数学建模算法与应用

第13章 数字图像处理

数字图像处理是一门迅速发展的新兴学科，发展的历史并不长。由于图像是视觉的基础，而视觉又是人类重要的感知手段，故数字图像成为心理学、生理学、计算机科学等诸多方面学者研究视觉感知的有效工具。随着计算机的发展，以及应用领域的不断加深和扩展，数字图像处理技术已取得长足的进展，出现了许多有关的新理论、新方法、新算法、新手段和新设备，并在军事公安、航空、航天、遥感、医学、通信、自控、天气预报以及教育、娱乐、管理等方面得到广泛的应用。

所以，数字图像处理是一门实用的学科，已成为电子信息、计算机科学及其相关专业的一个热门研究课题，相应的图像处理技术也是一门重要的课程，是一门多学科交叉、理论性和实践性都很强的综合性课程。

数字图像处理是计算机和电子学科的重要组成部分，是模式识别和人工智能理论的的中心研究内容。数字图像处理的内容包括：数字图像处理的基本概念，数字图像显示，点运算，代数运算和几何运算等概念。二维傅立叶变换、离散余弦变换、离散图像变换的基本原理与方法。图像的增强方法，包括空间域方法和变换域方法。图像恢复和重建基本原理与方法。图像压缩编码的基本原理等内容。

13.1　数字图像概述

13.1.1　图像的基本概念

图像因其表现方式的不同分为连续图像和离散图像两大类。

连续图像：是指在二维坐标系中具有连续变化的图像，即图像的像点是无限稠密的，同时具有灰度值（即图像从暗到亮的变化值）。连续图像的典型代表是由光学透镜系统获取的图像，如人物照片和景物照片等，有时又称之为模拟图像。

离散图像：是指用一个数字序列表示的图像。该阵列中的每个元素是数字图像的一个最小单位，称为像素。像素是组成图像的基本元素，是按照某种规律编成系列二进制数码（0 和 1）来表示图像上每个点的信息。因此，又称之为数字图像。

以一个我们身边的简单例子来说，用胶片记录下来的照片就是连续图像，而用数码相机拍摄下来的图像是离散图像。

13.1.2　图像的数字化采样

由于目前的计算机只能处理数字信号，我们得到的照片、图纸等原始信息都是连续的模拟信号，必须将连续的图像信息转化为数字形式。我们可以把图像看作是一个连续变化的函数，图像上各点的灰度是所在位置的函数，这就要经过数字化的采样与量化。下面简单介绍图像数字化采样的方法。

图像采样就是按照图像空间的坐标测量该位置上像素的灰度值。方法如下：对连续图像$f(x,y)$进行等间隔采样，在(x,y)平面上，将图像分成均匀的小网格，每个小网格的位置可以用整数坐标表示，于是采样值就对应了这个位置上网格的灰度值。若采样结果每行像素为M个，每列像素为N个，则整幅图像就对应于一个$M \times N$数字矩阵。

这样我们就获得了数字图像中关于像素的两个属性：位置和灰度。位置由采样点的两个坐标确定，也就对应了网格行和列；而灰度就表明了该像素的明暗程度。

把模拟图像在空间上离散化为像素后，各个像素点的灰度值仍是连续量，接着我们就需要把像素的灰度值进行量化，把每个像素的光强度进行数字化，也就是将 $f(x,y)$ 的值划分成若干个灰度等级。

一幅图像经过采样和量化后便可以得到一幅数字图像。通常可以用一个矩阵来表示。

$$f(x,y)=\begin{bmatrix} f(0,0) & f(0,1) & \cdots & f(0,N-1) \\ f(1,0) & f(1,1) & \cdots & f(1,N-1) \\ \vdots & \vdots & & \vdots \\ f(M-1,0) & f(M-1,1) & \cdots & f(M-1,N-1) \end{bmatrix}$$

图 13.1 数字图像的矩阵表示

一幅数字图像在Matlab中可以很自然地表示成矩阵

$$g=\begin{bmatrix} g(1,1) & g(1,2) & \cdots & g(1,N) \\ g(2,1) & g(2,2) & \cdots & g(2,N) \\ \vdots & \vdots & & \vdots \\ g(M,1) & g(M,2) & \cdots & g(M,N) \end{bmatrix}$$

其中 $g(x+1,y+1)=f(x,y)$ ， $x=0,\cdots,M-1$ ， $y=0,\cdots,N-1$。

矩阵中的元素称作像素。每一个像素都有x和y两个坐标，表示其在图像中的位置。另外还有一个值，称灰度值，对应于原始模拟图像在该点处的亮度。量化后的灰度值，代表了相应的色彩浓淡程度，以 256 色灰度等级的数字图像为例，一般由 8 位，即一个字节表示灰度值，由 0-255 对应于由黑到白的颜色变化。对只有黑白二值采用一个比特表示的特定二值图像，就可以用 0 和 1 来表示黑白二色。

将连续灰度值量化为对应灰度级的具体量化方法有两类，即等间隔量化与非等间隔量化。根据一幅图像具体的灰度值分布的概率密度函数来进行量化，但是由于灰度值分布的概率函数因图而异，不可能找到一个普遍适用于各种不同图像的最佳非等间隔量化公式，因此，在实际应用中一般都采用等间隔量化来进行量化的。

13.1.3　数据类

虽然我们处理的是整数坐标，但 Matlab 中的像素值本身并不是整数。表 13.1 中列出了 Matlab 和图像处理工具箱为表示像素值所支持的各种数据类。表中的前 8 项称为数值数据类，第 9 项称为字符类，最后一项称为逻辑数据类。

表 13.1　数据类

名称	描述
double	双精度浮点数，范围为[-10^{308},10^{308}]
uint8	无符号 8 比特整数，范围为[0,255]
uint16	无符号 16 比特整数，范围为[0,65535]
uint32	无符号 32 比特整数，范围为[0,4294967295]
int8	有符号 8 比特整数，范围为[-128,127]
int16	有符号 16 比特整数，范围为[-32768,32767]
int32	有符号 32 比特整数，范围为[-2147483648,2147483647]
single	单精度浮点数，范围为[-10^{38},10^{38}]
char	字符
logical	值为 0 或 1

13.1.4　图像类型

在计算机中，按照颜色和灰度的多少可以将图像分为二值图像、灰度图像、索引图像和真彩色 RGB 图像四种基本类型。目前，大多数图像处理软件都支持这四种类型的图像。

1. 二值图像

一幅二值图像的二维矩阵仅由 0、1 两个值构成，“0”代表黑色，“1”代表白色。由于每一像素（矩阵中每一元素）取值仅有 0、1 两种可能，所以计算机中二值图像的数据类型通常为 1 个二进制位。二值图像通常用于文字、线条图的扫描识别（OCR）和掩膜图像的存储。

二值图像在 Matlab 中是一个取值只有 0 和 1 的逻辑数组。因而，一个取值只有 0 和 1 的 uint8 类数组，在 Matlab 中并不认为是二值图像。使用 logical 函数可以把数值数组转换为二值数组。因此，若 A 是一个由 0 和 1 构成的数值数组，则可使用如下语句创建一个逻辑数组 B

B=logical(A)，

若 A 中含有除了 0 和 1 之外的其它元素，则使用 logical 函数就可以将所有非零的量变换为逻辑 1，而将所有的 0 值变换为逻辑 0。

2. 灰度图像

灰度图像矩阵元素的整数取值范围通常为[0，255]。因此其数据类型一般为 8 位无符号整数（int8），这就是人们经常提到的 256 灰度图像。“0”表示纯黑色，“255”表示纯白色，中间的整数数字从小到大表示由黑到白的过渡色。若灰度图像的像素是 uint16 类，则它的整数取值范围为[0，65535]。若图像是 double 类，则像素的取值就是浮点数。规定双精度型归一化灰度图像的取值范围是[0，1]，0 代表黑色，1 代表白色，0 到 1 之间的小数表示不同的灰度等级。二值图像可以看成是灰度图像的一个特例。

3. RGB 彩色图像

一幅 RGB 图像就是彩色像素的一个$m \times n \times 3$数组，其中每一个彩色像素点都是在特定空间位置的彩色图像相对应的红、绿、蓝三个分量。RGB 也可以看成是一个由三幅灰度图像形成的“堆”，当将其送到彩色监视器的红、绿、蓝输入端时，便在屏幕上产生了一副彩色图像。按照惯例，形成一副 RGB 彩色图像的三个图像常称为红、绿或蓝分量图像。

分量图像的数据类决定了它们的取值范围。若一幅 RGB 图像的数据类是 double，则它的取值范围就是[0,1]，类似地，uint8 类或 uint16 类 RGB 图像的取值范围分别是[0,255]或[0,65535]。

4. 索引图像

索引图像有两个分量，即数据矩阵 X 和彩色映射矩阵 map。矩阵 map 是一个大小为$m \times 3$且由范围在[0,1]之间的浮点值构成的 double 类数组。map 的长度m同它所定义的颜色数目相等。map 的每一行都定义单色的红、绿、蓝三个分量。索引图像将像素的亮度值“直接映射”到彩色值。

每个像素的颜色由对应矩阵 X 的值作为指向 map 的一个指针决定。若 X 属 double 类，则其小于或等于 1 的所有分量都指向 map 的第 1 行，所有大于 1 且小于等于 2 的分量都指向第 2 行,依次类推。若 X 为 uint8 类或 uint16 类图像，则所有等于零的分量都指向 map 的第 1 行，所有等于 1 的分量都指向第 2 行，依次类推。

13.1.5　数据类与图像类型间的转换

在 Matlab 图像处理工具箱中，数据类与图像类型间的转换是非常频繁的。工具箱中提供了数据类之间进行转换的函数见表 13.2。

表 13.2　数据类之间的转换函数

名称	将输入转换为	有效的输入图像数据类
im2uint8	uint8	logical，uint8，uint16 和 double
im2uint16	uint16	logical，uint8，uint16 和 double
mat2gray	double，范围为[0,1]	double
im2double	double	logical，uint8，uint16 和 double
im2bw	logical	uint8，uint16 和 double

图像类型之间的转换函数有 ind2gray，gray2ind；rgb2ind，ind2rgb；ntsc2rgb，rgb2ntsc 等。可以使用 imtool 命令看一个图像文件的信息。

13.2　亮度变换与空间滤波

这里的空间指的是图像平面本身，在空间域（简称）内处理图像的方法是直接对图像的像素进行处理。当处理单色（灰度）图像时，亮度和灰度这两个术语是可以相互换用的。当处理彩色图像时，亮度用来表示某个彩色空间中的一个彩色图像分量。

本节讨论的空域处理由表达式

$$g(x,y)=T[f(x,y)] \tag{13.1}$$

表示，其中$f(x,y)$为输入图像，$g(x,y)$为输出（处理后）图像，T是对图像f进行处理的操作符，定义在点(x,y)的指定邻域内。此外，T还可以对一组图像进行处理，例如为降低噪声而让K幅图像相加。

Matlab 中函数 imadjust 是对图像进行亮度变换的工具。其语法为

g=imadjust(f,[low_in;high_in],[low_out;high_out],gamma)

此函数将图像 f 中的亮度值映射到 g 中的新值，即将 low_in 至 high_in 之间的值映射到 low_out 至 high_out 之间的值。参数 gamma 为调节权重，若 gamma（或分量，彩色图片 gamma 为三维行向量）小于 1，则映射被加权至更高（更亮）的输出值；若 gamma（或分量）大于 1，则映射被加权至更低（更暗）的输出值。

例 13.1　图像翻转

```
f=imread('tu1.bmp');    %读原图像
g=imadjust(f,[0; 1],[1; 0]); %进行图像翻转
subplot(1,2,1), imshow(f) %显示原图像
subplot(1,2,2), imshow(g) %显示翻转图像
```

图 13.2 中同时显示了原图像和反转图像，可以作比较。

图 13.2　原图像与翻转图像

13.2.1　线性空间滤波器

下面讨论线性滤波技术。使用拉普拉斯滤波器增强图像的基本公式为

$$g(x,y)=f(x,y)+c\nabla^2 f(x,y),$$

其中$f(x,y)$为输入的退化图像，$g(x,y)$为输出的增强图像，c取 1 或-1（具体选择见下文）。

拉普拉斯算子定义为

$$\nabla^2 f(x,y) = \frac{\partial^2 f(x,y)}{\partial x^2} + \frac{\partial^2 f(x,y)}{\partial y^2},$$

对于离散的数字图像，二阶导数用如下的近似

$$\frac{\partial^2 f(x,y)}{\partial x^2} = f(x+1,y) + f(x-1,y) - 2f(x,y),$$

$$\frac{\partial^2 f(x,y)}{\partial y^2} = f(x,y+1) + f(x,y-1) - 2f(x,y).$$

因而有

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y). \tag{13.2}$$

从（13.2）式易见，拉普拉斯算子∇^2对图像f的作用就相当于如下矩阵T_1与f相乘，

$$T_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{13.3}$$

我们称T_1为滤波器、掩膜、滤波掩膜、核、模板或窗口。

还可以用如下的矩阵T_2近似拉普拉斯算子，

$$T_2 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}, \qquad (13.4)$$

该矩阵更逼近二阶导数，因此对图像的改善作用更好。上文中c的选取依赖于形如（13.3），（13.4）式的近似矩阵。当这些近似矩阵中心元素（如（13.3）中的-4，（13.4）中的-8）为负时$c = -1$，反之$c = 1$。

可以选取其它的拉普拉斯近似矩阵。Matlab 中函数 fspecial('laplacian', α)实现一个更为常见的拉普拉斯算子掩膜

$$\begin{bmatrix} \frac{\alpha}{1+\alpha} & \frac{1-\alpha}{1+\alpha} & \frac{\alpha}{1+\alpha} \\ \frac{1-\alpha}{1+\alpha} & \frac{-4}{1+\alpha} & \frac{1-\alpha}{1+\alpha} \\ \frac{\alpha}{1+\alpha} & \frac{1-\alpha}{1+\alpha} & \frac{\alpha}{1+\alpha} \end{bmatrix}. \tag{13.5}$$

13.2.2　图像恢复实例

利用Matlab中的图像处理工具箱可以方便地进行图像修复。

例 13.2　模糊图像修复

```
f=imread('tu2.bmp'); %读取原图像
h1=fspecial('laplacian',0); %式（13.3）的滤波器，等价于式（13.5）中参数为 0
g1=f-imfilter(f,h1); %中心为-4，c=-1,即从原图像中减去拉普拉斯算子处理的结果
h2=[1 1 1; 1 -8 1; 1 1 1]; %式（13.4）的滤波器
g2=f-imfilter(f,h2); %中心为-8，c=-1
subplot(1,3,1),imshow(f) %显示原图像
subplot(1,3,2),imshow(g1) %显示滤波器(13.3)修复的图像
subplot(1,3,3),imshow(g2) %显示滤波器(13.4)修复的图像
```

线性拉普拉斯滤波器对模糊图像具有很好的修复效果。图 13.3 给出了原始图像及利用滤波器（13.3）和（13.4）修复的图像。



原图像　　（13.3）的滤波图像　（13.4）的滤波图像

图 13.3　原图像及滤波效果图像

13.2.3 非线性空间滤波器

Matlab 中非线性空间滤波的一个工具是函数ordfilt2，它可以生成统计排序滤波器。其响应是基于对图像邻域中所包含的像素进行排序，然后使用排序结果确定的值来替代邻域中的中心像素的值。函数ordfilt2 的语法为

g=ordfilt2(f, order, domain)，

该函数生成输出图像 g 的方式如下：使用邻域的一组排序元素中的第 order 个元素来替代 f 中的每个元素，而该邻域则由 domain 中的非零元素指定。

统计学术语中，最小滤波器（一组排序元素中的第一个样本值），称为第 0 个百分位，它可以使用语法

```
g=ordfilt2(f,1,ones(m,n))
```

来实现。同样，第 100 个百分位指的就是一组排序元素中的最后一个样本值，即第mn个样本，它可以使用语法

```
g=ordilt2(f,m*n,ones(m,n))
```

实现。

数字图像处理中最著名的统计排序滤波器是中值滤波器，它对应的是第 50 个百分位。可以使用

g=ordfilt2(f,median(1:m*n),ones(m,n))

来创建中值滤波器。基于实际应用的重要性，工具箱提供了一个二维中值滤波函数

g=medfilt2(f,[m,n])，

数组[m,n]定义一个大小为 m×n 的邻域，中值就在该邻域上计算。该函数的默认形式为

g=medfilts(f)，

它使用一个大小为 3×3 的邻域来计算中值，并用 0 来填充输入图像的边界。

例 13.3　中值滤波

```
f=imread('Lena.bmp'); %读原图像
f1=imnoise(f,'salt & pepper',0.02); %加椒盐噪声
g=medfilt2(f1); %进行中值滤波
subplot(1,3,1),imshow(f),title('原图像')
subplot(1,3,2),imshow(f1),title('被椒盐噪声污染的图像')
subplot(1,3,3),imshow(g),title('中值滤波图像')
```

图 13.3 给出了原图像，被椒盐噪声污染的图像及滤波后的图像。

（a）原图像（b）被椒盐噪声污染的图像（c）中值滤波图像

图 13.4　中值滤波对比图

13.3 频域变换

为了快速有效地对图像进行处理和分析，如进行图像增强、图像分析、图像复原、图像压缩等，常常需要将原定义在图像空间的图像以某种形式转换到频域空间，并利用频域空间的特有性质方便地进行一定的加工，最后再转换回图像空间，以得到所需要的效果。

13.3.1　傅立叶变换

傅立叶变换是对线性系统进行分析的一个有力工具，它将图像从空域变换到频域，使我们能够把傅立叶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。

1　二维连续傅立叶变换

二维傅立叶变换的定义如下

$$F(u,v)=\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty} f(x,y)e^{-jux}e^{-jvy}dxdy, \qquad (13.6)$$

其中，j为虚数单位，u和v是频率变量，其单位是弧度/采样单位；$F(u,v)$通常称为$f(x,y)$的频率表示。$F(u,v)$是复值函数，在u和v上都是周期的，且周期为2π。因为其具有周期性，通常只显示$-\pi \le u,v \le \pi$的范围。

注意$F(0,0)$是$f(x,y)$的所有值之和，因此，$F(0,0)$通常称为傅立叶变换的恒定分量或 DC 分量（DC 表示直流）。如果$f(x,y)$是一幅图像，则$F(u,v)$是它的谱。

定义二维傅立叶变换的频谱、相位谱和功率谱（频谱密度）如下

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

$$\Phi(u,v) = \arctan[I(u,v) / R(u,v)],$$

$$P(u,v) = |F(u,v)|^2 = R^2(u,v) + I^2(u,v),$$

其中$R(u,v)$和$I(u,v)$分别为$F(u,v)$的实部和虚部。

二维傅立叶逆变换定义如下

$$f(x,y)=\frac{1}{4\pi^2}\int_{-\infty}^{+\infty}\int_{-\infty}^{+\infty}F(u,v)e^{jux}e^{jvy}dudv. \quad (13.7)$$

在 Matlab 工具箱中（可以参看 Matlab 图像工具箱的 pdf 帮助文件），由于对象是离散函数，二维傅立叶变换定义为

$$F(u,v)=\sum_{x=-\infty}^{\infty}\sum_{y=-\infty}^{\infty}f(x,y)e^{-jux}e^{-jvy},$$

逆变换定义为

$$f(x,y)=\frac{1}{4\pi^2}\int_{-\pi}^{+\pi}\int_{-\pi}^{+\pi}F(u,v)e^{jux}e^{jvy}dudv,$$

也记作 $f(x,y)=F^{-1}(F(u,v))$。

2　二维离散傅立叶变换（DFT）

令 $f(x,y)$ 表示一幅大小为 $M\times N$ 的图像，其中 $x=0,1,\cdots,M-1$ 和 $y=0,1,\cdots,N-1$，$f(x,y)$ 的二维离散傅立叶变换定义如下

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)e^{-j2\pi(ux/M+vy/N)},\qquad(13.8)$$

其中 $u=0,1,\cdots,M-1$ 和 $v=0,1,\cdots,N-1$，u 和 v 用作频率变量，x 和 y 用作空间变量。由 $u=0,1,\cdots,M-1$ 和 $v=0,1,\cdots,N-1$ 定义的 $M\times N$ 矩形区域常称为频率矩形。显然，频率矩形的大小与输入图像的大小相同。

二维离散傅立叶逆变换由下式给出

$$f(x,y)=\frac{1}{MN}\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)e^{j2\pi(ux/M+vy/N)}，(13.9)$$

其中$x=0,1,\cdots,M-1$和$y=0,1,\cdots,N-1$。因此，给定$F(u,v)$，我们可以借助于 DFT 逆变换得到$f(x,y)$。在这个等式中，$F(u,v)$的值有时称为傅立叶系数。

$F(0,0)$称为傅立叶变换的直流（DC）分量，不难看出，$F(0,0)$等于$f(x,y)$的平均值的MN倍。$F(u,v)$满足

$$F(u,v)=F(u+M,v)=F(u,v+N)=F(u+M,v+N),$$

即 DFT 在u和v方向上都是周期的，周期由M和N决定。周期性也是 DFT 逆变换的一个重要属性

$$f(x,y)=f(x+M,y)=f(x,y+N)=f(x+M,y+N)$$

我们可以简单地认为这是 DFT 及其逆变换的一个数学特性。还应牢记的是，DFT 实现仅计算一个周期。

3　基于离散傅立叶变换的频域滤波

在频域中滤波首先计算输入图像的傅立叶变换$F(u,v)$，然后用滤波器$H(u,v)$对$F(u,v)$作变换，最后对其得到的变换结果作逆傅立叶变换就得到频域滤波后的图像。具体到离散情况，主要包括以下 5 个步骤

（1）用$(-1)^{x+y}$乘以输入图像进行中心变换得到

$$f_c(x,y)=(-1)^{x+y}\,f(x,y).$$

（2）计算图像$f_c(x,y)$的离散傅立叶变换

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f_c(x,y)e^{-j2\pi(ux/M+vy/N)}.$$

（3）用滤波器$H(u,v)$作用$F(u,v)$，得到

$$G(u,v)=H(u,v)F(u,v)。$$

（4）计算$G(u,v)$的离散傅立叶逆变换，并取实部，得到

$$g_p(x,y)=\text{real}\{F^{-1}(G(u,v))\}.$$

（5）用$(-1)^{x+y}$乘以$g_p(x,y)$得到中心还原滤波图像

$$g(x,y)=(-1)^{x+y}g_p(x,y).$$

理想低通滤波器具有传递函数

$$H(u,v)=\begin{cases}1, & \text{若}D(u,v)\le D_0,\\ 0, & \text{若}D(u,v)>D_0,\end{cases}$$

其中D_0为指定的非负数，$D(u,v)$为点(u,v)到滤波器中心的距离。

$D(u,v) = D_0$的轨迹为一个圆。注意，若滤波器H乘以一幅图像的傅立叶变换，我们会发现理想滤波器切断（乘以 0）了圆外F的所有分量，而圆上和圆内的点不变（乘以 1）。虽然这个滤波器不能使用电子元件来模拟实现，但通常用来解释折叠误差等问题。

n阶巴特沃兹低通滤波器（在距离原点D_0处出现截止频率）的传递函数为

$$H(u,v)=\frac{1}{1+[D(u,v)/D_0]^{2n}},$$

与理想低通滤波器不同的是，巴特沃兹低通滤波器的传递函数并不是在D_0处突然不连续。对于具有平滑传递函数的滤波器，我们通常要定义一个截止频率，在该点处$H(u,v)$会降低为其最大值的某个给定比例。

高斯低通滤波器的传递函数为

$$H(u,v)=e^{-\frac{D^2(u,v)}{2\sigma^2}},$$

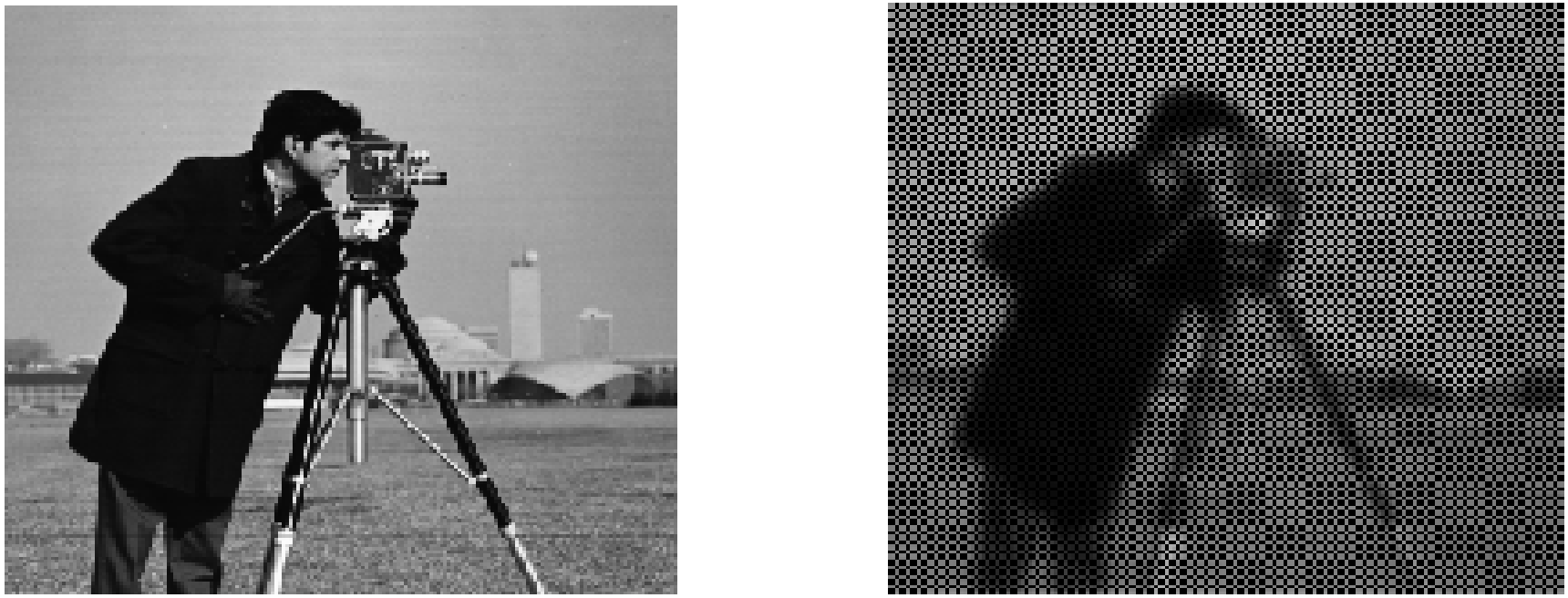
其中σ为标准差。通过令$\sigma=D_0$，我们可以根据截止参数D_0得到表达式

$$H(u,v)=e^{-\frac{D^2(u,v)}{2D_0^2}}.$$

下面我们给出一个低通滤波器的例子。

例 13.4　1 阶巴特沃兹低通滤波器

取截止频率$D_0 = 15$，原图像和 1 阶巴特沃茨滤波后的图像见图 13.5。



（a）原图像　　（b）低通滤波后的图像

图 13.5　原图像和低通滤波后的图像对比图

13.3.2　离散余弦变换

DCT（Discrete Cosine Transform）变换的全称是离散余弦变换，是图像处理中经常使用的变换算法。通过 DCT 变换可以将图像空间域上的信息变换到频率域上，它较好地利用了人类视觉系统的特点。

对于一个$M \times N$图像$f(x,y)$的二维 DCT 变换定义为

$$F(u,v)=\alpha(u)\beta(v)\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)\cos\frac{(2x+1)u\pi}{2M}\cos\frac{(2y+1)v\pi}{2N},$$

(13.10)

其逆变换为

$$f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\alpha(u)\beta(v)F(u,v)\cos\frac{(2x+1)u\pi}{2M}\cos\frac{(2y+1)v\pi}{2N},$$

(13.11)

其中，$x,u=0,1,\cdots,M-1$；$y,v=0,1,\cdots,N-1$，

$$\alpha(u)=\begin{cases}\dfrac{1}{\sqrt{M}}, & u=0,\\ \sqrt{\dfrac{2}{M}}, & u=1,\cdots,M-1,\end{cases}\qquad \beta(v)=\begin{cases}\dfrac{1}{\sqrt{N}}, & v=0,\\ \sqrt{\dfrac{2}{N}}, & v=1,\cdots,N-1,\end{cases}$$

令变换核函数

$$h(x,y,u,v)=\alpha(u)\beta(v)\cos\frac{(2x+1)u\pi}{2M}\cos\frac{(2y+1)v\pi}{2N},$$

则 DCT 变换公式又可写为

$$F(u,v)=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}f(x,y)h(x,y,u,v)\ ,$$

$$u=0,1,\cdots,M-1,\ \ v=0,1,\cdots,N-1,$$

DCT 逆逆变换公式写为

$$f(x,y)=\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}F(u,v)h(x,y,u,v)\ ,x=0,1,\cdots,M-1,$$

$$y=0,1,\cdots,N-1,$$

式中，u,v代表 DCT 变换矩阵内某个数值的坐标位置，$F(u,v)$代表 DCT 变换后矩阵内的某个值，x,y代表数据矩阵内某个数值的坐标位置，$f(x,y)$代表图像矩阵内的某个数据。

DCT 变换相当于将图像分解到一组不同的空间频率上，$\alpha(u)$和$\beta(v)$即为每一个对应的空间频率成分在原图像中所占的比重；而逆变换则是一个将这些不同空间频率上的分量合成为原图像的过程，变换系数$\alpha(u)$和$\beta(v)$在这个精确、完全的重构过程中规定了各频率成分所占分量的大小。

DCT 变换的实现有两种方法，一种是基于快速傅立叶变换（FFT）的算法，这是通过工具箱提供的 dct2 函数实现的；另一种是 DCT 变换矩阵（transform matrix）方法。变换矩阵方法非常适合做8×8或16×16的图像块的 DCT 变换，工具箱提供了 dctmtx 函数来计算变换矩阵。一个$M\times M$的 DCT 变换矩阵$T=(T_{pq})_{M\times M}$定义为

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

对于一个$M \times M$的矩阵A，TA是一个$M \times M$矩阵，它的每一列是矩阵A的对应列的一维 DCT 变换，A的二维 DCT 变换可以由$B = TAT^T$计算；由于T是实正交矩阵，它的逆阵等于它的转置矩阵，B的二维逆 DCT 变换可以由$T^T BT$给出。

由式（13.11）可知，原始图像 f 可表示为一系列函数

$$\sum_{u=0}^{M-1}\sum_{v=0}^{N-1}\alpha(u)\beta(v)\cos\frac{(2x+1)u\pi}{2M}\cos\frac{(2y+1)v\pi}{2N},$$

其中 $x=0,\cdots,M-1$，$y=0,\cdots,N-1$ 的加权组合，这组函数就是 DCT 基函数。

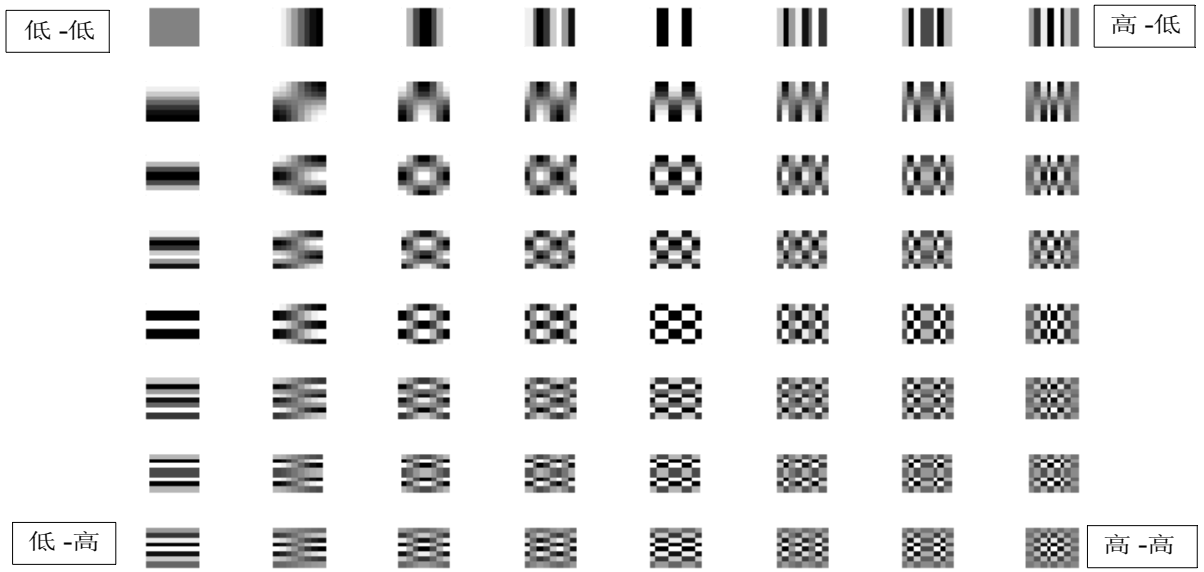


图 13.6　DCT 变换的 8×8 基函数

图 13.6 是用图像方式显示8×8DCT 基函数矩阵。水平频率从左到右依次变大，垂直频率从上往下依次变大。

在下面的例子中，计算输入图像的8×8子块的二维DCT 变换，由于图像的能量主要集中在低频区域，在每个子块中，只利用 64 个 DCT 系数中的 10 个低频系数，其余都置为 0。然后用每个子块的二维 DCT 逆变换重构图像，从而实现对图像的压缩，在这里使用了变换矩阵方法。原始图像和经压缩—解压后的图像分别如图 13.7（a）和（b）所示。

（a）原图像　　（b）压缩—解压后的图像

图 13.7　灰度图像压缩对比图

例 13.5

```
I = imread('cameraman.tif'); %cameraman.tif 是 Matlab 自带的图像文件
I = im2double(I); %数据转换成 double 类型
T = dctmtx(8); %T 为 8×8 的 DCT 变换矩阵
dct = @(block_struct) T * block_struct.data * T'; %定义正 DCT 变换的匿名函数，这里 block_struct 是 Matlab 内置的结构变量
B = blockproc(I,[8 8],dct); %做正 DCT 变换
```

```
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]; %给出掩膜矩阵
B2 = blockproc(B,[8 8],@(block_struct) mask .* block_struct.data); %提取低频系数
invdct = @(block_struct) T' * block_struct.data * T; %定义 DCT 逆变换的匿名函数
I2 = blockproc(B2,[8 8],invdct); %做逆 DCT 变换
subplot(1,2,1), imshow(I) %显示原图像
subplot(1,2,2), imshow(I2) %显示变换后的图像
```

例 13.6　用 DCT 变换对 RGB 彩色图像做压缩。

```
clc, clear
f0=imread('tu3.bmp'); %读入图像
f1=double(f0); %数据转换成 double 类型
for k=1:3
g(:,:,k)=dct2(f1(:,:,k));　%对 R，G，B 各个分量分别作离散余弦变换
end
```

```
g(abs(g)<0.1)=0;    %把 DCT 系数小于 0.1 的变成 0
for k=1:3
f2(:,:,k)=idct2(g(:,:,k)); %作逆 DCT 变换
end
f2=uint8(f2); %把数据转换成 uint8 格式
imwrite(f2,'tu4.bmp'); %把 f2 保存成 bmp 文件
subplot(1,2,1),imshow(f0)
subplot(1,2,2),imshow(f2)
```

对于通常的图像来说，大多数的DCT 系数的值非常接近于0，如果舍弃这些接近于0的值，在重构图像时并不会带来图像画面质量的显著下降。所以利用DCT进行图像压缩可以节约大量的存储空间。图13.8给出了一幅原始图像和经压缩—解压后的图像。

（a）原始图像

（b）压缩—解压后的图像

图 13.8　彩色图像压缩对比图

13.3.3　图像保真度和质量

在图像压缩中为增加压缩率有时会放弃一些图像细节或其它不太重要的内容。在这种情况下常常需要有对信息损失的测度以描述解码图像相对于原始图像的偏离程度，这些测度一般称为保真度（逼真度）准则。常用的主要准则可分为两大类

（1）客观保真度准则；

（2）主观保真度准则。

1　客观保真度准则

当所损失的信息量可用编码输入图与解码输出图的函数表示时，可以说它是基于客观保真度准则的。最常用的一个准则是输入图和输出图之间的均方根误差。

令 $f(x,y)$代表输入图，$\hat{f}(x,y)$代表对 $f(x,y)$先压缩后解压得到的 $f(x,y)$的近似，对任意 x和 y，$f(x,y)$和 $\hat{f}(x,y)$之间的误差定义为

$$e(x,y)=\hat{f}(x,y)-f(x,y),$$

如两幅图尺寸均为 $M\times N$，则它们之间的总误差为

$$\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\left|\hat{f}(x,y)-f(x,y)\right|,$$

$f(x,y)$和 $\hat{f}(x,y)$之间的均方根误差 e_{rms}为

$$e_{\text{rms}}=\left[\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y)-f(x,y)]^2\right]^{\frac{1}{2}}. \qquad (13.12)$$

另一个客观保真度准则是均方信噪比（signal-to-noise ratio，SNR）。如果将$\hat{f}(x,y)$看作原始图$f(x,y)$和噪声信号$e(x,y)$的和，那么输出图的均方根信噪比为

$$\mathrm{SNR}_{\mathrm{rms}}=\sqrt{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}\hat{f}^2(x,y)\Big/\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y)-f(x,y)]^2} \tag{13.13}$$

实际使用中常将SNR归一化并用分贝（dB）表示，令

$$\bar{f} = \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y),$$

则有

$$\mathbf{SNR} = 10\ln\left[\frac{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[f(x,y)-\bar{f}]^2}{\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y)-f(x,y)]^2}\right]. \quad (13.14)$$

如果令

$$f_{\max} = \max\{ f(x,y), x = 0,1,\cdots,M-1, y = 0,1,\cdots,N-1\}$$

，则可得到峰值信噪比

$$\mathrm{PSNR} = 10\ln\left[\frac{f_{\max}^2}{\frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\hat{f}(x,y) - f(x,y)]^2}\right] \tag{13.15}$$

均方根误差e_{rms}越小，峰值信噪比PSNR越大，处理的图像质量越好。

2　主观保真度准则

尽管客观保真度准则提供了一种简单和方便的评估信息损失的方法，但很多图像是供人看的。在这种情况下，用主观的方法来测量图像的质量常更为合适。一种常用的方法是对一组（常超过 20 个）精心挑选的观察者展示一幅典型的图像并将他们对该图的评价综合平均起来以得到一个统计的质量评价结果。

评价也可通过将 $\hat{f}(x,y)$和 $f(x,y)$比较并按照某种相对的尺度进行。如果观察者将 $\hat{f}(x,y)$和 $f(x,y)$逐个进行对比，则可以得到相对的质量分。例如可用{-3，-2，-1,0,1,2,3}来代表主观评价{很差，较差，稍差，相同，稍好，较好，很好}。

主观保真度准则使用起来比较困难。

13.4　数字图像的水印防伪

随着数字技术的发展，Internet 应用日益广泛，数字媒体因其数字特征极易被复制、篡改、非法传播以及蓄意攻击，其版权保护已日益引起人们的关注。因此，研究新形势下行之有效的版权保护和认证技术具有深远的理论意义和广泛的应用价值。

数字水印技术，是指在数字化的数据内容中嵌入不明显的记号，从而达到版权保护或认证的目的。被嵌入的记号通常是不可见或不可察觉的，但是通过一些计算操作可以被检测或被提取。因此，数字图像的内嵌水印必须具有下列特点：

（1） 透明性：水印后图像不能有视觉质量的下降，与原始图像对比，很难发现二者的差别；

（2） 鲁棒性：加入图像中的水印必须能够承受施加于图像的变换操作（如加入噪声、滤波、有损压缩、重采样、D／A或A／D转换等），不会因变换处理而丢失，水印信息经检验提取后应清晰可辨；

（3） 安全性：数字水印应能抵抗各种蓄意的攻击，必须能够唯一地标志原始图像的相关信息，任何第三方都不能伪造他人的水印图像。

在过去的十多年里，数字水印技术的研究取得了诸多成就。而针对图像水印技术的研究，主要体现在空间域和频率域两个层面上，所谓空间域水印，就是将水印信息嵌入到载体图像的空间域特性上，例如图像像素的最低有效位。而频率域水印技术，又称为变换域水印技术，是将水印信息嵌入到载体图像的变换域系数等特征上，例如在图像的DFT 或DCT 或小波变换系数上嵌入水印信息。

13.4.1　基于矩阵奇异值分解的数字水印算法

1　矩阵的奇异值分解（SVD）与图像矩阵的能量

矩阵的奇异值分解变换是一种正交变换，它可以将矩阵对角化。我们知道任何一个矩阵都有它的奇异值分解，对于奇异值分解可用下面的定理来描述。

定理13.1　设A是一个秩为r的$m\times n$矩阵，则存在正交矩阵U和V，使得

$$U^TAV=\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix},\tag{13.16}$$

其中$\Sigma=\mathrm{diag}\{\sigma_1,\cdots,\sigma_r\}$，这里$\sigma_1\geq\cdots\geq\sigma_r>0$，$\sigma_1^2,\cdots,\sigma_r^2$是矩阵$A^TA$对应的正特征值。称

$$A=U\begin{bmatrix}\Sigma & 0\\ 0 & 0\end{bmatrix}V^T\tag{13.17}$$

为A的奇异值分解，σ_i（$i=1,\cdots,r$）称为A的奇异值。

矩阵的F（Frobenius）范数定义为

$$\|A\|_F^2 = \mathbf{tr}(A^T A) = \sum_{i=1}^{m}\sum_{j=1}^{n} a_{ij}^2 , \tag{13.18}$$

由于

$$\mathbf{tr}(A^T A) = \mathbf{tr}\left(V \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix}^T U^T U \begin{bmatrix} \Sigma & 0 \\ 0 & 0 \end{bmatrix} V^T \right),$$

$$= \mathbf{tr}\left(V \begin{bmatrix} \Sigma^2 & 0 \\ 0 & 0 \end{bmatrix} V^T \right) = \sum_{i=1}^{r} \sigma_i^2$$

所以

$$\|A\|_F^2 = \sum_{i=1}^{r} \sigma_i^2 . \tag{13.19}$$

上式表明矩阵F范数的平方等于矩阵的所有奇异值的平方和。对于一幅图像，通常用图像矩阵的F范数来衡量图像的能量，所以图像的主要能量集中在矩阵哪些数值较大的奇异值上。

例13.7　图像的奇异值分解。

为了说明一幅图像矩阵的奇异值与图像能量的对应关系，我们以图片 Lena 为例（图 13.9（a）），对图像矩阵进行奇异值分解，得到其奇异值的分布如图 13.10 所示。

可以看出，矩阵的最大奇异值和最小奇异值相差很大。最大的奇异值为30908，而最小的为0.0028，接近于零。在所有的256个奇异值中，如果只保留其中最大的20个，得到的压缩图片如图13.9（b），在质量上它虽然与原图片有一定差异，但是基本上能反映其真实面貌和特性，损失掉的这部分能量或信息都集中在哪些被忽略的较小的奇异值当中。

（a）Lena原图像　　（b）只保留20个奇异值的Lena

图13.9　奇异值压缩图像对比图

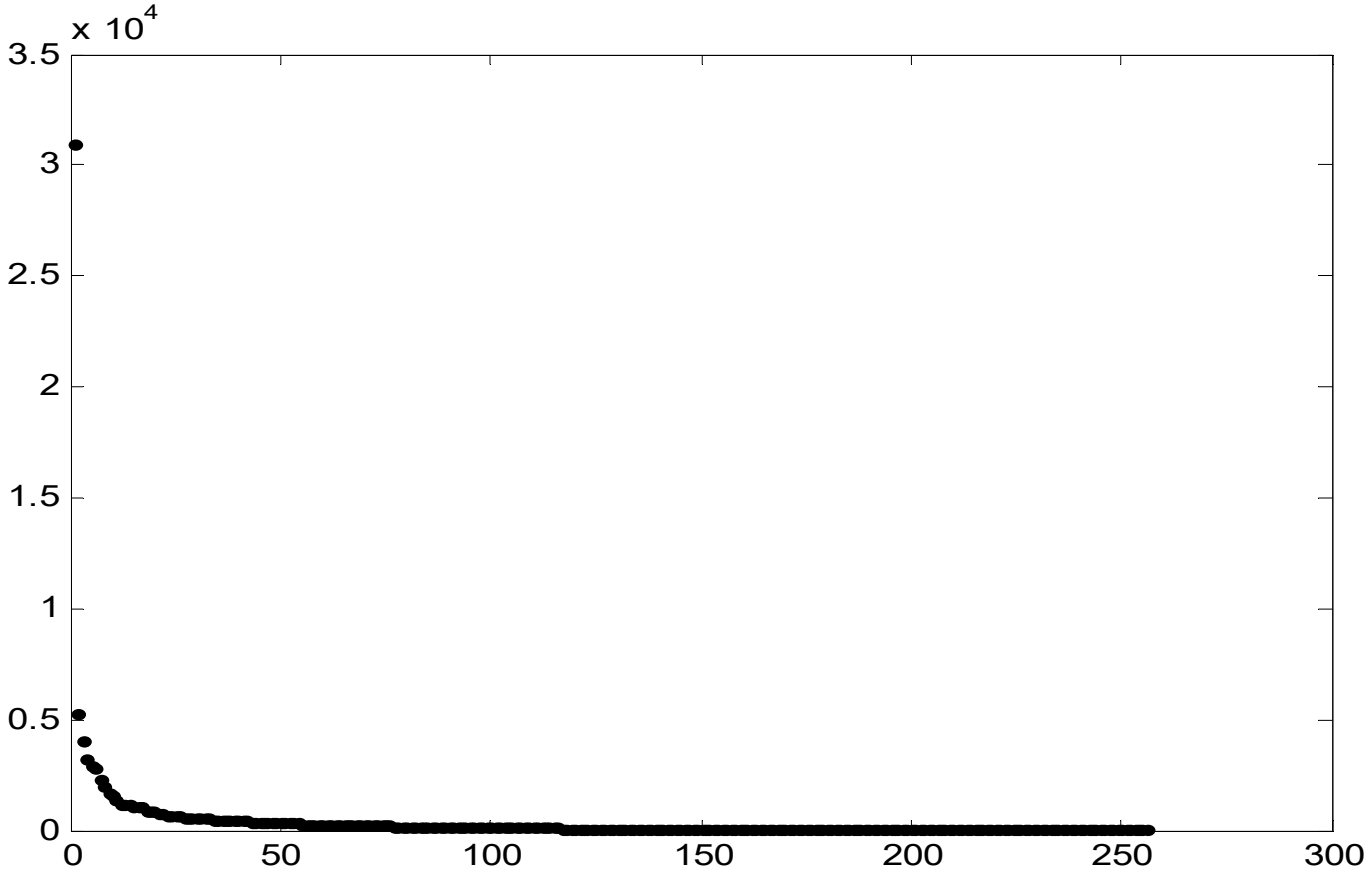


图13.10　Lena的奇异值分布情况

由于在实际应用中，图像总是带有一定噪声的，也就是说待分解的图像矩阵一般都是扰动的，因此了解噪声对矩阵奇异值的影响具有重要意义。Weyl在1912年给出了受噪声扰动的矩阵奇异值与未受噪声扰动的奇异值之差的一个上界，它充分证明了矩阵奇异值分解的稳定性，这个性质可用如下定理来描述。

定理13.2（Weyl定理）　设A为一个大小为$m\times n$的矩阵，$B=A+\delta$，δ是矩阵A的一个扰动，假设矩阵A，B的奇异值分别为$\sigma_1^{(1)}>\cdots>\sigma_r^{(1)}$和$\sigma_1^{(2)}>\cdots>\sigma_r^{(2)}$，$\sigma^*$是矩阵$\delta$的最大奇异值，则有$\left|\sigma_i^{(1)}-\sigma_i^{(2)}\right|<\|\delta\|_2=\sigma^*$，其中$\|\cdot\|_2$表示2—范数。

由Weyl定理可知，当图像被施加小的扰动时图像矩阵的奇异值的变化不会超过扰动矩阵的最大奇异值，因此基于矩阵奇异值分解的数字水印算法具有很好的稳定性，能够有效地抵御噪声对水印信息的干扰。由上述矩阵奇异值分解的性质可知，图像矩阵奇异值分解的稳定性非常好。

当图像加入小的扰动时，其矩阵奇异值的变换不超过扰动矩阵的最大奇异值。基于矩阵奇异值分解的数字水印算法正是将想要嵌入的水印信息嵌入到图像矩阵的奇异值中，如果在嵌入水印的过程中选择一个嵌入强度因子来控制水印信息嵌入的程度，那么当嵌入强度因子足够小时，图像在视觉上不会产生明显的变化。

2　水印嵌入

设一副图像对应的矩阵A大小为$M \times N$，需要嵌入的水印对应的矩阵W大小为$m \times n$，自然地有$M > m$，$N > n$。在矩阵A的左上角取一个大小为$m \times n$的子块A_0。

首先对A_0进行奇异值分解，得到$A_0 = U_1 S_1 V_1^T$，其中S_1是A_0的奇异值矩阵。我们的目标就是将水印W嵌入到矩阵S_1中，在这里定义一个描述水印嵌入过程的参数a，称为嵌入强度因子，则水印嵌入的过程表示为$A_1 = S_1 + aW$。

可以看出，矩阵A_1包含了所有的水印信息，水印信息的能量反映在A_1的奇异值当中。对A_1进行奇异值分解，得到$A_1 = U_2 S_2 V_2^T$，则S_2反映了嵌入水印的图像的全部信息，由此得到子块A_0嵌入水印后的图像子块$A_2 = U_1 S_2 V_1^T$，这样就完成了水印的嵌入。

上述水印嵌入算法的基本过程可以表示为

$$\boldsymbol{A}_0 = \boldsymbol{U}_1 \boldsymbol{S}_1 \boldsymbol{V}_1^T, \tag{13.20}$$

$$\boldsymbol{A}_1 = \boldsymbol{S}_1 + a\boldsymbol{W}, \tag{13.21}$$

$$\boldsymbol{A}_1 = \boldsymbol{U}_2 \boldsymbol{S}_2 \boldsymbol{V}_2^T, \tag{13.22}$$

$$\boldsymbol{A}_2 = \boldsymbol{U}_1 \boldsymbol{S}_2 \boldsymbol{V}_1^T. \tag{13.23}$$

在上面的公式中，矩阵U_1,U_2,V_1,V_2都是正交矩阵。对一个矩阵进行正交变换后它的奇异值保持不变，因此矩阵A_2与A_1有相同的奇异值。设$\sigma_i(A_0)$为矩阵A_0的第i个奇异值，$i=1,\cdots,r$，通过 Weyl 定理可得原图像矩阵子块和嵌入水印后图像矩阵子块的奇异值之间的如下关系

$$\begin{aligned}\left|\sigma_i(A_0)-\sigma_i(A_2)\right| &= \left|\sigma_i(S_1)-\sigma_i(A_1)\right| \\ &= \left|\sigma_i(S_1)-\sigma_i(S_1+aW)\right| \\ &\le \left\|S_1-(S_1+aW)\right\|_2 \\ &= \left\|aW\right\|_2 = a\left\|W\right\|_2 .\end{aligned}$$

嵌入强度因子a的意义在上式中一目了然，它衡量了水印对原图像的扰动情况。在水印嵌入时，选择合适的嵌入强度因子是十分重要的。小的嵌入强度因子有利于水印的透明性，但嵌入的水印信息容易受到外界噪声的干扰，如果噪声强度足够大，则可能使水印信息被噪声淹没而完全丢失，导致提取水印时无法得到水印的全部信息。

大的嵌入强度因子有利于增强算法的鲁棒性，即使在噪声较强的情况下水印信息也不会受到很大的影响，但是过大的嵌入强度因子可能对原矩阵的奇异值产生较大的影响，有可能破坏水印的透明性，影响图像的质量。因此，在水印嵌入时要选择适当的嵌入强度因子使得水印图像的不可觉察性与鲁棒性达到最佳。

3　水印提取

水印提取是上述水印嵌入过程的逆过程。在水印提取时，假设我们得到的是受扰动的图像矩阵A_2^*，首先对A_2^*进行奇异值分解

$$A_2^* = U_3 S_2^* V_3^T , \qquad (13.24)$$

由此得到包含有全部水印信息的奇异值矩阵S_2^*，然后利用水印嵌入时的矩阵U_2, V_2，得到

$$A_1^* = U_2 S_2^* V_2^T , \qquad (13.25)$$

由上述水印嵌入的算法可得

$$W^* = \frac{1}{a}(A_1^* - S_1). \qquad (13.26)$$

这样就得到了水印的信息W^*，其中a为水印嵌入时所用的嵌入强度因子，S_1为原图像A_0的奇异值矩阵。

例13.8　水印嵌入与提取。

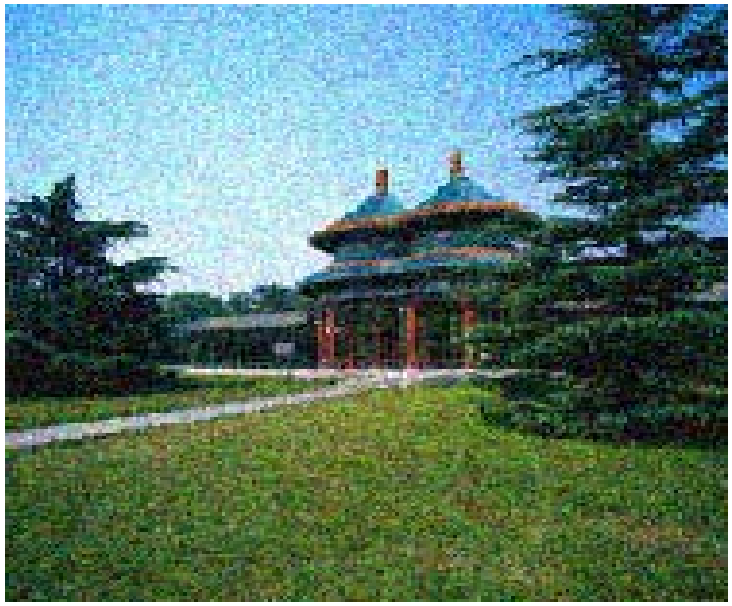
以图 13.11（a）中图像作为载体图像，图 13.11（b）中图像作为水印图像，选择嵌入强度因子为 $a = 0.05$，由于我们选取的是彩色图片，计算时分别对 R,G,B 层做奇异值分解。嵌入水印后的图像见图 13.11(C)。


（a）载体图像　（b）水印图像（c）嵌入水印后的图像

图 13.11　原图像与嵌入水印的对比图像

比较图13.11（a）和（c）可知，在嵌入强度因子较小时对图像的扰动很少，图像基本上没有明显的变化。

为了考察算法的稳定性，将得到的嵌入了水印的图像引入一定的高斯噪声后再提出水印，并对提出的水印图像进行中值滤波。被噪声污染的嵌入水印图像见图13.12（a)，提取的水印图像见图13.12(b)，可以看出，此时较小的扰动并没有导致水印信息的丢失，所以这种基于奇异值分解的数字水印算法确实具有较强的鲁棒性。



（a）引入高斯噪声的合成图像（b）引入高斯噪声后提取的水印

图 13.12　引入高斯噪声的水印提出

13.4.2　基于DCT变换的水印算法

在图像的 DCT 系数上嵌入水印信息具有诸多优势，首先，DCT 变换是实数域变换，对实系数的处理更加方便，且不会使相位信息发生改变。第二，DCT 变换是有损图像压缩 JPEG 的核心，基于 DCT 变换的图像水印将兼容 JPEG 图像压缩。

最后，图像的频域系数反映了能量分布，DCT 变换后图像能量集中在图像的低频部分，即 DCT 图像中不为零的系数大部分集中在一起（左上角），因此编码效率很高，将水印信息嵌入图像的中频系数上具有较好的鲁棒性。

1　水印嵌入算法

水印嵌入算法是通过调整载体图像子块的中频 DCT 系数的大小来实现对水印信息的编码嵌入。算法描述如下

（1）读取原始载体图像A，对A进行8×8分块，并对每块图像进行DCT变换。

（2）在8×8的子块中，中频系数的掩膜矩阵取为

$$H = \begin{bmatrix} 0 & 0 & 0 & 0 & 1 & 1 & 0 & 0 \\ 0 & 0 & 0 & 1 & 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 1 & 0 & 0 & 0 & 0 \\ 0 & 1 & 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{bmatrix},$$

每个子块的中频系数总共有 11 个位置。每个子块的中频位置可以嵌入 11 个像素点的亮度值，对应地，我们将水印图像按照 11 个像素点一组，进行分块，水印图像的最后一个分块如果不足 11 个像素点，则通过把亮度值置 0 进行扩充。

对载体图像DCT系数进行修改

$$g_i' = g_i + \alpha f_i，\ i = 1,\cdots,11，\qquad (13.27)$$

其中g_i是载体图像中频系数的 DCT 值，g_i'是变换后的 DCT 值，α为水印嵌入的强度，这里取$\alpha = 0.05$，f_i为对应位置的水印图像的灰度值（或亮度值）。

2　水印提取算法

水印提取是水印嵌入的逆过程，具体算法描述如下

（1）计算合成图像和原始载体图像的差图像ΔA。

（2）对差图像ΔA进行8×8分块，并对每个小块做DCT变换。

（3）从DCT小块中提取可能的水印序列

$$f_i = (g_i^{'} - g_i)/\alpha,\ \ i = 1,\cdots,11,$$

(4)用下列函数计算可能的水印W和原嵌入水印W^*的相关性

$$C(W^*,W)=\sum_{i=0}^{L-1}(f_i^*f_i)\Big/\sqrt{\sum_{i=0}^{L-1}f_i^2}\,, \tag{13.28}$$

这里f_i和f_i^*分别为图像W和W^*的灰度值(或亮度值),L为像素点的个数。

根据相似性的值就可以判断图像中是否含有水印，从而达到版权保护的目的。判定准则为，事先设定一个阈值T，若$C(W^{*},W)>T$，可以判定被测图像中含有水印，否则没有水印。在选择阈值时，既要考虑误检也要考虑虚警。

例13.9　基于DCT变换的水印嵌入和提取。

以图13.13（a）中图像作为载体图像，图13.13（b）中图像作为水印图像，选择嵌入强度因子为$\alpha = 0.05$，由于我们选取的是彩色图片，计算时分别对R，G，B层做奇异值分解。嵌入水印后的合成图像见图13.14（a），提取的水印图像见图13.14（b）。

（a）载体图像　　（b）水印图像

图 13.13　载体图像与水印图像

（a）水印合成图像　　（b）提取的水印图像

图 13.14　水印合成图像与提取的水印图像

13.5　图像的加密和隐藏

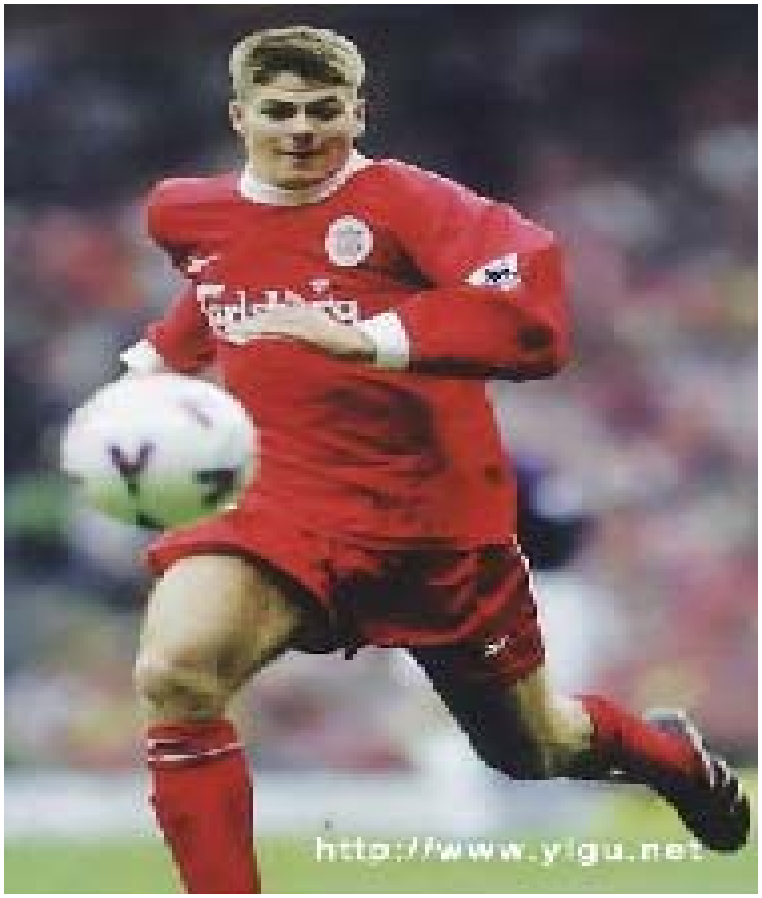
13.5.1　问题的提出

当今时代，信息网络技术在全世界范围内得到了迅猛发展，它极大方便了人们之间的通讯和交流。借助于计算机网络，人们可以方便、快捷的将数字信息（如数字化音乐、图像、影视等方而的作品）传到世界各地，而且这种复制和传送几乎可以无损地进行。

但是，这样的数据传输并不能保证信息的隐秘性，由此以来，保证网络上传送的信息的安全性便成为一个具有重要意义的问题，国内外的一些学者在不断探索信息的隐秘传输，并取得较为丰硕的成果。

信息隐藏技术是利用人类感觉器官的不敏感（感觉冗余），以及多媒体数字信号本身存在的冗余（数据特性冗余），将秘密信息隐藏于掩护体（载体）中，不被觉察到或不易被注意到，而且不影响载体的感觉效果，以此来达到隐秘传输秘密信息的目的。

现有两幅图片，为了保密，需要将图 13.15（a）中的图像进行加密，然后隐藏在图 13.15（b）的图像之中，供将来进行传输。



（a）保密图片　　（b）载体图片

图 13.15　保密图片和载体图片

13.5.2　加密算法

图像加密有很多方法，下面我们利用 Hénon 混沌序列打乱图像矩阵的行序和列序。Hénon 混沌序列为

$$x_{n+1} = 1 - \alpha x_n^2 + y_n, \tag{13.29}$$

$$y_{n+1} = \beta x_n. \tag{13.30}$$

其中$\alpha = 1.4$，$\beta = 0.3$。

设保密图像A的大小为$M_1 \times N_1$，载体图像的大小为$M_2 \times N_2$，为了方便处理，利用 Matlab 软件把载体图像处理成与保密图片同样的大小。记$L = \max\{M_1, N_1\}$，不妨设$L = N_1$，使用密钥 key= -0.400001作为混沌序列（13.29）和（13.30）的初始值x_0, y_0，生成两个长度为L的混沌序列X_L和Y_L，截取X_L的前M_1个分量，并把得到的子序列按照从小到大的次序排列，利用该子序列的排序地址打乱保密图像矩阵A的行序。

同样地，把混沌序列$\boldsymbol{Y_L}$按照从小到大的次序排列，利用该序列的排序地址打乱保密图像矩阵$\boldsymbol{A}$的列序。

图 13.15(a)图像加密以后得到的图像见图 13.16。

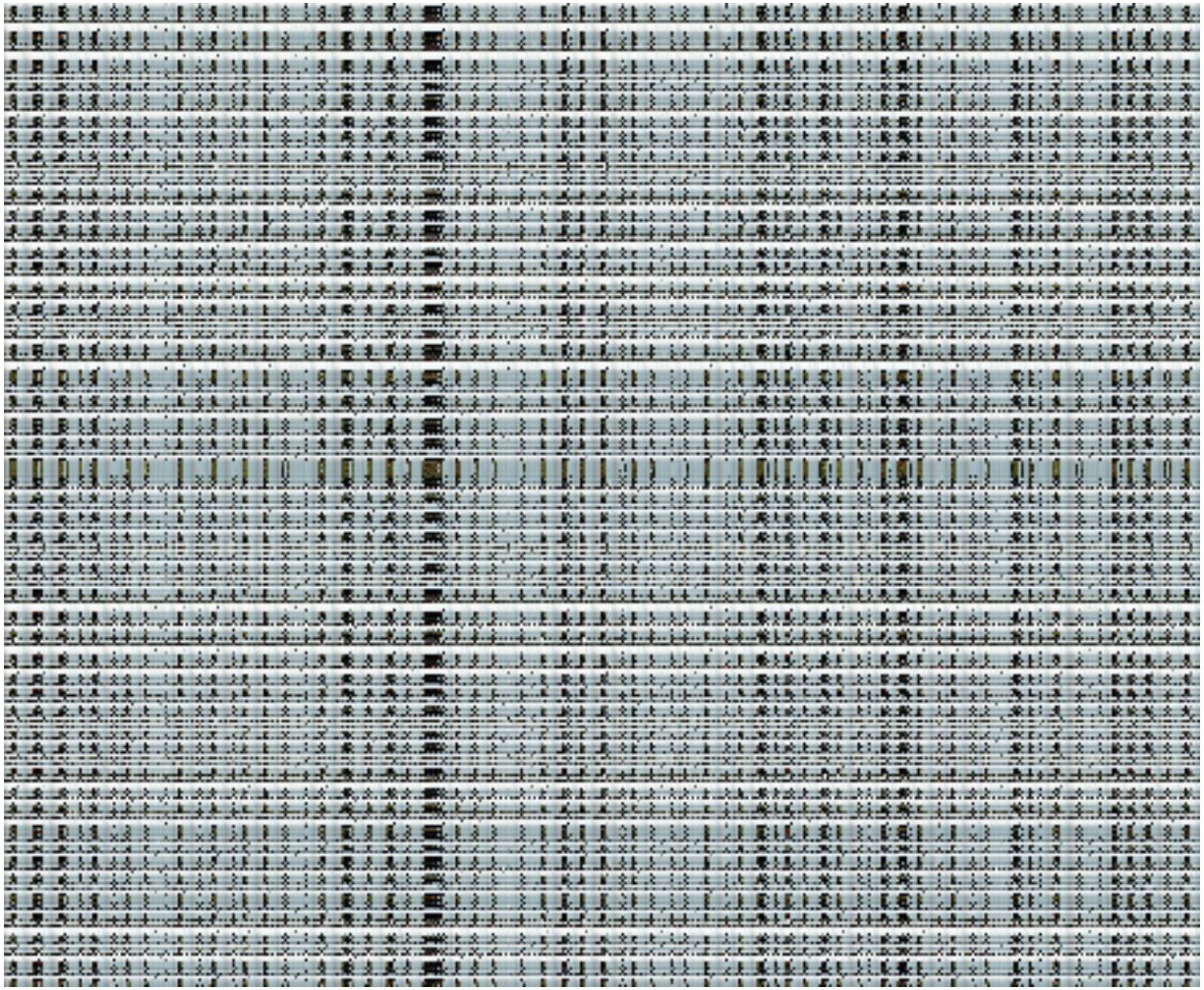


图 13.16　保密图像加密后得到的图像

13.5.3　图像的隐藏

上面讲过的水印算法可以用于图像隐藏，我们这里就不重复了。下面给出基于空域 LSB 的图像隐藏算法的基本思路。

对于 uint8 格式的灰度图像，每个像素点有 256 个灰度级别，可以用 8 位二进制码来表示。我们把它从高到低分成 8 个位平面，每个平面均可以用二值图像来形象表示。由于人的视觉对低位平面不敏感，所以可以利用低 4 位进行信息隐藏。

对于 LSB 嵌入算法，我们简单举例如下，载体图像的像素点 $f(x,y)$=146，其二进制码为 10010010，我们需把它的低 4 位置 0，可以用二进制码 11110000 进行按位与运算即可得

$$f'(x,y)=10010000.$$

加密图像的像素点$g(x,y)$=234，其二进制码为 11101010，同样与二进制码 11110000 进行按位与运算，在转换为十进制后除以 16，相当于二进制码向右移 4 位，我们得到

$$g'(x,y)=00001110,$$

合成的像素点

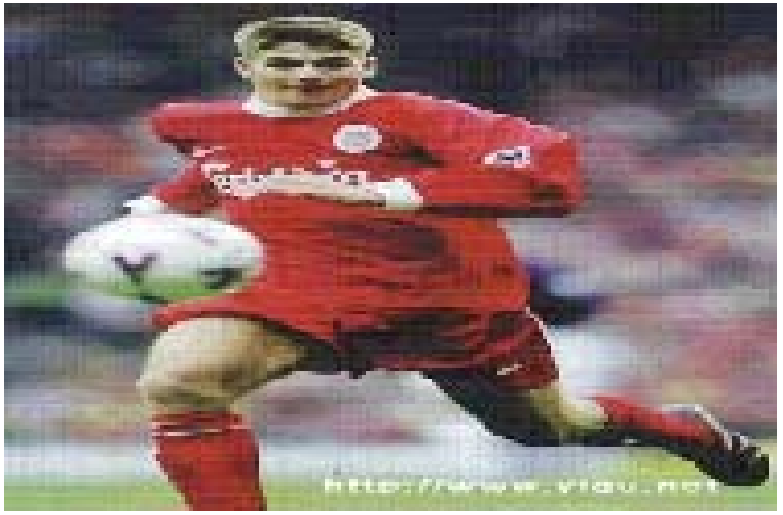
$$I(x,y)=f'(x,y)+g'(x,y),$$

将$I(x,y)$转换为十进制数为 160。像素值 160 既包含载体图像的信息，又包含加密图像的信息，即把加密图像嵌入到了载体图像中。

使用 Matlab 软件很容易实现上述有关的运算，Matlab 中位操作的命令有 bitand，bitor，bitset，bitget。

13.5.4　仿真结果

利用空域 LSB 隐藏算法，我们把加密后的图像隐藏到载体图像中，最终嵌入加密图像的合成图像效果见图 13.17（b）。保密图像的提出过程就是上面隐藏和加密的逆过程，从合成图像中提出的保密图像效果见图 13.17（a）。



（a）提取的保密图像（b）嵌入加密图像的保密图像

13.17　合成图像及提取的保密图像