	-1.9647	-1.589	0.075121
13	1.0227	0.81999	-0.027273	-0.18311	0.0013332	0.12829	-0.71932	-0.357	0.20833
14	-0.11799	-0.050877	-0.27769	-0.069928	0.16341	-0.48668	-1.7466	-0.97666	0.44036
15	2.7263	-0.3721	-0.55398	-0.4147	-0.20954	-1.4153	-0.49399	0.49629	-0.070697
16	-0.52134	-0.10154	-0.073403	0.25936	0.27123	1.3156	-0.04928	0.32516	0.53803

图 3.9　DCT 系数矩阵

前面已经说过,对于 DCT 变换来说,图像的主要能量是集中在其 DCT 系数的一小部分。这“一小部分”就是指的低频部分。同样对照式(3.3),随着 p,q 阶数的不断增大,图像信号在两组正交函数上的投影值出现了大量的正负相抵消的情景,从而导

致得到的频率系数在数值(绝对值)上的不断减小。当 p = 0, q = 0 时,得到的频率系数与余弦函数无关(cos0 = 1),完全就是图像抽样信号(像素)的均值,也是最大的一个值,被称为 DCT 变换的直流(DC)系数,其他的频率系数由于都由余弦函数参与得到,所以被称为交流(AC)系数。中、低频系数所含有的原始信号的成分较多,所以由其反变换重构图像就能得到图像的近似部分。高频系数是在众多正交的余弦函数上投影的加权,是这些不同频率的余弦信号一起来刻画原始信号的结果,图像近似的部分在这些函数上都有反映从而被相互抵消了,剩下的就是原始信号的细节部分了。

接下来对 lenna 图像做 8× 8 的分块 DCT。由式(3.6),我们必须先产生一个阶数为 8 的正交 DCT 变换矩阵。利用前面说的 dctmtx 函数,输入 dctmtx(8),得到阶数为 8 的正交 DCT 变换矩阵如图 3.10 所示。

Array Editor: dctmtx8

File Edit View Web Window Help

Numeric format: shortG　Size: 8 by 8

	1	2	3	4	5	6	7	8
1	0.35355	0.35355	0.35355	0.35355	0.35355	0.35355	0.35355	0.35355
2	0.49039	0.41573	0.27779	0.097545	-0.097545	-0.27779	-0.41573	-0.49039
3	0.46194	0.19134	-0.19134	-0.46194	-0.46194	-0.19134	0.19134	0.46194
4	0.41573	-0.097545	-0.49039	-0.27779	0.27779	0.49039	0.097545	-0.41573
5	0.35355	-0.35355	-0.35355	0.35355	0.35355	-0.35355	-0.35355	0.35355
6	0.27779	-0.49039	0.097545	0.41573	-0.41573	-0.097545	0.49039	-0.27779
7	0.19134	-0.46194	0.46194	-0.19134	-0.19134	0.46194	-0.46194	0.19134
8	0.097545	-0.27779	0.41573	-0.49039	0.49039	-0.41573	0.27779	-0.097545

图 3.10　8 阶 DCT 变换矩阵

继而利用 blkproc 函数完成分块操作,blkproc 函数的调用格式如下:

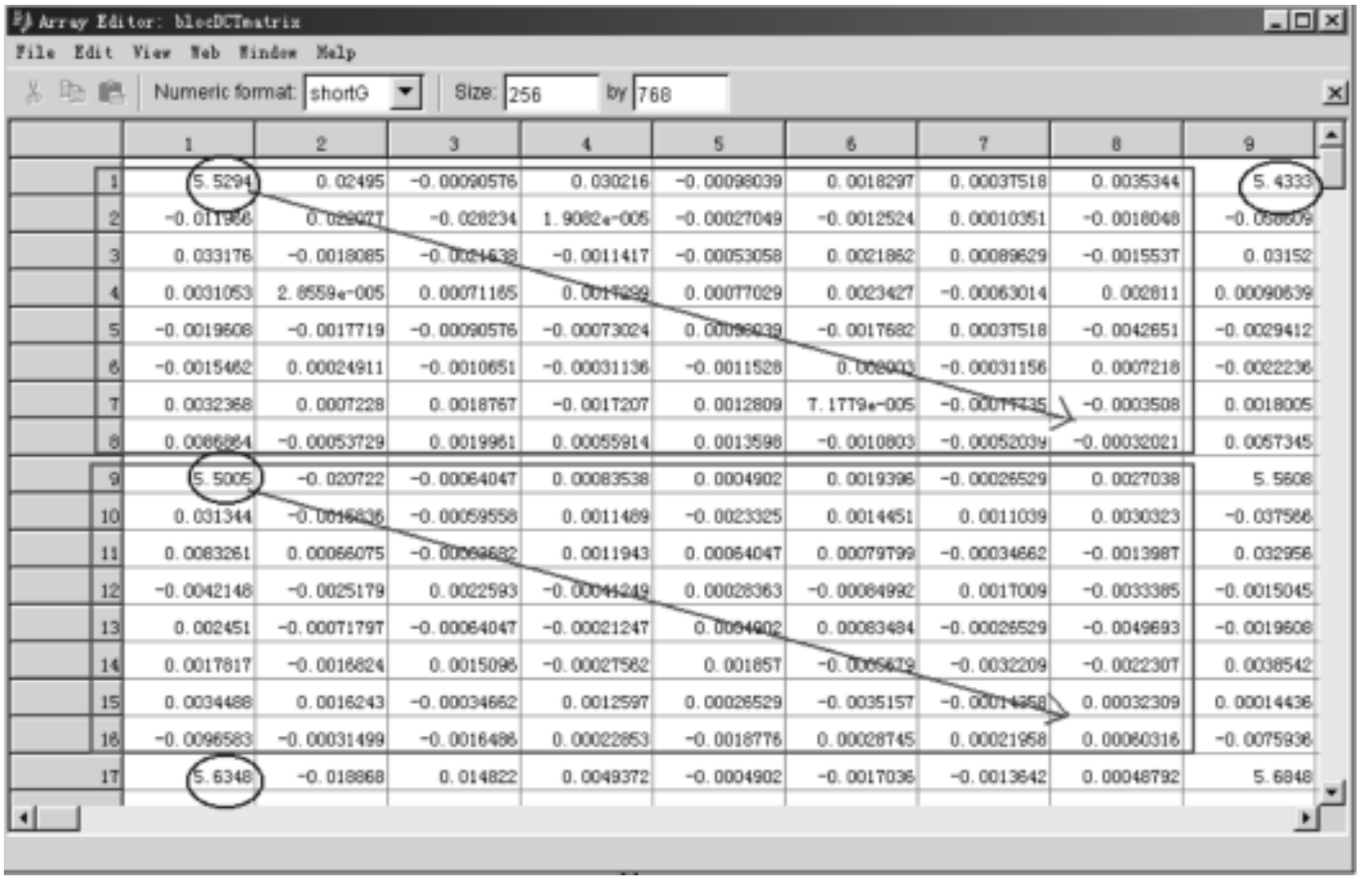
$$B = \text{blkproc}(A, [m\ n], \text{fun}, P_1, P_2, \cdots)$$

其中 A 为原始信号矩阵,[m n] 为分块的大小,fun 为对每一个分块 x 的操作规则,P_i 是 fun 中调用的参数。对图像进行 8× 8 DCT 分块操作的代码如下:

```
>> RGB = imread('lenna.jpg');
>> RGB = double(RGB)/255;
>> RGB = reshape(RGB,256,256*3); % 将三维 RGB 矩阵变为二维
>> T = dctmtx(8);
>> blocDCTmatrix = blkproc(RGB,[8 8],'P1*x*P2',T,T');
```

得到的系数矩阵如图 3.11 所示。

对比图 3.11 和图 3.9 可以发现,对图像进行分块 DCT 后,在每一个 8× 8 范围



	1	2	3	4	5	6	7	8	9
1	5.5294	0.02495	-0.00090576	0.030216	-0.00098039	0.0018297	0.00037518	0.0035344	5.4333
2	-0.011966	0.028027	-0.028234	1.9082e-005	-0.00027049	-0.0012524	0.00010351	-0.0018048	-0.008609
3	0.033176	-0.0018085	-0.0024638	-0.0011417	-0.00053058	0.0021862	0.00089629	-0.001553T	0.03152
4	0.0031053	2.8659e-005	0.00071165	0.0014299	0.00077029	0.0023427	-0.00063014	0.002811	0.00090639
5	-0.0019608	-0.0017719	-0.00090576	-0.00073024	0.00098039	-0.0017682	0.00037518	-0.0042651	-0.0029412
6	-0.0015462	0.00024911	-0.0010651	-0.00031136	-0.0011528	0.000903	-0.00031156	0.0007218	-0.0022236
7	0.0032368	0.0007228	0.0018767	-0.0017207	0.0012809	7.1779e-005	-0.00074435	-0.0003508	0.0018005
8	0.0086864	-0.00053729	0.0019961	0.00055914	0.0013598	-0.0010803	-0.00052039	-0.00032021	0.0057345
9	5.5005	-0.020722	-0.00064047	0.00083538	0.0004902	0.0019396	-0.00026529	0.0027038	5.5608
10	0.031344	-0.0016836	-0.00059558	0.0011489	-0.0023325	0.0014451	0.0011039	0.0030323	-0.037566
11	0.0083261	0.00066075	-0.00094682	0.0011943	0.00064047	0.00079799	-0.00034662	-0.001398T	0.032956
12	-0.0042148	-0.0025179	0.0022593	-0.00041249	0.00028363	-0.00084992	0.0017009	-0.0033385	-0.0015045
13	0.002451	-0.00071797	-0.00064047	-0.00021247	0.0004902	0.00083494	-0.00026529	-0.0049693	-0.0019608
14	0.0017817	-0.0016824	0.0015096	-0.00027562	0.001657	-0.0005479	-0.0032209	-0.002230T	0.0038542
15	0.0034488	0.0016243	-0.00034662	0.0012597	0.00026529	-0.0035157	-0.00014358	0.00032309	0.00014436
16	-0.0096583	-0.00031499	-0.0016486	0.00022853	-0.0018776	0.00028745	0.00021958	0.00060316	-0.0075936
17	5.6348	-0.018868	0.014822	0.0049372	-0.0004902	-0.0017036	-0.0013642	0.00048792	5.6848

图 3.11　8× 8 分块 DCT 系数

内其频率系数仍然符合前面总结的 DCT 系数分布规律。

3.2.3　JPEG 压缩算法中的离散余弦变换(DCT) 编码

在介绍 JPEG 压缩算法之前, 先介绍一下变换编码的基本原理。变换编码就是在频域对图像进行编码。其基本思想是将空域中描述的图像数据经过某种正交变换(诸如 Fourier 变换、离散余弦变换等) 转换到另一个变换域(频率域) 中进行描述, 变换后的结果是产生一批变换系数, 然后对这些变换系数进行编码处理, 从而达到压缩图像数据的目的。变换编码、解码的工作流程图如图 3.12 所示。首先将原始图像分成子块, 每一子块经正交变换、量化、编码后由信道传输到接收端, 接收端作解码、反量化、逆变换, 恢复原图像。

图像数据经过正交变换后, 空域中的总能量在变换域中得到保持, 但像素之间的相关性下降, 能量将会重新分布, 并集中在变换域中少数的变换系数上, 以达到压缩数据的目的。以 Fourier 变换为例, 频谱幅值大的变换系数均集中在低频部分, 几乎包含了图像信息的 90%, 而高频部分的幅值均很小甚至趋于零。因此, 我们完全可以仅对低频部分的变换系数进行量化、编码和传输, 而高频部分则可以舍弃, 从而达到数据压缩的目的。

JPEG 压缩有两种方法, 在此我们只介绍基于 DCT 编码的方法。基于 DCT 的压缩编码算法是有失真的压缩编码, 工作原理如图 3.13 所示。JPEG 系统首先将要压

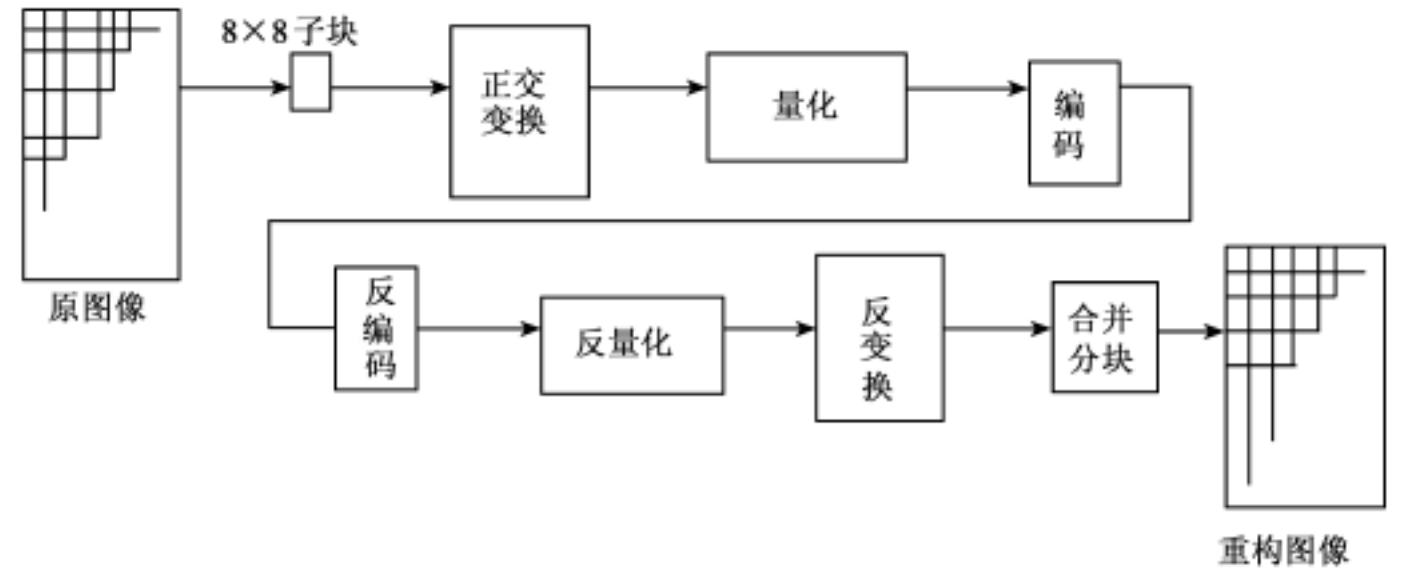


图 3.12　变换编码、解码工作流程图

缩的图像转换为 YCbCr 颜色空间，并把每一个颜色平面分成 8× 8 的像素块。然后，对所有的块进行 DCT 变换。在量化阶段，对所有的 DCT 系数除以一预定义的量化值(如表 3.1 所示) ，并取整到最接近的整数。这个处理的目的是调整图像中不同频率成分的影响，尤其是减小了最高频的 DCT 系数，它们主要是噪声并且不代表图像的主体部分(直流系数的差分编码和交流系数的行程编码) 。最终获得的 DCT 系数通过熵编码器进行压缩编码。在 JPEG 译码时，反量化所有的 DCT 系数(也就是乘以编码阶段中使用的量化值) ，然后执行逆 DCT 变换重构数据。恢复后的图像很接近但不同于原始图像。但是如果适当地设置量化值，得到的图像凭人眼是观察不到差异的。

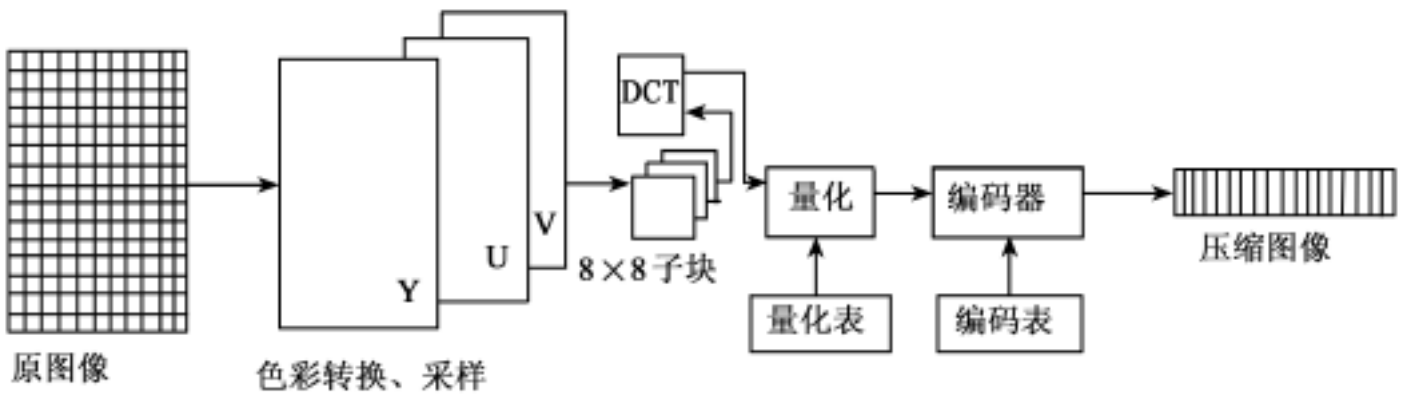


图 3.13　DCT 变换编码原理

下面详细说明 DCT 变换编码的每个主要步骤。

(1) 颜色空间转换和采样

JPEG 压缩只支持 YCbCr 颜色模式，其中 Y 代表亮度，CbCr 代表色度，所以在将彩色图像进行数据压缩之前必须对颜色模式进行转换，将 RGB 模式转为 YCbCr 颜色模式。下面的转换公式我们曾在 1.5.2 节中也给出过。

$$\begin{bmatrix} Y \\ Cb \\ Cr \end{bmatrix} = \begin{bmatrix} 0.299 & 0.587 & 0.114 \\ -0.169 & -0.3316 & -0.50 \\ 0.50 & -0.4186 & -0.0813 \end{bmatrix} \begin{bmatrix} R \\ G \\ B \end{bmatrix}$$

对转换后的数据进行采样，采样比例一般是4∶2∶2 或 4∶1∶1 方式，按 4∶2∶2 采样后的图像，其色度数据比原来减少一半。选择这样的采样方式是因为人的视觉对亮度要比对色度更敏感，而重建后的图像与原图像的差异是人所不易察觉到的。

(2) DCT 变换

在进行 DCT 变换之前，把图像顺序分割成 8× 8 子块。对每一子块，将用 p 位表示的图像数据(一般用 8 位表示一个像素的颜色分量)，即在 $[0, 2^{p-1}]$ 范围内表示的无符号整数，变成 $[-2^{p-1}, 2^{p-1}-1]$ 范围内表示的有符号整数，作为 DCT 变换的输入量。经过 DCT 变换，将空域中表示的图像数据转换到频域中进行，并获得 64 个变换系数。

(3) 量化

为了达到进一步压缩数据的目的，对 DCT 系数进行量化。在 JPEG 中采用线性均匀量化器，并提供了亮度、色差两个量化表，如表 3.1 所示。量化定义为对 64 个变换系数分别除以量化表中所对应的量化分量 Q(u, v)，并四舍五入取整，见式(3.7)。量化的作用是在保证图像质量的前提下，丢掉那些对视觉影响不大的信息。

$$F^q(u, v) = \text{IntegerRound}(F(u, v)/Q(u, v)) \tag{3.7}$$

表 3.1　　亮度与色度量化表

亮度量化表								色差量化表							
16	11	10	16	24	40	51	61	17	18	24	47	99	99	99	99
12	12	14	19	26	58	60	55	18	21	26	66	99	99	99	99
14	13	16	24	40	57	69	56	24	26	56	99	99	99	99	99
14	17	22	29	51	87	80	62	47	66	99	99	99	99	99	99
18	22	37	56	68	109	103	77	99	99	99	99	99	99	99	99
24	35	55	64	81	104	113	92	99	99	99	99	99	99	99	99
49	64	78	87	103	121	120	101	99	99	99	99	99	99	99	99
79	92	95	98	112	100	103	99	99	99	99	99	99	99	99	99

(4) 直流系数 DC 与交流系数 AC 的编码

对 64 个变换系数经过量化后，其中(0,0) 为直流系数 DC，其余的 63 个为交流系数 AC。由于 8× 8 的相邻图像子块之间 DC 系数有很强的相关性，所以对 DC 系数采用差分编码，即对 $DC_i - DC_{i-1}$ 的差值进行编码，如图 3.14 所示。

对 63 个交流系数 AC 则采用行程编码。由于 AC 系数通常会有很多零值，为了增加零的行程长度采用 Z 字形(Zigzag) 行程扫描，如图 3.15 所示。Z 字形行程扫描的另一个重要目的是将低频系数置于高频系数之前。

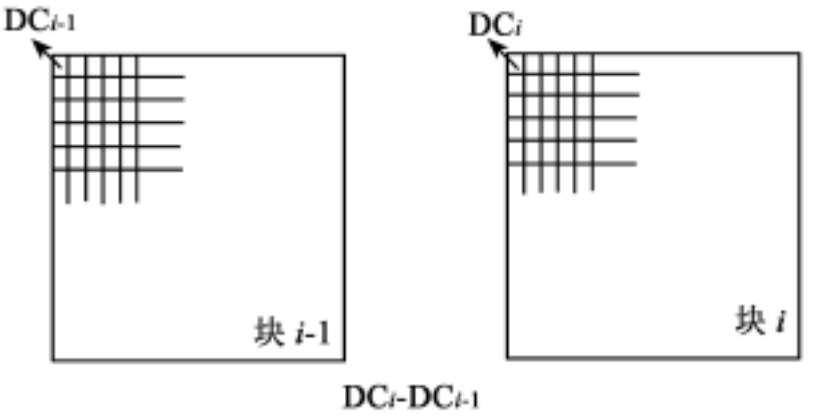


图 3.14　直流系数

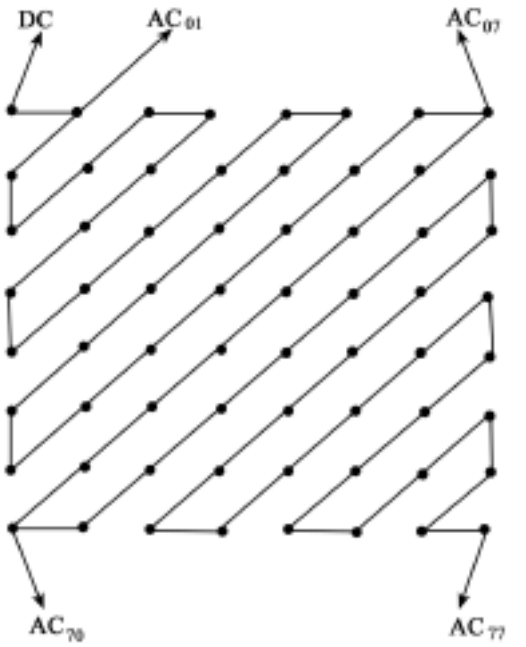


图 3.15　Zigzag 排列

Zigzag 排列是非常重要的一个知识点。在今后的 DCT 域数字水印中我们仍将用到该排列抽取 DCT 系数。下面我们给出一个 8× 8 矩阵的 Zigzag 排列抽取的算法实现方法，供大家参考。

我们知道在算法分析领域有两个很重要的概念：时间复杂度和空间复杂度，二者往往是矛盾的。在信息安全领域，我们更追求的是算法的时间复杂度低而相对不太在意其空间复杂度，于是用空间换时间成了一个主流思想。如在 AES 加密程序中，其 S 变换使用查表的方法得到的程序，其加密效率远远高于其他算法的加密效率。又如在 DES 芯片的大规模集成电路设计中，使用大量重复的流水线语句虽然使得程序的代码量大大增加却很好地提高了芯片的加密效率。这种例子很多，在这里，我们也使用查表的方法完成 Zigzag 排列。

8× 8 的 Zigzag 排列的系数表形式由表 3.2 和表 3.3 给出：

表 3.2　Zigzag 变换表：原始索引表

1, 1	1, 2	1, 3	1, 4	1, 5	1, 6	1, 7	1, 8
2, 1	2, 2	2, 3	2, 4	2, 5	2, 6	2, 7	2, 8
3, 1	3, 2	3, 3	3, 4	3, 5	3, 6	3, 7	3, 8
4, 1	4, 2	4, 3	4, 4	4, 5	4, 6	4, 7	4, 8
5, 1	5, 2	5, 3	5, 4	5, 5	5, 6	5, 7	5, 8
6, 1	6, 2	6, 3	6, 4	6, 5	6, 6	6, 7	6, 8
7, 1	7, 2	7, 3	7, 4	7, 5	7, 6	7, 7	7, 8
8, 1	8, 2	8, 3	8, 4	8, 5	8, 6	8, 7	8, 8

表 3.3-1　Zigzag 变换表: 行变换索引表

1	1	2	3	2	1	1	2
3	4	5	4	3	2	1	1
2	3	4	5	6	7	6	5
4	3	2	1	1	2	3	4
5	6	7	8	8	7	6	5
4	3	2	3	4	5	6	7
8	8	7	6	5	4	5	6
7	8	8	7	6	7	8	8

表 3.3-2　Zigzag 变换表: 列变换索引表

1	2	1	1	2	3	4	3
2	1	1	2	3	4	5	6
5	4	3	2	1	1	2	3
4	5	6	7	8	7	6	5
4	3	2	1	2	3	4	5
6	7	8	8	7	6	5	4
3	4	5	6	7	8	8	7
6	5	6	7	8	8	7	8

函数源代码如下:

```
% 文件名: zigzag. m
% 程序员: 郭迟
% 编写时间: 2004. 2. 12
% 函数功能: 本函数将完成对输入的 8× 8 矩阵按照 zigzag 排列抽取数据
% 输入格式举例: zigdone = zigzag( A, 1)
% 参数说明:
% matrix 为输入矩阵
% select =1 为正变换, select =2 为反变换
% zigdone 为输出矩阵
function zigdone = zigzag( matrix, select) ;
if select == 1
row = [ 1   1   2   3   2   1   1   2
        3   4   5   4   3   2   1   1
        2   3   4   5   6   7   6   5
        4   3   2   1   1   2   3   4
        5   6   7   8   8   7   6   5
        4   3   2   3   4   5   6   7
        8   8   7   6   5   4   5   6
        7   8   8   7   6   7   8   8] ;

col = [ 1   2   1   1   2   3   4   3
```

```
        2   1   1   2   3   4   5   6
        5   4   3   2   1   1   2   3
        4   5   6   7   8   7   6   5
        4   3   2   1   2   3   4   5
        6   7   8   8   7   6   5   4
        3   4   5   6   7   8   8   7
        6   5   6   7   8   8   7   8] ;
end
if select  == 2;
row = [ 1   1   1   1   2   2   4   4
        1   1   1   2   3   4   4   6
        1   2   2   3   4   4   6   6
        2   2   3   4   4   6   6   7
        2   3   3   5   5   6   7   7
        3   3   5   5   6   7   7   8
        3   5   5   6   7   8   8   8
        5   5   7   7   8   8   8   8] ;

col = [ 1   2   6   7   7   8   4   5
        3   5   8   6   1   3   6   3
        4   1   5   2   2   7   2   4
        2   4   3   1   8   1   5   6
        3   4   8   1   8   6   5   7
        5   7   2   7   7   4   8   5
        6   3   6   8   3   1   4   6
        4   5   1   2   2   3   7   8] ;
end

zigdone = zeros( 8) ;
for i = 1∶8
    for j = 1∶8
        zigdone( i, j)  = matrix( row( i, j) , col( i, j) ) ;
    end
end
```

同理, Zigzag **逆变换的变换表如表** 3. 4 **所示。通过表** 3. 2、**表** 3. 3 **和表** 3. 4 **可以非常方便地完成** Zigzag **变换**。

表 3.4　　Zigzag 逆变换行、列索引表

1	1	1	1	2	2	4	4
1	1	1	2	3	4	4	6
1	2	2	3	4	4	6	6
2	2	3	4	4	6	6	7
2	3	3	5	5	6	7	7
3	3	5	5	6	7	7	8
3	5	5	6	7	8	8	8
5	5	7	7	8	8	8	8

1	2	6	7	7	8	4	5
3	5	8	6	1	3	6	3
4	1	5	2	2	7	2	4
2	4	3	1	8	1	5	6
3	4	8	1	8	6	5	7
5	7	2	7	7	4	8	5
6	3	6	8	3	1	4	6
4	5	1	2	2	3	7	8

(5) 熵编码

为了进一步压缩图像数据,有必要对直流系数 DC 和交流系数 AC 进行熵编码,这是 DCT 变换编码的最后一步。熵编码一般采用哈夫曼编码。编码时 DC 系数与 AC 系数分别采用不同的哈夫曼编码表,对于亮度和色度也需要不同的哈夫曼编码表。所以,对图像数据进行编码时,同时需要 4 张不同的哈夫曼编码表。对直流系数 DC 和交流系数 AC 来说,熵编码过程分为两步进行:

①将已量化的变换系数,按 Zigzag 字形顺序变成一种中间符号格式序列(Intermediate sequence of Symbols)。

②将这些中间符号格式序列转变成变字长代码。

由于过程比较复杂,在这里就不再阐述了,有兴趣的读者可参考数字图像处理的相关书籍。

3.2.4　DCT 变换在图像压缩上的应用示例

下面具体举例进行图像 DCT 变换压缩,编写函数 dctcom.m 完成对输入图像的 8× 8分块的二维 DCT,然后丢弃块中那些近似于 0 的频率数值,只保留 64 个 DCT 系数中的 10 个,最后对每一个块使用二维反 DCT 重构图像,函数的代码如下:

```
% 文件名: dctcom.m
% 程序员: 李巍
% 编写时间: 2004.1.12
% 函数功能: 本函数将利用 DCT 变换完成对输入图像进行压缩
% 输入格式举例: comimage = dctcom('c:\lenna.jpg','jpg')
% 参数说明:
% image 为输入的灰度图像
% permission 为图像类型
```

```
% comimage 为压缩后的图像矩阵
function comimage = dctcom( image, permission)
f = imread ( image, permission) ;
f = double( f) /255;
T = dctmtx( 8) ;
B = blkproc( f, [ 8 8] , 'P1* x* P2 ', T, T') ; % T 和 T 的转置是 DCT 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] ;   %1 表示保留, 0 表示舍弃
B2 = blkproc( B, [ 8 8] , 'P1. * x', mask) ; %" . * "表示矩阵对应元素向量乘法
I2 = blkproc( B2, [ 8 8] , 'P1* x* P2 ', T', T) ; %" * "表示矩阵乘法
subplot( 221) , imshow( f) ; title( '原始图像') ;
subplot( 222) , imshow( I2) ; title( '压缩后的图像') ;
M = I2 - f;    % 压缩前后图像数据矩阵相减
subplot( 223) , imshow( mat2gray( M) ) , title( '图像细节')    % 构成并显示相减图像
```

以灰度的 lenna 图像为输入, 得到的结果如图 3.16 所示。

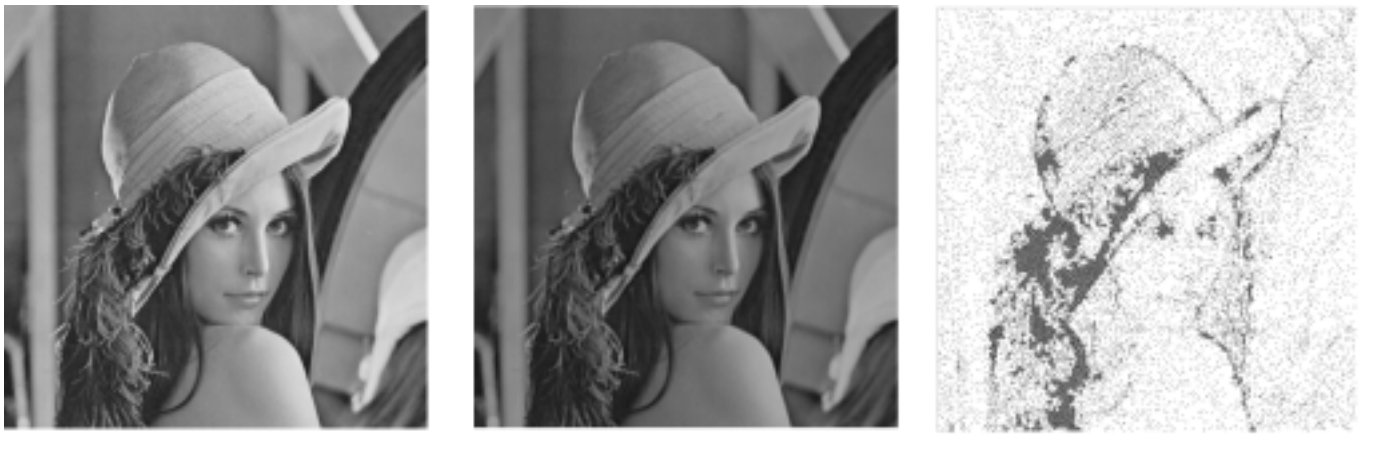


图 3.16　DCT 域图像压缩

对比图 3.16 中的两幅图像可以看出, 虽然几乎 85% 的 DCT 系数都被丢弃, 导

致重建的图像有一些质量损失,但是图像仍然是清晰可辨的。

3.3 小波分析初步

1981 年,法国地质学家 Jean. Morlet 首次提出了“小波分析”的概念,建立了以其自己名字命名的 Morlet 小波,在地质信号处理中取得了巨大成功。此后,经过 Meyer, Mallat, Daubechies 等人的不断深入研究,奠定了小波分析的基础。由于小波分析有助于人们区分信号的平坦部分与敏感变换部分,如今它已经在自然科学诸多领域被使用。一般地说,小波分析的内容可以理解为对小波本身的研究(如低通滤波器的冲激响应 h(t) 的确定、如何快速实现既定信号的小波变换等)和小波应用场合的合理把握两个方面。对于信息隐藏这门学科来说,小波变换域的信息隐藏算法具有透明性高、鲁棒性高等优点,现在已经成为主流算法之一。事实证明,基于不同的小波基水印的性能是不同的。我们认为,在信息隐藏中的小波分析更主要的应该是努力寻找最适合信息隐藏这一任务的小波,通过理论和实验分析得到适合信息隐藏的小波的一般描述。

3.3.1 小波函数存在的空间

这里所谓的“空间”是数学意义上的数学空间。一般地说,“空间”是指某种对象(函数、向量、状态等)的一个集合以及在这个集合中建立的关于对象间的一种或几种数学结构的总称。“空间”通过数学结构建立了元素与元素之间的关系,如大家熟悉的线性空间,就是在集合 X 上定义了加和乘两种线性运算得到的。在小波分析这门学科中,大家往往会遇到诸如 L^2 空间、l^2 空间、Hilbert 空间等概念。这里,我们先简单了解一下这些空间结构。

定义 3.1 (内积空间) 设 X 是数域 K 上的线性空间。如果对于每一元素对 $x, y \in X$,存在 K 中的数 $<x, y>$ 与它们惟一对应,并且对于任意的 $x, y, z \in X$, $a, b \in K$,有:

① $<x, y> = \overline{<y, x>}$(共轭对称性。这里我们讨论的都是实值函数,所以共轭的符号可以去掉,下同)

② $<ax + by, z> = a<x, z> + b<y, z>$(线性)

③ $<x, x> \geq 0$ 且有 $<x, x> = 0 \quad x = \theta$, θ 为线性空间 X 中的零元。(非负性)

则称 $<x, y>$ 为 x 和 y 的内积,X 为内积空间。

内积空间的元素为点或者向量。当元素为向量时,n 维内积空间的内积定义为 $<x, y> = \sum_{n=1}^{\infty} x_n y_n$。向量 x 的长度称为范数,记为 $\|x\| = \sqrt{<x, x>}$。很好理解,当 K 为实数域 R 时,X 为实内积空间,又叫做实 Euclid 空间。在内积空间中,有我们非常熟悉的 Cauchy-Schwarz 不等式:

$$\| x \| \| y \| = \sqrt{< x, x >} \sqrt{< y, y >} \geq | < x, y > |$$

定义 3.2　(**赋范线性空间**)　　所谓赋范线性空间简单地说就是定义了元素范数的线性空间。设 X 是数域 K 上的线性空间。如果对于每一元素 $x \in X$, 存在 K 中数 $\| x \|$ 与它惟一对应, 并且对于任意的 $x, y \in X, a \in K$, 有:

① $\| x \| \geq 0$ 且有 $\| x \| = 0 \quad x = \theta$, θ为线性空间 X 中的零元。(非负性)

② $\| ax \| = | a | \| x \|$ (绝对齐性)

③ $\| x + y \| \leq \| x \| + \| y \|$ (满足三角不等式性)

则称 $\| x \|$ 为 x 的范数, X 为赋范线性空间。

显然, 对于内积空间, 由于其元素的范数可以由内积诱导出 ($\| x \| = \sqrt{< x, x >}$), 当然它也就是赋范线性空间。反之, 赋范线性空间中不一定能定义一个内积使元素的范数恰好能由其内积导出, 所以, 赋范线性空间不一定是内积空间。

定义 3.3　(Banach 空间)　　设 X 是赋范线性空间, $\{ x_n \} \quad X$ 是一个点列。如果

$$" \ \varepsilon > 0, \forall \ A \in N_+, \text{使得} " \ m, n > A, \text{恒有} \ \| x_m - x_n \| < \varepsilon$$

则称$\{ x_n \}$是空间 X 中的一个 Cauchy 列。

直观地说, 也就是当 n 充分大时, $\{ x_n \}$中的点的距离(范数) 可以任意小, 即点列收敛。若 X 中的每个 Cauchy 列都收敛于 X 中的点, 则称 X 是完备的。对于一个完备的赋范线性空间, 我们又称为 Banach 空间。

定义 3.4　(Hilbert 空间)　　若 X 是内积空间, 由内积诱导出的范数构成 Banach 空间, 则称 X 是完备的内积空间, 又称为 Hilbert 空间。

显然, 大家熟悉的 n 维 Euclid 空间 R^n 是 Hilbert 空间。下面, 我们再来看一种典型的 Hilbert 空间, l^2 空间。

定义 3.5　(**平方可和数列空间**, l^2 **空间**)　　l^2 空间中的每个元素都是一个实数列$\{ x_n \}$, 记为 $x = \{ x_1, x_2, \cdots, x_n, \cdots \}$, 并满足 $\sum_{n=1}^{\infty} | x_n |^2 < + \infty$。$l^2$ 空间中的加、乘、内积和范数分别定义为:

$$x + y = (x_1 + y_1, x_2 + y_2, \cdots, x_n + y_n, \cdots)$$

$$\lambda x = (\lambda x_1, \lambda x_2, \cdots, \lambda x_n, \cdots)$$

$$< x, y > = \sum_{n=1}^{\infty} x_n y_n$$

$$\| x \| = \sqrt{< x, x >} = \sqrt{\sum_{n=1}^{\infty} | x_n |^2}$$

l^2 空间是一个无限维空间, 最早是由德国科学家 Hilbert 提出的。

在高等数学中我们系统地学习了黎曼(Riemann) 积分。我们知道, Riemann 积分可积的函数必须是连续的, 或有有限个间断点的。Riemann 积分要求被积函数 f 在积

分区间[a, b]的变化不能“太剧烈”,或者急剧变化的点不能太多。所以,真正能够使Riemann积分可积的函数是非常少的。Riemann积分的几何意义是曲线围成的面积,其基础是建立在对曲线长度的分割上的。随着自然科学的发展,人们开始注意到如何将长度、面积等概念扩展到更为广泛的点集类上,将积分的概念置于集合测度论的理论框架中去。法国数学家勒贝格(H. L. Lebesgue)提出的Lebesgue积分很好地扩充了Riemann积分。

Lebesgue积分改变了Riemann积分对积分区间分割的思路,采用对被积函数值域进行分割的方法。相应的定义域被分割为互不相交的子集,从而使被积函数的值在这些区间上变化不大。H. L. Lebesgue本人对于其积分思想和Riemann积分的区别有一个很精辟的比喻:我必须偿还一笔钱。如果我从口袋中随意的摸出各种不同面值的钞票,逐一地还给债主直到全部还清,这就是Riemann积分。不过,我还有另一种方法,就是将我口袋的钱全部拿出来,并把相同面值的钞票放在一起,然后计算每一类面值的总额,最后相加在一起还给债主,这就是我的积分! 有关Lebesgue积分的具体内容涉及到测度论中点集的测度和可测函数等概念,并不是我们这里要讨论的。有兴趣的读者可以参考一些实变函数方面的书。

建立Lebesgue积分的方法很多。例如,对于Riemann可积的全体函数构成的赋范线性空间R([a, b])是不完备的。将R([a, b])中的Cauchy列的极限函数(不在R([a, b])中,因而不完备)作为新的点添加到R([a, b])中,就得到了完备的R([a, b])空间,也就是Lebesgue可积函数空间L([a, b])。由此,我们引入L^p和L^2空间。

定义3.6 (L^p([a, b])空间) 设[a, b]是R上的一个闭区间,$1 \le p \le \infty$,若$|f|^p$在[a, b]上Lebesgue可积,则称f为[a, b]上的p方可积函数。类似向量范数的定义为:

$$\| f \|_p = \sqrt[p]{\int_{[a,b]} | f(x) |^p dx}$$

使$\| f \|_p < +\infty$的[a, b]上的全体p方可积函数构成的空间记为L^p([a, b])空间。

L^p([a, b])空间满足范数公理,同时是完备的,所以是一个Banach空间。

随着Lebesgue积分的建立,在L^p([a, b])空间中,L^2空间(p = 2)的重要性逐渐被人们所认识。L^2空间与l^2空间具有同构的特点。同时,L^2空间紧密联系着Fourier级数及其展开式,成为信号处理领域中的一个数学理论基础。

定义3.7 (L^2([a, b])空间是Hilbert空间) 对于L^p([a, b])空间,当p = 2时,设$f, g \in L^2([a, b])$,定义$< f, g > = \int_{[a,b]} f(x)\ \overline{g(x)}\, dx$。则由内积诱导的范数满足Cauchy- Schwarz不等式,即$| < f, g > | \le \| f \|_2 \| g \|_2$,显然$< f, g >$满足内积空间的要求,所以$L^2$([a, b])是Hilbert空间。

定义3.8 (L^2([a, b])空间的正交基) 若$f, g \in L^2([a, b])$,且有$< f, g > = 0$,则称f与g正交。若对$f \in L^2([a, b])$,存在点列$\lambda[n]$使得$\lim_{N\to\infty} \left\| f - \sum_{n=0}^{N} \lambda[n] e_n \right\| = 0$

成立，则$\{e_n\}_{n\in N}$为一组正交基。若对于一切 n，都有$\| e_n \|_2 = 1$，则称$\{e_n\}_{n\in N}$是标准正交基。

定义 3.9　(Hilbert 空间的 Riesz 基)　我们削弱无限维空间正交性的要求称$\{e_n\}_{n\in N}$是 Hilbert 空间的一组 Riesz 基，如果它是线性无关的，对于 A，B > 0，任意的 $f\in H$，总有 λ[n] 使得 $f = \sum_{n=0}^{\infty} \lambda[n] e_n$ 且$\frac{1}{B}\| f \|^2 \leq \sum_n | \lambda[n] |^2 \leq \frac{1}{A}\| f \|^2$。

可以证明存在e_n'使得 $\lambda[n] = < f, e_n' >$，$f = \sum_{n=0}^{\infty} < f, e_n' > e_n = \sum_{n=0}^{\infty} < f, e_n > e_n'$。称$\{e_n'\}_{n\in N}$是$\{e_n\}_{n\in N}$的对偶 Riesz 基，且 $< e_n, e_m' > = \delta[n - m]$，存在双正交关系。

后面我们将知道，小波函数就是存在于 $L^2(R)$ 空间的，其相应的积分运算也都是 Lebesgue 积分。本节我们不加证明地用定义的方式给出了一些结论，作为小波分析的一点数学基础，接下来我们就具体认识一下什么是小波和小波分析在信息隐藏上的运用。

3.3.2　小波与小波变换简述

通俗地讲，小波(Wavelet)是一种在有限(小)区域内存在的波，是一种其函数表达式具有紧支集，即在有限范围内函数 f(x) 不等于零的特殊波形。假设存在一个时域函数 $\phi(t)$，满足：

$$\phi(t) \overset{f}{\longleftrightarrow} \Phi(\omega) \quad (f \text{ 表示 fourier 变换})$$

$$\int_{-\infty}^{\infty} \phi(t)\, dt = 0$$

或
$$C_\phi = \int_0^{\infty} \frac{| \Phi(\omega) |^2}{\omega} d\omega < +\infty \tag{3.8}$$

则称 $\phi(t)$ 为一个母小波函数(Mother Wavelet Function)。从式(3.8)可以看出，一个母小波函数有如下几个特点：

①因为 $\Phi(\omega) = \int \phi(t)\, e^{-\omega t j} dt$，而$\int_{-\infty}^{\infty} \phi(t)\, dt = 0$，故而 $\Phi(0) = \int_{-\infty}^{\infty} \phi(t)\, dt = 0$。也就是说，一个母小波函数的直流分量(Direct Current Components)为 0。换句话说，就是母小波函数具有正负交替的特点，其均值为 0。

②由式(3.8)的频域条件式可以看出，其频域函数在整个频带上的积分是有限的，也就是说，一个母小波函数是一个带通信号。

③母小波函数随 t 绝对值的变大而最终衰减为 0，即其函数表达式具有紧支集。

通信学上有名的 Shannon 函数 $\phi(t) = \frac{\sin(\pi t/2)}{\pi t/2}\cos(\frac{3\pi}{2})$ 就是一个典型的母小波，图 3.17 给出了它的时、频域图像。

如果我们将 Fourier 变换看成是一个棱镜折射的过程，很显然，Fourier 变换就是将

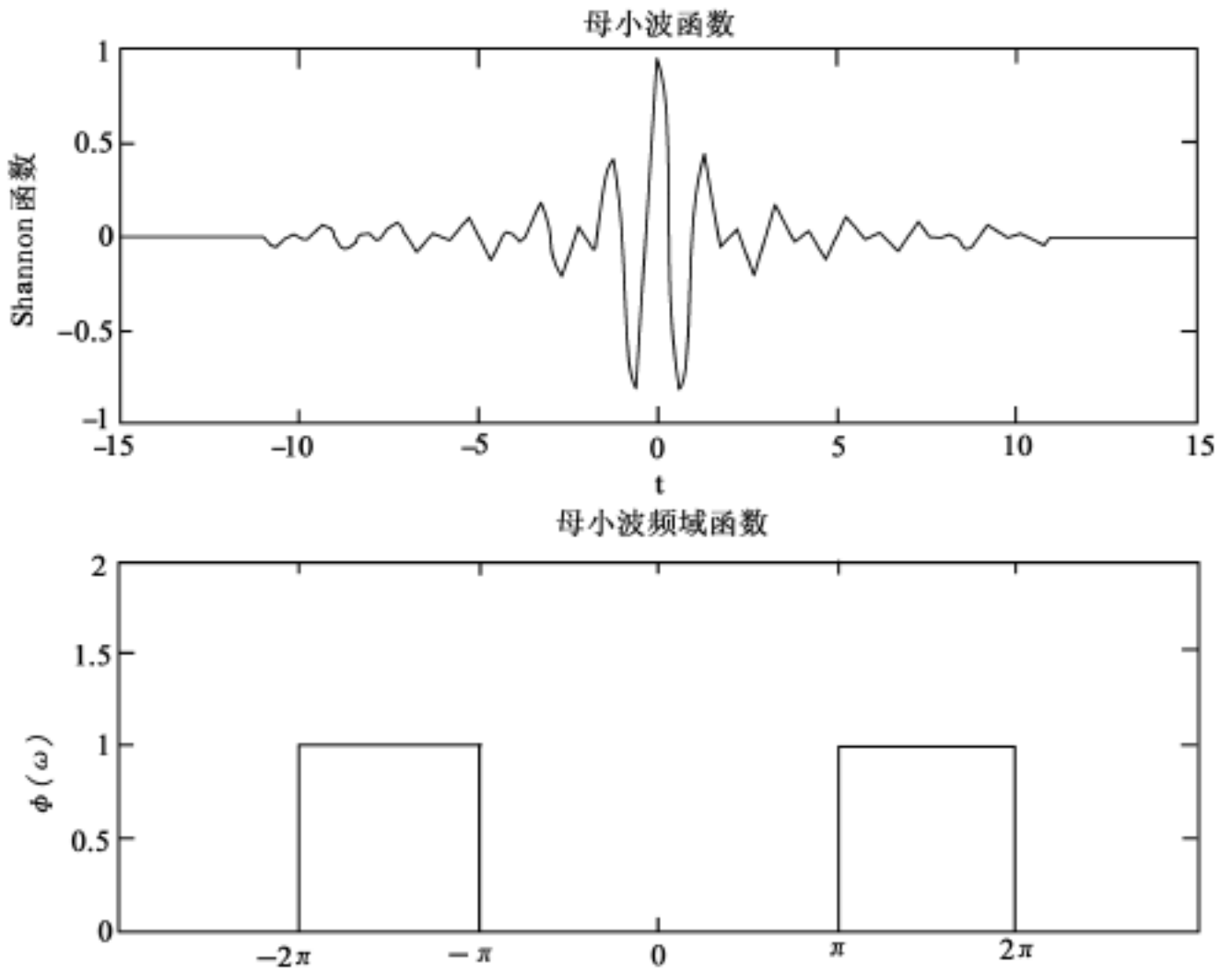


图 3.17　Shannon 小波的时、频图像

信号投影到一组正弦和余弦函数构成的正交基上，将信号等同于一系列正弦和余弦波的叠加。同样地，小波变换也是将一个棱镜折射的过程，它是将信号投影到一组小波函数构成的正交基上，使信号等同于一系列小波函数的叠加，如图 3.18 所示。

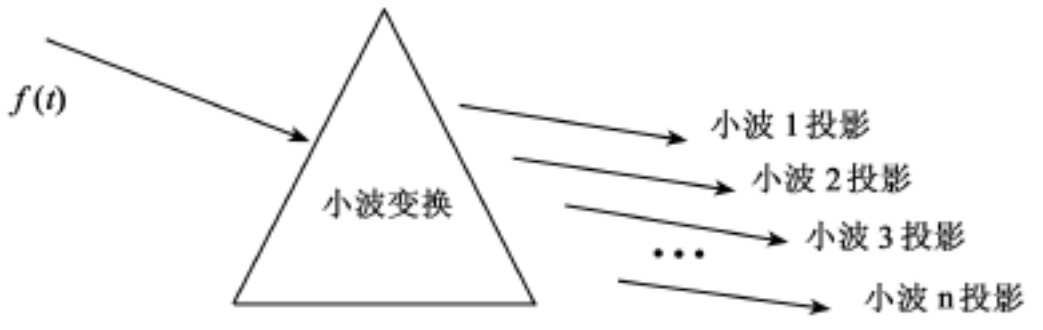


图 3.18　小波变换的简单描述

那么这系列小波函数是怎样得到的呢？这就要用到前文所述的母小波函数。对于满足式(3.8) 的母小波函数 φ(t) 作尺度伸缩(scaling) 和时间平移(shifting) , 得到:

$$\phi_{a,b}(t) = |a|^{\frac{-1}{2}}\phi\left(\frac{t-b}{a}\right) \quad a \in R-\{0\}, b \in R \tag{3.9}$$

$\phi_{a,b}(t)$ 称为小波函数或小波。在式(3.9)中，a 反映函数的尺度，b 反映了小波沿 t 平移的位置。显然，$\phi_{a,b}(t)$ 与母小波 $\phi(t)$ 之间具有同样的性质。对于母小波 $\phi(t)$，其能量集中在原点，而小波 $\phi_{a,b}(t)$ 的能量集中在 b 点。图 3.19 给出了 Shannon 母小波在 $a=0.5, b=1$ 和 $a=2, b=1$ 时的小波函数图像。

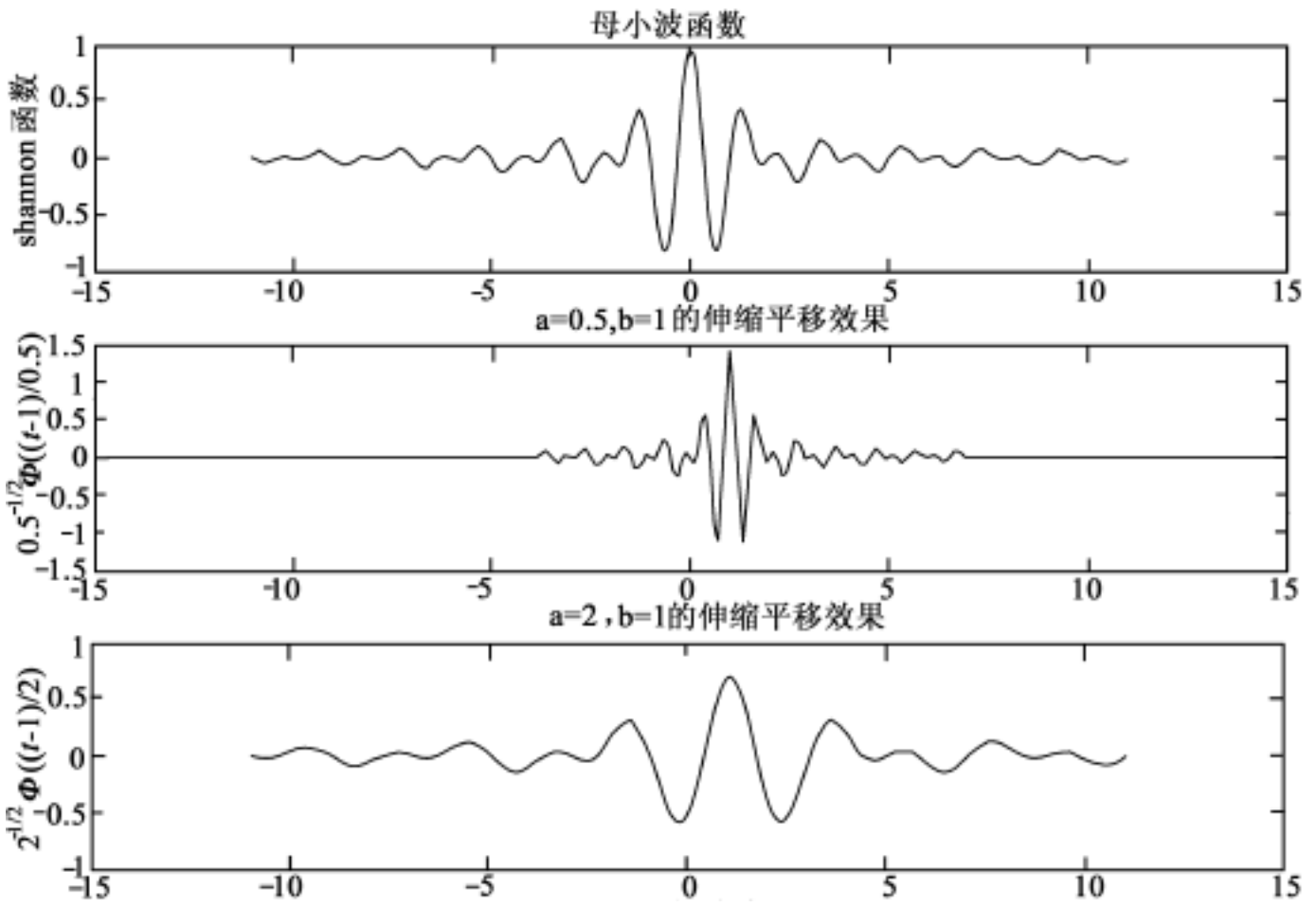


图 3.19　母小波函数与小波函数

下面，我们结合图 3.18 的小波变换过程，以一维信号的连续小波变换为例，作出如下直观解释：

①选择一个小波函数并与信号起点对准。

②计算二者的逼近程度，即将信号与小波函数求内积，得到小波变换系数：

$$CWT(a,b) = \langle f(t), \phi_{a,b}(t) \rangle = \int f(t)\,|a|^{\frac{-1}{2}}\,\overline{\phi\left(\frac{t-b}{a}\right)}\,dt \tag{3.10}$$

系数越大，表明此时刻信号与小波函数越相像。

③将小波函数沿 t 平移一个单位，重复②，直到处理完整个信号。

④将小波函数尺度伸缩一个单位，重复①，②，③，直到所有尺度的小波函数都参与计算。

这样一来，我们就得到了不同尺度下的小波函数对信号在不同时间段的评估值。

这些小波变换系数(评估值)就反映了信号在这些小波上的投影大小。显然,尺度越大,意味着小波函数在时间上越长,被参与求内积的原始信号区间越大,则相应的频率分辨率越低,获取的是信号的低频特性。反之,尺度越高,获取的是信号的高频特征。式(3.10)给出了信号 f 的变化位置(b + at),变化速度(a)和信号量值。

对于式(3.10),其逆变换构成了小波重构函数,表示为:

$$f(t) = \frac{1}{C_\phi}\int_0^\infty\int_{-\infty}^{\infty} CWT(a,b) \mid a \mid^{\frac{-1}{2}} \phi\left(\frac{t-b}{a}\right) db \frac{da}{a^2}$$

$$C_\phi = \int_0^\infty \frac{\mid \Phi(\omega) \mid^2}{\omega} d\omega \tag{3.11}$$

我们假设母小波函数 $\phi(t)$ 及其频域函数 $\Phi(\omega)$ 分别构成的时间窗和频率窗的中心和半径为(t_0, Δt),(ω_0, $\Delta\omega$),由 Fourier 变换的线性和移位性质可知,$\phi\left(\frac{t-b}{a}\right)$ 的时间窗、频率窗的中心和半径为($b + at$, $a\Delta t$),(ω_0/a, $\Delta\omega/a$)。于是就形成了时间 t 和频率 ω 之间的局部化的时间—频率窗(CWT 时间—频率窗):

$$[b + at - a\Delta t, b + at + a\Delta t] \times \left[\frac{\omega_0}{a} - \frac{\Delta\omega}{a}, \frac{\omega_0}{a} + \frac{\Delta\omega}{a}\right]$$

通过 CWT 时间—频率窗能对信号的局部进行精确定位和描述,这也就是小波变换优于 Fourier 变换的地方。Fourier 分析没有能够反映出频率随时间变换的关系,由于一个信号的频率与其周期长度成反比,要取得高频的信息必须将时间间隔取小,取得低频的信息必须将时间间隔取大,Fourier 分析无法做到这一点。而小波分析则可以产生一个灵活可变的时间—频率窗,在高频部分变窄,低频部分变宽,在高频部分取得较高的时间分辨率,在低频部分取得较高的频率分辨率,非常符合实际信号处理的需要。

同时,Fourier 分析适合非常平稳的周期信号(不是周期信号的进行周期延拓),而小波分析适合处理急剧变化的不稳定信号。显然,用不规则的小波函数去逼近急剧变化的信号肯定比用平滑的正、余弦曲线要好。信号的局部特征用小波函数去表征其效果要远大于用正、余弦信号表征的结果。在信息隐藏领域,通过小波分析去提取载体信号的特征分量,将秘密信息隐藏在其中并与信号压缩等算法相适应,这就是 DWT 域信息隐藏的一般思路。

与连续 Fourier 变换(CFT)和离散 Fourier 变换(DFT)的关系一样,虽然连续小波变换可以很好地提取信号特征,但由上述算法可以发现其计算工作量是巨大的,于是便自然地引出了只取部分尺度和时刻进行运算,最大可能的、准确反映信号特征且能大大减少工作量的离散小波变换(DWT)。

离散小波变换方法是对尺度按幂进行离散化,取为 $a^0, a^1, a^2, \cdots, a^k, k \in N$,得到二进制小波。由图 3.19 可以看到,尺度越大小波函数在时间上越长,有利于表征信号的缓慢变换部分,频率分辨率越低;反之,小波函数在时间上越短,有利于表征信号的尖锐变换的细节部分,频率分辨率越高。当尺度扩大 a^k 时,频率就降低了 a^k 倍,则采样间

隔可以扩大 a^k 倍。根据 Nyquist 采样定理, 将小波函数沿 t 也以 a^k 倍进行平移做均匀采样, 可以不丢失原信号的信息。为了与计算机运算相适应, 一般取 $a=2$, 所以式(3.9)一般又写为:

$$\Phi_{j,k}(t) = 2^{\frac{-j}{2}}\Phi(2^{-j}t - k) \qquad j,k \in Z \tag{3.12}$$

由式(3.12)得到的小波函数构成了 $L^2(R)$ 空间的一组正交小波基。表明了按 2^k 伸缩的小波承载了信号在分辨率 2^{-j} 上的变化。

对应于式(3.10), 式(3.11)可以得到一维 DWT 和 IDWT 的变换式: 设信号 f(t) 的离散取值序列为 f(k)。f(k) 属于典型的 Hilbert 空间(平方可和数列空间, $l^2(Z)$), 即 $\sum_{k=1}^{\infty} |f(k)|^2 < +\infty$。二进制小波如式(3.12), 则:

$$DWT(j,k) = 2^{\frac{-j}{2}} \sum_k f(k)\Phi(2^{-j}t - k) \tag{3.13}$$

$$f(k) = \sum_j \sum_k DWT(j,k)\Phi_{j,k}(k) \tag{3.14}$$

3.3.3　小波分析方法及应用示例

(1) 理论描述

小波分析的方法是多分辨分析(MultiResolution Analysis, MRA)。与 Fourier 变换中 DFT 和 FFT 的关系一样, 在小波变换中也存在一个快速小波算法(FWA), 称为 Mallat 算法。

定义 3.10　空间 $L^2(R)$ 中的一列闭子空间 $\{V_j\}_{j\in Z}$ 称为 $L^2(R)$ 中的一个 MRA, 当且仅当 $\{V_j\}$ 满足:

①" $j \in Z$, $\cdots \quad V_{j-1} \quad V_j \quad V_{j+1} \quad \cdots$;　(单调性)

②$\bigcap_{j\in Z} V_j = \{0\}$, $\bigcup_{j\in Z} V_j = L^2(R)$;　(逼近性)

③" $j,k,a \in Z$, $f(x) \in V_j \quad f(ax-k) \in V_j$;　(线性)

④存在 $g \in V_0$, 使得 $\{g(x-k) \mid k \in Z\}$ 构成 V_0 的 Riesz 基。　(Riesz 基存在性)

对于上面的定义, 我们发现, 一个 MRA 与我们日常生活中观察事物的实际情况是非常相似的, 如图 3.20 所示。假设现在 Cindy 在观察前方的一座山, 她现在离山的距离(尺度)为 j, 观察到的山的信息则为 V_j。改变距离(尺度)为 j+1, Cindy 继续观察, 得到的山的信息则为 V_{j+1}。显然, j 距离(尺度)比 j+1 距离(尺度)离山更近, 则观察到的山的地貌特点信息 V_j 比 V_{j+1} 要多, 即 $V_j \quad V_{j+1}$。反之, 改变距离(尺度)为 j-1, 有 V_{j-1} V_j。所以, 改变观察的尺度, 就可以获得观察对象的不同精度的信息, 当尺度选择合适时, 就可以得到观察对象非常细微精度的信息, 这也就是小波分析为什么叫做‘数字显微镜’的原因。

我们可以从 MRA 上来寻找小波。对于一个属于 $L^2(R)$ 的信号 f, 其小波系数的部

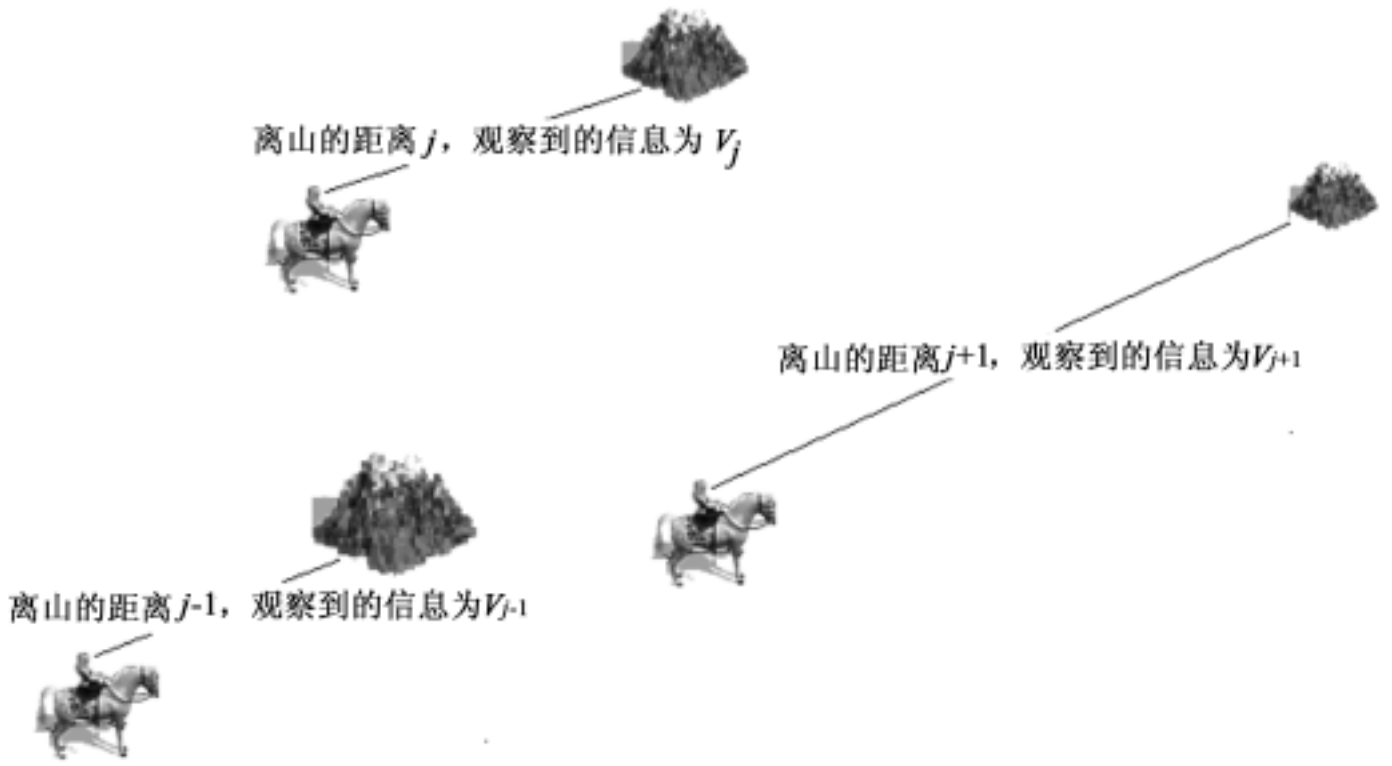


图 3.20　Cindy 观察山的示意图

分和 $\sum_{n=-\infty}^{\infty} \langle f, \phi_{j,n} \rangle \phi_{j,n}$ 其实可以理解为 f 在分辨率 2^{-j+1} 与 2^{-j} 上的两个逼近之差。通过在不同的空间 $\{V_j\}_{j\in Z}$ 上的正交投影，多分辨地计算出信号在不同分辨率上的逼近。f 在分辨率 2^{-j} 上的逼近定义为它在 V_j 上的正交投影。为了计算这个投影，我们必须找到 V_j 的一组标准正交基。而要描述一个空间，只需给出该空间的一组基就可以了。

定义 3.11　如果 $\{V_j\}_{j\in Z}$ 称为 $L^2(R)$ 中的一个 MRA，则存在惟一的函数 $\psi(t)$ 使得 $\{\psi_{j,k}(x) \mid k \in Z\}$ 构成 V_j 的一个标准正交基。其中 $\psi(t) = \sum_{k\in Z} h(k)\psi(2t-k)$，经平移和伸缩后有 $\psi_{j,k}(x) = 2^{\frac{-j}{2}}\psi(2^{-j}x-k)$。函数 $\psi(t)$ 被称为尺度函数。

尺度函数应该满足的方程 $\psi(t) = \sum_{k\in Z} h(k)\psi(2t-k)$。$h(k)$ 就是低通滤波器的冲激响应。MRA 完全由尺度函数所生成的每个空间 V_j 的一组正交基所刻画，而任何尺度函数都被一个滤波器所确定。前面说过，小波分析的一个重要方面就是关于滤波器的设计。要设计能够精确重构的滤波器必须是共轭的，其频率特性必须满足精确重构条件：$|H(\omega)|^2 + |H(\omega+\pi)|^2 = 1$。例如，Meyer 小波使用的滤波器 $|H(\omega)|^2 = 1 - \frac{(2N-1)!}{[(N-1)!]^2 2^{2N-1}}\int_0^{\omega} \sin^{2N-1}x\,dx$ 就是满足这种条件的。

可以发现，这里的 $\psi_{j,k}(x)$ 与前面的小波函数 $\phi_{j,k}(x)$（式(3.12)）有着相似的结构。同样地，小波函数也对应一个滤波器 $\phi(t) = \sum_{k\in Z} g(k)\psi(2t-k)$，其中 $g(k) = (-1)^{1-k}h(1-k)$，即 $G(\omega) = e^{-\omega j}\overline{H(\omega+\pi)}$。前面说过，小波函数 $\phi_{j,k}(x)$ 构成 $L^2(R)$

中的一组正交基。事实上，由前面的关系，$\phi_{j,k}(x)$ 还构成了 $L^2(R)$ 的一个子空间 W_j 的正交基。那么由小波函数 $\phi_{j,k}(x)$ 决定的空间 W_j 和由尺度函数 $\psi_{j,k}(x)$ 决定的空间 V_j 有什么联系呢?结论就是，W_j 是 V_{j-1} 与 V_j 的正交补空间。形象地描述为：$W_j = V_{j-1} - V_j$。也就是说，对于一切 j，V_j 描述的是近似部分，W_j 描述的是细节部分，是相邻两个尺度下观察到的对象的细微差异。结合图3.20我们再来说明一下，如 Cindy 在远处看到一座山，走近了发现山顶有棵松树，那么这棵松树就是属于 $\{W_j\}\mid_{j\in Z}$ 这个空间的元素。

事实上，一个 $L^2(R)$ 中的一个 MRA 就是由小波函数、尺度函数、$\{V_j\}\Big|_{j\in Z}$ 和 $\{W_j\}\Big|_{j\in Z}$ 四者描述的。完全的得到以上四个元素，就完成了对对象的小波分析。

最后，我们来看一下快速小波分析(FWT, Mallat 算法)的数学描述。设一个 MRA 描述为 $\cdots\quad V_{j-1}\quad V_j\quad V_{j+1}\quad \cdots,\cdots\quad W_{j-1}\quad W_j\quad W_{j+1}\quad \cdots$，设信号 $f(x)$ 在尺度 j 下得到的平滑信号(近似部分)为：

$$A_j^d f(x) = \langle f(x), \psi_{j,k}(x)\rangle = 2^{-\frac{j}{2}}\int f(x)\,\psi(2^{-j}x - k)\,dx \tag{3.15}$$

信号 $f(x)$ 在尺度 j 下得到的细节信号为：

$$D_j f(x) = \langle f(x), \phi_{j,k}(x)\rangle = 2^{-\frac{j}{2}}\int f(x)\,\phi(2^{-j}x - k)\,dx \tag{3.16}$$

Mallat 算法的实质就是并不对信号在每一个尺度下作完整的分析，而是对信号由细到粗的分解和由粗到细的重构，即将 $A_{j-1}^d f(x)$ 分解为 $A_j^d f(x)$ 和 $D_j f(x)$。其关系为：

$$\begin{cases} A_j^d f(x) = \sum_k h(k-2n)\,A_{j-1}^d f(x) \\ D_j f(x) = \sum_k g(k-2n)\,A_{j-1}^d f(x) \end{cases} \tag{3.17}$$

式(3.15)～式(3.17)的是一维信号的 Mallat 算法描述。二维信号的 Mallat 算法可以类似的推导出：

对于一个二维信号的 MRA，同样是由小波函数、尺度函数、近似描述空间和细节描述空间四者决定的，记为：$\{\ (x), \Psi(x), \{V_j^2\}\mid_{j\in Z}, \{W_j^2\}\mid_{j\in Z}\}$

其中，尺度函数有：$\Psi(x,y) = \psi(x)\,\psi(y)$ (3.18)

小波函数有：$\begin{cases} \phi^1(x,y) = \psi(x)\,\phi(y) \\ \phi^2(x,y) = \phi(x)\,\psi(y) \\ \phi^3(x,y) = \phi(x)\,\phi(y) \end{cases}$

即：$(x,y) = \{\phi^n(x,y)\mid n = 1,2,3\}$ (3.19)

空间 $\{V_j^2\}$ 有：$V_{j-1}^2 = V_{j-1}\times V_{j-1} = V_j^2 + W_j^2$ (3.20)

空间 $\{W_j^2\}$ 有：$W_j^2 = (V_j\times W_j) + (W_j\times V_j) + (W_j\times W_j)$ (3.21)

可以看到，式(3.19)和式(3.21)具有对应关系。在低的水平频率 ω_1 和高的垂直

频率 ω_2 处，$|\phi^1(\omega_1,\omega_2)|$ 大。在高的水平频率 ω_1 和低的垂直频率 ω_2 处，$|\phi^2(\omega_1,\omega_2)|$ 大。在高的水平和高的垂直频率 ω_1,ω_2 处，$|\phi^3(\omega_1,\omega_2)|$ 大。

如此一来，二维信号的 Mallat 算法就可以写为：

$$A_{j-1}f(x,y) = A_jf(x,y) + D_j^1f(x,y) + D_j^2f(x,y) + D_j^3f(x,y) \quad (3.22)$$

其中，A 为低频分量，D 可以看为水平、垂直和对角三个方向上的高频分量。二维信号的小波分解如图 3.21 所示。

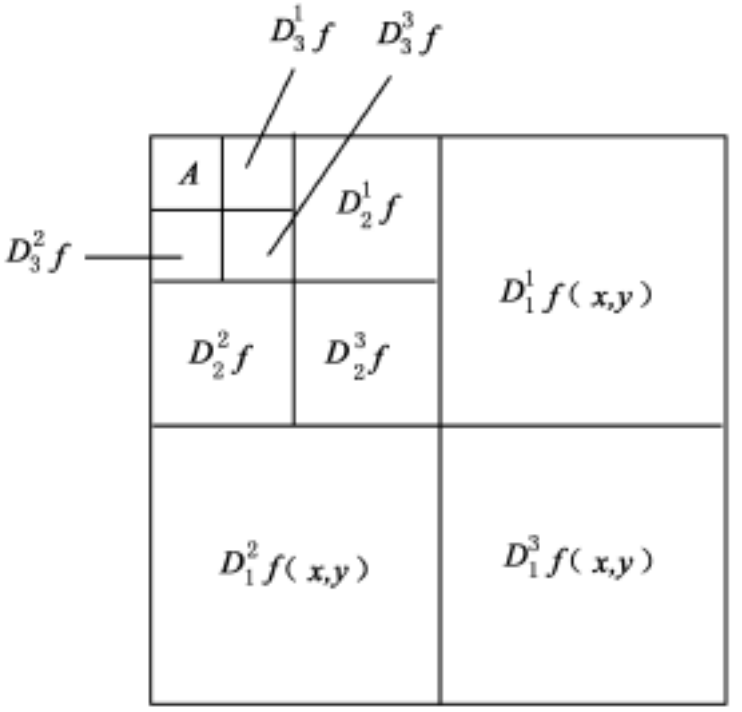


图 3.21　二维信号的小波分解

(2) 应用示例

前面我们粗略地叙述了什么是小波和小波分析，下面我们就一维信号的小波分解、二维信号的小波分解、小波对图像进行压缩和降噪以及小波对图像进行融合四个方面进行实验，进一步了解小波分析的特点。

1) 一维信号的小波分解

我们选择 MATLAB 自带的电网检测信号 leleccum 作为分析对象(如图 3.22 所示)。编写函数 wavelet1D.m 完成实验。函数代码如下：

```
% 文件名: wavelet1D.m
% 程序员: 郭迟
% 编写时间: 2004.1.20
% 函数功能: 本函数将完成对输入的一维信号进行多尺度离散小波分解
% 输入格式举例: load leleccum; [lowf,highf] = wavelet1D(leleccum,'db1',3)
% 参数说明:
% lowf 为最大尺度分解后的低频部分
% highf 为最大尺度分解后的高频部分
```

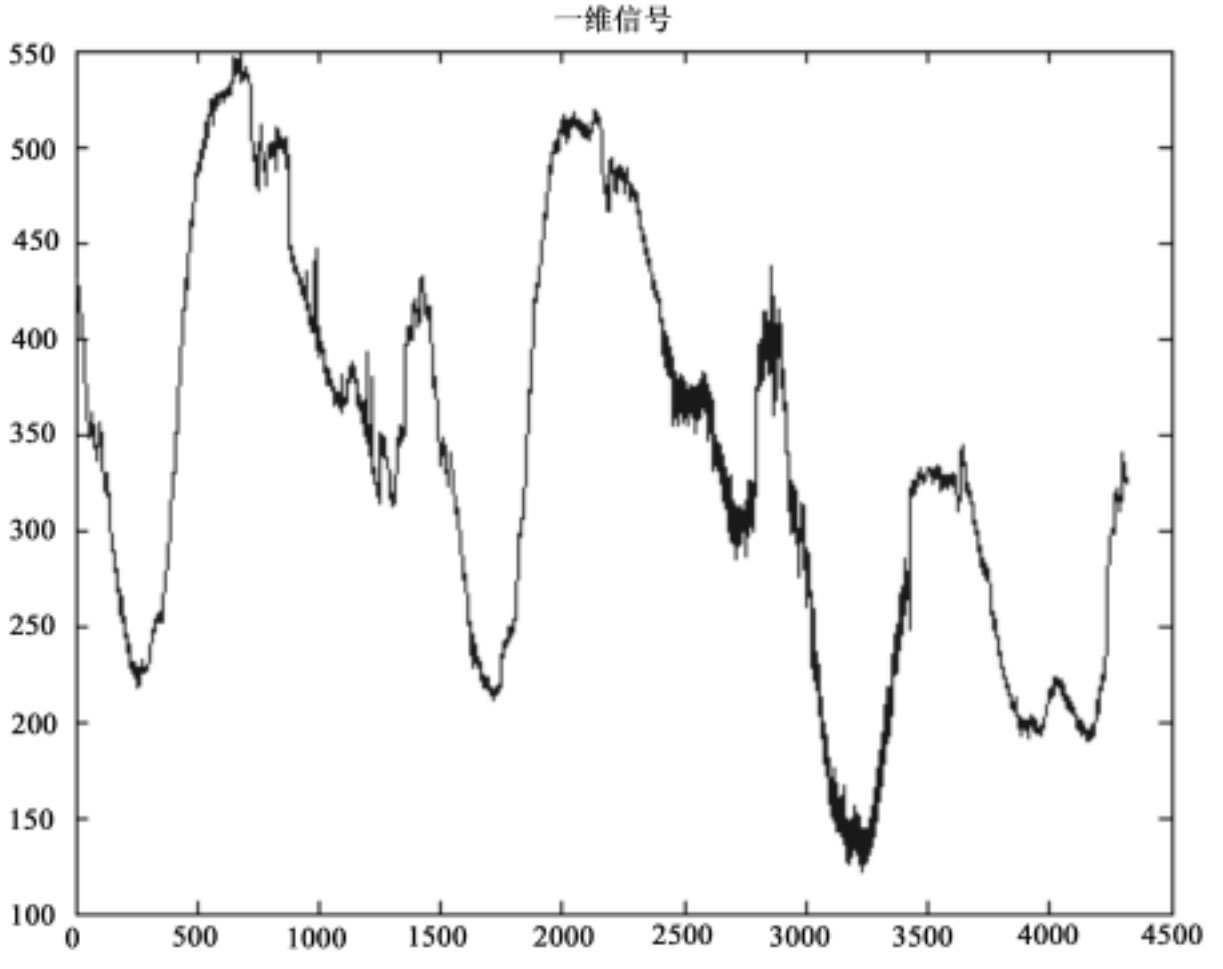


图 3.22　一维信号 leleccum

```
% signal 为输入的原始一维信号
% wavelet 为使用的小波类型
% level 为分解的尺度
function [ lowf, highf] = wavelet1D( signal, wavelet, level) ;
is = length( signal) ;
im = max( signal) ;
% 一维小波分解
[ C, S] = wavedec( signal, level, wavelet) ;
% 提取最大尺度分解后的低频部分
lowf = appcoef( C, S, wavelet, level) ;
% 提取最大尺度分解后的高频部分
highf = detcoef( C, S, level) ;
% 重构最大尺度下分解的低频信号
A = wrcoef( 'a', C, S, wavelet, level) ;
% 重构各尺度下分解得到的高频信号
for i = 1: level
        D( i,:) = wrcoef( 'd', C, S, wavelet, i) ;
```

```
end
% 显示重构后的效果
subplot( 2, 2, 1) ; plot( A) ; axis( [ 0 is 0 im] ) ; title( '低频平滑信号') ;
for i = 1: level
    subplot( 2, 2, i + 1) , plot( D( i,:) ) ; title( [ int2str( i) , '尺度下的高频细节信号'] ) ;
end
```

我们使用 db1 小波做 3 尺度分解, 输入:

```
>> load leleccum;
>> [ lowf, highf]  = wavelet1D( leleccum, 'db1', 3) ;
```

实验结果如图 3.23 所示。

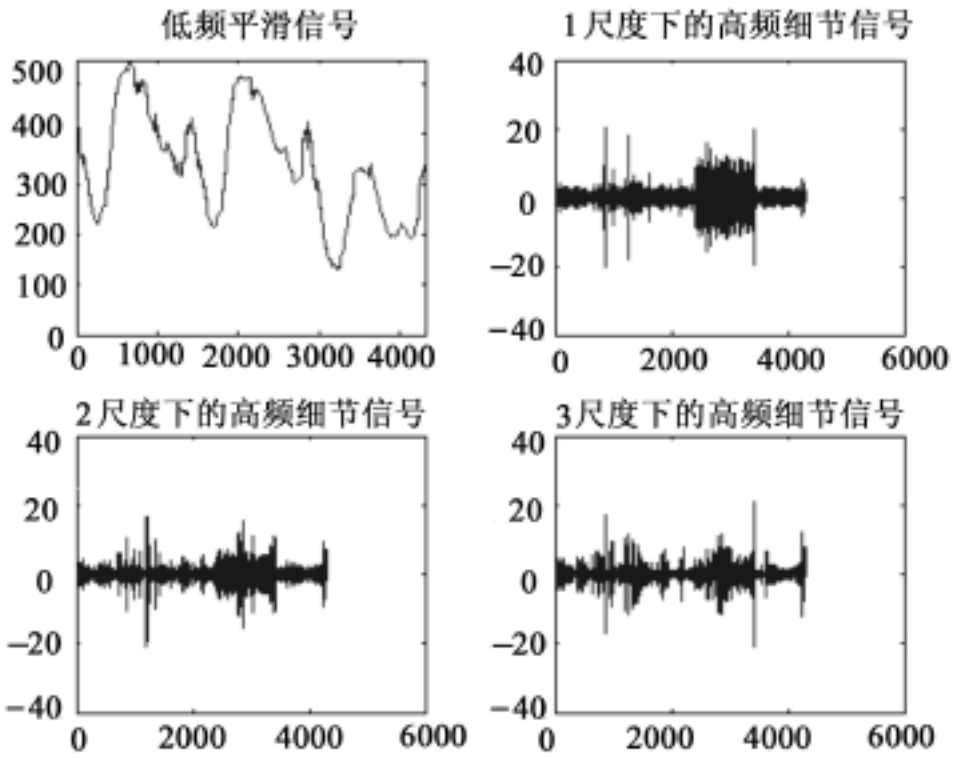


图 3.23　一维信号用 db1 小波做 3 层分解

将图 3.23 的结果与前文对应, 得到如下的分解树, 如图 3.24 所示。

不难发现, 低频平滑信号与原始信号非常相似, 而通过小波分析得到的各尺度小的细节信号在这里基本上可以看做是高斯白噪声。所以, 如果我们将图 3.23 所示的低频平滑信号取代原始信号, 实际上就完成了一个简单的对信号降噪的过程。如果这里的原始信号是一个语音信号的话, 那么经过小波处理后的低频信号与原始信号是很难用耳朵分辨的。图 3.25 是对应图 3.24 分解树中的 f(x) , 和 $D_3 f(x)$、$A_3^d f(x)$ 在未重构之前的矩阵表示, highf 和 lowf 分别表示了 3 尺度下的原始信号的高频系数和低频系数。

2) 二维信号的小波分解

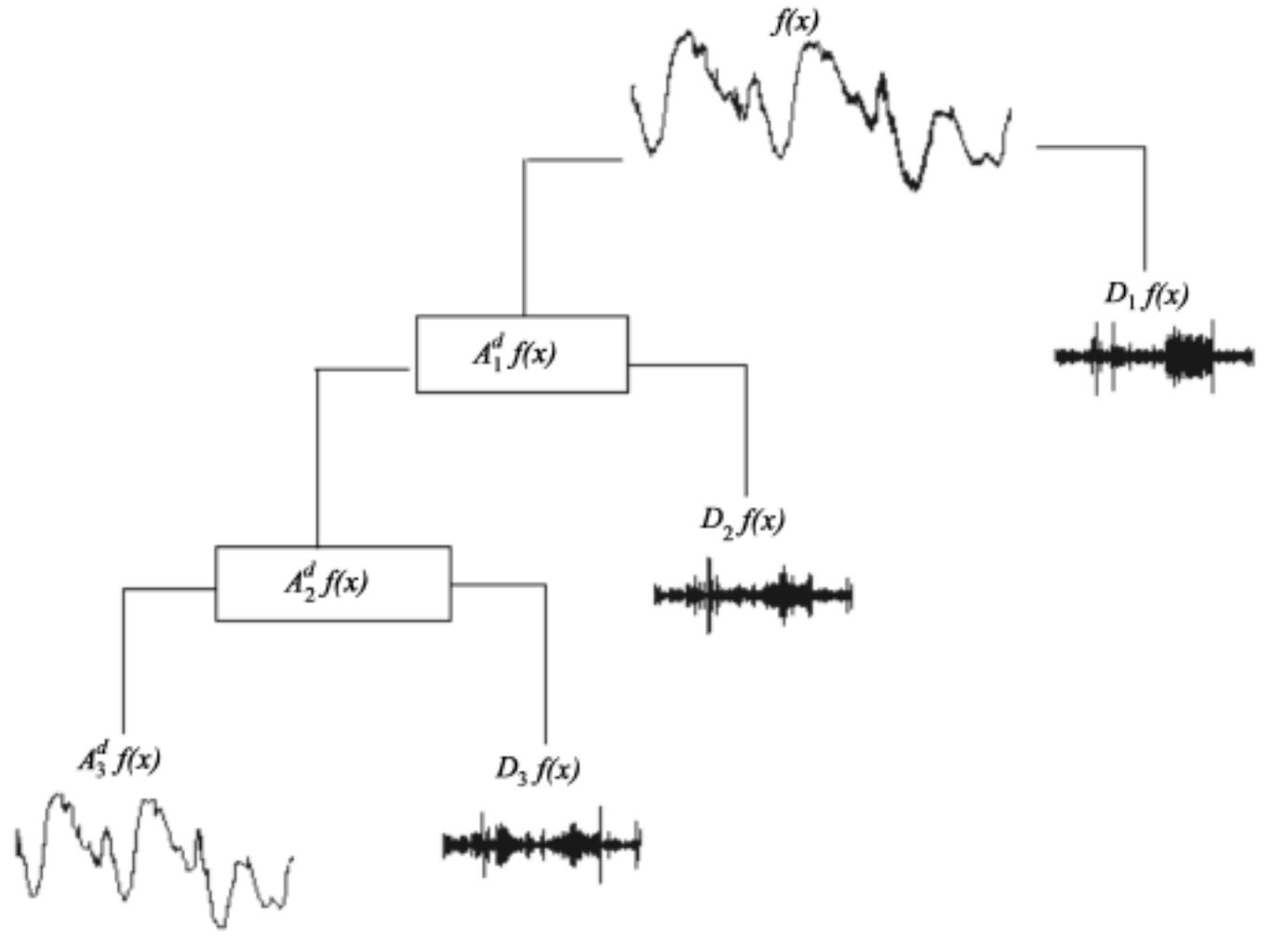


图 3.24　一维信号用 db1 小波做 3 层分解的分解树

Array Editor: leleccum

File Edit View Web Window Help

Numeric format: shortG　Size: 1 by 4320

	1	2	3	4	5	6	7	8	9	10
1	420.2	423.53	423.52	423.35	424.52	428	430.79	428.92	427.24	427.91

Array Editor: highf

File Edit View Web Window Help

Numeric format: shortG　Size: 1 by 540

	1	2	3	4	5	6	7	8	9
1	-7.6467	5.6745	9.515	7.0483	9.655	12.545	12.004	0.39379	-0.2119

Array Editor: lowf

File Edit View Web Window Help

Numeric format: shortG　Size: 1 by 540

	1	2	3	4	5	6	7	8	9	10
1	1203.1	1199.4	1169.1	1139.6	1097.1	1046.4	1005.6	992.85	1012.5	993.31

图 3.25　一维信号用 db1 小波做 3 层分解的高、低频系数

与一维信号的小波分解一样，我们同样对图像做多尺度的二维小波分解。这里，我们选用的二维图像信号仍然是 lenna. jpg。由于 lenna 是一个 RGB 图像，我们仅对其 R 层进行实验。编写函数 wavelet2D. m 来完成实验。函数代码如下：

```
% 文件名: wavelet2D. m
% 程序员: 郭迟
% 编写时间: 2004. 1. 20
% 函数功能: 本函数将完成对输入的二维信号进行多尺度离散小波分解
% 输入格式举例: [ lowf, highH, highV, highD, C, S] = wavelet2D( lennaR, 'db1', 3)
% 参数说明:
% lowf 为最大尺度分解后的低频系数
% highfH 为最大尺度分解后的水平方向高频系数
% highfV 为最大尺度分解后的垂直方向高频系数
% highfD 为最大尺度分解后的对角方向高频系数
% C 为全部分解的频率系数
% S 为各尺度分解下得到的频率系数的长度
% signal 为输入的原始二维信号
% wavelet 为使用的小波类型
% level 为分解的尺度
function [ lowf, highH, highV, highD, C, S] = wavelet2D( signal, wavelet, level) ;
% 二维小波分解
[ C, S] = wavedec2( signal, level, wavelet) ;
% 提取最大尺度分解后的低频部分
lowf = appcoef2( C, S, wavelet, level) ;
% 提取最大尺度分解后的高频部分
highH = detcoef2( 'h', C, S, level) ;
highV = detcoef2( 'v', C, S, level) ;
highD = detcoef2( 'd', C, S, level) ;
% 重构最大尺度下分解的低频信号
A = wrcoef2( 'a', C, S, wavelet, level) ;
% 重构最大尺度下分解得到的高频信号
Dh = wrcoef2( 'h', C, S, wavelet, level) ;
Dv = wrcoef2( 'v', C, S, wavelet, level) ;
Dd = wrcoef2( 'd', C, S, wavelet, level) ;
% 显示重构后的效果
subplot( 2, 2, 1) , image( A) ; title( '低频平滑信号') ;
subplot( 2, 2, 2) , imshow( Dh) ; title( [ int2str( level) , '尺度下的水平高频细节信
```

号 ') ;

subplot(2, 2, 3) , imshow(Dv) ; title([int2str(level) , '尺度下的垂直高频细节信号 ') ;

subplot(2, 2, 4) , imshow(Dd) ; title([int2str(level) , '尺度下的对角高频细节信号 ') ;

我们使用 db1 小波做 2 尺度分解, 输入:

```
>> lenna = imread( 'C: \lenna. jpg ' , 'jpg ' ) ;
>> lennaR = lenna( : , : , 1) ;
>> [ lowf, highfH, highfV, highfD, C, S] = wavelet2D( lennaR, 'db1 ' , 2) ;
```

实验结果如图 3. 26 所示。

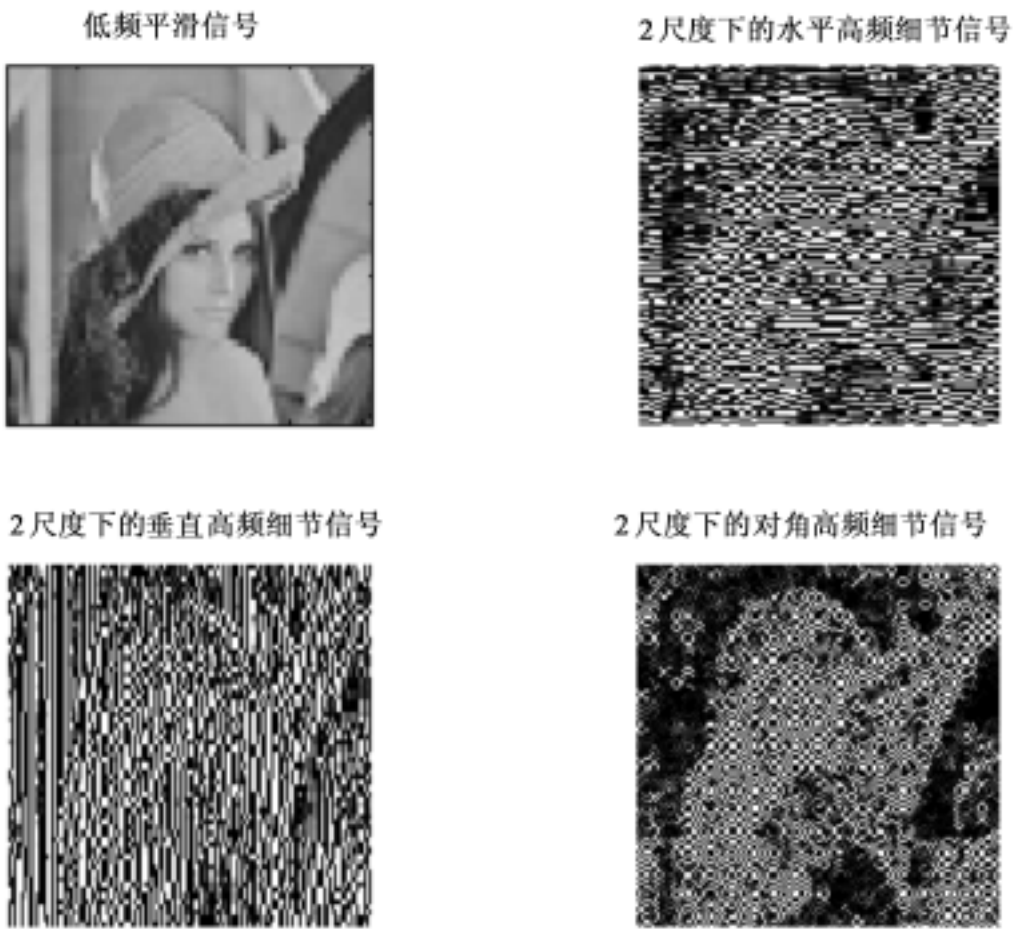


图 3. 26　lenna 图像用 db1 小波做 2 层分解得到的结果

图 3. 26 清晰地反映了两重小波分解后的各个频率段信号重构成的图像。可以发现, 低频图像与原始图像是非常近似的, 而高频部分也可以认为是冗余的噪声部分。所以, 图像载体下的小波分解信息隐藏算法一般都是将信息隐藏于分解后的低频部分, 从而获得高的鲁棒性。将信息隐藏于高频系数中, 可以获得很好的不可见性。不可见性与鲁棒性是信息隐藏算法性能好坏的重要判定依据, 二者可以看成是一对矛盾。解决这一矛盾的方法就是“ 折中 ”。具体的情况我们将在后面专门论述。

最后, 结合 MATLAB 代码, 我们来看一下这些频率系数的具体内容: lennaR 是一

个 256× 256 的二维信号, 对其做 1 层小波分解, 得到的 C 是一个 1× 65536 的行向量, 记录的是低频、水平高频、垂直高频和对角高频

$$(A_1 f(x, y), D_1^1 f(x, y), D_1^2 f(x, y), D_1^3 f(x, y))$$

四个部分的频率系数。S 是一个 3× 2 的矩阵(如图 3.27 所示), 其第一行表明尺度 1 下的低频系数为 128× 128 长度; 第二行表明尺度 1 下的高频系数为 128× 128 长度; 第三行表明 lennaR 是一个 256× 256 的二维信号。lowf 和 highfH, highfD 和 highfV 分别对应分离出来的四个部分的系数矩阵。

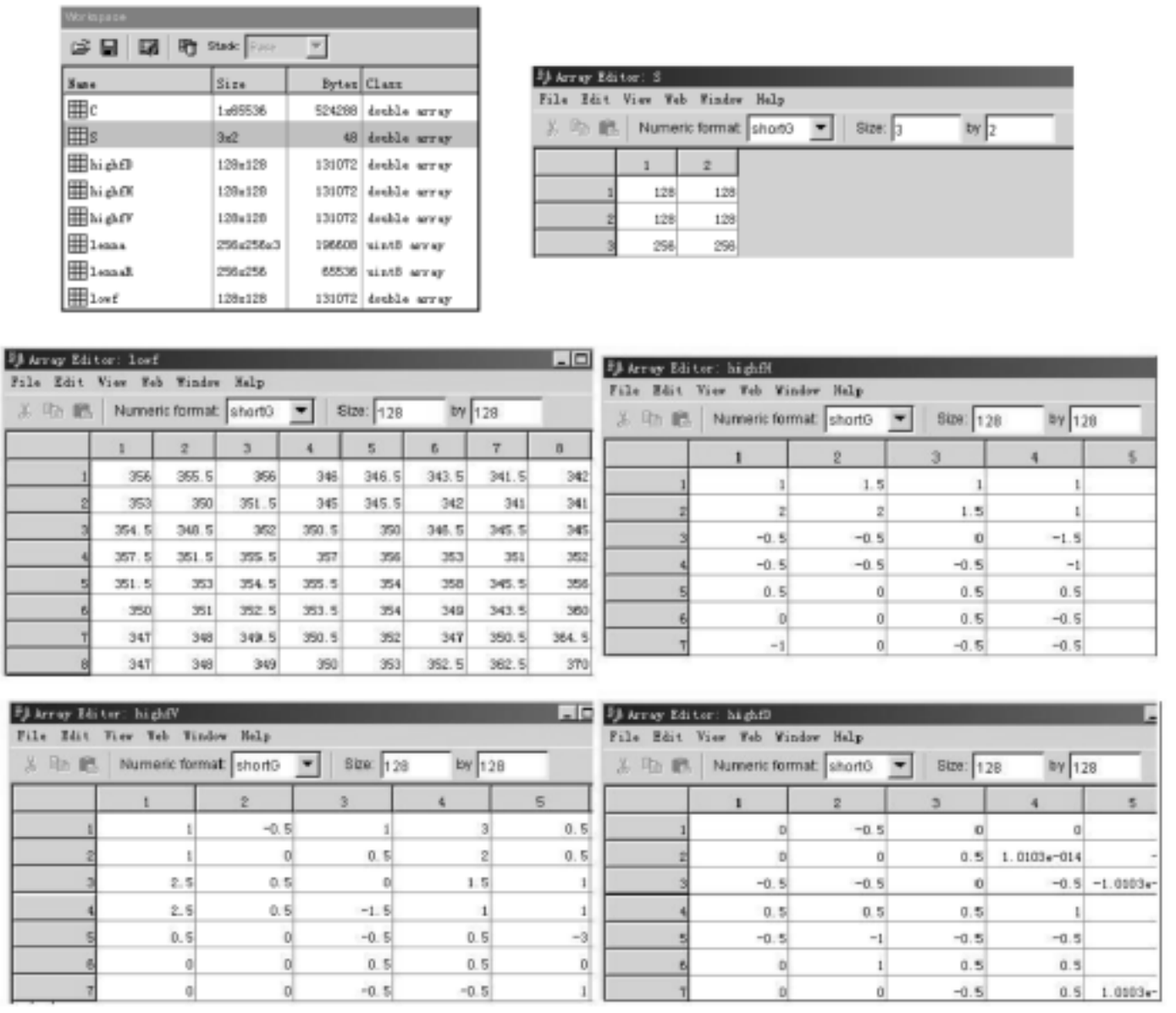


图 3.27　lenna 图像用 db1 小波做 1 层分解得到的频率系数

对 lennaR 做 2 层小波分解, 得到的 C 也是一个 1× 65536 的行向量, 记录的是 2 尺度低频、2 尺度水平高频、2 尺度垂直高频、2 尺度对角高频、1 尺度水平高频、1 尺度垂直高频和 1 尺度对角高频

$$(A_2 f(x, y), D_2^1 f(x, y), D_2^2 f(x, y), D_2^3 f(x, y), D_1^1 f(x, y), D_1^2 f(x, y), D_1^3 f(x, y))$$

七个部分的频率系数。S 是一个 4× 2 的矩阵(如图 3.28 所示), 其第一行表明尺度 2

下的低频系数为 64× 64 长度; 第二行表明尺度 2 下的高频系数为 64× 64 长度; 第三行表明尺度 1 下的高频系数为 128× 128 长度; 第四行表明 lennaR 是一个 256× 256 的二维信号。lowf 和 highfH, highfD 和 highfV 分别对应 2 尺度下分离出来的四个部分的系数矩阵。

比较图 3.27 和图 3.28, 可以发现 MATLAB 中的二维 DWT 有如下规律:

①返回的频率系数(在 C 向量中) 以如下形式存放:

C = [A(level) | H(level) | V(level) | D(level) H(level - 1) V(level - 1)
D(level - 1) … | H(1) | V(1) | D(1)]

②返回频率系数的同时, 返回一个长度记录矩阵 S。S 的格式为:

S(1, :) = 尺度 level 下的低频系数长度

S(i, :) = 尺度 level - i + 2 下的低频系数长度

S(level + 2, :) = 原始信号的大小

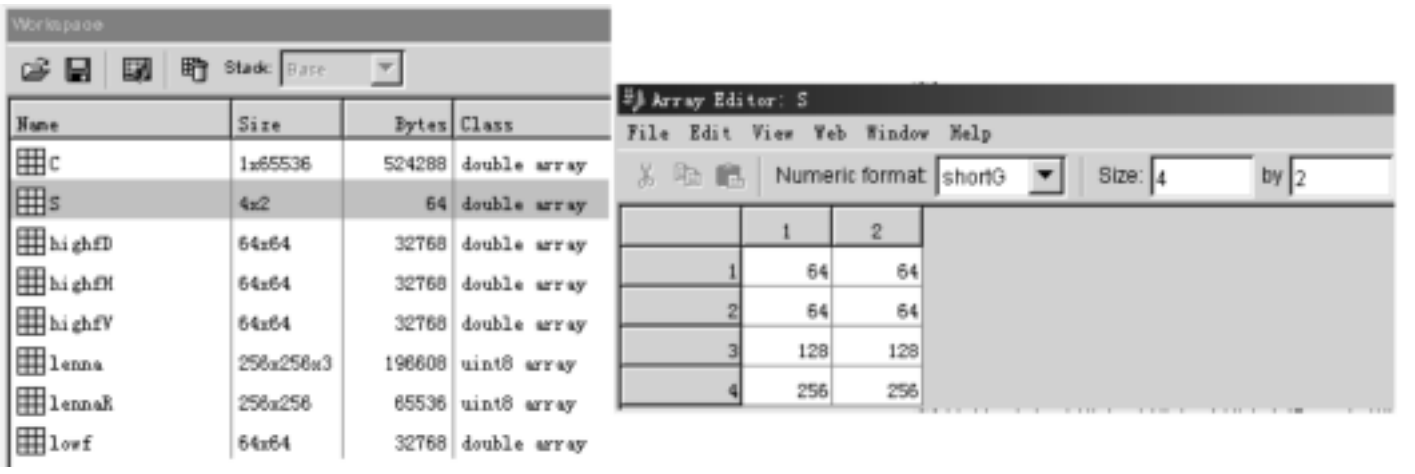


图 3.28　lenna 图像用 db1 小波做 2 层分解得到的结果

③原始信号通过两个共轭滤波器后, 得到高、低频两路信号。假设原始信号抽取 256 个点参与计算, 那么将得到 512 个频率数据, 如此下去显然冗余太大。所以, 在滤波后还需要进一步抽样以减少冗余。通行的方法是隔一数丢弃一个数, 从而保证滤波后的两路信号与原始信号数据长度一致。对照图 3.27 和图 3.28 可以发现 2 尺度下的频率系数与 1 尺度下的频率系数在长度上就满足这种关系。

了解以上三个结论, 对今后的小波信息隐藏实验有非常大的帮助。

3) 小波对图像进行压缩和降噪

结合前面一节在 DCT 运用中的图像压缩, 我们很容易得到小波图像压缩的算法。小波分析将图像分为低频和高频两部分。低频部分对应的是图像的近似部分, 具有原始图像信号主要的能量; 高频部分则对应图像的细节。因此, 利用小波分解直接除去图像的高频部分而仅用其低频进行重构就是最为直接的一种图像压缩方法。当然, 高频部分并不全是噪声, 这样做会丢失一些图像的重要细节, 所以, 一般采用一个阈值判断的方法, 仅将低于这个阈值的高频系数认为全部是噪声而置 0(丢弃) 。

那么这个阈值如何选定呢？我们给出一个最简单的方法：

①对于图像压缩，阈值等于1尺度下对信号分解获得的高频系数的绝对值的中值。如果这个值算出为0，那么阈值就等于1尺度下对信号分解获得的高频绝对值系数中最大的那个值的5%。用伪代码描述为：

```
threshold = median( abs( frequency at level 1) ) ;
if ( threshold == 0)
    threshold = 0. 05*  max( abs( frequency at level 1) )
```

②对于图像降噪，阈值等于 $s\times\sqrt{2\times \lg^{n}}$。其中s为噪声的评估级别，n是图像矩阵的像素数量。

编写函数 imagecom. m 和 imagenr. m 分别完成压缩和降噪的实验。二者代码基本相似，这里我们仅给出 imagenr. m，函数代码如下：

文件名：imagenr. m

% 程序员：郭迟

% 编写时间：2004. 1. 24

% 函数功能：本函数将完成对输入的 RGB 图像用小波分析的方法进行自动降噪，得到高频系数阈值，降噪效果百分比和结果

% 输入格式举例：[comimage, perf0, perf1, thr] = imagenr('c: \lenna. jpg', 'c: \lenna2. jpg', 'jpg', 'db6', 2)

```
% 参数说明:
% image 为输入的含噪声的 RGB 图像地址
% permission 为图像的文件类型
% addr 为处理后的图像存放的地址
% wavelet 为使用的小波类型
% level 为分解的尺度
% comimage 为降噪后的结果
% perf0 为高频系数置 0 的百分比
% perf1 为降噪后的能量百分比
% thr 为降噪量化选择的阈值
function [ comimage, perf0, perf1, thr] = imagecom( image, addr, permission, wavelet, level) ;
signal = imread( image, permission) ;
[ row, col] = size( signal) ;
signal2 = double( signal) /255;
signal2 = reshape( signal2, row, col) ;
% 对图像进行小波分解
[ C, S] = wavedec2( signal2, level, wavelet) ;
```

```
% 计算量化阈值
[ thr, sorh, keepapp] = ddencmp( 'den', 'wv', signal2) ;
% 压缩
[ comimage, cxc, lxc, perf0, perf1] = wdencmp( 'gbl', C, S, wavelet, level, thr, sorh, keepapp) ;
comimage = reshape( comimage, row, col/3, 3) ;
imwrite( comimage, addr, permission) ;
% 显示结果
subplot( 221) , imshow( image) ; title( '含噪声的原始图像') ;
comimage = imread( addr, permission) ;
subplot( 222) , imshow( comimage) ; title( '降噪后的图像') ;
disp( '高频系数置 0 的百分比:') ; perf0
disp( '压缩后的能量百分比:') ; perf1
```

实验的结果如图 3.29 所示。

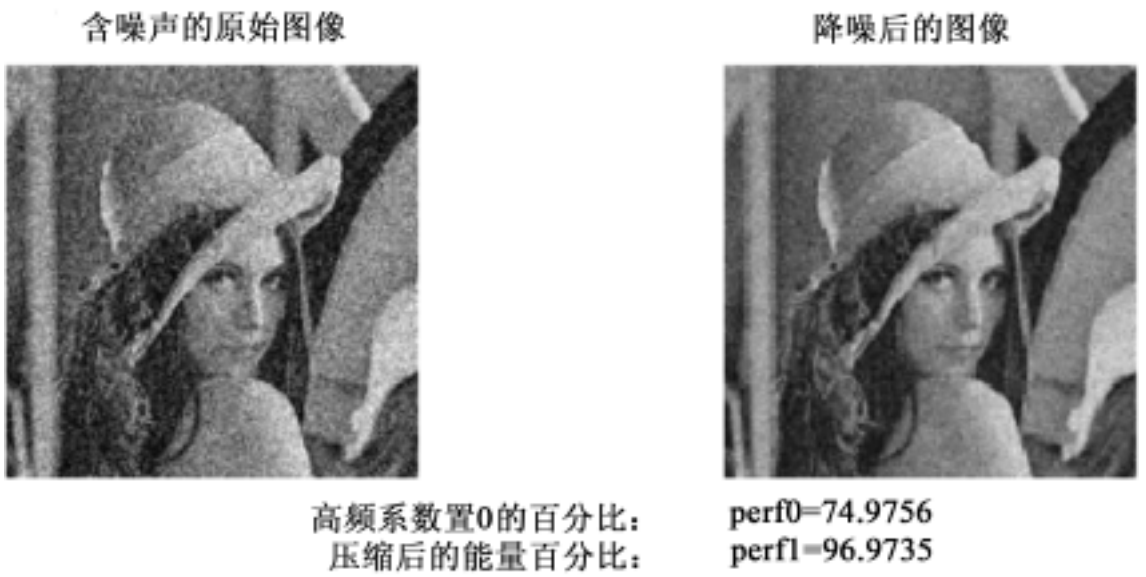


图 3.29　lenna 图像 1 层分解降噪的结果

特别说明一下 MATLAB 中的两个函数。MATLAB 中的 ddencmp. m 函数就是用上述算法实现自动计算阈值的，同时该函数还返回两个参数 sorh 和 keepapp，分别表示阈值的类型(硬阈值和软阈值)和对于等于该值的高频系数是否也应去掉(keepapp = 1 为去掉)。

MATLAB 中的 wdencmp. m 函数为压缩函数，返回的 perf0 与 perf1 分别为根据 cxc 和 lxc 计算出的高频系数置 0 的百分比和压缩后的能量百分比。

若要完成图像压缩，只需将上面程序中 ddencmp 函数的参数“den”改为“cmp”就可以了。

4) 小波对图像进行融合

首先必须明确的一点是,这里的图像融合是可见融合,也就是利用小波分析的方法将两个图像合并在一起,当然两个图像的基本信息都是清晰可见的。如果想要将一个图像不可见地融合到另一图像中,实际上就是图像隐藏了,我们会在后面的章节详细论述。

编写函数 imagecbe. m 完成实验,函数代码如下:

```
% 文件名: imagecbe. m
% 程序员: 郭迟
% 编写时间: 2004. 1. 24
% 函数功能: 本函数将完成对输入的两幅 RGB 图像用小波分析的方法进行图像融合,要求第一幅图像的大小大于等于第二幅图像
% 输入格式举例: cbeimage = imagecbe( 'c: \lenna. jpg', 'c: \cindy. jpg', 'jpg', db6')
% 参数说明:
% image1 为第一幅图像地址
% image2 为第二幅图像地址
% permission 为图像的文件类型
% wavelet 为使用的小波类型
% cbeimage 为融合的图像
function cbeimage = imagecbe( image1, image2, permission, wavelet) ;
% 读取两幅待融合的图像
im1 = imread( image1, permission) ;
im2 = imread( image2, permission) ;
im1 = double( im1) /255;
im2 = double( im2) /255;
[ row1, col1] = size( im1) ;
[ row2, col2] = size( im2) ;
a = reshape( im1, row1, col1) ;
b = reshape( im2, row2, col2) ;
% 对图像进行 1 尺度下的小波分解,提取频率系数
[ C1, S1] = wavedec2( a, 1, wavelet) ;
[ C2, S2] = wavedec2( b, 1, wavelet) ;
size1 = size( C1) ;
size2 = size( C2) ;
% 对图像进行系数调整
for i = 1: size1( 2)
    C1( i) = 1. 2* C1( i) ; % 这里的 1. 2 和后面的 0. 8 可自行调整。
```

```
end
for i = 1: size2(2)
        C2(i) = 0.8 * C2(i);
end
% 对图像进行融合
C = 0.5 * (C1 + C2);
% 对图像进行重构
cbeimage = waverec2(C, S1, wavelet);
cbeimage = reshape(cbeimage, row1, col1/3, 3);
cbeimage = uint8(cbeimage * 255);
% 显示结果
subplot(131), imshow(im1); title('第一幅图像');
subplot(132), imshow(im2); title('第二幅图像');
subplot(133), imshow(cbeimage); title('融合图像');
```

实验结果如图 3.30 所示。

图 3.30　lenna 图像与 Wbarb 图像融合的结果

以上是我们从实验的角度完成的对一维和二维信号的小波分析。同时，通过三个具体的应用示例进一步说明了小波分析在图像处理上的重要性。之所以选择这三个例子，是因为它们与我们要谈到的信息隐藏有很密切的联系。对这里的有些实验稍加改变就可以达到信息隐藏的目的；对有些实验加深认识则可以在后面的数字水印实验中取得很好的实验效果。

3.3.4　常用的小波函数族

前面的实验我们已经使用了 db1 和 db6 两类小波。之所以称之为“两类”而不是“两个”是因为这里的 db1 或 db6 不仅包括了小波母函数(小波函数)，还包括了尺度函数和相应的满足精确重构条件的滤波器等。事实上，db1 和 db6 小波都是属于 Daubechies 小波系，是由比利时学者 Daubechies 创建的。

谈到小波系列,我们首先解释一下几个相关的概念。

① 支集长度。小波是指在很小的有限区间内不为0的波,那么这个很小的有限区间就称为小波母函数的支集长度。小波的支集长度越短,其对信号的局部表征能力就越强。

② 小波的消失矩阶数。小波母函数 $\varphi(t)$ 的k阶矩是指 $m = \int_{-\infty}^{\infty} \varphi(t)\, t^k dt$。所谓消失矩阶数就是指使m为0的k的取值。高阶消失矩意味着被处理信号的平滑部分的低频小波系数大多为0,从而更好地获得信号的细节特征。所以,消失矩的实际意义是将信号能量相对集中在少数的小波系数中。然而小波有k阶消失矩,则其支集长度至少为2k - 1。支集长度越高,计算复杂度也随之增高。

③ 正则性。小波的正则性主要对因阈值变化或小波系数的量化而引起的误差产生表面化的影响。如果小波是正则的,则引起的误差是光滑的,不易被察觉。

④ 正交性。用正交小波基进行的重构代价是最小的。正交小波基比较难构造,所以人们往往使用近似正交小波(双正交小波)。对于一个小波 $\varphi(t)$,如果它满足:

$$< \varphi_{j,k}, \varphi_{a,b} > = 0, j \neq a, j, k, a, b \in Z$$

则就是一个双正交小波。双正交小波是用两类小波集合完成小波分析任务的。一类(包括小波函数和尺度函数)用于信号分解,一类用于信号重构。可以看到,双正交小波设计的核心是平移,比较容易实现。

⑤ 对称性。小波函数的对称性和正交性往往是一对矛盾。一个具有对称性的小波首先给人以主观上的美感。同时,对于图像小波分析,对称小波能尽可能地避免图像的边界失真问题,降低量化误差。

下面,我们就从以上5个方面来认识一下在工程应用上常见的几类小波系列。

(1) Daubechies 小波

Daubechies 小波在 MATLAB 中记为 dbN。N = 2, 3, …, 10, 是小波序号。Daubechies 小波的小波母函数和尺度函数的支集长度为2N - 1。小波的消失矩阶数为N。Daubechies 小波是正交小波,具有近似的对称性。图3.31给出了db4和db8小波的有关信息。

(2) Symlets 小波

Symlets 小波与 Daubechies 小波非常相似。在 MATLAB 中记为 symN。N = 2, 3, …是小波序号。Symlets 小波的小波母函数和尺度函数的支集长度为2N - 1,滤波器长度为2N。小波的消失矩阶数为N。Symlets 小波是双正交小波,具有近似的对称性。图3.32给出了sym4和sym8小波的有关信息。

(3) Biorthogonal 小波

Biorthogonal 小波是具有线性相位的双正交小波,一般成对出现。在 MATLAB 中通常表示为 bioNr. Nd 的形式。Nr 为重构小波,Nd 为分解小波,其对应形式如下:

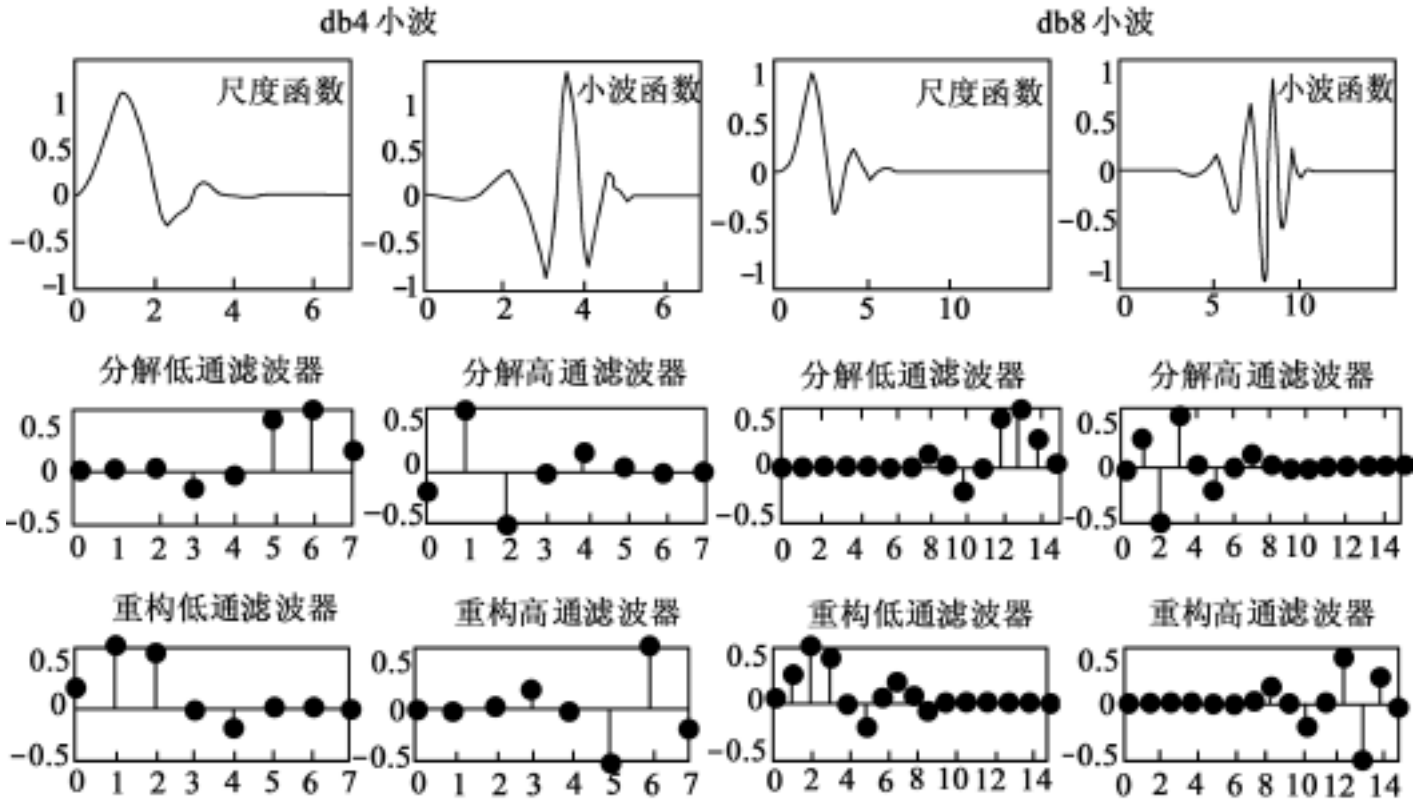


图 3.31　Daubechies 小波

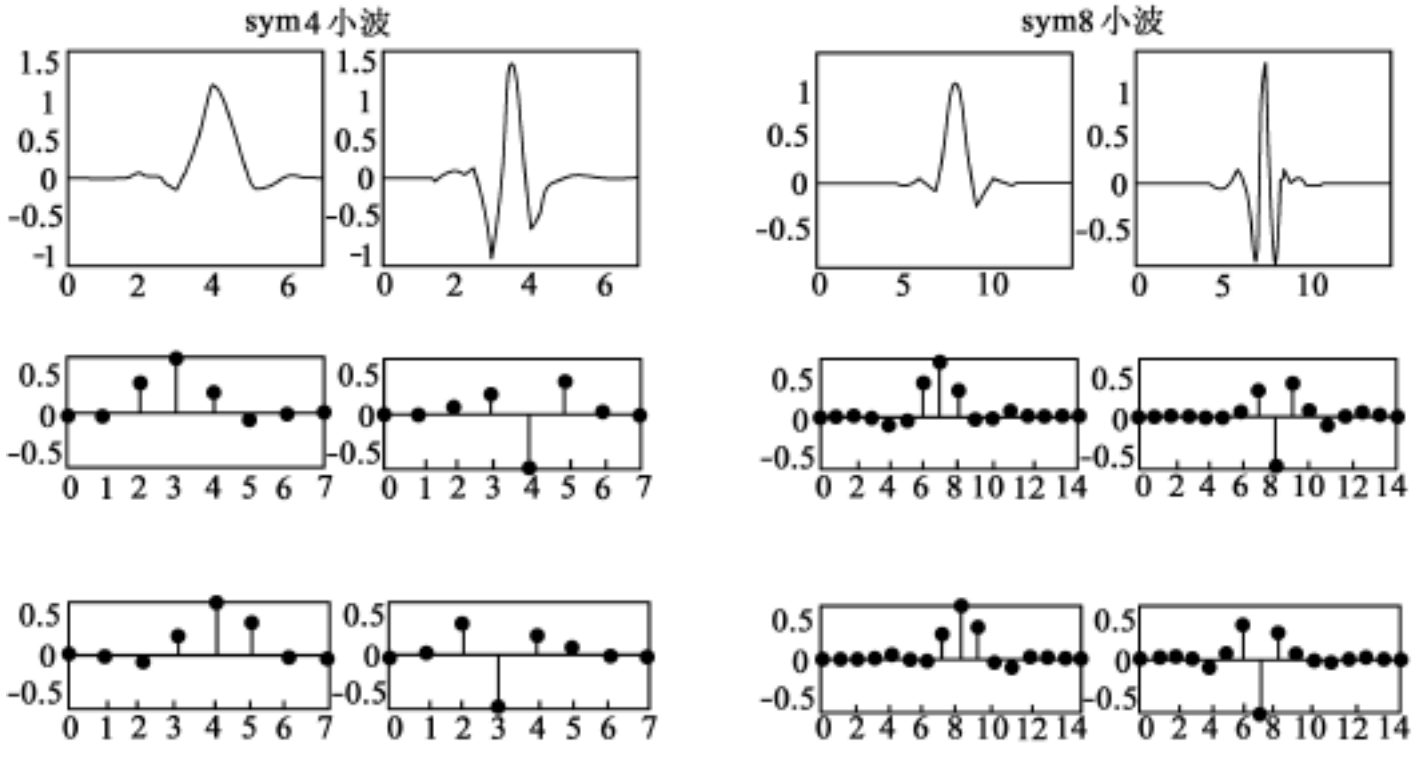


图 3.32　Symlets 小波

Nr	1	2	3	4	5	6
Nd	1, 3, 5	2, 4, 6, 8	1, 3, 5, 7, 9	4	5	8

图 3.33 给出了 bio2.4 和 bio4.4 小波的有关信息。

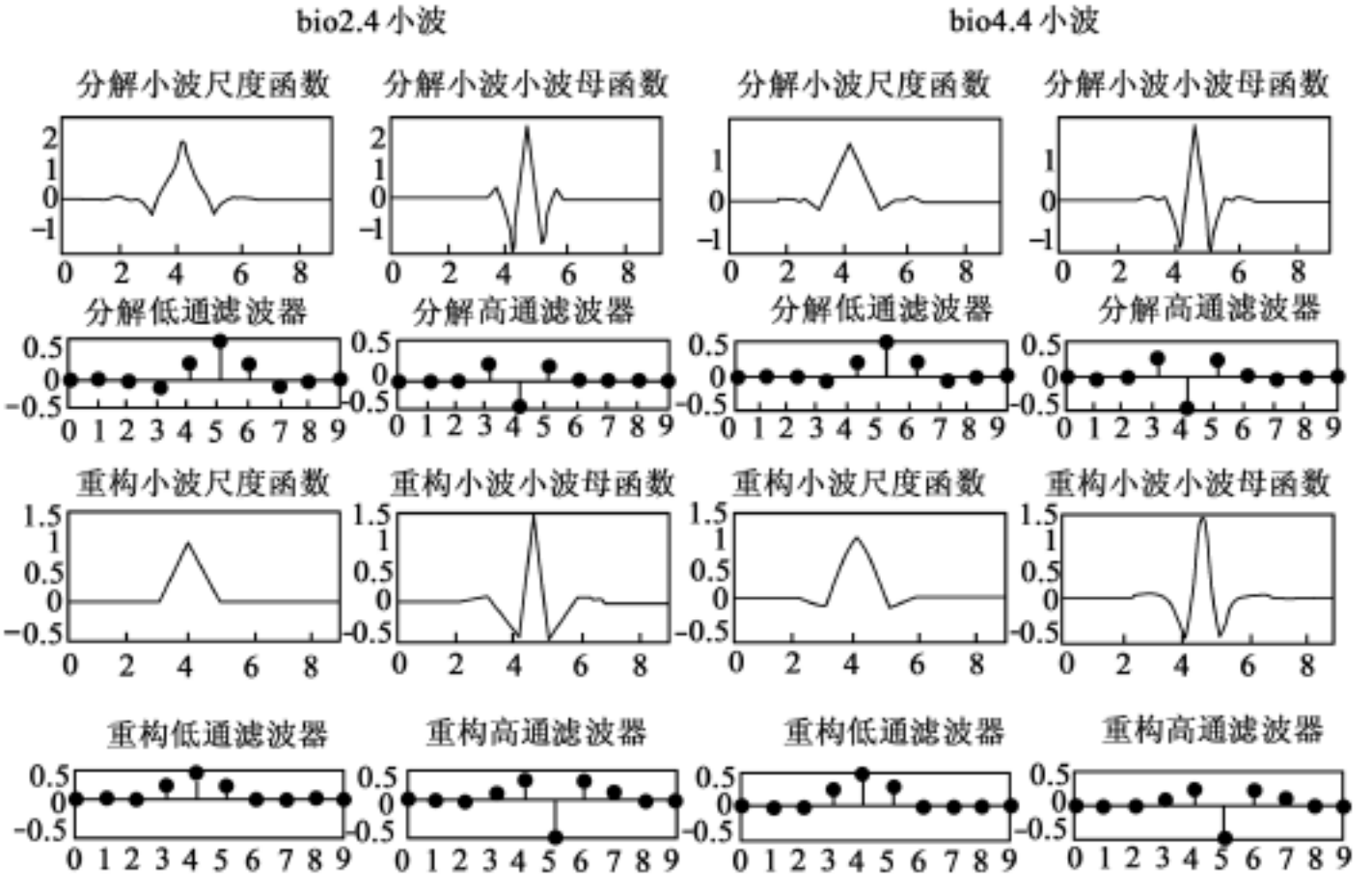


图 3.33　Biorthogonal 小波

前面已经说过，在信息隐藏领域中小波分析的主要任务应该是选择合适的小波基去适应各种 DWT 域的隐藏算法。这是因为对于同样的隐藏算法，选用不同的小波产生的数字水印或隐藏结果，其性能有着很大的差异。同时，选用同一小波的不同尺度下的频率系数作为隐藏载体，得到的结果其性能也是不同的。图 3.34 是我们用 W-SVD 算法产生的数字水印在抗 JPEG 压缩方面的性能曲线图。左侧使用的是 db1

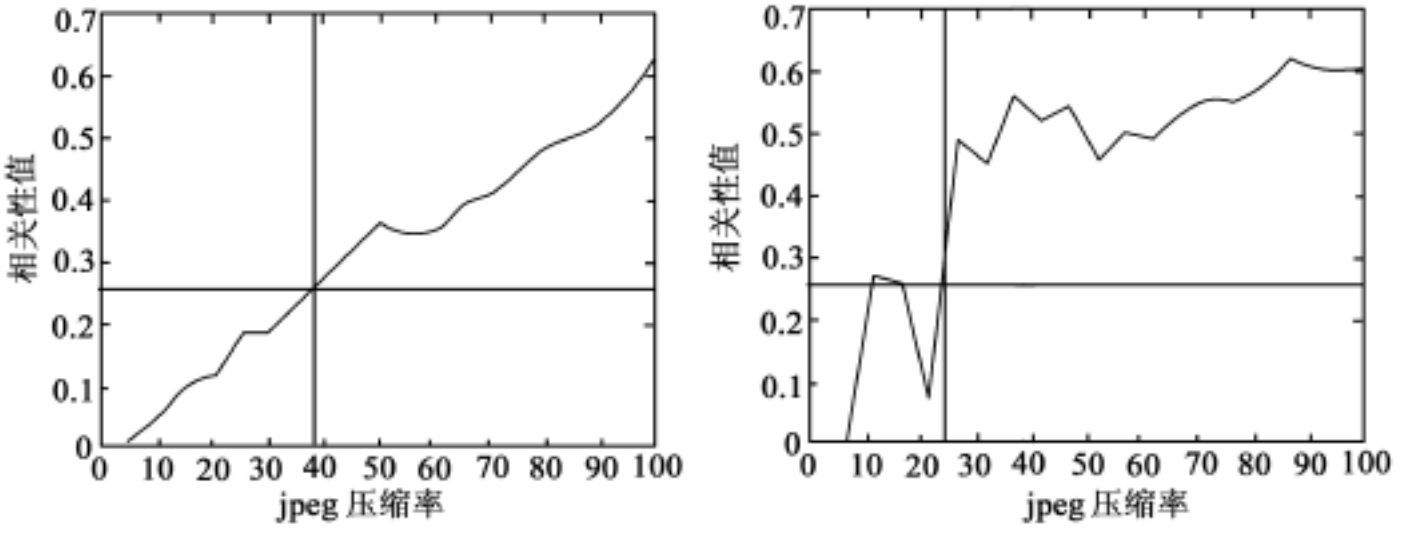


图 3.34　不同小波在信息隐藏算法上的差异

小波，右侧使用的是 db6 小波，取一个同样的足够大的检测阈值，可以发现其性能上的明显差异。

小波的选择是一个非常复杂的问题。它不仅与小波本身的性能有关，还涉及到隐藏算法、性能评定的标准等几个方面。在一般的实际运用中，最好的方法就是多实验、多比较。当然，除了实验的方法外，从理论上也是能够得到一些适合信息隐藏的小波的一般描述的。