第八章　视觉感知与基于感知的数字水印

在水印性能评价中，我们有必要分析水印的不可见性。一个良好的水印系统，其基本要求就是在鲁棒性、不可见性以及诸如安全性等多方面都达到较高的水平。对水印不可见性的评价是一个多学科复合的研究问题，我们这里仅对图像视觉信号作一定的分析和评估。在数字水印之前，大量的图像视觉感知评估方法就已经应用于图像纹理分割和图像压缩等领域。对于水印图像的感知质量评估可以参照这些研究成果进行，当然鉴于数字水印的特殊性，有些评价手段也是不适用的。

谈到图像感知质量不可回避地要涉及到人类的视觉系统（HVS）。本章我们就由 HVS 展开，先对其作一个简单的了解。在阐述 HVS 之后，我们开始对图像的感知质量进行评估。评估分为主观和客观两方面进行。主观方面采用二选一迫选实验为实验方法，涉及到了物理心理学上有关 JND 和图像感知级别等方面的问题。客观方面，我们选取了几种常规的又具有代表性的图像视觉感知评价方法加以介绍和实现。最后，我们以经典的 Watson 视觉模型为基础，完成了一个基于感知的数字水印实验，供大家参考。

8.1　人类视觉系统

研究视觉感知不能不提到人类视觉系统（Human Visual System, HVS）。对于 HVS 的研究，学术界表现出了极大的热情。研究内容本身也构成了一个结合生物学、心理学、数学和工程信号处理等多学科的庞大体系。在图像数字水印中，视觉感知能力是一个非常重要的性能评价方向。将人类视觉的某些特征与水印算法本身结合起来就成了当前数字水印研究的热点，而这一切的基础当然就是 HVS。

8.1.1　空间频率

在接触 HVS 之前，我们先来看一种新的频率定义：空间频率（Spatial Frequency）。在日常生活以及我们以前的研究中，大量出现了“频率”的概念。这些频率基本上都是定义在时间上的，所以也叫做“时间频率（Temporal Frequency）”。简单地形容时间频率，就是一个信号在单位时间内幅度变化的快慢，我们用赫兹（Hz）表示它，1Hz = 1/1s。也就是说，时间频率一般由三个方面组成：时间、幅度变化和时间单位（秒）。将这三个概念作牵引，在空间上，我们也可以定义一组度量：空间、空间中的

变化和空间单位。

空间是一个很广泛的表述。一幅放在我们面前的图像就是可以看做一个平面二维空间。它也是我们研究的对象。我们知道,在时间频率上,我们强调的变化一般是指信号幅度的变化,那么在空间上有哪些变化呢?

在视觉上,我们提到的变化是亮度强度的变化。不同的亮度强度对我们的视觉系统构成了不同的刺激(Stimulus)。亮度的变化我们以"对比(Contrast)"的概念来描述。图 8.1 给出了一组黑白间隔的条纹,其黑白间隔的条纹排列起来在水平方向不断重复变化,也就是亮度上对比的不断变化。对比变化一次称为一个 cycle。

从图 8.1 上我们可以很直观地感受到,图 8.1(c)的变化最快,图 8.1(a)的变化最慢,这是我们的第一感觉。这种快慢是如何计算出来的呢?很显然,我们需要一个空间上的单位,在这个单位中对比变化的次数越多,相应的空间频率就越高。考虑到是"人类视觉",我们当然不能用 m, cm, m^2 等这类一般的空间度量单位。

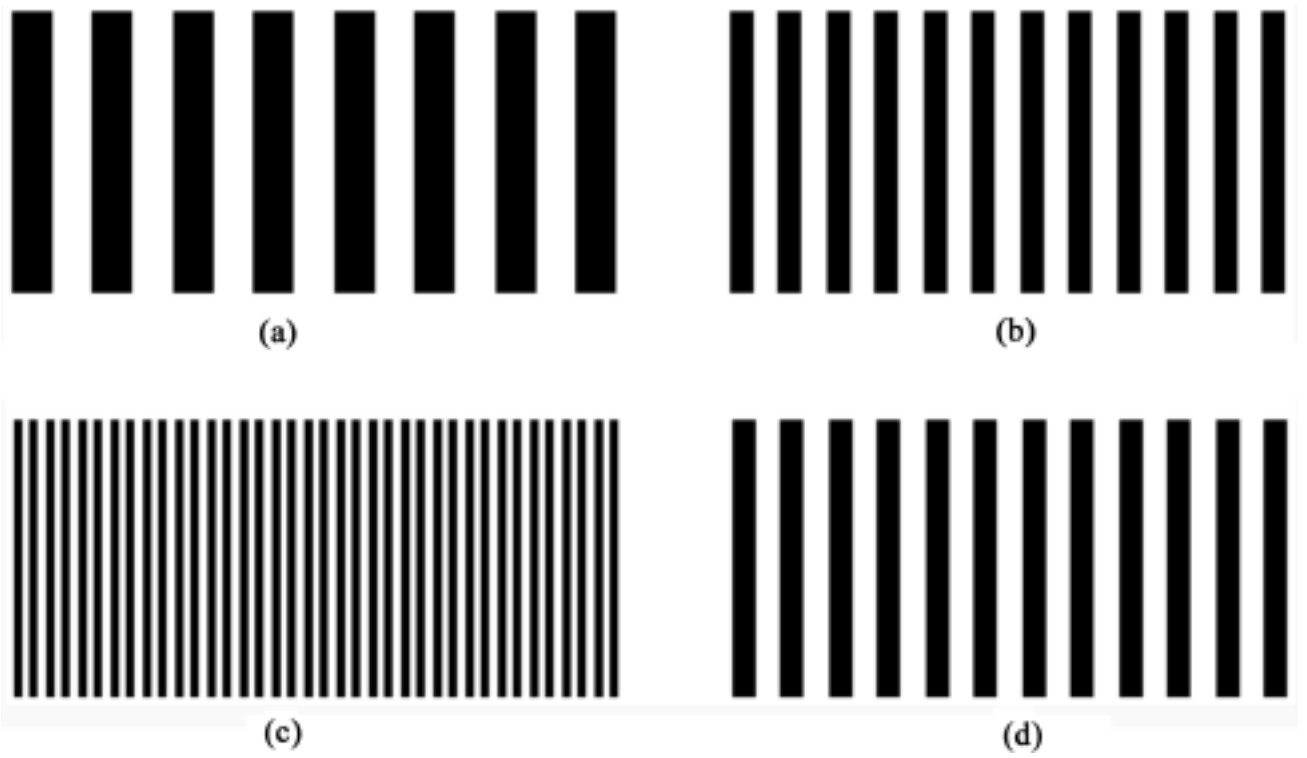


图 8.1　亮度栅栏

在视觉上,我们有一种特殊的度量单位:视角(Visual Angle),如图 8.2 所示。视角的单位也是度(degree)。大家将手臂向前伸直并竖起拇指,看到一个拇指的宽度大概需要 2°的视角。将视角作为空间的单位,就如同秒作为时间单位一样,可以很准确地描述整个空间的大小。

有了空间、空间中的变化(对比变化)和空间单位(视角),我们就可以定义空间频率了。空间频率是指在单位视角中对比变化的次数,记为 cycle/degree, cpd。

在日常生活中,我们似乎很难与以上定义联系起来,因为我们看到的多彩世界绝对不是像一组黑白栅栏那么简单。在视觉感知的研究中,我们如何分析人类的视觉

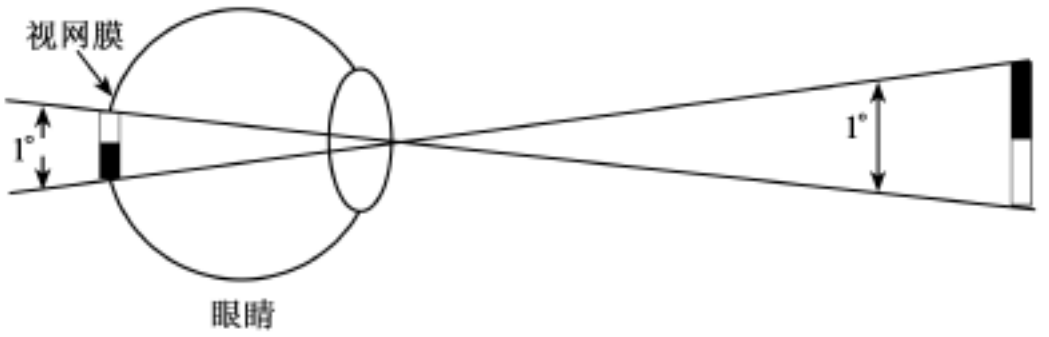


图 8.2　视角

系统接收到的不同刺激呢? 在生物学上,生物学者是将电极插入神经元旁边来收集刺激信号的。他们发现在人类的大脑视觉皮层中,有对不同空间频率感应的细胞。也就是说,尽管我们的视觉集中反应出来的是通常所说的“五颜六色”,但细化到每一个脑细胞,仍然是对空间频率的一种感应,空间频率是一个并不“直觉”的物理量。

这里有一个生物视觉研究上经常做的实验。大家先注视图 8.1(a) 和图 8.1(c) 半分钟以上,然后再去瞟一眼图 8.1(b) 和图 8.1(d),图 8.1(b) 和图 8.1(d) 谁的空间频率高? 一般大家都会说是图 8.1(b) 高(图 8.1(b) 的栅栏看上去细一些)。事实是这样吗? 请眺望一下远方休息一下再看,或者用尺量一下栅栏宽度,原来图 8.1(b) 和图 8.1(d) 根本上就是一幅图。为什么会这样呢? 这是因为长时间注视图 8.1(a) 和图 8.1(c),导致对位于上方的图案低频感应细胞疲劳,对位于下方的图案高频感应细胞也疲劳而造成的。再去看图 8.1(b) 和图 8.1(d),疲劳的细胞感应灵敏度就降低了,所以将位于上方的图 8.1(b) 认为是高频而将位于下方的图 8.1(d) 认为是低频了。这种“认为”也称做适应(adaptation)。

其实,在日常生活中,空间频率有时还是可以被“直觉”到的。图 8.3(a) 是将 lenna 经过马赛克处理的图像。就大家现在看书的这一距离,一定感到左右两幅图像很不一样。那么把书拿远一点呢? 左边的 lenna 伴随着观察距离的增大,似乎越来越清晰了! 事实上,马赛克处理的实质就是用一个小范围内的平均颜色替换整个范围内的像素,所以处理后的图像看上去是一块一块的。将这种变化理解为是对比变化,显然在观察距离近的时候空间频率低,而观察距离远的时候空间频率高。当提升观察距离时,图 8.3 左右两图在单位视角中的变化均比较高,所以比较接近。

在工程上,我们经常要使用到空间频率滤波。这时,我们所谓的“对比变化”就过于抽象了,所以,我们一般是将空间上正弦波的变化理解为空间频率。根据物理 Fourier 光学的有关知识,光从物面到频谱以及从频谱到像面的传播都可以用 Fourier 变换描述。对于图像的二维 Fourier 变换,有:

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

$$p = 0,1,2,\cdots,M-1 \quad q = 0,1,2,\cdots,N-1$$

其中,p,q 就是水平和垂直方向上的空间频率。这样一来,图像的空间频率滤波就与

图 8.3　lenna 的马赛克处理

我们熟悉的数字信号处理的有关内容结合到一起了。当然,在不同的视觉模型中,我们也可以定义其他的空间频率计算方法。

8.1.2　人类视觉系统的一般描述

下面我们从空间频率、时间频率、方向、光谱特性、掩蔽和误差合并六个方面来认识 HVS。

(1)空间频率与对比敏感度

研究表明,人们对于高空间频率的敏感度较低而对于低空间频率的敏感度较高。如图 8.3 所示,在相对低空间频率的情况下,我们很容易发现左右两幅图的不同,而在相对高空间频率的情况下则就不那么容易发现左右两幅图的不同了。人类眼睛的视网膜有 X 和 Y 两种神经细胞。Y 细胞对空间频率呈低通特性,X 细胞也呈低通特性但低频衰减快,高频有延伸,截止频率较 Y 细胞高。事实上,HVS 的空间频率响应可以看做是一个窄带通滤波器,由亮度对比的敏感度与空间频率组成的函数来描述。峰值亮度对比敏感度对应的空间的频率是 4 cpd。

一个信号只有当其对比度高于某个对比度阈值(contrast threshold)的时候才能被 HVS 感受到。对比阈值随着空间频率的变化而变化,构成一个对比阈值函数(Contrast Threshold Function, CTF)。该函数在整个可见光的空间频率(0 ~60 cpd)内度量,在每一个空间频率下的对比阈值是在给定的空间频率下刚好能够感受到一个正弦波所需要的最小正弦幅值。对比敏感度(contrast sensitiveness)就是对比阈值的倒数。由对比敏感度和空间频率组成的函数被称为对比敏感函数(Contrast Sensitivity Function, CSF)。CSF 是研究视觉感知的基础中的基础。一个典型的 CSF 如图 8.4 所示。

可以看到,HVS 对空间频率的最大响应区为 2 ~10 cpd。CSF 受到方向性、图像亮度、图像空间的大小和观察距离等多方面的影响,一般将 CSF 认为是一个带通滤波器(band pass filter)。前面已经具体说明过,当观察距离增大时,低频的失真将进

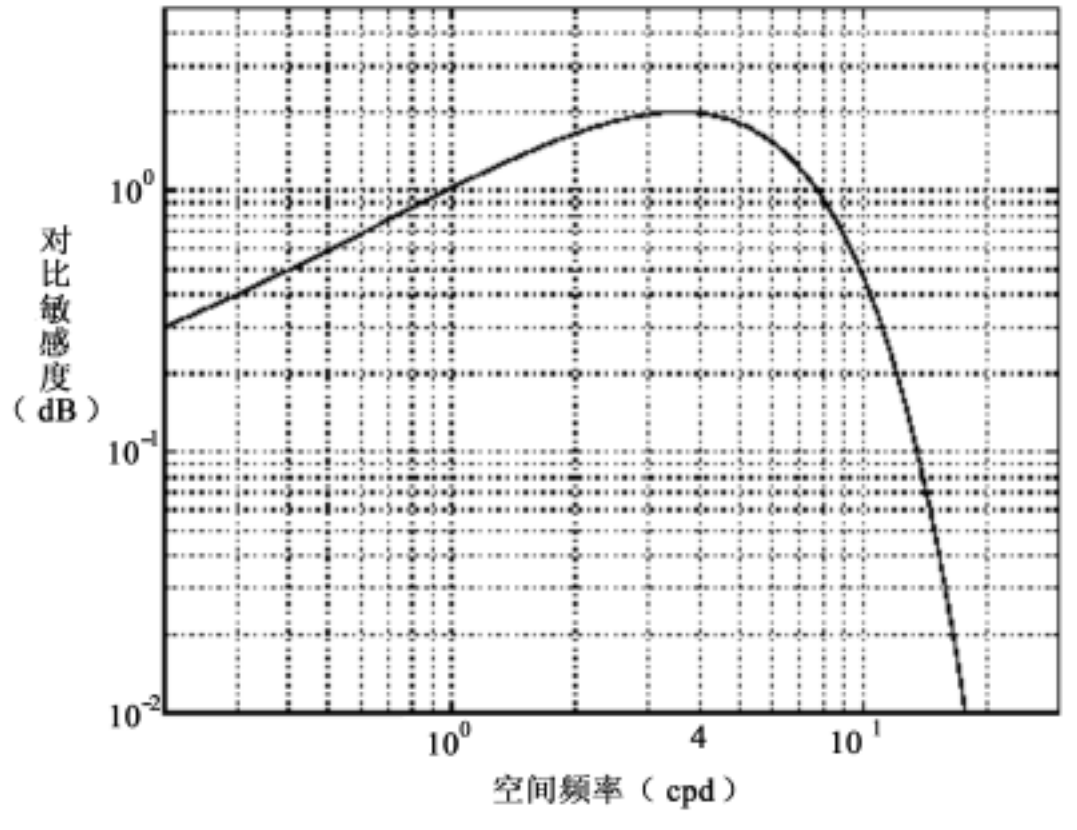


图 8.4　对比敏感函数 CSF

入最大响应区。所以，有时也将 CSF 定义为低通滤波器(low pass filter) 。

(2) 视觉时间频率特性

时间频率通常是以抖动、运动等形式被我们感知的。人眼对于不同时间频率有不同的反应。图 8.5 表示时间频率敏感度与时间频率构成的函数图像。

研究表明，当频率高于 30Hz 时，HVS 敏感度就会迅速衰减。所以，电影、电视的传输速度都不会超过每秒 60 帧。

(3) 视觉感知在方向上的分解

研究表明，视觉感知对水平(horizontal) 和垂直(vertical) 方向上的刺激最敏感，对对角(diagonal) 方向尤其是 45°(135°) 方向上的刺激最不敏感。一个 HVS 可以由多个并列的视觉通道(channel) 组成，这些通道的空间频率带宽约为一个倍频程，方向上的选择宽度在 15°~60°之间。为了模拟实现 HVS 这种在空间频率和方向上的分解特性，人们使用了大量方法，最典型的就是用一组 Gabor 多分辨滤波器建模或对信号使用小波变换。

(4) HVS 的光谱特性

HVS 对于不同的颜色其敏感度也不同。眼睛对各种颜色的敏感度可以用标准相关光谱亮度功效函数来描述。标准相关光谱亮度功效函数如图 8.6 所示。由图 8.6 可以看出，对于三原色 R, G, B，人眼对绿色最敏感，对蓝色最不敏感。联想到前面我们多次给出的亮度方程，一个像素点的亮度 Y 有:

$$Y = 0.299R + 0.587G + 0.144B$$

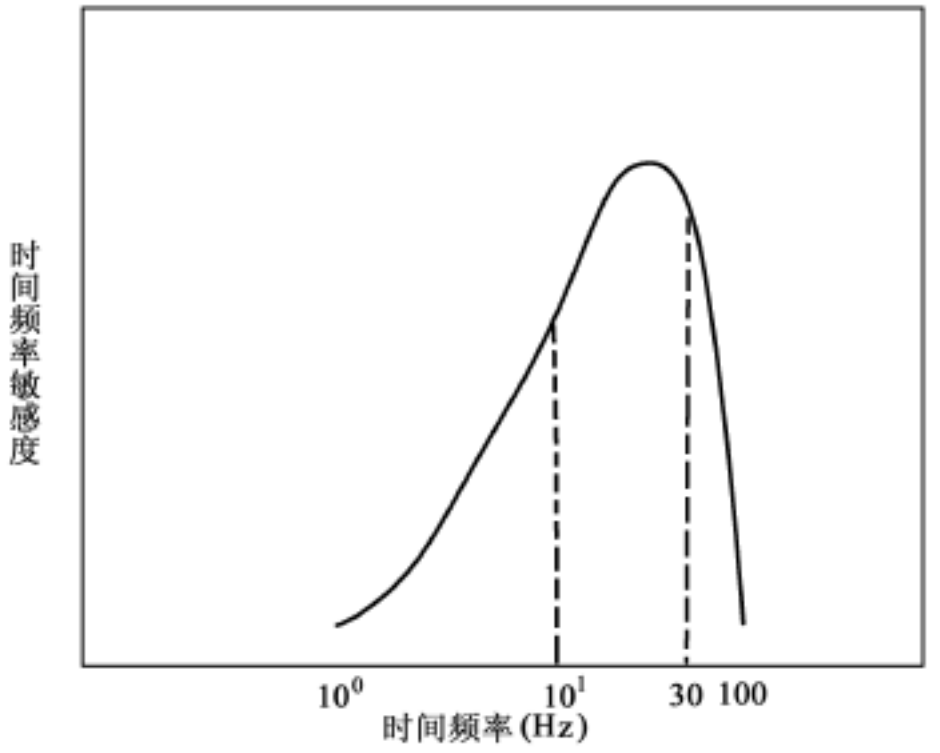


图 8.5　时间频率敏感度

其各项系数的确定也是根据 HVS 得到的。

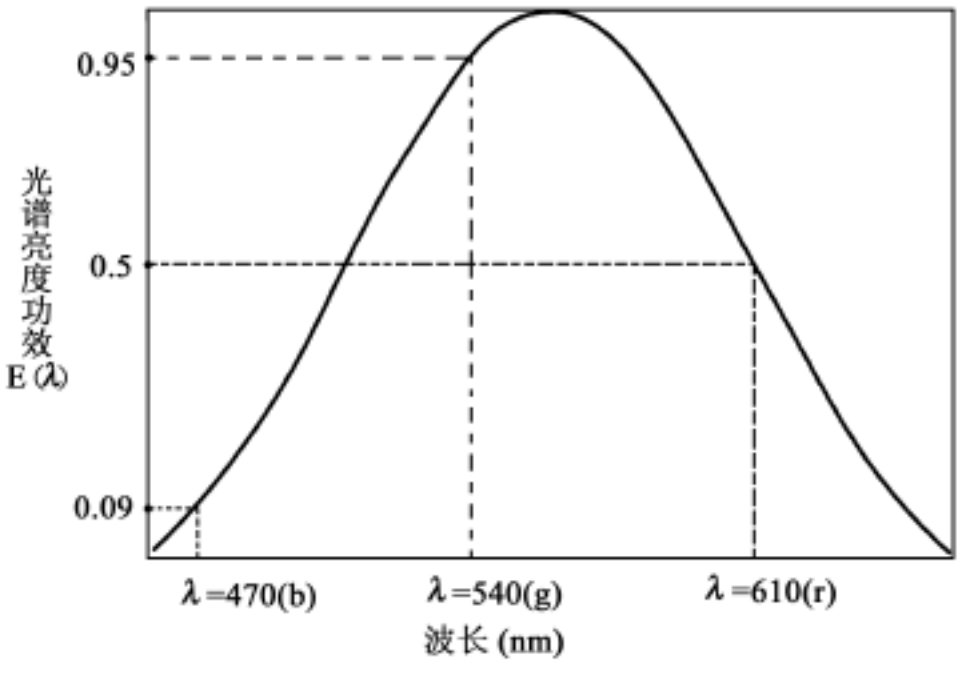


图 8.6　标准相关光谱亮度功效函数

(5) 掩蔽

众所周知, 一个信号的存在能淹没或掩蔽另一个信号, 如两个人在十分吵闹的足球场说悄悄话, 球场的噪声足以掩蔽他们的对话。在视觉上, 有两种掩蔽经常出现: 对比度(频率) 掩蔽与亮度掩蔽。我们主要谈对比度掩蔽。

HVS 对刺激信号的响应并不取决于刺激信号的绝对亮度, 而是取决于刺激相对

于背景(刺激信号的平均亮度)的局部变化,即刺激信号的背景对比度。人眼在不同背景对比度下可以感知的最小亮度差被称为可见度阈值(visible threshold)。研究表明,当背景(background)和刺激(stimulus)在方向、空间频率和时间频率非常接近时掩蔽效果最大。我们定义一个 C_{T0} 为从 CSF 上获得的无掩蔽条件下的可见度阈值。C_T 为在有掩蔽条件下的可见度阈值。C_M 为背景对比度,表示为背景信号与刺激信号的比例。图 8.7 是一个在对数坐标下的掩蔽与可见度阈值构成的函数图像。

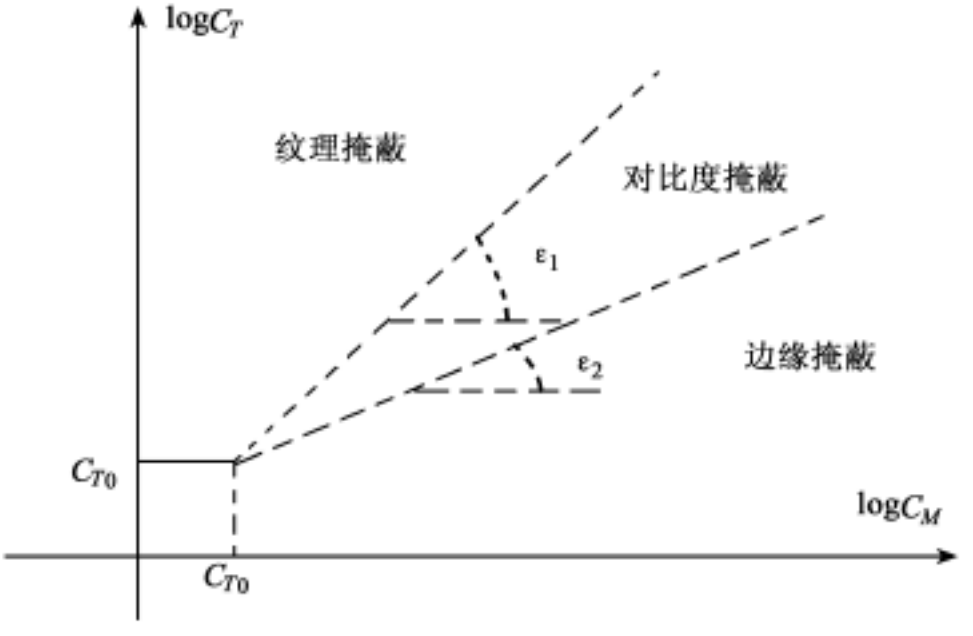


图 8.7　掩蔽

可以看到,当 C_M 较低时,可见度阈值 $C_T = C_{T0}$。事实上,当 C_M 逐渐接近 C_{T0} 时,可见度阈值开始提升,由于其处于一个短暂阶段,在图 8.7 中我们忽略这一情况。当 C_M 大于 C_{T0} 且继续不断增大时,可见度阈值在对数坐标下呈现线性增长的态势。一般情况下,可以将 C_T 描述为:

$$C_T = \begin{cases} C_{T0} & C_M < C_{T0} \\ C_{T0}(C_M / C_{T0})^{\varepsilon} & \text{其他} \end{cases} \tag{8.1}$$

式 8.1 中 ε为图 8.7 所示曲线的斜率。有学者进一步研究了背景与刺激的关系,认为可见度阈值与背景对比度的关系曲线受到多方面的影响。并根据引起掩蔽的原因(如局部对比度、边缘和局部活动性),将掩蔽分为对比度掩蔽区、边缘掩蔽区与纹理(噪声)掩蔽区等。由于纹理掩蔽中的噪声比较复杂,观察者对其缺乏更多的先验认识,所以相应的 ε要大一些。

(6)误差合并

前面提到的频率敏感度、方向分解和掩蔽等均是从不同的层面衡量一个视觉感知质量。事实上,我们对一个图像的直观质量评价就是 HVS 对多方面评估结果的一种合并。为了模拟这种合并,一般采用明科斯基和(Minkowski summation)(L^P 范数)来计算。明科斯基和可以表示为:

$$E = \left| \sum_i |S_i|^{\beta} \right|^{\frac{1}{\beta}} \tag{8.2}$$

S_i 为观察者可以得到的各种评估质量，β是求和指数，对于图像在一般情况下 β取为4比较恰当。

8.1.3 CSF 的实现方式

前面已经阐述了 CSF 的来源及其基本特点。由于观察角度不同，对于 CSF 的实现也不尽相同(或者说是用 CSF 模拟 HVS 的程度和方式不同)。本小节我们给出两种低通 CSF 的形式表示并完成其 MATLAB 实现。

(1) Mannos 的 CSF

经典的低通 CSF 是由 J·Mannos 和 D·Sakrison 提出的，如图 8.8(a)所示。通过该 CSF 能够成功地反映图像的感知质量，其频率响应为：

$$H(f_r) = 2.6(0.0192 + 0.114f_r) \cdot \exp(-(0.114f_r)^{1.1}) \tag{8.3}$$

其中，f_r 是径向频率(radial frequency)，由以下公式计算获得：

$$f_r = \frac{\pi}{180\arcsin\left(\frac{1}{\sqrt{1+d^2}}\right)} \sqrt{u^2 + v^2} \tag{8.4}$$

径向频率是一种视觉频率，由水平空间频率 u 和垂直空间频率 v 以及观察距离 d 共同计算得到。由式(8.3)和式(8.4)构成的 CSF 频率响应的峰值响应频率是 8cpd。

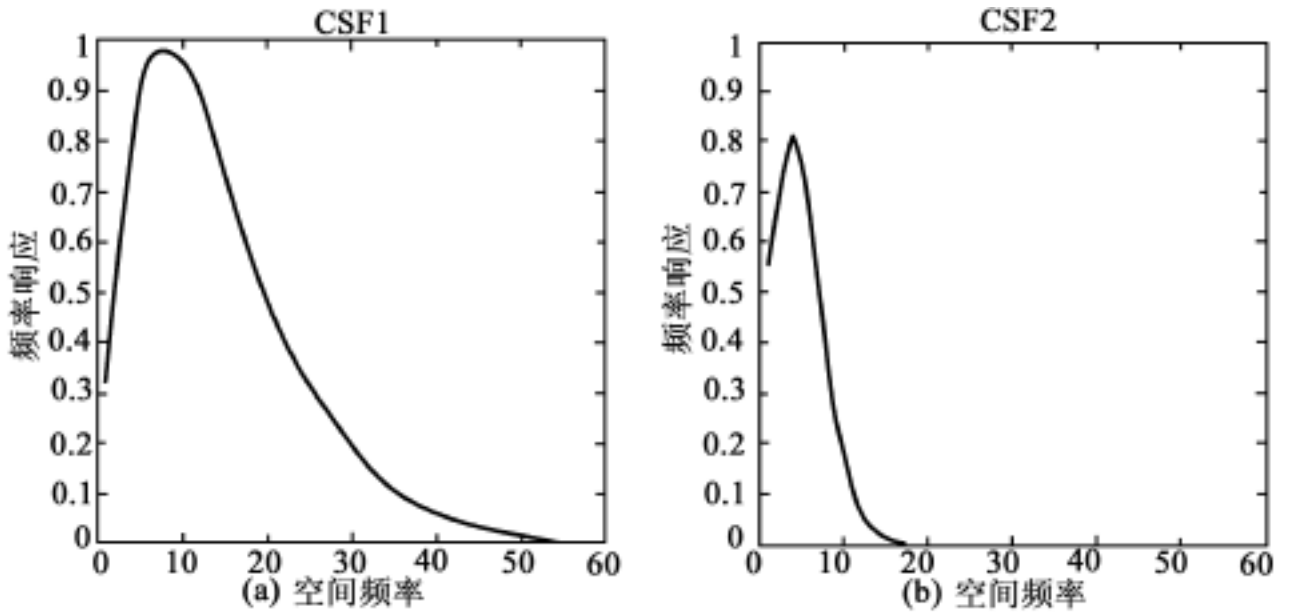


图 8.8 Mannos 和 Makoto Miyahera 的 CSF

(2) Makoto Miyahara 的 CSF

Makoto Miyahara 在其有关编码图像的质量尺度(Picture Quality Scale, PQS)的文章中提出了另一种 CSF 的实现方式，如图 8.8(b)所示。其 CSF 的空间频率响应为：

$$S(\omega) = 1.5e^{\frac{-\sigma^2\omega^2}{2}} - e^{-2\sigma^2\omega^2} \tag{8.5}$$

其中，$\sigma=2$，$\omega=\frac{2\pi f}{60}$，$f=\sqrt{u^2+v^2}$。u，v 分别是水平和垂直方向上的空间频率。同时，Makoto Miyahara 等还认为，式(8.5)所给出的 CSF 比较注重对图像边界重构起重要作用的低频系数的量化，而弱化了高频系数。所以，在式(8.5)的基础上，还给出了一个针对高频的频率响应：

$$O(\omega,\theta) = \frac{1+e^{\beta(\omega-\omega_0)}\cos^4 2\theta}{1+e^{\beta(\omega-\omega_0)}} \tag{8.6}$$

其中，$\theta=\arctan(\frac{v}{u})$，$\beta=8$，$f_0=11.13$cpd。将以上两个频率响应结合起来，就得到了最终的 CSF 滤波器：

$$S_a(u,v) = S(\omega)O(\omega,\theta) \tag{8.7}$$

通过式(8.7)所示的 CSF 滤波器对图像失真进行量化，可以得到需要的失真度量。

(3)两种 CSF 的比较和实现

比较两种 CSF 的实现形式，前者(CSF1)关注到了观察距离(viewing distance)对滤波结果的影响，后者(CSF2)在高频部分引入了观察方向的影响。与图 8.4 所示的理想 CSF 比较而言，CSF1 在峰值响应频率上并不十分准确。事实上，有关 CSF 的实现形式有很多。Nill 和 Nagn 等就分别设计了峰值响应频率为 5cpd 和 3cpd 的低通 CSF。此外，由于 HSV 在对比敏感上呈现带通的特性，所以有不少学者如 Barten 等也设计出了许多带通 CSF。低通 CSF 与带通 CSF 各有优缺点，有兴趣的读者可以参考相关的文献。

MATLAB 可以根据频率采样法创建滤波器。我们重点完成对 CSF2 的实现。其核心思想是给出一个所需的频率响应幅值矩阵，根据频率响应设计的滤波器频率响应将经过所有的给定点，从而确定滤波器的系数。研究表明，人眼可感受到的视觉频率大概为 0～60cpd，所以，我们将滤波器的水平和垂直空间频率分别从 -20cpd 取到 20cpd。在 20×20 的笛卡儿平面上，利用式(8.7)计算出各点对应的频率响应(如图 8.9 所示)。将所得到频率响应利用 MATLAB 的 fsamp2.m 函数计算出滤波器系数。fsamp2.m 可以根据在笛卡儿平面上定义的各点的频率响应设计一个 FIR 滤波器，其最基本的调用格式为：

```
filtercoefficients = fsamp2(frequencyresponse)
```

其中，frequencyresponse 是所输入的频率响应。如果 frequencyresponse 是 M×N 的，返回的滤波器系数相应的也是 M×N 大小。

编写函数 csf.m 完成滤波器设计，函数代码如下：

```
% 文件名: csf.m
% 程序员: 郭迟
% 编写时间: 2004.3.22
```

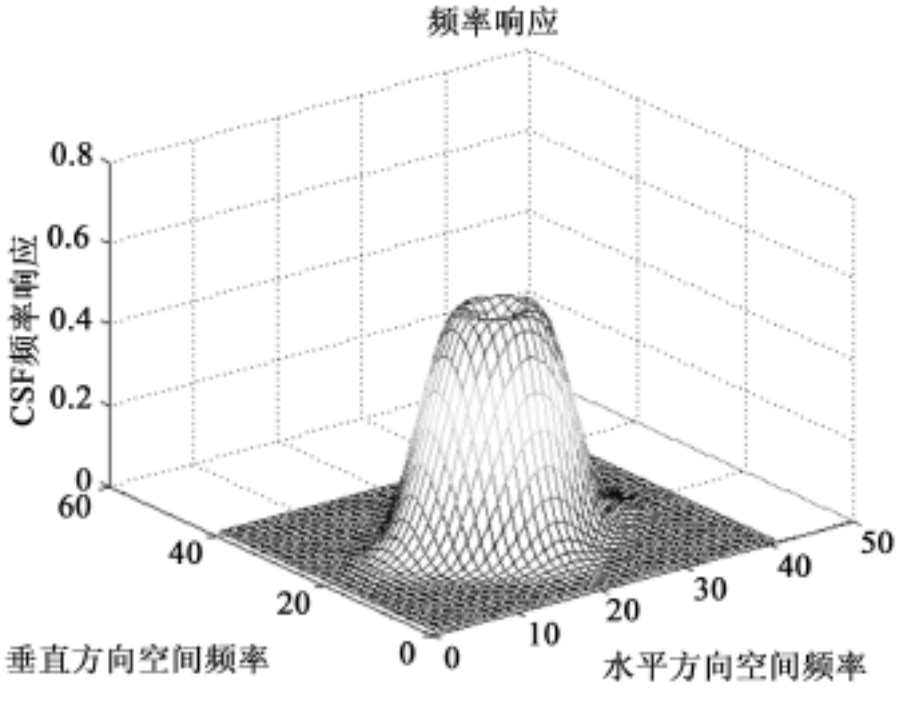


图 8.9　CSF2 频率响应

```
% 函数功能: 本函数将完成一个 CSF 的设计
% 输入格式举例: filtercoefficients = csf
% 参数说明:
    % filtercoefficients 为 CSF 的滤波器系数
    function filtercoefficients = csf( )
    % 调用子函数计算频率响应矩阵
    Fmatrix = csfmat;
    % 画出频率响应
    % figure(1); mesh(Fmatrix), title('频率响应'), xlabel('水平方向空间频率');
ylabel('垂直方向空间频率'); zlabel('CSF 频率响应');
    % 利用 FSAMP2 函数计算频率系数
    filtercoefficients = fsamp2(Fmatrix);
    % 子函数, 计算频率响应矩阵
    function Fmatrix = csfmat( )
    u = -20:1:20;
    v = -20:1:20;
    n = length(u);
    Z = zeros(n);
    for i = 1:n
        for j = 1:n
            Z(i, j) = csffun(u(i), v(j)); % 调用子函数计算相应空间频率下的
频响
```

```
        end
    end
    Fmatrix = Z;
    % 子函数, 计算 u, v 下的频率响应
    function Sa = csffun(u, v)
    % CSF 频率响应
    sigma = 2;
    f = sqrt(u.* u + v.* v);
    w = 2* pi* f/60;
    Sw = 1.5* exp(- sigma^2* w^2/2) - exp(- 2* sigma^2* w^2/2);
    % 高频修正
    sita = atan(v./(u + eps)); % eps = 2^-52, 是避免 0 的一种修正
    bita = 8;
    f0 = 11.13;
    w0 = 2* pi* f0/60;
    Ow = (1 + exp(bita* (w - w0)) * (cos(2* sita))^4) / (1 + exp(bita*
(w - w0)));
    % 最终结果
    Sa = Sw * Ow;
```

图 8.10 是根据图 8.9 定义的频率响应构建的 FIR 滤波器系数。在后面的章节中, 我们会利用这里设计的 CSF 滤波器完成对图像失真的度量。

Array Editor: filtercoefficients

File Edit View Web Window Help

Numeric format: shortG　Size: 41 by 41

	1	2	3	4	5	6	7	8	9
1	2.7151e-008	2.3708e-008	1.5338e-008	1.358e-009	-1.6855e-008	-3.7738e-008	-6.2346e-008	-9.407e-008	-1.3436e-007
2	2.3708e-008	1.94e-008	1.1903e-008	2.2176e-009	-1.0846e-008	-3.1288e-008	-6.2731e-008	-1.033e-007	-1.4421e-007
3	1.5338e-008	1.1903e-008	7.9304e-009	3.6074e-009	-6.902e-009	-3.0262e-008	-6.4774e-008	-9.8881e-008	-1.2175e-007
4	1.358e-009	2.2176e-009	3.6074e-009	6.621e-010	-1.2971e-008	-3.4543e-008	-5.0936e-008	-5.5396e-008	-6.3762e-008
5	-1.6855e-008	-1.0846e-008	-6.902e-009	-1.2971e-008	-2.414e-008	-2.3136e-008	-5.1426e-009	1.5896e-011	-5.186e-008
6	-3.7738e-008	-3.1288e-008	-3.0262e-008	-3.4543e-008	-2.3136e-008	1.3161e-008	3.6066e-008	-1.1017e-008	-1.2204e-007
7	-6.2346e-008	-6.2731e-008	-6.4774e-008	-5.0936e-008	-5.1426e-009	3.6066e-008	4.962e-009	-8.5157e-008	-1.0248e-007
8	-9.407e-008	-1.033e-007	-9.8881e-008	-5.5396e-008	1.5896e-011	-1.1017e-008	-8.5157e-008	-5.7209e-008	1.905e-007
9	-1.3436e-007	-1.4421e-007	-1.2175e-007	-6.3762e-008	-5.186e-008	-1.2204e-007	-1.0248e-007	1.905e-007	5.0962e-007
10	-1.7818e-007	-1.7473e-007	-1.3544e-007	-1.0521e-007	-1.6648e-007	-2.0394e-007	4.7176e-008	4.3994e-007	2.8135e-007

图 8.10　由 CSF 频率响应构建的 FIR 滤波器系数

8.1.4 Gabor 滤波器设计

在许多中文文献上,“Gabor”被翻译为“伽柏”或“盖博”等。1946 年 Dennis Gabor 创建了将 Fourier 变换用于分析一个特定短时间内信号的方法——短时 Fourier 变换(Short-Time Fourier Transform, STFT)。在小波分析产生以前,Gabor 变换一直是一种通用的加窗 Fourier 变换,能够将一个信号映射到时间和频率两个域上,弥补了 Fourier 变换的许多不足。

在图像处理和视觉感知分析领域,人们经常使用 Gabor 多通道滤波来获取不同分辨率下的图像数据特征。这种多通道分解极大地符合了 HVS 感知多通道的特点,成功地模拟了视觉多分辨的现象。相对于单通道的感知质量测试方法,多通道测试有着极大的优势。

下面先来解释什么是“通道”。

前文已经说过,科学研究者经过大量的生理和心理实验都证实了人类大脑拥有一个分别机制,即对不同的频率有不同的细胞去感应。这就要求我们在设计模拟 HVS 的滤波器时,应该与 HVS 一样,能够将收集到的视觉信号按照空间频率、观察方向等分解到不同的频带中去。这里每一个频带就称为一个通道。

一般来说,将空间频率和方向分解到 4 ~8 个频带是合适的,尽管这种分解可以是线性的。许多研究者都采用以下方法:整个空间滤波频带由 17 个 Gabor 滤波器组成,整个频率平面将根据径向频率和方向两个方面被划分。5 个径向频率根据 1 个倍频程取为 0, 2, 4, 8 和 16cpd。所谓频率 f_1 与 f_2 相差 N 个倍频程是指 f_1 与 f_2 满足 $\log_2(\frac{f_1}{f_2}) = N$。四个方向分别为 0, π/4, π/2, 3π/4。每个方向与不同的空间频率滤波器结合,可以构建 17 个通道(中心频率 0 对应的通道是一样的)对应 17 个 Gobar 滤波器,形成一个多分辨滤波器组。

下面我们来具体定义这 17 个 Gabor 滤波器。

一个二维 Gabor 滤波器的冲击响应(impulse response)可以表示为:

$$h(x, y) = \frac{1}{2\pi\sigma^2} e^{\frac{-(x^2+y^2)}{2\sigma^2}} \cos(2\pi u_0(x \cdot \cos\theta + y \cdot \cos\theta)) \qquad (8.8)$$

其中 u_0 是径向频率,θ表示方向。σ是 Gabor 函数中的高斯(Gauss)窗宽度(一个形如式(8.8)的 Gabor 函数可以看成一个二维 Gauss 函数调制的有向复正弦栅格。其中的 $\frac{1}{2\pi\sigma^2} \cdot e^{\frac{-(x^2+y^2)}{2\sigma^2}}$ 部分就是一个被复正弦调制的 Gauss 函数)。事实上,式(8.8)是一个极坐标下的函数,u_0 和 θ都是由水平和垂直的空间频率 u, v 决定的。$\theta = \arctan(\frac{v}{u})$ 看做是滤波器的方向,$u_0 = \sqrt{u^2 + v^2}$ 为径向频率。此外,我们将 Gauss 窗定义为等方性的(isotropic),所以,其垂直和水平宽度相同,均记为 σ(否则 Gauss 函数将比

这里的复杂)。根据我们前面的分析,取 $\theta=\{0, \pi/4, \pi/2, 3\pi/4\}$, $u_0=\{0, 2, 4, 8, 16\}$则可以得到需要的 Gabor 滤波器组。

图 8.11 是被 17 个通道划分的空间频带。

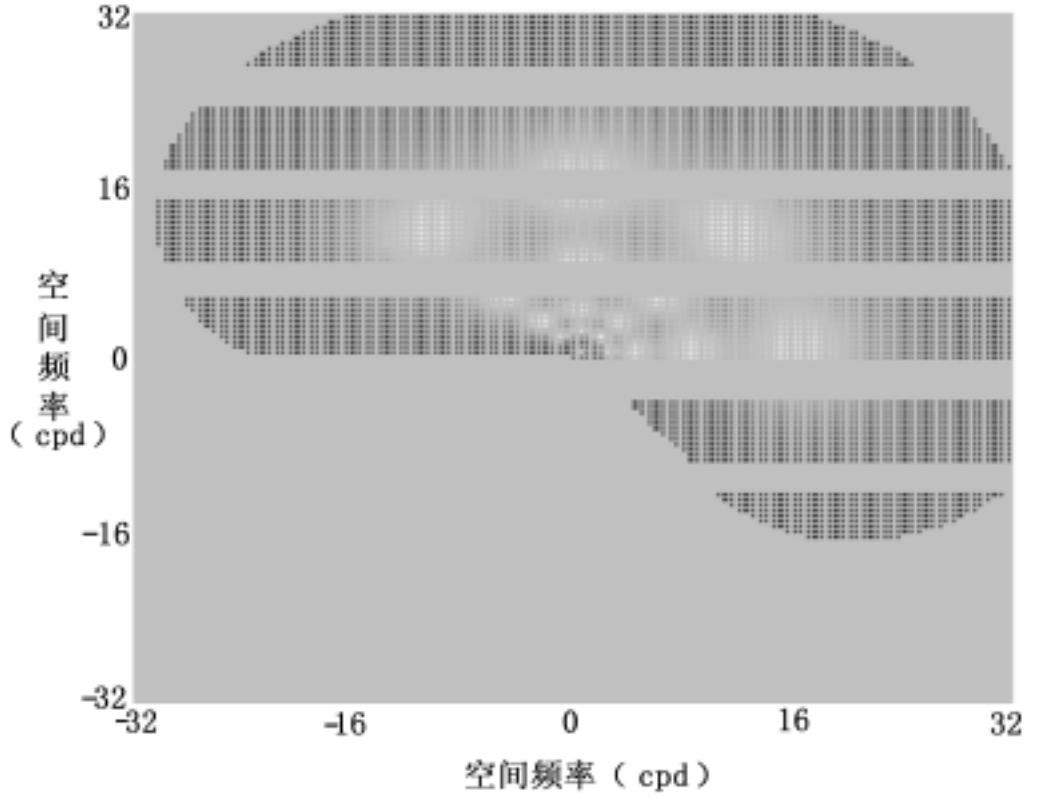


图 8.11　划分的空间频带

有了 Gabor 滤波器的冲激响应,再根据二维卷积的定义,则 Gabor 滤波器 O_h 可以表示为:

$$m(x, y) = O_h(i(x, y)) = i(x, y) * h(x, y) \tag{8.9}$$

其中 $m(x, y)$ 为滤波输出, $i(x, y)$ 为图像, $*$ 为二维卷积运算。

如此一来,决定 Gabor 滤波器的就剩下高斯窗宽度 σ这一个参数了。σ不同,多分辨分析的效果是不同的。如在有关图像纹理分割(texture segmentation)的研究中,人们的研究重点就是确定 σ。

8.2　常用的感知评价方法

首先,由于中英文翻译上的差异和计算机专业的特定性,我们有必要澄清几个词。

在数字水印的性能评价中,水印的不可见性是十分重要的一个评价方面。在许多文献中,将不可见性也称之为透明性(invisibility)。但事实上,invisibility 是“看不见”的意思,而“透明”的英文是 transparence。由于在计算机专业中,“透明(transparence)”一词往往有特定的含义:所谓透明的,就是指由机器自动完成的而不需要人关心和了解的意思。比如在 CPU 中,我们说描述符寄存器是透明装入的,这里的透明

就是上述意思。我们在水印和隐秘性能的评价中所谈到的“透明”显然不是不需要人关心和了解的意思,而指的是不可感知,不容易被人发觉,对原始图像(载体)破坏不大等。所以,本书中我们为了避免歧义,将感知方面的性能统一称为不可见性(imperceptible),这是与 invisibility 的意思一致的。

此外,在数字水印部分,不可见性是针对原始作品和加有水印的作品两者之间的一个评价方面。如果将加有水印的作品和受到攻击的水印作品进行比较,对不可见性的评价往往衍生为对保真性(fidelity)的评价。总而言之,透明性、不可见性和保真性只是用词上的差异,三者并无较大区别。一般地说,不可见性和鲁棒性(robustness)是对立的,一个良好的水印系统一定是追求二者都达到相对最优的系统。

8.2.1 主观评价

所谓感知,从严格意义上讲本身就是一个属于人主观意识范畴的概念。所以对于图像的感知性能,人类的主观评价是十分有效和重要的。我们在前几章涉及到不可见性的评价中,都是给出一组结果,通过直接观察判断孰好孰坏,这实际上就是一种主观评价。在水印性能的主观评价中,有必要说明的是由于每个人视觉敏感度是不同的,同一个人在不同时间、不同年龄、不同环境下体现出的敏感度也是不同的。所以,主观评价往往需要大量的人群进行大量的实验才能说明问题,而且研究的结果也只能是针对特定人群有效。在本书中,我们请了 100 位同学帮助我们进行了水印不可见性的主观评价,他们的年龄都在 20 ~23 岁之间。

我们进行水印不可见性主观评价的实验是二选一迫选实验(Two Alternatives Forced Choice, TAFC, 参见附录 4)。在该实验中,我们使用 W-SVD 算法生成 10 幅加有水印的 lenna 图像,其水印强度依次提高(即在算法中 α从 0.05 取到 0.9,其他参数相同)。每一幅图像与原始的 lenna 图像构成一组,要求观察者必须在不知道两者差异的情况下判断出加有水印(感知质量低)的那一幅。当两幅图的品质接近时,被正确选择的概率是近乎 50% 的,也就是说预期得到的反应是随机的,观察者不能明确作出判断。随着水印强度的增大,被正确选择的概率会越来越大,最终达到 100%。

显然,随着水印强度的增大,原始作品与加有水印的作品之间的感知差异也越来越大。为了衡量这种差异,我们引入一个心理物理学上有关感知差异的概念——JND(Just Noticeable Difference,临界可觉察差异)来说明。

感知总是由外界刺激引起的,刺激的存在以及它的变化是感知产生和发生变化的重要条件。研究刺激和心理量之间的关系的学科称为心理物理学(psychological and physical research)。心理量与物理量之间的关系是用感受性的大小来说明的。检验感受性大小的基本指标称感觉阈值。感觉阈值是人感到某个刺激存在或刺激发生变化所需刺激强度的临界值,又分为绝对感觉阈值和差别感觉阈值。绝对感觉阈值指最小可觉察的刺激量,也就是有 50% 的可能被觉察到的最小刺激量。早期心理

物理学家研究总结得出，一般人视觉绝对感觉阈限相当于 30 英里（约 48 公里）以外的一烛光，那是相当小的。差别感觉阈值是指那种刚能引起差别感觉的两个刺激之间的最小差异量。研究发现，为了辨别一个刺激出现了差异，所需差异大小与该刺激本身的大小有关。心理物理学上的韦伯定律（Weber's law）指出，在一个刺激上发现一个最小可觉察的感觉差异所需要的刺激变化量与原有刺激的大小有固定的比例关系。这个固定比例对不同感觉是不同的。所以，差别感觉阈值又表示为刺激变化量与原有刺激量之间的固定比例关系。在刺激变化时所产生的最小感觉差异就是 JND。无论是绝对感觉阈值和差别感觉阈值都是因人而异的，它可以因对观察者训练或其他方式而改变。

在水印不可见性评价中，人们通常将在 TAFC 中有 50% 正确率的那一组原始图像和加有水印的图像之间的差异称为零 JND。由于 JND 是对刺激的一个最小变化的觉察量，那么就可以用它作为测量知觉变化的单位。具体计算出是几倍 JND 的方法并不统一。比如在后面介绍 Watson 水印算法时我们将了解到 Watson 对多倍 JND 的定义。

表 8.1 是我们找 100 个同学对 10 组 lenna 图像进行 TAFC 的结果，相应的曲线图如图 8.12 所示。图 8.13 是 α取 0.1 那一组的两幅图像，图 8.13（b）是加有水印的图像。两者的差异可以认为是零 JND。

表 8.1　　100 个同学进行 TAFC 的结果

水印强度（α取值）	正确人数	错误人数	正确率
0.05	48	52	48%
0.1	53	47	53%
0.2	61	39	61%
0.3	75	25	75%
0.4	96	4	96%
0.5	95	5	95%
0.6	100	0	100%
0.7	100	0	100%
0.8	99	1	99%
0.9	98	2	98%

前面我们已经说过，将 JND 看做是测量知觉变化的单位时，具体多倍 JND 的确定方法是不统一的。如我们可以将在 TAFC 中正确率为 75% 的那组图像的差异定义为 1 倍 JND。图 8.14 表明了这种定义下的感知差异。

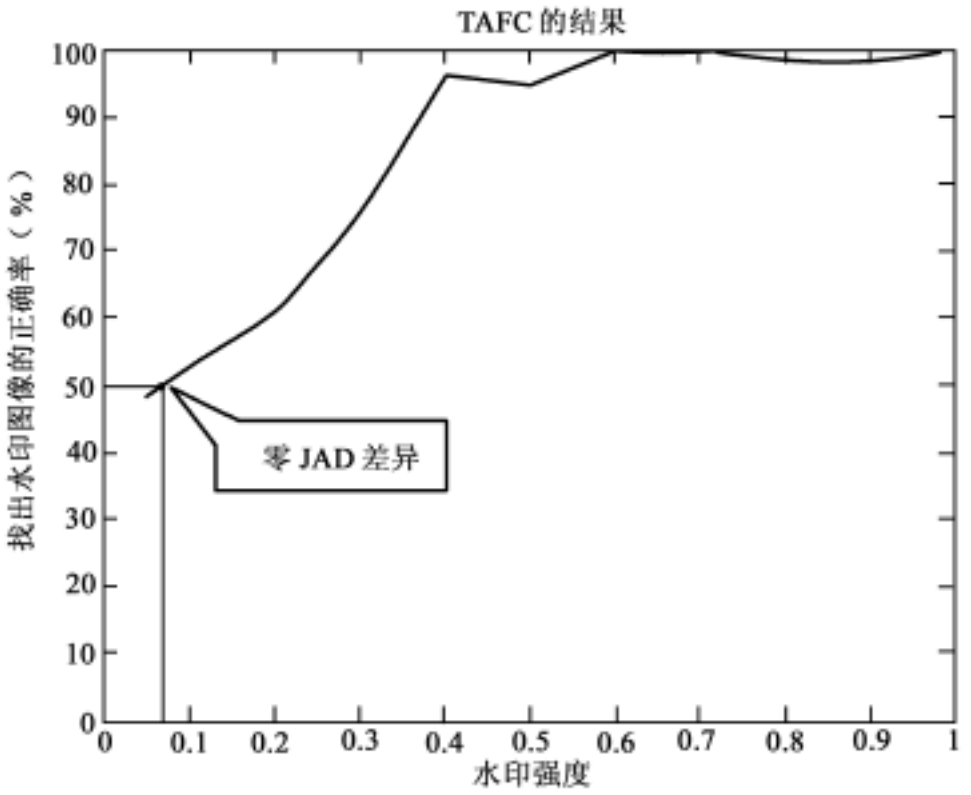


图 8.12　TAFC 的结果

图 8.13　(a) 与(b) 之间有零 JND 的差异(TAFC 的第二组实验图)

图 8.14　(a) 与(b) 之间有 1 倍 JND 的差异(TAFC 的第四组实验图)

除了 TAFC, 主观评价的另一个典型实验就是要求观察者将作品按照感知质量优劣排成等级。如对于电视图像的感知质量, ITU-R Rec. 500 有如表 8.2 的等级定义。

表 8.2　　ITU-R Rec. 500 中对图像品质的定义

等级(rating)	损伤程度(impairment)	品质(quality)
5	不可感知的(imperceptible)	Excellent
4	可感知, 但不妨碍观看 (perceptible, not annoying)	Good
3	轻微妨碍观看的(slightly annoying)	Fair
2	妨碍观看的(annoying)	poor
1	十分妨碍观看的(very annoying)	Bad

8.2.2　客观评价

主观评价存在的最大问题就是因人而异。在衡量一个水印系统的不可见性上, 我们希望有一种(多种) 客观的评价方法, 能够将感知差异量化为一定的数值, 通过数值大小的比较直接评定感知质量。我们将这种量化后的数值称为感知距离。这里, 我们选择性地介绍一些常用的和具有代表意义的图像感知质量客观评价策略。

(1) 均方差 MSE

MSE(Mean Square Error) 是最普遍的使用于图像感知质量评价的手段之一。MSE 可以直接反映出评估对象发生的改变。对于衡量加有水印信息的图像与原始图像在品质上的差异, 我们完全可以使用 MSE 作为一种估计的手段, 得到图像质量变化的客观指标。图像之间的 MSE 由以下公式计算得到:

$$MSE = \frac{1}{MN}\sum_{x=1}^{M}\sum_{y=1}^{N}\left(I(x,y) - I_w(x,y)\right)^2 \tag{8.10}$$

其中, $I(x,y)$ 表示原始图像各像素, $I_w(x,y)$ 表示加有水印的图像各像素。$M \times N$ 是图像的尺寸。

值得注意的是, 我们通常使用的图像是 RGB 图像。考虑到 RGB 像素是以一个 3 维矢量的形式出现的, 所以在本节的各种评价方法的实现中, 我们都是先将图像转化为灰度图像再进行计算的。换句话说, 我们计算的误差实际上是评价对象在亮度上的误差, 这种误差在图像感知质量中也是尤为值得重视的。

另外, 在 MATLAB 中, 对于图像的像素值, 我们在转换为 double 型计算时并不将其缩小到[0, 1] 区间, 即使用的是‘ image = double(image) ’命令而不是前面经常使用的‘ image = double(image) /255 ’命令, 这是为了保证计算结果有一定的数量级, 便于

区分。

编写函数 mse. m 完成两幅图像的 MSE 计算, 函数代码如下:

```
% 文件名: mse. m
% 程序员: 郭迟
% 编写时间: 2004. 3. 20
% 函数功能: 本函数将完成对输入图像的 MSE 计算
% 输入格式举例: msevalue = mse( 'c: \lenna. jpg', 'c: \test. jpg')
% 参数说明:
% original 为原始图像
% test 加有水印的图像
% msevalue 为两者均方差
function msevalue = mse( original, test) ;
% 读取图像并处理到亮度关系
A = imread( original) ;
A = rgb2gray( A) ;
A = double( A) ;
B = imread( test) ;
B = rgb2gray( B) ;
B = double( B) ;
% 判断输入图像是否有效
[ m, n] = size( A) ;
[ m2, n2] = size( B) ;
if m2 ~= m || n2 ~= n
    error( '图像选择错误') ;
end
% 计算 MSE
msevalue = 0;
for i = 1: m
    for j = 1: n
        msevalue = msevalue + ( A( i, j) - B( i, j) ) ^2;
    end
end
msevalue = msevalue / ( m* n) ;
disp( [ '输入数据的 MSE 为: ', num2str( msevalue) ] ) ;
```

结合我们在上一小节中进行 TAFC 的 10 组样本(每组有一幅原始图像和一幅加有水印的图像), 我们计算出其相应的 MSE 如表 8.3 所示。显然, 表 8.3 的数据与表

8.1 中我们进行主观评价的结果是一致的。

表 8.3　　10 组嵌有不同强度的水印图像与原始图像的 MSE

组号(水印强度)	MSE
1(0.05)	0.026672
2(0.1)	0.18658
3(0.2)	0.78499
4(0.3)	1.8336
5(0.4)	3.0933
6(0.5)	4.6069
7(0.6)	6.2051
8(0.7)	7.9844
9(0.8)	10.0265
10(0.9)	12.3536

MSE 的优点就是简单,便于理解。但在实际应用中,利用 MSE 去对图像感知质量进行评价存在很大的缺点。简而言之就是 MSE 会对图像的感知质量进行低估或高估。

所谓低估图像的感知质量,是指对在主观评价中普遍认为品质高的图像计算出的 MSE 过大,从而造成客观评价与主观评价不一致的现象。图 8.15 给出了四幅图像。其中图 8.15(a)是原始 lenna 图像,图 8.15(b)是对图 8.15(a)进行压缩率为 6%的 JPEG 压缩后的图像,图 8.15(c)是在 lenna 图像中加入高斯白噪声并利用第三章小波降噪的方法对其进行降噪后的图像,图 8.15(d)是在 lenna 图像的高频(包括部分中频) DCT 系数中加入随机噪声(类似蓝噪声)后的图像。经过计算我们发现:

$$MSE(a,b) = 151.7369$$
$$MSE(a,c) = 176.9037$$
$$MSE(a,d) = 148.9751$$

单从 MSE 这一指标上看,三者的感知质量是差不多的。但事实上,图 8.15(d)由于是对高频系数进行的调整,图像品质是明显好于图 8.15(b)和图 8.15(d)的。换句话说,MSE 在评定图 8.15(d)的感知质量时,过于低估了图 8.15(d)的品质。

所谓高估图像的感知质量,是指对在主观评价中普遍认为品质低的图像计算出的 MSE 过小,从而造成客观评价与主观评价不一致的现象。图 8.16 给出了两幅图

原始 lenna 图像

（a）原始图像

JPEG 压缩率为 6% 的 lenna 图像

（b） MSE=151.7369

全频加入高斯白噪并适当降噪后的 lenna 图像

（c） MSE=176.9037

加入高频噪的 lenna 图像

（d） MSE=148.9751

图 8.15　MSE 低估了图像的品质

像。图 8.16(a) 是对 lenna 图像进行压缩率为 8% 的 JPEG 压缩后的图像，图 8.16(b) 是将 lenna 图像整体向下平移一行(即图 8.16(b) 的第 2 行是原始 lenna 图像的第 1 行……依此进行的) 后的图像。经过计算得到:

$$MSE(lenna, a) = 116.5538$$

$$MSE(lenna, b) = 127.8253$$

单从 MSE 这一指标上看，图 8.16(a) 的感知质量应高于图 8.16(b)，而事实上这是明显不符合实际的，也就是说，MSE 过于高估了图 8.16(a) 的品质。

所谓“低估”与“高估”实际上是一个相对的概念，主要是评价的对象不同罢了。当我们用 MSE 衡量一个主观评价低的图像时就可能会造成高估，反之就可能造成低估。

(2) 信噪比 SNR 和峰值信噪比 PSNR

在一定意义上说，SNR(Signal to Noise Ratio) 与 PSNR(Peak Signal to Noise Ratio)

图 8.16　MSE 高估了图像的品质

是最通行的评定信号品质的指标。如在通信系统中，输出 SNR 与输入 SNR 的比例就构成了系统增益，增益越大系统越优。由于水印模型是与通信系统模型紧密联系的（这一点在第七章已经阐述），相对原始作品来说，水印信号可以认为是随机噪声。有噪声就会影响原始作品的品质，也自然存在 SNR 和 PSNR 这些指标。理论上的 SNR 与 PSNR 都应该是针对信号功率和噪声功率来说的，而我们这里的定义有所不同。在图像处理和水印不可见性评价中，用以下公式定义加有水印的图像的 SNR 和 PSNR：

$$\mathrm{SNR(dB)} = 10\log_{10}\frac{\sum_{x=1}^{M}\sum_{y=1}^{N} I(x,y)^2}{\sum_{x=1}^{M}\sum_{y=1}^{N}\left(I(x,y) - I_W(x,y)\right)^2} \tag{8.11}$$

$$\mathrm{PSNR(dB)} = 10\log_{10}\frac{D^2 MN}{\sum_{x=1}^{M}\sum_{y=1}^{N}\left(I(x,y) - I_W(x,y)\right)^2} \tag{8.12}$$

其中，D 是信号的峰值。对于一个 8 位图像来说，每个像素值的峰值就是 255，我们这里 D 都等于 255（如果将像素值归一化到[0,1]区间，则有 D=1；若是针对 16 位图，则 D=65 535）。式(8.11)与式(8.12)均取 10 倍的以 10 为底的对数，目的是将计算出来的值换算成分贝(dB)这一标准单位。

在具体应用中，我们一般采用 PSNR 代替 SNR，这是因为 SNR 的计算复杂度要大一些。主观上可以容忍的图像的 PSNR 值都在 20dB 以上。通过计算原始图像与加有水印的图像之间的 SNR 或 PSNR，可以在一定程度上对不可见性作出评价。

稍加留意不难发现，SNR 与 PSNR 与 MSE 有十分相同的部分。事实上，式(8.11)与式(8.12)用 MSE 作为参数可以直接表示为：

$$SNR(MSE) = 10\log_{10}\frac{\sum_{x=1}^{M}\sum_{y=1}^{N} I(x,y)^2}{MN \cdot MSE}$$

$$PSNR(MSE) = 10\log_{10}\frac{D^2}{MSE} \tag{8.13}$$

可以看到, MSE 在评定图像品质的时候出现的问题, SNR 与 PSNR 同样具有。此外, 在有些文献上, 将 PSNR 定义为:

$$PSNR(MSE) = 10\log_{10}\frac{D}{\sqrt{MSE}}$$

但由于我们关心的只是一个比例, 所以, 这两种定义不存在谁对谁错的问题, 尽管二者之间存在一个 2 倍关系。

编写函数 snr. m 完成计算图像 SNR 和 PSNR 的实验, 函数代码如下:

```
% 文件名: snr. m
% 程序员: 郭迟
% 编写时间: 2004. 3. 21
% 函数功能: 本函数将完成对输入图像的信噪比、峰值信噪比计算
% 输入格式举例: [ snrvalue, psnrvalue] = snr( 'c: \lenna. jpg', 'c: \test. jpg')
% 参数说明:
% original 为原始图像
% test 为加有水印的图像
% snrvalue 为两者信噪比
% psnrvalue 为两者峰值信噪比
function [ snrvalue, psnrvalue] = snr( original, test) ;
% 调用函数, 计算 MSE
msevalue = mse( original, test) ;
if msevalue == 0
        error( '图像选择错误') ;
end
% 计算原始图像的信号功率
A = imread( original) ;
A = rgb2gray( A) ;
A = double( A) ;
[ m, n] = size( A) ;
signal = 0;
for i = 1: m
        for j = 1: n
```

```
                signal = signal + A(i,j)^2;
            end
        end
        signal = signal/(m*n);
        % 计算信噪比,峰值信噪比
        snrvalue = signal/msevalue;
        snrvalue = 10*log10(snrvalue);
        psnrvalue = 255^2/msevalue;
        psnrvalue = 10*log10(psnrvalue);
        disp(['待测图像的信噪比为:',num2str(snrvalue),'dB']);
        disp(['待测图像的峰值信噪比为:',num2str(psnrvalue),'dB']);
```

与表 8.3、图 8.15 和图 8.16 对应,我们将前面计算过 MSE 的图像用 SNR 和 PSNR 衡量一次,结果如表 8.4 所示。

表 8.4　　MSE 与 SNR、PSNR

内　　容		MSE	SNR(dB)	PSNR(dB)
TAFC 的 10 组图像(与表 8.1 对应)	1	0.026672	57.2072	63.8702
	2	0.18658	48.7591	55.422
	3	0.78499	42.5192	49.1822
	4	1.8336	38.8348	45.4978
	5	3.0933	36.5636	43.2266
	6	4.6069	34.8338	41.4967
	7	6.2051	33.5403	40.2033
	8	7.9844	32.4454	39.1084
	9	10.0265	31.4563	38.1193
	10	12.3536	30.5499	37.2129
图 8.15	b	151.7369	19.6569	26.3199
	c	176.9037	18.9905	25.6534
	d	148.9751	19.7367	26.3997
图 8.16	a	116.5538	20.8026	27.4655
	b	127.8253	20.4017	27.0646

(3)加权信噪比 WSNR

无论是 MSE 还是 SNR,PSNR,在衡量图像感知质量时只关注了图像信号本身固有的一些统计特性,而没有与 HVS 联系起来,达到主客观评价完全一致的要求。如何将客观评价方法尽可能地与 HVS 结合起来,是当前在图像感知评价领域非常受关注的一个问题。下面我们将介绍一种有对比敏感函数 CSF 滤波参与的信噪比计算方法。得到的信噪比就称为 WSNR(Weighted Signal to Noise Ratio)。

在早期的图像感知质量评价中,通常使用的方法基本上都是按照以下几个步骤进行的:

① 将原始图像与待测图像的区别计算出来,这种区别被称为错误图像(error image)。

② 将错误图像经过 CSF 滤波。

③ 按照信噪比的形式计算一个度量值。

根据选用的 CSF 的不同,这种度量方式可以考虑到观察距离、亮度变化等方面的影响,但对于对比掩蔽、亮度掩蔽等方面则无能为力。WSNP 就是比较典型的一种。

对于 WSNR 的具体实现方式,也有不同的理解。我们是利用前面的 CSF2 完成 WSNR 的计算实验,函数代码如下:

```
% 文件名: wsnr. m
% 程序员: 郭迟
% 编写时间: 2004. 3. 21
% 函数功能: 本函数将完成对输入图像的 WSNR 计算
% 输入格式举例: wsnrvalue = wsnr( 'c: \lenna. jpg', 'c: \test. jpg')
% 参数说明:
% original 为原始图像
% test 为加有水印的图像
% wnrvalue 为两者加权信噪比
function wsnrvalue = wsnr( original, test)
% 读取图像并处理到亮度关系
A = imread( original) ;
A = rgb2gray( A) ;
A = double( A) ;
B = imread( test) ;
B = rgb2gray( B) ;
B = double( B) ;
% 判断输入图像是否有效
[ m, n] = size( A) ;
[ m2, n2] = size( B) ;
if m2 ~= m || n2 ~= n
    error( '图像选择错误') ;
end
if A == B
    error( '图像选择错误') ;
```

```
end
% 计算失真度
e = A - B;
% CSF 滤波
filtercoefficients = csf;
result = filter2( filtercoefficients, e) ;
% 计算信噪比
wsnrvalue = 10* log10( ( 255^2) /( mean( mean( result^2) ) ) ) ;
disp( [ '待测图像的 WSNR 为: ' num2str( wsnrvalue1) , 'dB' ] ) ;
```

下面我们来看看 WSNR 相对于 PSNR 改进的地方。前面已经说过，由于 MSE 存在感知高(低)估的问题，PSNR 并不能满足主客观一致的要求。仍然以表 8.4 所列的图像为评价对象，我们具体来看一下计算出的 WSNR 是否对 PSNR 有一定改观(如表 8.5 所示)。

表 8.5　　PSNR 和 WSNR

评价对象		PSNR	WSNR
图 8.15	b	26.3199	36.3125
	c	25.6534	35.2346
	d	26.3997	60.4151
图 8.16	a	27.4655	37.6706
	b	27.0646	52.4243

很明显，WSNR 在主客观一致的方面明显强于 PSNR。

WSNR 粗略地应用了部分的视觉特性，评价的结果与主观评价更为接近。但事实上，对于不同类型的图像以及不同的图像编码方式，它还不具备足够优秀的图像感观质量预测能力。WSNR 的基础是 CSF，而 CSF 可以认为是 HVS 的线性空间不变(linear spatially invariant)的系统，对于非线性和空间变化的影响无法估量。它不能模拟由于其他空间频率的振幅带来的对比变化，同时也忽略了局部背景带来的掩蔽效应。

对于加噪图像，Thomas D·kite 等人曾分析了错误图像与原始图像的相关系数对 WSNR 的影响。从理论上讲，错误图像只应该是由噪声构成的，其和原始图像的相关系数应该为 0。但事实上，通过计算可以发现，相关系数并不为 0，这就是信噪比计算不精确的原因。如果用以下的方式进行图像加噪

$$J = \alpha I + N$$

其中，I 为原始图像，N 为白噪声模板，α是比例因子，J 是加噪后的图像，那么错误图像就应该为：

$$E = J - I = (\alpha - 1)I + N$$

实验表明,当 α从 1.0 变化到 1.03 时,计算出的 J 与 I 的 SNR 和 WSNR 都变化了 2 ~3dB。而当 α等于 1.03 时其相关系数仅为 0.115。也就是说,错误图像与原始图像越相关,计算出的 WSNR(SNR, PSNR) 越小,越容易造成评价低估。

在第七章的水印基本模型中,我们提到过有一种水印模型本身就是根据原始图像的有关性质生成水印。将水印信号看做是噪声,则此时的错误图像与原始图像的相关系数肯定是不为 0 的。所以,在水印不可见性的评价中,无论是 SNR, PSNR 还是 WSNR,我们都不能认为它们计算出的结果是精确的,只有当其与主观评价一致时,才能说明问题。

(4) 掩蔽峰值信噪比 MPSNR

Christian J. Van 等人在有关运动图像的感知质量评价体制(Moving Pictures Quality Metric, MPQM) 中提出了 MPSNR(Masked Peak Signal to Noise Ratio) 的概念。由于运动图像需要考虑到视觉的时间特性等因素,其评价方法比较复杂。Iván Kopilovic 等人也将其修改后用于静止图像的评价中。顾名思义, MPSNR 考虑到了图像的掩蔽效应,因而是更为理想的图像感知质量评价的方法。事实上, MPSNR 利用了多通道滤波、掩蔽计算、误差求和等多种手段,很符合当前图像质量评价的趋势和特点。我们简要的说明一下 MPSNR 的计算。MPSNR 的计算机制,如图 8.17 所示。

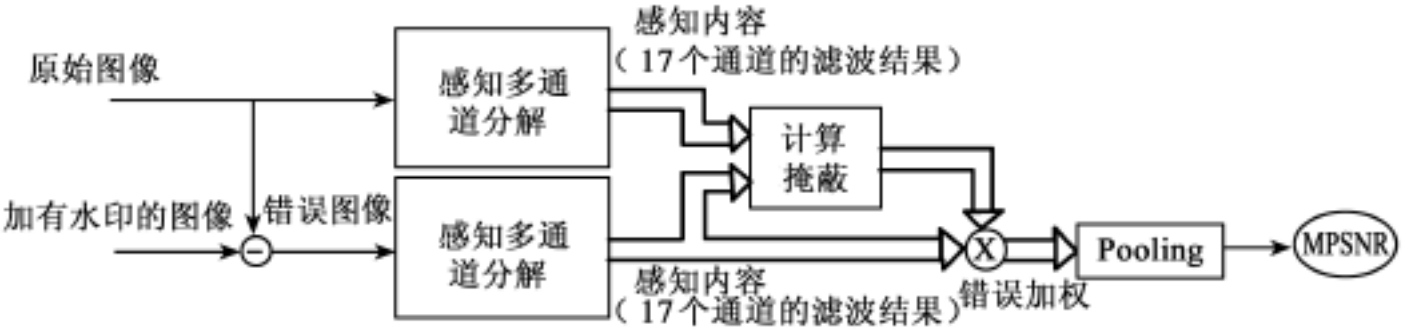


图 8.17　MPSNR 的计算机制

1) 感知多通道分解

Christian J. Van 等人使用的感知多通道分解的方法就是前面我们提到的 Gabor 滤波(在 MPQM 中除了 17 个 Gabor 滤波器外还使用了暂态、稳态两个时间滤波器) ,滤波器共有 17 个,径向频率按倍频程取为 0, 2, 4, 8, 16,方向为 0, π/4, π/2, 3π/4。

2) 计算掩蔽

对于多通道分解出来的结果,逐一对应地计算掩蔽。即对同一通道的原始图像分解结果和错误图像分解结果进行计算,这里是将原始图像看做是错误图像的掩蔽。也就是将错误图像看做刺激,原始图像看做背景。对于错误图像经过分解滤波后的每一个通道的结果,取其中的每一个像素逐一与可见度阈值 C_T 进行比较,将可见度阈值 C_T 认为是一个 JND 单位,求得一个 JND 的倍数关系。结合图 8.7 和式(8.1) 所

示有关 HVS 的掩蔽特性,我们具体来分析掩蔽的计算。

对应式(8.1),每个像素的可见度阈值 C_T 有:

$$C_T(x,y) = \begin{cases} C_{T0} & (C_M < C_{T0}) \\ C_{T0}(C_M(x,y)/C_{T0})^{\varepsilon} & (C_M > C_{T0}) \end{cases}$$

其中,C_{T0} 是信号在没有掩蔽情况下的最小感知阈值,我们通过一个与具体通道有同样的径向频率和方向等参数的 CSF 可以得到。$C_M(x,y)$ 是原始滤波图像与错误滤波图像逐一地将像素相比得到的对比图像。从而我们可以计算获得 C_T。对于错误图像的每一个像素(x,y),都计算:

$$\frac{(x,y)}{C_T(x,y)} = N(JNDS)$$

N 是得到的一个具体的实数。得到了每一个像素的 JND 倍数,也就实际上得到了原始图像对错误图像的视觉影响。相应地,将这些数据乘到错误图像上,完成错误加权。

3) Pooling

Pooling 是一个集合数据进行高水平质量评价的一种操作,一般来说包括以下几个步骤:首先,Pooling 对加权后的错误图像进行分块。所分的块在运动图像中是一个三维分量(空间水平、空间垂直和时间)。时间分量关注的是信号在视网膜上停留的时间,空间分量关注的是我们对图像观察时的视觉焦点。在静止图像中,所分的块当然只包括两个空间分量。原则上,块的大小与视觉焦点要统一,一般为在水平和垂直方向上覆盖 2°视角,这是由人眼视网膜中央凹的特性所决定的。Iván Kopilovic 在其研究中采用的是这样一组参数,可供大家参考,如表 8.6 所示。

表 8.6　　Iván Kopilovic 用于图像质量评价的观察指标

项　　目	指　　标
观察距离	0.5m
像素/英寸	72
像素/度	24.74
像素大小(度)	0.04042

视觉真正关注的实际上是图像中的一块,参与计算的也是其中的一块。当然,为了简便,也可以不分块。

Pooling 对每个通道的加权错误图像的每块进行误差合并。合并的方法是求明科斯基和。

$$E = \left(\frac{1}{N} \sum_{c=1}^{N} \left(\frac{1}{N_x N_y} \sum_{x=1}^{N_x} \sum_{y=1}^{N_y} \left| e[x,y,c] \right| \right)^{\beta} \right)^{\frac{1}{\beta}} \qquad (8.14)$$

其中, e[x, y, c] 是(x, y) 在通道 c 下当前块中的处于(x, y) 位置的加权错误像素值。N_x 和 N_y 是块或图像的大小。N 是通道的数量。对应我们前面的例子, N = 17。β为求和指数, 一般取 4。

4) 计算掩蔽峰值信噪比

E 为类似 MPQM 计算出的具有掩蔽特性的失真度。为了便于理解, 一般再经过以下处理, 将 E 转化为信噪比的形式, 称为 MPSNR。因为 MPSNR 仍然取 dB 为单位, 所以同时也将以 dB 为单位的 MPSNR 称为视觉分贝(visual decibels, vdB 或) 。视觉分贝有:

$$\mathrm{MPSNR(dB)} = 10\log_{10}\frac{255^2}{E^2} \tag{8.15}$$

5) 计算视觉感知质量

将误差合并后的这一客观评价的结果与 ITU-R Rec5.00 的主观评价等级制度相结合, 就得到了所谓的视觉感知质量 Q:

$$Q = \frac{5}{1 + \gamma E} \tag{8.16}$$

这里, Q 是视觉质量, 取值在 1 ~5 之间。γ是一标准常数。γ的选取一般取决于参照的视觉模型。一般地说, 假设水印图像在每一个通道中的每一块的每一个像素都只含有一个 JND 的感知误差, 这时的水印在理论上就有最小的不可见性, 此时的视觉质量 Q 就应该处于一个高等级(4 级以上) 。假设将这时的视觉质量定义为 4.99, Christian J. Van 给出了一个 γ的取值, 此时, γ= 0. 623。

MPSNR 的求解比较复杂。大家可以到国际信号处理协会(ITS) 的官方网站上下载有关 MPQM 的评价程序, 其地址为: http: / /ltswww. epfl. ch / 。

8.3 Watson 基于 DCT 的视觉模型

Watson 提出的基于 DCT 的视觉模型是一个经典的综合敏感度、掩蔽和误差合并的感知模型。与前面我们提到的方法不同的是, Watson 采用 DCT 取代了多通道分解, 与 JPEG 和一些水印算法能很好地结合。

Watson 模型由以下几个部分组成: 对比敏感表、亮度掩蔽、对比度掩蔽和感知质量度量。

8.3.1 对比敏感表

该模型定义了一个对比敏感表, 表中每一个数字代表在不存在掩蔽的情况下可被感知的最小 DCT 系数, 并将这个系数认为是 1 个 JND。数值越小说明人眼对该频率越敏感。对比敏感表由以下方法(可以认为也是一种 CSF) 计算出来。

对于每一个表元素 $t_{i,j}$, 有:

$$\log_{10} t_{i,j} = \log_{10} \frac{T_{min}}{r_{i,j}} + k(\log_{10} f_{i,j} - \log_{10} f_{min})^2 \tag{8.17}$$

其中，$r_{i,j} = r + (1 - r)\cos^2 \theta_{i,j}$。式(8.17)中有几个参数 T_{min}，k，f_{min} 和 $f_{i,j}$ 分别由式(8.18)，式(8.19)，式(8.20)和式(8.21)给出，它们都是平均亮度 L 的函数。具体地说，T_{min} 是在阈值频率 f_{min} 下的亮度阈值，k 决定了抛物线的开口。

$$T_{min} = \begin{cases} \dfrac{L}{S_0} & L > L_T \\ \dfrac{L}{S_0}\left(\dfrac{L_T}{L}\right)^{1-a} & L \leq L_T \end{cases} \tag{8.18}$$

其中，L_T 取为 $13.45 cd/m^2$。cd/m^2 是亮度的单位，以每单位面积上的发光强度来表示。发光强度的单位就是国际单位坎得拉(cd)。$S_0 = 94.7$，$a = 0.649$。

$$k = \begin{cases} k_0 & L > L_k \\ k_0\left(\dfrac{L}{L_k}\right)^b & L \leq L_k \end{cases} \tag{8.19}$$

其中，$L_k = 300 cd/m^2$，$k_0 = 3.125$，$b = 0.0706$。

$$f_{min} = \begin{cases} f_0 & L > L_f \\ f_0\left(\dfrac{L}{L_f}\right)^c & L \leq L_f \end{cases} \tag{8.20}$$

其中，$L_f = 300 cd/m^2$，$f_0 = 6.78 cpd$，$c = 0.182$。

$f_{i,j}$ 是空间频率，与 DCT 变换结合，可以得到：

$$f_{i,j} = \frac{1}{16}\sqrt{\left(\frac{i}{W_x}\right)^2 + \left(\frac{j}{W_y}\right)^2} \tag{8.21}$$

其中，W_x 和 W_y 分别是水平和垂直方向上 1 个像素占有的视觉度数。

最后，r 一般取 0.7。$\theta_{i,j}$ 与 $f_{i,j}$ 有关，表示为：

$$\theta_{i,j} = \arcsin \frac{2 \cdot f_{i,0} \cdot f_{0,j}}{f_{i,j}^2} \tag{8.22}$$

Ingemar J. Cox 等人给出了一个具体的对比敏感表，如表 8.7 所示。

表 8.7　　Watson 模型下的一种对比敏感表(对应 8× 8DCT 块)

1.40	1.01	1.16	1.66	2.40	3.43	4.79	6.56
1.01	1.45	1.32	1.52	2.00	2.71	3.67	4.93
1.16	1.32	2.24	2.59	2.98	3.64	4.60	5.88
1.66	1.52	2.59	3.77	4.55	5.30	6.28	7.60
2.40	2.00	2.98	4.55	6.15	7.46	8.71	10.17
3.43	2.71	3.64	5.30	7.46	9.62	11.58	13.51
4.79	3.67	4.60	6.28	8.71	11.58	14.50	17.29
6.56	4.93	5.88	7.60	10.17	13.51	17.29	21.15

8.3.2 亮度掩蔽

Watson 认为，在 8× 8 块中，如果像素的平均亮度大，那么 DCT 系数可以进行较大幅度的修改。他利用每个 DCT 块的 DC 系数（直流系数代表平均能量）和对比敏感表，求得了相应的亮度掩蔽阈值。对于每个 DCT 块，有：

$$t_{L_{i,j}} = t_{i,j}\left(\frac{C_k}{\overline{DC}}\right)^a \tag{8.23}$$

其中 $t_{i,j}$ 就是表 8.7 中的数据，C_k 是当前 DCT 块的 DC 系数，$\overline{DC}$ 为原始图像的 DCT 变换矩阵中的平均 DC 系数或者一个预期的显示亮度。a 与式(8.18)一致，一般取为 0.649。

从式(8.23)可以看出，在一幅图像中，比较明亮的区域可以进行较大的修改而不被感知。编写函数 lummask.m 完成亮度阈值的计算，函数代码如下：

```
% 文件名: lummask.m
% 程序员: 郭迟
% 编写时间: 2004.4.2
% 函数功能: 本函数将完成 Watson 模型下图像亮度阈值的计算
% 输入格式举例: lumthreshold = lummask('c:\lenna.jpg')
% 参数说明:
% image 为输入图像
% lumthreshold 为输出矩阵
function lumthreshold = lummask(image);
% 读取图像转亮度
i = imread(image);
i = double(i);
i = rgb2gray(i);
% 分块 DCT 变换
T = dctmtx(8);
DCTcoef = blkproc(i,[8 8],'P1*x*P2',T,T');
% 计算平均 DC 系数
[m,n] = size(DCTcoef);
meandc = 0;
count = 0;
for i = 0:ceil(m/8 - 1)
    for j = 0:ceil(n/8 - 1)
        meandc = meandc + DCTcoef(8*i + 1,8*j + 1);
```

```
            count = count + 1;
        end
end
meandc = meandc/count;
% 计算亮度掩蔽
fun = @blocklum; % 调用子函数
lumthreshold = blkproc( DCTcoef, [8 8], fun, meandc);
function result = blocklum( matrix, meandc);
% 敏感表
t = [1.40  1.01  1.16  1.66  2.40  3.43  4.79  6.56
     1.01  1.45  1.32  1.52  2.00  2.71  3.67  4.93
     1.16  1.32  2.24  2.59  2.98  3.64  4.60  5.88
     1.66  1.52  2.59  3.77  4.55  5.30  6.28  7.60
     2.40  2.00  2.98  4.55  6.15  7.46  8.71  10.17
     3.43  2.71  3.64  5.30  7.46  9.62  11.58 13.51
     4.79  3.67  4.60  6.28  8.71  11.58 14.50 17.29
     6.56  4.93  5.88  7.60  10.17 13.51 17.29 21.15];
% 计算
for i = 1:8;
    for j = 1:8
        result(i, j) = t(i, j) * (matrix(1, 1)/meandc)^0.649;
    end
end
```

图 8.18 直观地显示了 lenna 的亮度掩蔽阈值，明亮的地方表示亮度掩蔽高的块。

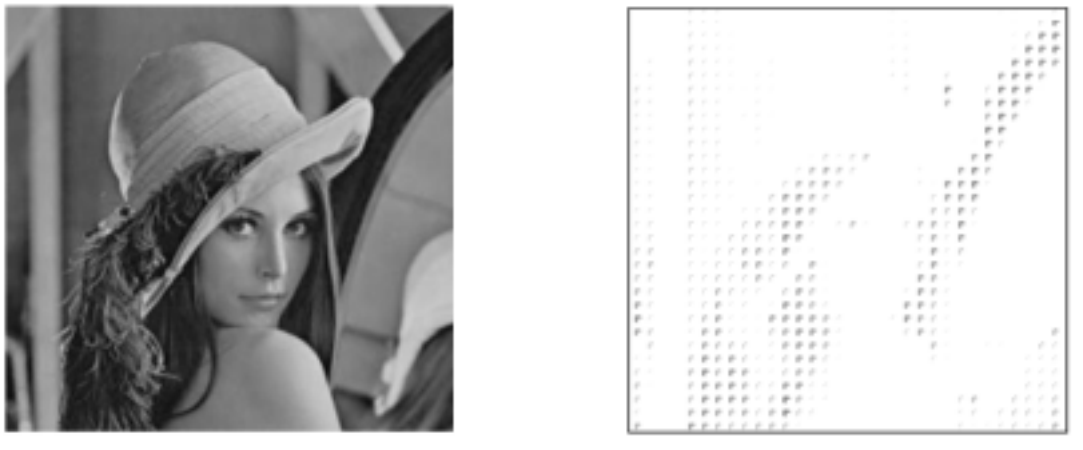


图 8.18　lenna 的亮度掩蔽

8.3.3 对比度掩蔽

亮度掩蔽阈值的取值要受到对比度掩蔽的影响，对比度掩蔽存在一个对比掩蔽阈值。对于每一个 DCT 块，有：

$$t_{C_{i,j}} = \max\{t_{L_{i,j}}, |DCTcoef_{i,j}|^{w_{i,j}} \cdot t_{L_{i,j}}^{1-w_{i,j}}\} \tag{8.24}$$

其中，$w_{i,j}$是一个常数，因频率系数不同而不同，决定对比掩蔽的度数。一般情况下，w_{00}取为 0，其他的 $w_{i,j}$取为 0.7（也可以全取为 0.7）。$DCTcoef_{i,j}$是每块的 DCT 系数。最终求得的对比阈值是亮度阈值和经 $w_{i,j}$调整的系数结果中最大的一个。$t_{C_{i,j}}$表明了每块 DCT 系数中各项在一个 JND 范围内可调整的大小，一般地又将这些阈值称为间隙。

编写函数 Watsonthreshold.m 完成图像间隙的计算，函数代码如下：

```
% 文件名: Watsonthreshold.m
% 程序员: 郭迟
% 编写时间: 2004.4.2
% 函数功能: 本函数将完成 Watson 模型下图像对比阈值的计算
% 输入格式举例: threshold = Watsonthreshold('c:\lenna.jpg')
% 参数说明:
% image 为输入图像
% threshold 为输出矩阵
function threshold = Watsonthreshold(image)
% 读取图像转亮度
i = imread(image);
i = double(i);
i = rgb2gray(i);
% 分块 DCT 变换
T = dctmtx(8);
DCTcoef = blkproc(i,[8 8],'P1*x*P2',T,T');
% 调用函数计算亮度阈值
lumthreshold = lummask(image);
% 计算 W(ij) 的修正值
[m,n] = size(DCTcoef);
for i = 1:m
    for j = 1:n
        another(i,j) = (abs(DCTcoef(i,j))/0.7) * (lumthreshold(i,j)/0.3);
    end
end
```

```
% 计算对比阈值
threshold = zeros( [ m n] ) ;
for i = 1: m
    for j = 1: n
        if lumthreshold( i, j) < = another( i, j)
            threshold( i, j) = another( i, j) ;
        else
            threshold( i, j) = lumthreshold( i, j) ;
        end
    end
end
```

8.3.4　感知质量度量

Watson 的感知质量度量方法分以下几个步骤完成:

① 计算原始图像与待测图像的 DCT 系数差, 求得 DCT 系数的错误图像 E。

② 将 E 除以间隙, 得到每一个像素的感知误差 $d_{i,j}$。$d_{i,j}$表明了第(i, j) 个频率以 JND 为单位下的误差, 是 JND 的倍数或分数。

③ 用明科斯基和对误差进行合并。合并公式为:

$$D(I, I_w) = \left(\sum_{i,j} | d_{i,j} |^{\beta} \right)^{\frac{1}{\beta}} \tag{8.25}$$

与前面提到的一样, β取 4。

编写函数 Watsondistorsion. m 完成图像的感知质量度量实验。函数代码如下:

```
% 文件名: Watsondistorsion. m
% 程序员: 郭迟
% 编写时间: 2004. 4. 2
% 函数功能: 本函数将完成 Watson 模型下图像感知质量度量
% 输入格式举例: result = Watsondistorsion( 'c: \lenna. jpg', 'c: \test. jpg')
% 参数说明:
% original 为原始图像
% test 加有水印的图像
% result 为两者误差估计
function result = Watsondistorsion( original, test)
% 读取图像转亮度
io = imread( original) ;
io = rgb2gray( io) ;
```

```
io = double( io) ;
iw = imread( test) ;
iw = rgb2gray( iw) ;
iw = double( iw) ;
% 分块 DCT 变换求错误图像
T = dctmtx( 8) ;
DCTcoefo = blkproc( io, [ 8 8] , 'P1* x* P2', T, T') ;
DCTcoefw = blkproc( iw, [ 8 8] , 'P1* x* P2', T, T') ;
e = DCTcoefw - DCTcoefo;
% 求间隙
threshold = Watsonthreshold( original) ;
% 计算 JND 倍数
[ m, n] = size( e) ;
d = zeros( [ m n] ) ;
for i = 1: m
        for j = 1: n
            d( i, j) = e( i, j) / threshold( i, j) ;
        end
end
% 误差合并
distortion = 0;
for i = 1: m
        for j = 1: n
            d( i, j) = d( i, j) ^4;
            distortion = distortion + d( i, j) ;
        end
end
result = sqrt( sqrt( distortion) ) ;
disp( [ '图像的感知误差为:', num2str( result) ] ) ;
```

图 8.19 是图 8.15 在 Watson 模型下评价出来的感知质量。Watson 模型下求出来的感知质量本身就可以理解为一个以 JND 为单位的量值。与信噪比方法不一样，这里数值越小表明评估的图像质量越高。

图 8.19　Watson 模型对图像质量的评估

8.4　感知自适应水印初步

自适应(self-adaptive) 水印是当前数字水印研究的热点。一般来说, 自适应水印都是与 HVS 相结合的, 称为感知自适应水印。我们知道, 一般的水印算法都会有一个(组) 水印强度的参数, 我们简单地记为 α。α越大水印的鲁棒性越强, 当然其不可见性就越差。早期的水印算法都是通过大量实验的结果, 人为地指定 α的取值, 以达到鲁棒性和不可见性相统一。随着研究的不断深入, 人们发现既然图像的感知质量可以用一些客观评价方法计算出来, 那何不将这种客观评价方法与水印算法结合起来, 让算法自动选取比较合适的水印强度参数 α呢? 我们下面介绍的就是利用 Watson 感知模型设计的简单自适应数字水印算法以及其实现。

在谈“ 自适应 ”之前, 我们先来简单地介绍一下这里使用的水印算法。

我们的水印模板取长度为 64 的高斯随机数以模拟高斯白噪。水印模型采用第七章介绍的基本模型。将原始图像做 8× 8 的分块 DCT, 并将 DCT 系数按照 zigzag 方法重新排列, 将生成的高斯噪声加入到 DCT 系数中去。设原始图像为 I, 加有水印的图像为 I_w, 水印信号为 W, 珚珚分别表示其 DCT 变换后的 DCT 矩阵, 则水印算法可以由式(8.26) 描述:

$$珚 = 珚 + \alpha W \tag{8.26}$$

其中, α就是我们重点要考虑的强度因子。

在 8.3 节我们详细地介绍了 Watson 提出的感知模型和图像质量评估方法, 现作以下推导:

由式(8.25) 得 $$D(I, I_w) = \left[\sum_{i,j} \left(\frac{珚 - 珚}{t_{C_{i,j}}} \right)^4 \right]^{\frac{1}{4}}$$

又因为 $$珚 = 珚 + \alpha W$$

所以 $$D(I, I_w) = \alpha \left[\sum_{i,j} \left(\frac{W}{t_{C_{i,j}}} \right)^4 \right]^{\frac{1}{4}}$$

$$= \alpha \cdot D(I, I_W')$$

其中 I_W'是当 α= 1 时的水印图像。可以发现, Watson 感知质量 $D(I, I_w)$ 实际上是 α的函数。对于种子一定的高斯水印模板, $D(I, I_W')$ 是一定的, 要改变水印强度就必须调整 α。当我们预先设定一个期望的感知质量 D_{equal} 时, α就可以由以下公式计算出来:

$$\alpha = \frac{D_{equal}}{D(I, I_W')} \tag{8.27}$$

根据期望的感知质量 D_{equal} 自动的对水印强度 α进行调整, 这就是全局自适应算法。

在具体实现中, 这里的全局自适应算法应注意以下两个问题:

①期望的感知质量 D_{equal} 如何选取。一般的根据主观评价的结果, 将零 JND 对应的图像与原始图像进行计算感知质量, 经过一定样本计算后求平均, 可以得到一个相对较优的期望质量。如我们在 8.2 节对 lenna 的主观评价中就得到过零 JND 的图像, 其与原始图像的 Watson 质量经过计算为 7 左右。在这里, 我们取 D_{equal} = 4 ～7 比较合适。

②考虑到舍入和修剪等细微操作都会对 α强度下的水印图像带来不同的 JND 值, 所以有必要在求得 α后, 再从 0.2α到 1.1α之间求取一定数量的水印图像, 找到视觉质量最好的结果。

编写函数 globaladaptive. m 完成实验, 函数代码如下:

```
% 文件名: globaladaptive. m
% 程序员: 郭迟
% 编写时间: 2004. 4. 7
```

```
% 函数功能: 本函数将完成 Watson 模型下全局自适应水印
% 输入格式举例: [ result, alf]  = globaladaptive( 'c: \lenna. jpg', 0. 1, 1983, 'c: \wm. jpg')
% 参数说明:
% image 为输入原始图像
% equal 为期望的感知质量
% seed 为随机数种子
% goal 为结果存放的地址
% result 为加有水印的结果
% alf 为求得的全局强度
function [ imagergb, alf]  = globaladaptive( image, equal, seed, goal)
% 读取图像做 DCT 变换
imagergb = imread( image) ;
imagergb = double( imagergb) ;
imagergb1 = imagergb; % 求基本误差时使用
imagergb2 = imagergb/255; % 显示时使用
r = imagergb(:,:, 1) ; % 往 R 层加水印
T = dctmtx( 8) ;
DCTcoef = blkproc( r, [ 8 8] , 'P1* x* P2', T, T') ;
% 生成长度为 64 的高斯随机数作为水印模板
rand( 'seed', seed) ;
wm = randn( 1, 64) ;
% 调用子函数, 给图像加 α为 1 时的水印
fun = @ wmadd;
result = blkproc( DCTcoef, [ 8 8] , fun, wm, 1) ;
% DCT 反变换
T = dctmtx( 8) ;
r = blkproc( result, [ 8 8] , 'P1* x* P2', T', T) ;
imagergb1(:,:, 1)  = r;
imagergb1 = imagergb1 /255; % 结果归一化才能存储
imwrite( imagergb1, 'temp. jpg') ;
% 计算 α为 1 时的感知误差
basic = Watsondistorsion( image, 'temp. jpg') ;
disp( '( α为 1 时的基本误差)') ;
% 计算全局强度因子 α
alf = equal/basic;
```

```
% 正式加水印
fun = @wmadd;
result = blkproc( DCTcoef, [ 8 8] , fun, wm, alf) ;
T = dctmtx( 8) ;
r = blkproc( result, [ 8 8] , 'P1* x* P2', T', T) ;
imagergb(:,:, 1) = r;
imagergb = imagergb /255; % 结果归一化才能存储
imwrite( imagergb, goal) ;
imagergb = double( imread( goal) ) /255;
Watsondistorsion( image, goal) ;
disp( '( α经过适应后的感知误差)' ) ;
% 输出结果
disp( [ '全局强度因子 α=', num2str( alf) ] ) ;
subplot( 121) , imshow( imagergb2) , title( '原始图像' ) ;
subplot( 122) , imshow( imagergb) , title( [ '加有水印强度 α=', num2str( alf) , '的图像' ] ) ;
function block = wmadd( test, wm, alf)
zigdone = zigzag( test, 1) ; % zigzag 排列
for i = 3:8
    for j = 1:8
        zigdone( i, j) = zigdone( i, j) + alf* ( wm( ( i - 1) * 8 + j) ) ; % 低频系数是不加水印的
    end
end
block = zigzag( zigdone, 2) ;
```

由于我们在添加水印时回避了每块最低频的 16 个系数,所以实际算出来的水印图像感知质量是要高于期望的感知质量的。

水印的检测仍然采用非盲相关检测法进行。检测的框架与 W-SVD 类似。这里我们就不重复叙述了。需要说明的是,我们的实验仅仅关心的是水印不可见性的适应。在实际应用中,还必须结合鲁棒性来一起考虑,而鲁棒性的评价往往就是分析检测错误的概率以及在水印图像受到攻击的前提下水印被检测出来的可能性。将鲁棒性与不可见性统一分析,求出的水印强度适应值才是理想的。

第九章　水印攻击与性能评价

前面两章我们分别从水印系统的基本原理和图像感知质量度量两方面认识了数字水印。由于数字水印的主要功能是保护作品的版权，在诸如拷贝跟踪等方面提供判断和辨别的依据。不法者要侵犯数字作品版权就一定会对其中的水印进行攻击，以达到其不法目的。这样一来，作为版权持有者除了要给作品添加水印外，还必须考虑所使用的水印系统的性能如何。

那么我们如何知道一个具体的水印系统的性能好坏呢？评价一个数字水印系统的好坏应包括多个方面，这是因为数字水印系统本身是一个复杂的综合体。水印不同、载体不同、嵌入的方法不同等都决定了各个水印系统的性能是不同的，具体来说包括以下的几个因素：

①嵌入信息的数量。它显然是一个重要的因素，在较小的载体内嵌入太多的信息，不仅仅是容量的问题，它还会直接影响到水印的鲁棒性。

②水印嵌入的强度。在嵌入水印的方法中，大多数会引入强度系数来施加水印，对于强度系数的影响我们可以在前面的章节中直观地了解到，嵌入的强度高会影响水印的不可见性，强度太低水印就变得脆弱。

③数据的大小和种类。

④密钥控制机制。这一点主要考虑的是水印的安全性。无论是直接用随机数来充当水印还是利用随机数来决定水印嵌入的位置，都需要保证这些秘密信息（密钥）不会被攻击者穷举出来或用其他的方法求出来，也就是说要遵循密码学基本的原则。

⑤对抗攻击的能力。

对含有水印图像的常见的攻击方法分为有意攻击和无意攻击两大类：有意攻击最常见的是解释攻击，这不是单纯加强水印策略能够解决的问题，它涉及到有关协议的制定。无意攻击是可以通过改善水印系统来解决的，依照 Stirmark 和 Checkmark 等常用的水印测试基准程序，无意攻击通常有：

- 剪切。
- 增强，模糊和其他滤波算法。
- 放大、缩小和旋转。
- 有损压缩，如 JPEG 压缩。
- 在图像中添加噪声等。

对于这些攻击，好的水印系统是具有一定的鲁棒性的，即经过这些攻击还能够保

证水印的正确检测和提取。

需要明确的是,水印性能测试与水印攻击往往是一致的。我们在这里将水印的攻击和性能的评价放在一章中来论述,正是因为评价一个水印系统是和水印攻击分不开的,离开了具体的攻击方法,对水印系统的评价是无从谈起的。在很多时候,研究遭受某种攻击后水印的误差更能有效地说明水印系统的好坏。

9.1 检测错误和误比特率

9.1.1 检测错误

一个水印系统的应用包括三个方面:水印的生成策略、嵌入策略和检测策略。广义的水印检测应该解决两个问题,即判断一幅图像中是否藏有水印和用适当的方法将水印完整的、正确的提取出来。检测错误指的是在判断是否存在水印时出现的错误。对于一个水印系统的检测策略来说,其出现的检测错误越低,水印系统的性能是越优的。在一定程度上,水印系统的检测策略直接影响对水印作品鲁棒性的判断。

在检测水印时,可以根据不同的水印系统设计各自系统的水印检测器,由该检测器的输出值判断有无水印,而检测器的输入当然是需要判断的图像,可以根据是否为盲检测来决定是否输入原始图像。

常用的水印检测策略是相关检测。在相关检测中,检测阈值的取值对检测结果有着决定性的影响。这样的检测可以看成是一个二元的假设校验,它存在着两类错误:

第一类错误,该类错误是指在没有嵌入水印的接收图像中检测到了水印,也就是说水印检测器输出的相关性值大于给定的阈值。由于检测器错误地发出报警,因此,这类错误也被称为虚警错误(falce positive) 。

第二类错误,该类错误是指在有嵌入水印的接收图像中没有检测到水印,也就是说水印检测器的输出的相关性值小于给定的阈值。在这种情况下,检测器应该但没有报警,因此这类错误又被称为漏警错误(falce negative) 。

图 9.1 中可以直观地看出这两类错误产生的情况。图中的横坐标表示检测器所设的阈值,纵坐标表示数字作品的检测值概率分布。虚线表示没有嵌入水印的图像经过检测器得到的输出值的概率密度函数,实线表示嵌有水印的图像经过检测器得到的输出值的概率密度函数,垂直的直线表示检测阈值,横轴从左到右取值在 0 ~1。无论是嵌入水印的作品还是未嵌入水印的作品的检测值分布,其函数图像与阈值线右边所包围的面积就是检测有水印的概率。可以直观地看到,当阈值越靠右,其与数字作品的检测值概率密度分布函数所围成的面积越小,则越难检测到水印。图 9.1 中的阴影部分表示的是发生第一类错误的概率,大家可以分析,自己找出发生第二类

错误的概率对应的区域。

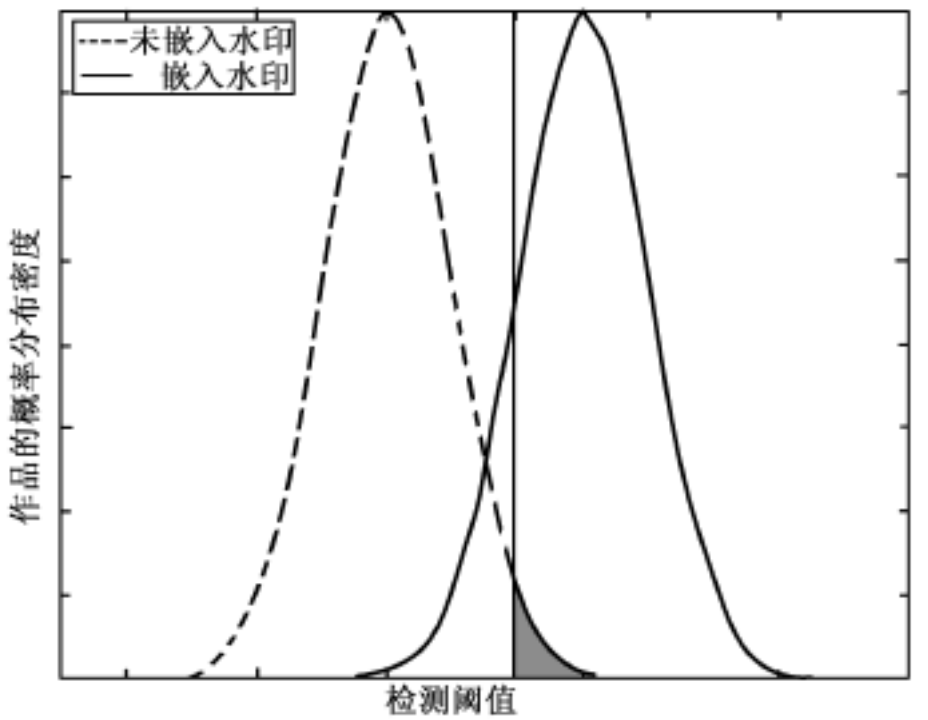


图 9.1　检测器输出分布和阈值

要注意的是,水印系统不同,嵌入的方法不同,检测器的设计方法不同,阈值的不同都将会影响到发生检测错误概率的大小。仅仅就阈值来看,阈值设定得越大,发生第二类错误的可能性越大;反之,则发生第一类错误的可能性越大。

为了进一步了解和分析这两类错误,我们引进利用 ROC(Receiver Operating Characteristic) 曲线的 ROC 分析法来说明它。ROC 曲线是关于检测阈值的函数曲线。它直观地反映了水印检测的灵敏度和特异性以及两类错误率之间的关系。ROC 曲线上每个点由两个分量构成,其纵坐标定义为 TPR(True Positive Ratio) ,我们也称之为灵敏度(sensitivity) ,用公式表示为:

$$TPR = \frac{TP}{TP + FN} \tag{9.1}$$

式(9.1) 中的 TP 表示正确接受测试结果的次数,也就是在嵌有水印的接收图像中检测到水印的次数; FN 表示错误拒绝测试结果的次数,也就是在嵌有水印的接收图像中没有检测到水印的次数。

同理, ROC 曲线上每个点的横坐标定义为 FPR(False Positive Ratio) ,它与特异性(specificity) 是互补的关系,即 FPR =1 - 特异性。同样用公式表示为:

$$FPR = \frac{FP}{TN + FP} \tag{9.2}$$

式(9.2) 中的 FP 表示错误报警测试结果的次数,即在没有嵌入水印的接收图像中检测到水印的次数; TN 表示正确拒绝测试结果的次数,即在没有嵌入水印的接收图像中没有检测到水印的次数。需要注意的是,我们在这里虽然用 ROC 曲线来表示两类

错误的关系,但是 ROC 曲线的两个轴并不是直接对应着两类错误,FPR 可以看做是对应的第一类错误,但是与第二类错误对应的不是 TPR,而是(1 - TPR)。

关于 ROC 曲线的含义,也就是如何去判断它的理想程度,可以用图 9.2 加以说明。

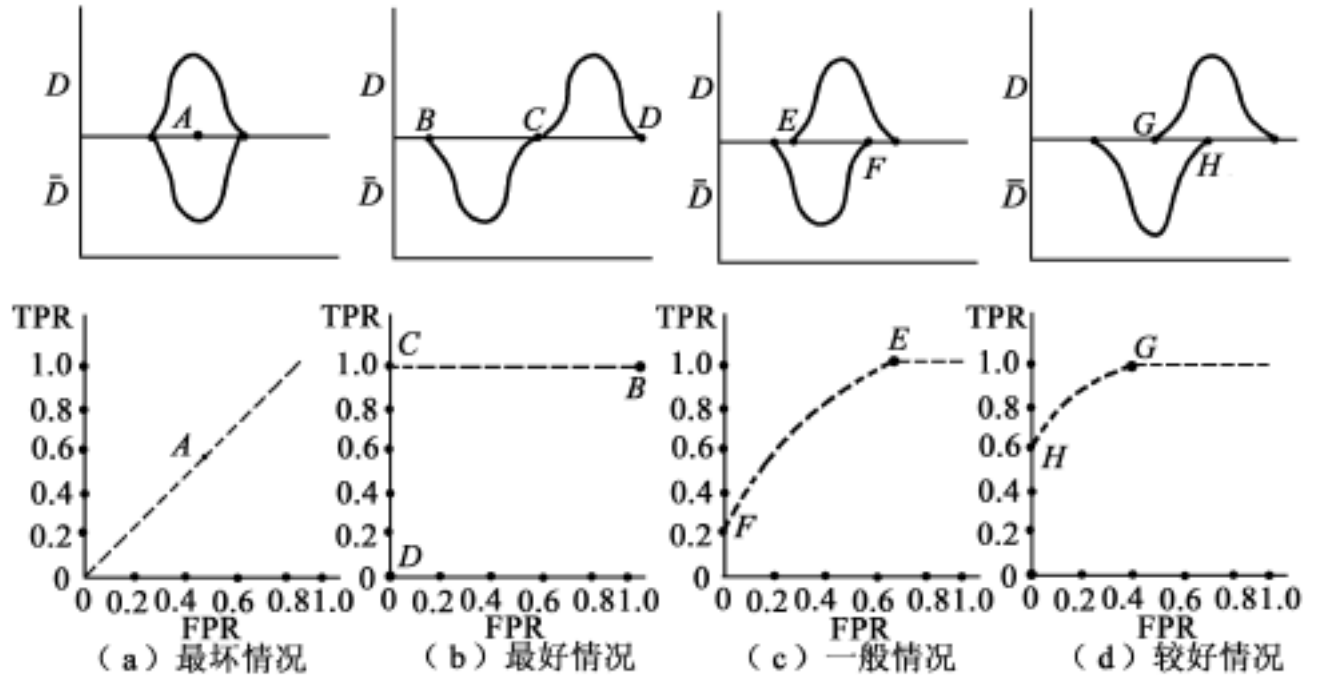


图 9.2　ROC 曲线的含义

ROC 曲线越向左上偏,曲线与坐标轴围成的面积越大,表明检测器性能就越好。图 9.2 上半部分所示的是 4 种加有水印的图像和没有加水印的图像的检测值分布的概率密度函数。图 9.2(a)中嵌有水印图像的检测值和没有嵌入水印图像的检测值分布完全相同,也就是说,不管阈值取在哪里都不能通过检测值来判断有没有水印存在。图 9.2(b)则是另一个极端,即嵌有水印图像的检测值和没有嵌入水印图像的检测值是在 C 点分开的,是完全不重叠的。我们如果将阈值定在 C 点和 D 点之间,则 FPR 为 0;一般情况是像图 9.2(c)和图 9.2(d)那样的,检测值分布是有重叠的,这样就会产生检测错误了。但二者的检测灵敏度和特异性是不同的。图 9.2 的下半部分是上半部分对应的 ROC 曲线,我们可以明显地看出,图 9.2(b)的曲线最理想。

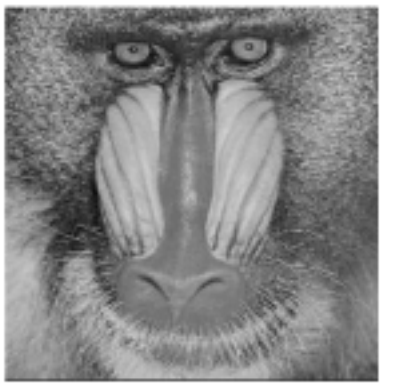
下面通过一个实例来绘制 ROC 曲线,使大家方便地掌握一些基本概念和方法。

我们使用的水印系统是第七章中介绍过的 W-SVD 水印系统。在标准图像库中我们选取如图 9.3 所示的 5 幅 JPEG 图像,其大小均为 256× 256。我们将使用它们来生成实验的样本空间。

检测的样本空间由以下几个方面构成:5 幅原始图像;每幅原始图像做以 10% 压缩率为间隔的压缩得到的 9 幅 JPEG 压缩图像,一共得到 45 幅图像;对每幅原始图像各自加入 10 个不同种子的水印,得到 50 幅加有水印的图像;再分别对 5 幅图像加入水印,并进行压缩,也得到 45 幅经过不同程度压缩的嵌有水印的图像和 5 幅嵌入水

kid.jpg

lenna.jpg



baboon.jpg

brandyrose.jpg

watch.jpg

图 9.3　ROC 样本生成图像

印但没有压缩的图像。这样共有 100 幅加有水印的图像(不管有没有压缩),以此作为后面检测漏警错误的样本空间。50 幅没有加入水印的图像,作为检测虚警错误的样本空间。

以某一个检测阈值(如 0.05)检测一幅图像,结果有 4 种组合:图像加有水印并被检测出来(TP)、图像加有水印没有被检测出来(FN)、图像没有加水印检测结果也无水印(TN)、图像没有加水印但检测出水印(FP)。

我们在前面已经说明了什么叫灵敏度和特异性,它们分别表示了在嵌有水印的图像中检测到水印的比例和在没有嵌入水印的图像中没有检测到水印的比例。对于一个检测方法的衡量我们就是用这两个指标。为了提高检测的灵敏度,我们只需将阈值取小就行了,要提高检测的特异性,只需将检测阈值取大就行了,可以看出,这两种特性是矛盾的。

我们采用 W-SVD 中的两种检测方法对水印进行检测,分别称为常规检测器和 DCT 检测器(参见 7.3)。每一幅图像检测都会得到两个检测相关性值 corr_coef 和 corr_DCTcoef。表 9.1 是以常规检测器检测得到的相关性值 corr_coef 的分布情况。表 9.2 是以 DCT 检测器检测得到的相关性值 corr_coef 的分布情况。

表 9.1　　常规检测器的检测结果

阈值	水印图像中检测到水印的累计数目(TP)	水印图像中没有检测到水印的累计数目(FN)	无水印图像中检测到水印的累计数目(FP)	无水印图像中没有检测到水印的累计数目(TN)	灵敏度(TPR)	特异性(TNR)
0.01	98	2	44	6	0.98	0.12
0.02	89	11	38	12	0.89	0.24
0.03	85	15	25	25	0.85	0.5
0.04	85	15	19	31	0.85	0.62
0.05	84	16	18	32	0.84	0.64
0.06	83	17	12	38	0.83	0.76
0.08	81	19	6	44	0.81	0.88
0.1	80	20	0	50	0.8	1
0.15	76	24	0	50	0.76	1
0.2	70	30	0	50	0.7	1
0.25	68	32	0	50	0.68	1
0.3	62	38	0	50	0.62	1
0.4	58	42	0	50	0.58	1
0.5	54	46	0	50	0.54	1
0.6	53	47	0	50	0.53	1
0.7	52	48	0	50	0.52	1

表 9.2　　DCT 检测器的检测结果

阈值	水印图像中检测到水印的累计数目(TP)	水印图像中没有检测到水印的累计数目(FN)	无水印图像中检测到水印的累计数目(FP)	无水印图像中没有检测到水印的累计数目(TN)	灵敏度(TPR)	特异性(TNR)
0.02	96	4	43	7	0.96	0.14
0.04	93	7	36	14	0.93	0.28
0.06	90	10	30	20	0.9	0.4
0.08	88	12	26	24	0.88	0.48
0.10	86	14	18	32	0.86	0.4
0.15	83	17	13	37	0.83	0.74
0.20	81	19	5	45	0.81	0.9
0.25	78	22	3	47	0.78	0.94
0.30	75	25	0	50	0.75	1
0.35	71	29	0	50	0.71	1
0.40	69	31	0	50	0.69	1
0.50	64	36	0	50	0.64	1
0.60	58	42	0	50	0.58	1
0.70	56	44	0	50	0.56	1
0.80	53	47	0	50	0.53	1

图 9.4 是根据表 9.1 和表 9.2 绘制的 ROC 曲线。我们使用 MATLAB 中的 trapz 函数来分别计算图 9.4 中两组 ROC 曲线与坐标轴所围成的曲边梯形的面积，得到两个面积分别为 0.7525 和 0.7459，由此可见，这两种检测器的性能是相当的。

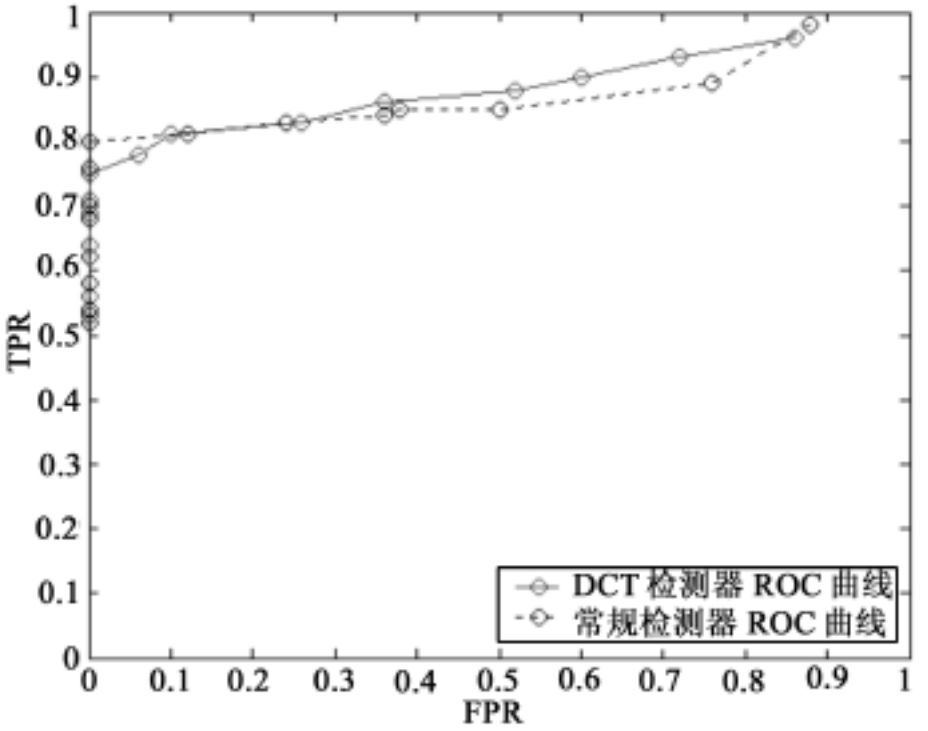


图 9.4　ROC 曲线

9.1.2　误比特率

误比特率也是用来检测水印系统好坏的参数，我们在这一节主要介绍一种误比特率的高斯模型，用到的水印系统是 7.2 节介绍的基本水印模型的改进系统。7.2 节所介绍的系统是实现水印嵌入的很简单的一种方法，但它只能实现对一位信息的嵌入，下面简单介绍一下对多位信息嵌入该如何改进。

要嵌入多位信息(以 8 位为例)，我们就按 Basic_wm 系统模型中的模板，不过这次的模板取为 8 个，基本参考模板记为 W_{r1}，W_{r2}，W_{r3}，…，W_{r8}，它们都是根据给定可以被当做水印密钥的种子伪随机产生的，而且服从独立的相同高斯分布，将每个模板标准化使其均值为 0，并归一化使其具有单位方差。8 位信息记为 m[1]，…，m[8]，编码方式一样，信息模板如下式：

$$W_{mi} = W_{ri} \quad m[i] = 1 \tag{9.3}$$

$$W_{mi} = -W_{ri} \quad m[i] = 0 \tag{9.4}$$

再取 W_{tmp} 为归一化之前的信息标志：

$$W_{tmp} = \sum_{i} W_{mi} \tag{9.5}$$

最后水印确定为 W_m：

$$W_m = \frac{W_{tmp}}{S_{W_{tmp}}} \tag{9.6}$$

$S_{W_{tmp}}$为样本标准差。按下式嵌入:

$$C_W = C_0 + \alpha W_m \tag{9.7}$$

检测器计算接收图像与 8 个基本模板的相关性来确定 8 位的值。

基于上述的水印系统,模板取为相互正交的白噪声,8 个模板组合作为信息嵌入模板,以$\frac{1}{\sqrt{8}}$为尺寸进行放缩使其具有单位方差。按式(9.7)嵌入后,则每位数据的嵌入强度为$\frac{\alpha}{\sqrt{8}}$。

假设对应于每位数据,检测器输出满足高斯分布,则该假设对于白噪声模板显然成立。对应于每一位的检测相关均值 μ为:

$$\mu = \frac{\alpha}{\sqrt{8}} \tag{9.8}$$

方差为 σ^2:

$$\sigma^2 = \sigma_{W_{ri}}^2 (\sigma_{C_0}^2 + \sigma_n^2) \tag{9.9}$$

每个模板的方差 $\sigma_{W_{ri}}^2$为 1, $\sigma_{C_0}^2$, σ_n^2 分别为作品和信道方差,此时的错误比特率为:

$$\begin{aligned} p &= \int_{-\infty}^{0} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{(x-\mu)^2}{2\sigma^2}} dx \\ &= \int_{-\infty}^{-\mu} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx \\ &= \int_{\mu}^{\infty} \frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}} dx \\ &= \operatorname{erfc}\left(\frac{\mu}{\sigma}\right) \end{aligned} \tag{9.10}$$

式(9.10)中 $\operatorname{erfc}(x) = \frac{1}{\sqrt{2\pi}} \int_x^{\infty} e^{-\frac{t^2}{2}} dt$, 在 MATLAB 中由 erfc 函数来实现。

这里要说明一下,误比特率就是指错误的比特占图像全部比特的百分比,上面提到的高斯模型是具体的一种实现,它是根据模板的性质用数学方法估计的误比特率,实际的误比特率用简单的计算百分比的方法就可以了。

在衡量水印作品的鲁棒性时,我们往往使用以下两种曲线来说明:“攻击—检测相关性值图”和“攻击—误比特率图”。二者的横轴均由同一攻击方式的不同攻击强度构成;纵轴一个是检测相关性值,另一个是误比特率。应该说,这两种曲线各有优劣。前者必须结合明确且相对正确的阈值才能分析出水印的抗攻击能力,后者则不需要额外的参数参与就能直观体现水印的性能。但考虑到在通信信道中产生的干扰,我们认为单纯的使用误比特率并不能真实反映水印的强度,所以在本书中不使用它。

9.2　几种常见的无意攻击

本节将重点阐述几种常见的无意攻击方法的原理及实现。同时,由于在衡量水印作品鲁棒性时,我们经常使用“攻击—检测相关性值图”。在本节中,还将重点说明在每种攻击下这种曲线的绘制方法,其中使用到的水印系统仍然为 W-SVD。

9.2.1　中值滤波

中值滤波(median filtering) 是基于排序统计理论的一种能有效抑制噪声的非线性信号处理技术。它的优点主要表现在以下几点:

①运算简单且速度较快;

②在滤除噪声的同时能很好地保护信号的细节信息(如边缘、锐角等) ;

③很容易自适应化,可进一步提高滤波的性能。

因此,中值滤波被广泛应用于数字图像处理场合。正因为如此,它也被我们看做是对水印的无意攻击中比较重要的一种,我们将简要地阐述中值滤波的理论,给出它的 MATLAB 实现方法和滤波结果以及它对数字水印的影响。

(1) 一维中值滤波

当 n 为奇数时,一个 n 个数的序列 $x_1, x_2, \cdots, x_n$ 按从小到大的顺序排列,处于中间位置的数就称为这个序列的中值,当 n 为偶数时,则定义处于中间两个数的平均值为这个序列的中值,用符号表示如下:

$$\mathrm{med}(x_1, x_2, \cdots, x_n) \tag{9.11}$$

例如: $\mathrm{med}(0,9,4,3,6,7,2) = 4$。因为在中值滤波器中的 n 通常为奇数,我们以后的讨论也仅仅涉及 n 为奇数的情况。我们再给出中值滤波器的定义。设中值滤波器的大小为 n,对序列$\{x_i \mid i \in Z\}$的标准中值滤波器有:

$$y_i = \mathrm{med}\{x_i, i \in Z\} = \mathrm{med}(x_{i-k}, \cdots, x_i, \cdots, x_{i+k}) \quad i \in Z \tag{9.12}$$

式(9.12) 中的 $k = (n-1)/2$, Z 为所有自然数的集合。这种滤波器也称为滑动滤波器,它的工作原理是:大小为 n 的滤波器的窗口的长度为 n, n 为奇数,对任一时刻该窗口内的所有值排序取中值,即为滤波器的输出。图 9.5 直观地给出了它的工作原理,在这个例子中的中值滤波器的大小为 $n = 3$,即它的窗口大小也为 3,此时的 $k = (n-1)/2 = 1$,信号的长度为 5,图中圆点的高度对应该点信号的值,为了让处于首尾的两点也能被滤波,在两端先扩展 1 点(一般地,扩展的点数为 k) 。

由图 9.5 还可以看到中值滤波器在去除脉冲噪声的同时有保护信号细节的性质,具体来说,中值滤波器窗口长度为 n 时,如果信号中的一个脉冲的长度大于或等于 $k+1$,滤波后该脉冲将得到保留,如果信号中的一个脉冲的长度小于或等于 k,滤波后该脉冲将被去除。对此,还是用图 9.6 来说明,该例中的 k 还是取 1。

由此性质,得到的结论是:中值滤波器保护边缘去除脉冲和振荡信号,而且长窗

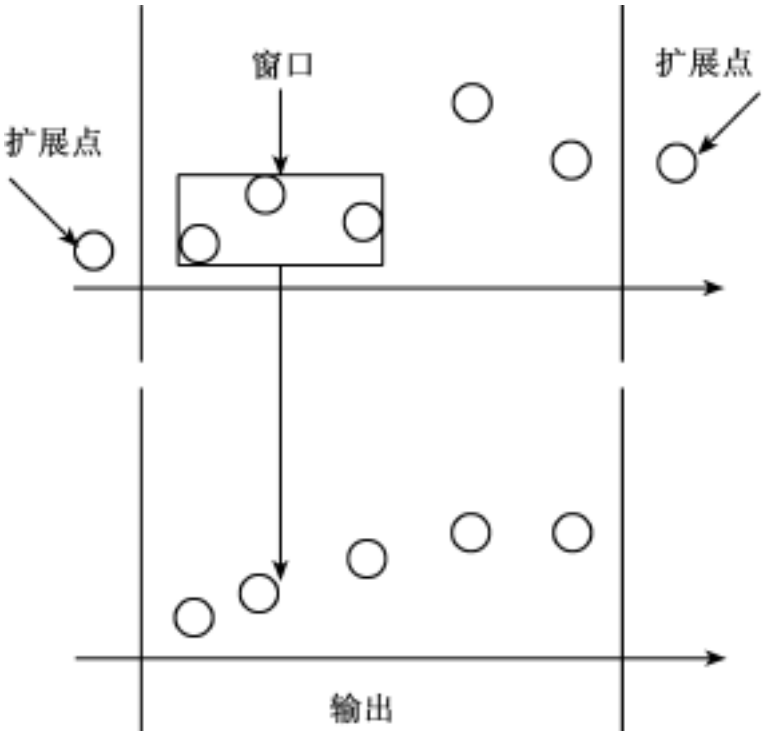


图 9.5　窗口为 3 的中值滤波图示

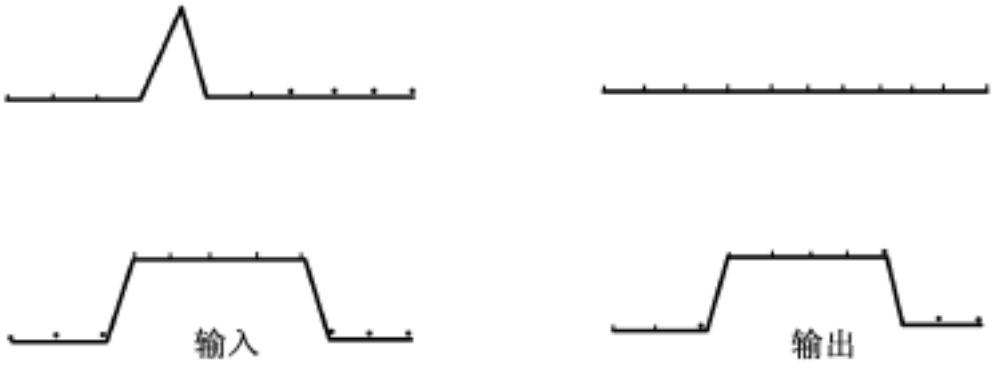


图 9.6　中值滤波去除脉冲

口滤波器去除脉冲和振荡信号比短窗口滤波器去除得多。

(2) 二维中值滤波

要应用到图像处理中的中值滤波器应该是二维的形式, 虽然一维的中值滤波器也能通过适当的运用来处理二维的图像, 方法我们将在下面略为提及, 但是那样相对比较麻烦。二维中值滤波器的窗口也应该是二维的, 下面来看看是如何定义的。

设若用数集$\{x_{ij}\}$表示一幅数字图像, 广泛地来说, 这里的(i, j)遍取Z^2或它的一个子集, 滤波器窗口 A 的尺寸为$N = (2k+1)(2k+1)$, 定义二维中值滤波器为:

$$y_{ij} = \mathrm{med}\{x_{i+r, j+s}, (r, s) \in A\} \tag{9.13}$$

与一维的中值滤波器的原理相似, 二维中值滤波器对图像的处理是用一个二维的窗口去依次成块地覆盖图像中的像素, 用覆盖的那些像素值的中值去取代窗口正中的那个像素的值。假设图像的大小为 K× L, 在用滤波器处理图像之前还要进行点扩展, 以保证输出的图像大小与原来的一致, 扩展点的个数取决于图像的大小和窗口的大小, 下面是一个图像大小为 3× 3, 窗口大小也为 3× 3 的点扩展的图示, 如图 9.7 所示。

45	32	66
23	76	67
21	71	28

45	45	32	66	66
45	45	32	66	66
23	23	76	67	67
21	21	71	28	28
21	21	71	28	28

图 9.7　二维中值滤波像素点扩展图示

对于窗口的选择，一维滤波器是不需要考虑的，它的窗口就只能是一维的形式，但是对于二维的中值滤波器，窗口的形状变得多种多样，这时也出现了子窗口的设计和选择的问题，二维中值滤波器保存边缘消除噪声的特性与此密切相关。我们采用图 9.8 所示的全方位子窗口来实现这种特性。

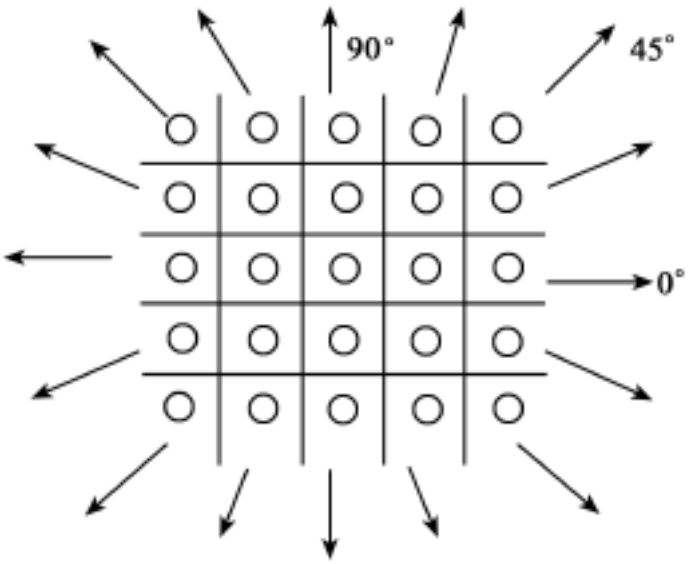


图 9.8　全方位子窗口

一般地，我们称在全窗口 A 下进行的二维中值滤波为常规二维中值滤波，它的计算方法我们在前面提到过，就是用该窗口下的像素的中值来取代处于中间的像素的值，它可以通过连续运用两次一维中值滤波来实现，即先按行再按列，或先按列再按行，表达式如下：

$$z_{ij} = \mathrm{med}(x_{i,j-k}, \cdots, x_{ij}, \cdots, x_{i,j+k}) \tag{9.14}$$

$$y_{ij} = \mathrm{med}(z_{i-k,j}, \cdots, z_{ij}, \cdots, z_{j+k,j}) \tag{9.15}$$

y_{ij}即为滤波器的输出。

(3) 二维中值滤波器的 MATLAB 实现

我们重点关注二维中值滤波的实现。编写函数 median16.m 完成实验，这里面主要用到了 MATLAB 中已有的 medfilt2 函数，函数代码如下。

% 文件名: median16.m

```
% 作者: 李鹏
% 创作时间: 2004. 3. 24
% 目的: 完成对图像的中值滤波
% 引用举例: image_opd = median16( 'test. png',3) ;
% 参数说明:
% image 为待做中值滤波的图像
% a 为二维中值滤波器的窗口尺寸参数, 窗口大小为 a× a, 这里的二维中值滤波器为常规中值滤波器
function image_opd = median16( image, a) ;
A = imread( image) ;
[ row, col] = size( A) ;
% 以下是对 8 位图的处理方式
% A = double( A) /255;
% 以下是对 16 位图的处理方式
A = double( A) /65535;
original = A;
B = reshape( A, row, col) ;
C = medfilt2( B, [ a a] ) ;
col = col/3;
image_opd = reshape( C, row, col, 3) ;
% 以下是对 8 位图的处理方式
% imwrite( image_opd, imagegoal) ;
% 以下是对 16 位图的处理方式
imwrite( image_opd, 'temp2. png', 'BitDepth',16) ; % 以 png 格式存储
```

图 9. 9 是 lenna 的原始图像和做 7× 7 的常规中值滤波后的结果。

原始图像　　7×7中值滤波后的结果

图 9. 9　真彩色图像的中值滤波

下面我们来说明“中值滤波—检测相关性图”(W-SVD 抗中值滤波攻击曲线)的绘制方法。

中值滤波是最常见的无意攻击之一,“中值滤波—检测相关性图”能刻画中值滤波攻击对嵌入水印图像的影响,其中的 x 轴表示中值滤波器的模板尺寸的大小,y 轴表示检测器检测出的相关性值。编写函数 plotmedian.m 完成实验,函数代码如下:

```
% 文件名: plotmedian.m
% 程序员: 李鹏
% 编写时间: 2004.3.29
% 函数功能: 本函数用于绘制对加有水印的图像做中值滤波(对 RGB 图像)后,检测相关性值与滤波程度的关系曲线
% 输入格式举例: plotmedian('test.png',21,'lenna.jpg',10,'db6',2,0.1,0.99);
% 函数说明:
% 横坐标表示模板的大小,纵坐标为相关性值
% 参数说明:
% test 为已经加入水印的待检测图像
% x 为处理图像模板的最大值
% original 为输入原始图像
% seed 为随机数种子
% wavelet 为使用的小波函数
% level 为小波分解的尺度
% alpha 为水印强度
% ratio 为算法中 d/n 的比例
function plotmedian(test, x, original, seed, wavelet, level, alpha, ratio)
quality = 1:2:x;
corr_coef = zeros(max(size(quality)),1);
count = 0;
for q = quality
    count = count + 1;
    image_opd = median16(test, q);
        [corr_coef(count), corr_DCTcoef(count)] = wavedetect('temp2.png', original, seed, wavelet, level, alpha, ratio);
end
plot(quality, abs(corr_DCTcoef));
xlabel('模板尺寸');
ylabel('相关性值');
```

从图 9.10 中可以明显地看出,随着中值滤波模板的增大,检测到的相关性值减小,说明对水印的破坏越大。

中值滤波器是最常用的非线性平滑滤波器,在实际应用中还存在另一种图像平

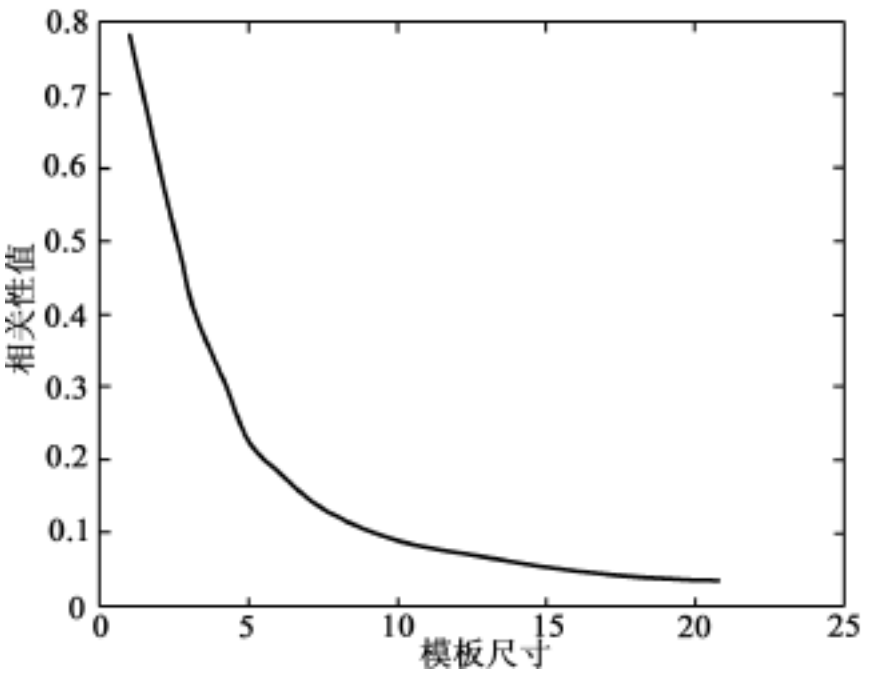


图 9.10　中值滤波-检测相关性值图

滑处理的方法——均值滤波。在 MATLAB 中均值滤波的模板用 h = fspecial('average') 语句来获得,用该模板和图像做二维卷积就得到均值滤波后的图像。中值滤波与均值滤波的不同之处在于,中值滤波器的输出是模板下像素的中间值,而均值滤波器输出的是模板下像素的平均值。中值滤波对极限像素值(也就是与周围像素值相差较大的像素值) 的平滑处理效果比均值滤波好。图 9.11 说明了这种情况。

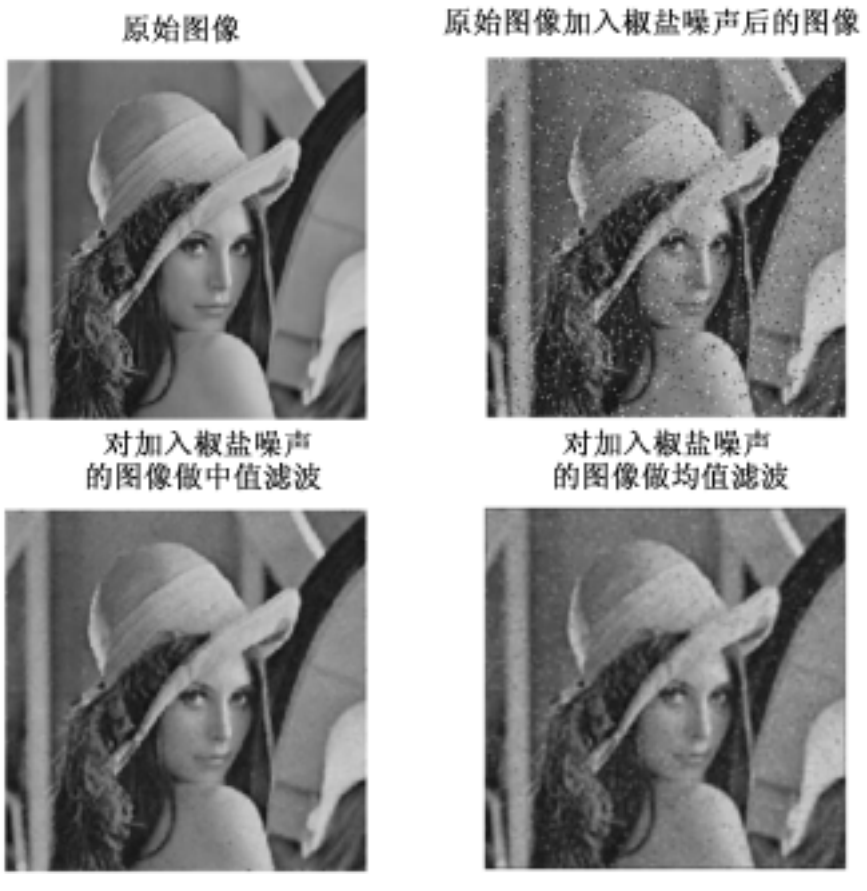


图 9.11　中值滤波与均值滤波

9.2.2　锐化滤波

锐化是空域滤波增强的一种方法，我们首先来看看什么叫空域滤波增强，然后再对锐化进行讨论并进一步给出锐化在 MATLAB 中的实现。

对于一个像素来说，它的邻域（也就是与之相邻的像素）和该像素之间是有密切联系的，我们用一个一般的式子来表示空域增强的方法：

$$g(x, y) = EH[f(x, y)] \tag{9.16}$$

式（9.16）中 $f(x, y)$，$g(x, y)$ 分别表示增强前后的图像，EH 表示增强操作。对于锐化这种空域滤波增强的方法来说，是基于一个模板的操作，如果用 s 和 t 分别表示 $f(x, y)$，$g(x, y)$ 在 (x, y) 处的值（对于灰度图像来说就是灰度值），并以 $n(s)$ 表示 $f(x, y)$ 在 (x, y) 邻域内像素的值，可以把式（9.16）写为：

$$t = EH[s, n(s)] \tag{9.17}$$

对于不同的滤波增强来说，它们的共同点是将图像与模板进行卷积，不同的是模板不同，图 9.12 可以直观地看到模板和图像的关系。

…	…	…	…	…
…	S4	S3	S2	…
…	S5	S0	S1	…
…	S6	S7	S8	…
…	…	…	…	…

图像像素

K4	K3	K2
K5	K0	K1
K6	K7	K8

滤波模板

图 9.12　锐化图像像素和模板

将模板作用在图像上，输出是每次模板作用的区域的中间像素经修改后的值，以 3× 3 的模板为例，输出用公式表示为：

$$R = K_0 S_0 + K_1 S_1 + \cdots + K_8 S_8 \tag{9.18}$$

（1）线性锐化滤波

线性锐化滤波通常就是一个高通滤波器，对于一个 3× 3 的模板来说，典型的系数是：

$$\begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix}$$

该模板所有系数之和为 0，这个滤波器将原图像中零频率分量去除。下面是采用拉普拉斯算子作为模板的线性锐化函数 sharpL. m。

```
% 文件名: sharpL. m
% 作者: 李鹏
% 创作时间: 2004. 3. 26
% 函数功能: 对图像做线性锐化
% 函数说明: 本函数的线性锐化滤波器采用‘ laplacian ’算子
% 引用举例: image_opd = sharpL( 'lenna. jpg', '1. jpg') ;
% 参数说明:
% image 为待做锐化的图像
% imagegoal 为锐化后的图像
function image_opd = sharpL( image, imagegoal)
A = imread( image) ;
[ row, col] = size( A) ;
A = double( A) /255; original = A;
B = reshape( A, row, col) ;
h = fspecial( 'laplacian') ;
C = filter2( h, B) ;
col = col/3;
image_opd = reshape( C, row, col, 3) ;
imwrite( image_opd, imagegoal) ;
```

图 9. 13 是 lenna 的原始图像和经过线性锐化后的图像。

原始图像

经过线性锐化后的图像

图 9. 13　真彩色图像的线性锐化

(2) 非线性锐化滤波

利用微分可以对图像进行锐化处理。对图像的微分我们通常用梯度完成。一个连续函数的梯度定义为:

$$\nabla f = \left[\frac{f}{x}, \frac{f}{y}\right] \tag{9.19}$$

它以 2 为模后:

$$\left| \nabla f_{(2)} \right| = \mathrm{mag}(\nabla f)\left[\left(\frac{f}{x} \right)^2 + \left(\frac{f}{y} \right)^2 \right]^{\frac{1}{2}} \tag{9.20}$$

简单点取 1 为模或以无穷大为模，分别表示为：

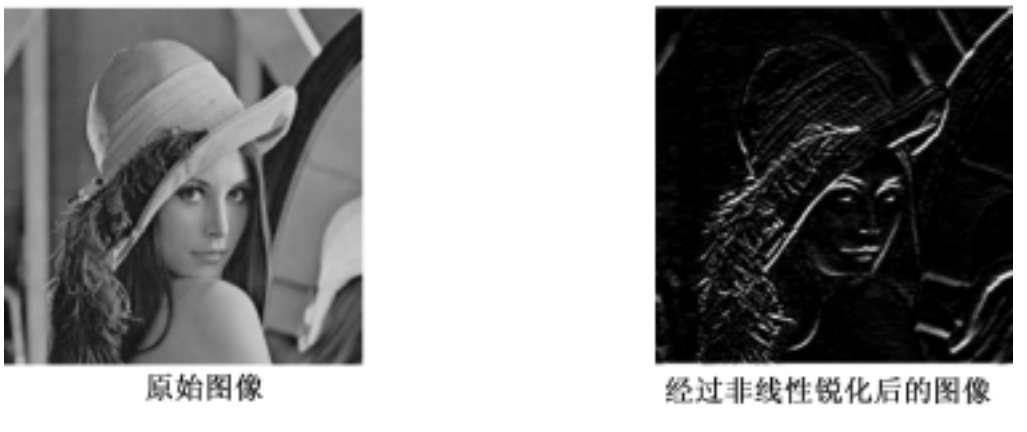
$$|\nabla f_{(1)}| = \left| \frac{f}{x} \right| + \left| \frac{f}{y} \right| \tag{9.21}$$

$$|\nabla f_{(\infty)}| = \max\left\{ \left| \frac{f}{x} \right|, \left| \frac{f}{y} \right| \right\} \tag{9.22}$$

同线性锐化一样，微分算子也有很多种，我们以 sobel 算子为例给出非线性锐化的函数 sharpS. m。

```
% 文件名: sharpS. m
% 作者: 李鹏
% 创作时间: 2004. 3. 26
% 函数功能: 对图像做非线性锐化
% 函数说明: 本函数的非线性锐化滤波器采用‘ sobel ’算子
% 引用举例:  image_opd = sharpS( 'lenna. jpg', '1. jpg');
% 参数说明:
% image 为待做锐化的图像
% imagegoal 为锐化后的图像
function image_opd = sharpS( image, imagegoal)
A = imread( image) ;
[ row, col]  = size( A) ;
A = double( A) /255; original = A;
B = reshape( A, row, col) ;
h = fspecial( 'sobel') ;
C = filter2( h, B) ;
col = col/3;
image_opd = reshape( C, row, col, 3) ;
imwrite(  image_opd, imagegoal) ;
```

图 9. 14 是 lenna 的原始图像和经过非线性锐化后的图像。



原始图像　　经过非线性锐化后的图像

图 9. 14　真彩色图像的非线性锐化

9.2.3 马赛克攻击

马赛克是我们平常生活中经常接触到的一个词。在装修房屋时使用马赛克的瓷砖,使厨房更加美观。在家中用影碟机欣赏电影时因为碟片划伤会出现马赛克现象,这让我们很头疼。在这里讲到的马赛克攻击也是图像经常会受到的无意攻击的一种,它将对水印产生怎样的影响是我们所关心的。

对于图像的马赛克攻击,原理很简单,就是将一幅图像中的像素按照一定大小的模板与相邻的像素一起取平均值,再将这个值赋给模板下的每一个像素,下面以 3×3 的模板为例,用图简单直观地分析,如图 9.15 所示。

12	14	16
13	15	17
14	16	18

处理前的像素值

15	15	15
15	15	15
15	15	15

处理后的像素值

图 9.15 马赛克处理示例

编写函数 mosaic16.m 完成对图像进行马赛克处理的函数,函数代码如下:

```
% 文件名: mosaic16.m
% 程序员: 李鹏
% 编写时间: 2004.3.20
% 函数功能: 本函数用于对图像做马赛克处理(对 RGB 图像)
% 输入格式举例: image_opd = mosaic16('test.png',3);
% 参数说明:
% image 为待处理图像
% x 为处理图像模板的大小
function image_opd = mosaic16(image, x);
A = imread(image);
[row, col] = size(A);
% 以下是对 8 位图的处理方式
%  A = double(A)/255;
% 以下是对 16 位图的处理方式
A = double(A)/65535;
original = A;
```

```
B = reshape( A, row, col) ;
r = x;
for i = 1: r: row
    for j = 1: r: col
        C( i: min( i + r - 1, row) , j: min( j + r - 1, col) ) = mean2( B( i: min( i + r - 1, row) , j: min( j + r - 1, col) ) ) ;
    end
end
col = col/3;
image_opd = reshape( C, row, col, 3) ;
% 以下是对 8 位图的处理方式
% imwrite( image_opd, imagegoal) ;
% 以下是对 16 位图的处理方式
imwrite( image_opd, 'temp2. png', 'BitDepth', 16) ;
% 以 png 格式存储
```

图 9. 16 是对 lenna 进行模板为 7× 7 的马赛克攻击的结果。

原始图像

经过 7×7 的马赛克处理的图像

图 9. 16　真彩色图像的马赛克处理

下面我们来说明“马赛克—检测相关性图”(W-SVD 抗马赛克处理攻击曲线) 的绘制方法。

“马赛克—检测相关性图”能刻画马赛克处理对嵌入水印图像的影响, 其中的 x 轴表示马赛克模板尺寸的大小, y 轴表示检测器检测出的相关性值。编写函数 plotmosaic. m 完成实验, 函数代码如下:

```
% 文件名: plotmosaic. m
% 程序员: 李鹏
% 编写时间: 2004. 3. 28
% 函数功能: 本函数用于绘制对加有水印的图像做马赛克处理( 对 RGB 图像) 后, 检测相关性值与马赛克程度的关系曲线
```

```
% 输入格式举例: plotmosaic( 'test. png',40, 'lenna. jpg',10, 'db6',2, 0. 1, 0. 99) ;
% 函数说明:
% 横坐标表示模板的大小, 纵坐标为相关性值
% 参数说明:
% test 为已经加入水印的待检测图像
% x 为处理图像模板的最大值
% original 为输入原始图像
% seed 为随机数种子
% wavelet 为使用的小波函数
% level 为小波分解的尺度
% alpha 为水印强度
% ratio 为算法中 d/n 的比例
function plotmosaic( test, x, original, seed, wavelet, level, alpha, ratio)
quality = 2: 2: x;
corr_coef = zeros( max( size( quality) ) , 1) ;
count = 0;
for q = quality
    count = count + 1;
    image_opd = mosaic16( test, q) ;
[ corr_ coef ( count) , corr _DCTcoef( count) ] = wavedetect ( 'temp2. png', original,
seed, wavelet, level, alpha, ratio) ;
end
plot( quality, abs( corr_DCTcoef) ) ;
xlabel( '模板尺寸') ;
ylabel( '相关性值') ;
```

从图 9.17 可以看到, 随着马赛克模板的增大, 检测值是逐渐减小的, 也就是说, 对图像的破坏是越来越大的。

9.2.4 加噪攻击

图像在传播的过程中最容易受到也是必然会受到的攻击就是加入的噪声了, 因此, 噪声也是一种典型的无意攻击, 它对嵌入的水印也会产生影响。通常最常见到的噪声是服从高斯分布的随机噪声。直接利用 MATLAB 中的 imnoise 函数, 我们编写向嵌入水印的图像中加入高斯噪声的函数, 函数代码如下:

```
% 文件名: noiseadd16. m
% 程序员: 李鹏
% 编写时间: 2004. 3. 25
```

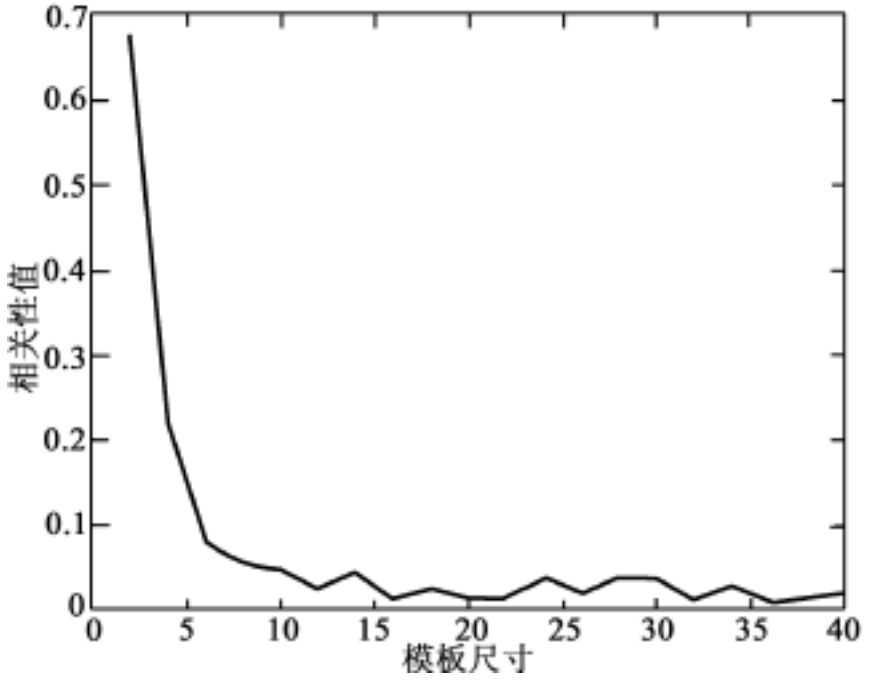


图 9.17　马赛克—检测相关性值图

```
% 函数功能: 本函数用于对 16 位图像加 gaussian 噪声
% 输入格式举例:  image_opd = noiseadd16('test.png',0,0.01);
% 函数说明: 本函数用到 MATLAB 中自带的 imnoise 函数,这里所加的噪声仅仅是
gaussian 噪声,其他噪声的参数不同,这里不再列举,同学们可以自己查看 imnoise 函数
    % 参数说明:
    % image 为待处理图像
    % M 为噪声的均值
    % V 为噪声的方差
    function image_opd = noiseadd16(image,M,V)
    A = imread(image);
    original = A;
    [row,col] = size(A);
    % 以下是对 8 位图的处理方式
    %  A = double(A)/255;
    % 以下是对 16 位图的处理方式
    A = double(A)/65535;
    B = imnoise(A,'gaussian',M,V);
    col = col/3;
    image_opd = reshape(B,row,col,3);
    % 以下是对 8 位图的处理方式
    % imwrite(image_opd,imagegoal);
    % 以下是对 16 位图的处理方式
```

imwrite(image_opd, 'temp2. png', 'BitDepth', 16) ;

图 9. 18 是对 lenna 加入高斯噪声的结果。

图 9. 18　真彩色图像的加噪处理

下面我们来说明“噪声—检测相关性图”(W-SVD 抗加噪处理攻击曲线) 的绘制方法。

“噪声—检测相关性图”能刻画噪声处理对嵌入水印图像的影响, 其中的 x 轴表示高斯噪声的方差, y 轴表示检测器检测出的相关性值。编写函数 plotaddnoise. m 完成实验, 函数代码如下:

```
% 文件名: plotaddnoise. m
% 程序员: 李鹏
% 编写时间: 2004. 3. 29
% 函数功能: 本函数用于绘制对加有水印的图像加入噪声( 对 RGB 图像) 后, 检测相关性值与加入噪声强度的关系曲线
% 输入格式举例: plotaddnoise( 'test. png', 0. 2, 'lenna. jpg', 10, 'db6', 2, 0. 1, 0.99) ;
% 函数说明:
% 横坐标表示方差, 纵坐标为相关性值
% 参数说明:
% test 为已经加入水印的待检测图像
% x 为方差最大值
% original 为输入原始图像
% seed 为随机数种子
% wavelet 为使用的小波函数
% level 为小波分解的尺度
% alpha 为水印强度
% ratio 为算法中 d/n 的比例
function plotaddnoise( test, x, original, seed, wavelet, level, alpha, ratio)
```

```
quality = 0.01:0.01: x;
corr_coef = zeros( max( size( quality) ) , 1) ;
count = 0;
for q = quality
    count = count + 1;
    image_opd = noiseadd16( test, 0, q) ;
[ corr_ coef ( count) , corr _DCTcoef( count) ] = wavedetect ( 'temp2. png', original,
seed, wavelet, level, alpha, ratio) ;
end
plot( quality, abs( corr_DCTcoef) ) ;
xlabel( 方差') ;
ylabel( 相关性值') ;
```

图 9.19 的总体趋势表明，随着所加噪声的方差的增大，图像被破坏的越厉害，检测出水印的难度越大。

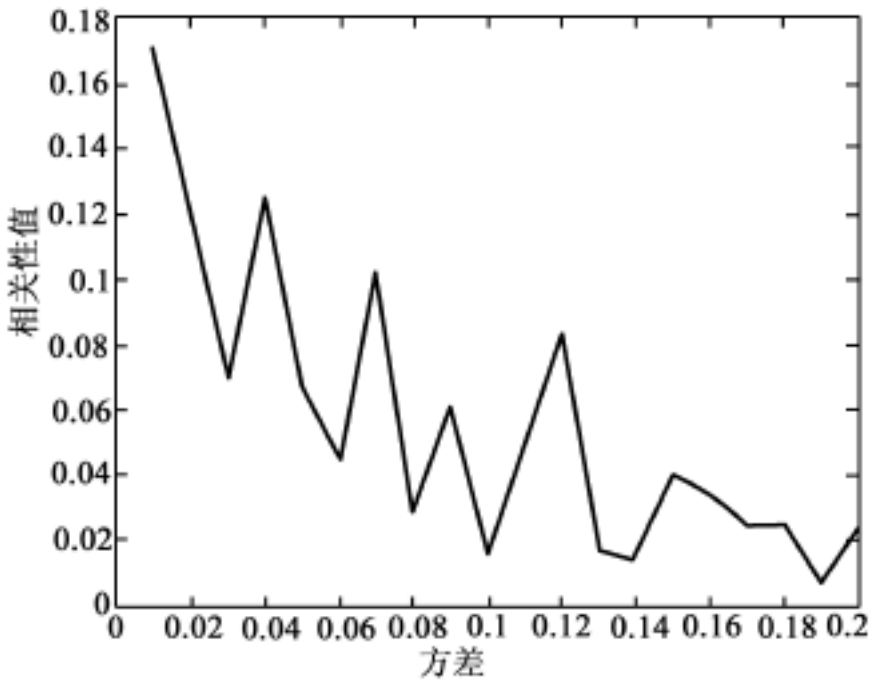


图 9.19　噪声—检测相关性值图

9.2.5　图像的旋转、剪切和改变大小

在图像的处理中，经常要对图像进行一系列的操作，包括图像的旋转、剪切和改变图像的大小，这些功能都能用一些简单的图像处理工具实现，在 MATLAB 中也有很方便简单的函数实现，我们在这一小节里把它们放在一起进行介绍。

旋转是指将图像按照一定的角度转动一定的度数，用 MATLAB 实现很方便，例如，下面的操作是将 lenna 旋转 45°，旋转后的结果如图 9.20 所示。

```
>> A = imread( 'lenna. jpg') ;
```

```
>> B = imrotate( A, 45, 'bilinear' );
>> imshow( B) ;
```

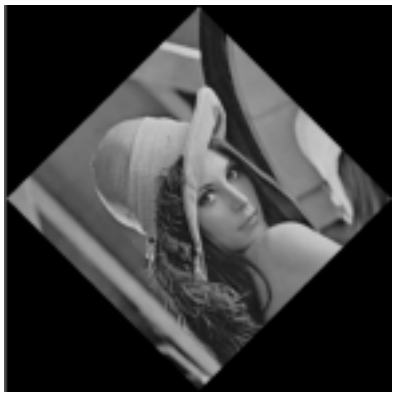


图 9.20　旋转 45 度以后的图像

其中的参数' bilinear '是指用双线性插补的方法来完成旋转操作,所谓双线性插补是指输出像素的赋值为 2× 2 矩阵所包含的有效点的加权平均值。

用 MATLAB 来改变图像的大小也是十分方便的,我们可以用以下的操作将图像改变为一个 100× 150 的图像,这里的大小是指图像的尺寸,不包含其他的意义。图 9.21 是改变的结果。

```
>> A = imread( 'lenna. jpg' );
>> B = imresize( A, [ 100 150] ) ;
>>  imshow( B) ;
```

也可以用 imresize 函数来缩放图像,下面的操作是放大图像为原来的两倍。

```
>>  A = imread( 'lenna. jpg' );
>> B = imresize( A, 2) ;
>> imshow( B) ;
```

图像的剪切有两种方法,一种是剪取鼠标左键拖动选取的矩形区域,另一种是给定一组参数,包括剪切图像起始坐标、剪切图像的长宽等,再按参数剪切,我们给出第二种方式的剪切函数。下面的操作是剪取起始坐标为(60, 40),长宽分别为 100 和 90 的一块图像的演示,图 9.22 是剪切的结果。

```
>> A = imread( 'lenna. jpg' );
>> B  =  imcrop( A, [ 40  60  100  90] ) ;
>> imshow( B) ;
```

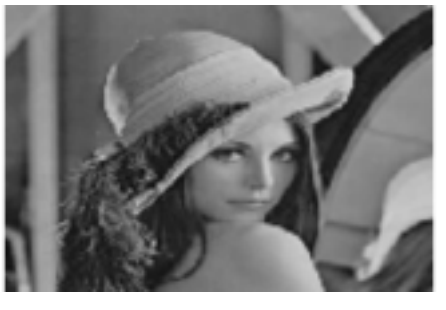


图 9.21　100× 150 的 lenna

图 9.22　剪切后的图像

9.2.6　JPEG 压缩

JPEG 压缩是对图像处理的最普遍的方法,由于我们在前面的章节中多次讲述它的概念及模型,也给出了实现压缩的函数(用 MATLAB 最简单的实现方式是在读取图像后,写入时用 imwrite(data, '1. jpg', 'jpg', 'quality', q),直接生成按压缩比 q 压缩

后的图像 1. jpg)。当然,用一些图像处理的工具也可以很方便地进行压缩。我们在这里只作为一种典型的无意攻击的方法提出来,图 9.23 是 lenna 的原始图像和经过 10% JPEG 压缩后的图像。

原始图像

压缩后的图像

图 9.23　lenna 经过 10% JPEG 压缩后的图像

下面我们来说明“JPEG 压缩—检测相关性图”(W-SVD 抗 JPEG 压缩攻击曲线)的绘制方法。

“JPEG 压缩—检测相关性图”能刻画 JPEG 压缩率对嵌入水印图像的影响,其中的 x 轴表示压缩率,y 轴表示检测器检测出的相关性值。编写函数 plotjpeg. m 完成实验,函数代码如下:

```
% 文件名: plotjpeg. m
% 程序员: 李鹏
% 编写时间: 2004. 3. 29
% 函数功能: 本函数用于绘制对加有水印的图像做 JPEG 压缩(对 RGB 图像)后,检测相关性值与 JPEG 压缩率的关系曲线
% 输入格式举例: plotjpeg( 'test. png', 'lenna. jpg',10, 'db6',2, 0. 1, 0. 99) ;
% 函数说明:
% 横坐标表示模板的大小,纵坐标为相关性值
% 参数说明:
% test 为已经加入水印的待检测图像
% original 为输入原始图像
% seed 为随机数种子
% wavelet 为所使用的小波函数
% level 为小波分解的尺度
% alpha 为水印强度
% ratio 为算法中 d/n 的比例
function plotmedian( test, original, seed, wavelet, level, alpha, ratio)
data = imread( test) ;
```

```
data = double( data) /65535;
[ M, N]  = size( data) ;
quality = 5: 5: 100;
corr_coef = zeros( max( size( quality) ) , 1) ;
count = 0;
for q = quality
        count = count + 1;
        imwrite( data, 'temp. jpg', 'jpg', 'quality', q) ;
        temp = imread( 'temp. jpg') ;
        temp = double( temp) /255;
        imwrite( temp, 'temp2. png', 'BitDepth', 16) ;
         [ corr _ coef( count) , corr _ DCTcoef( count) ]  = wavedetect( 'temp2. png',
original, seed, wavelet, level, alpha, ratio) ;
end
plot( quality, abs( corr_DCTcoef) ) ;
xlabel( 'jpeg 压缩率') ;
ylabel( '相关性值') ;
```

图 9. 24 表明, 对加有水印的图像压缩得越厉害, 对水印的破坏就越大, 这也是符合我们直观认识的。

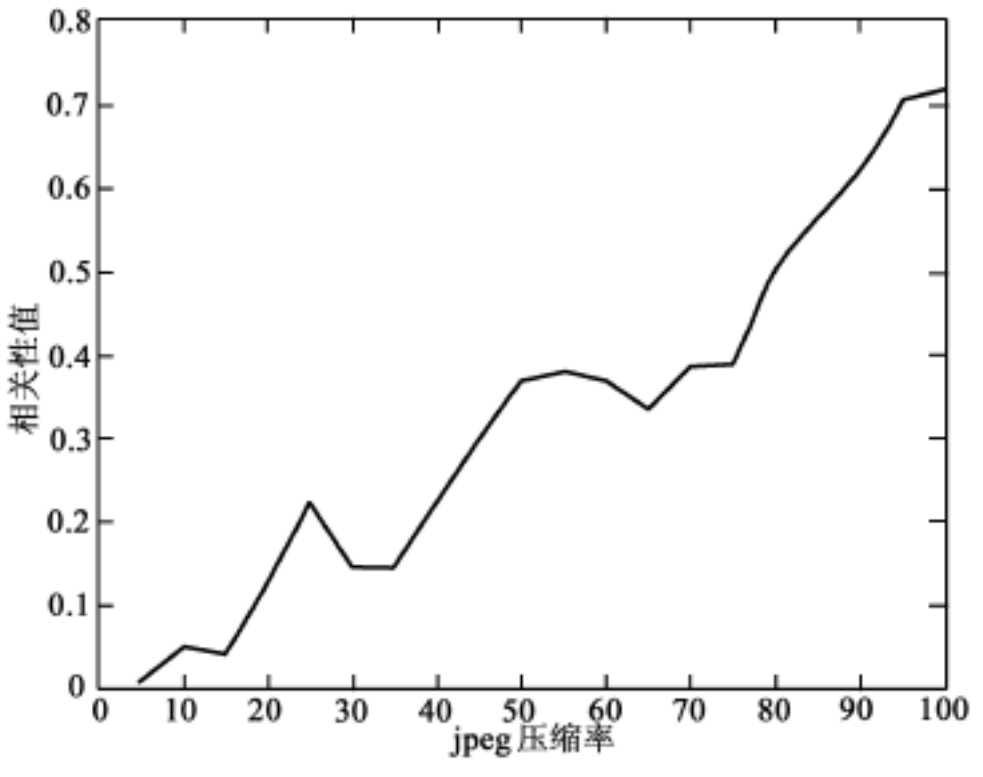


图 9. 24　JPEG 压缩—检测相关性值图

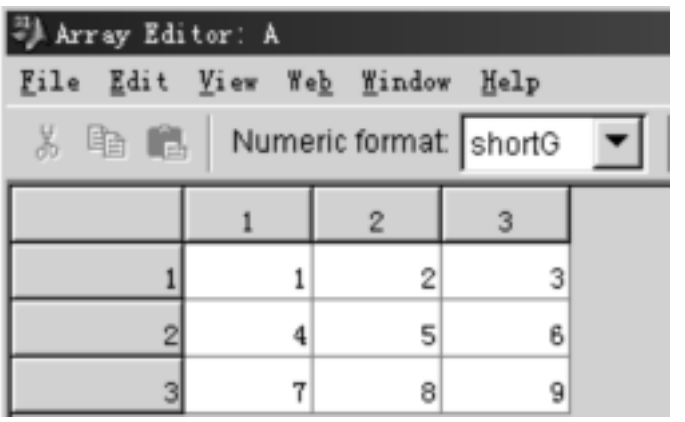
9.2.7　模糊处理

模糊处理(blurring) 技术也是经常用到的处理图像的方法。这种方法的原理十分简单, 其核心在于一个二维的卷积, 我们可以很方便地使用 MATLAB 中的 conv2 函数来实现这个卷积。卷积的模板通常取为如图 9. 25 所示的一个 5× 5 模板, 该模板又被称为模糊核(blurring kernel) 。

$$\frac{1}{44}\times\begin{bmatrix}1&1&2&1&1\\1&2&2&2&1\\2&2&8&2&2\\1&2&2&2&1\\1&1&2&1&1\end{bmatrix}$$

图 9. 25　模糊处理的模糊核

在将图像与模糊核做卷积之前, 我们需要像中值滤波那样先对图像矩阵做一定的处理, 也就是将图像矩阵按照一定的方法扩充。方法是: 首先生成一个比图像矩阵的行和列都多 4 的全 0 矩阵, 再将图像矩阵嵌入全 0 矩阵的正中, 然后, 按图像矩阵的边缘做扩充, 填满整个全 0 矩阵。下面以一个 3× 3 的矩阵为例, 假设它就是像素矩阵, 设为 A, 如图 9. 26 所示。扩充后的矩阵叫 Xe, 如图 9. 27 所示。



	1	2	3
1	1	2	3
2	4	5	6
3	7	8	9

图 9.26　原始像素

可见, 扩充是按照原始矩阵的边缘进行的, 最外面的行和列分别向两边沿扩展, 剩下的四个角上的位置分别由原始像素矩阵的四个顶上的值来填充。这样, 就从图像得到了做二维卷积的另一个矩阵。下面是用该模板做模糊处理的函数, 图 9. 28 是对 lenna 做处理的结果。

编写函数 blurringL16. m 完成模糊处理的实验, 函数代码如下:

```
% 文件名: blurringL16. m
% 程序员: 李鹏
```

Array Editor: Xe

File Edit View Web Window Help

Numeric format: shortG Size: 7 by 7

	1	2	3	4	5	6	7
1	1	1	1	2	3	3	3
2	1	1	1	2	3	3	3
3	1	1	1	2	3	3	3
4	4	4	4	5	6	6	6
5	7	7	7	8	9	9	9
6	7	7	7	8	9	9	9
7	7	7	7	8	9	9	9

图 9.27　扩展后的像素模板

模糊 1 次后的图像

模糊 20 次后的图像

图 9.28　对图像进行模糊处理

```
% 编写时间: 2004.3.20
% 函数功能: 本函数用于对图像做模糊处理(对 RGB 图像)
% 输入格式举例: image_opd = blurringL16('test.png',5);
% 参数说明:
% image 为待处理图像
% imagegoal 为处理后的图像
% x 为模糊的次数
function [image_opd, C] = blurringL16(image, x);
A = imread(image);
[M, N] = size(A);
% 下面是对 8 位图的处理方式
% A = double(A)/255;
```

```
% 下面是对 16 位图的处理方式
A = double( A) /65535;
original = A;
B = reshape( A, M, N) ;
blur = 1 /44*  [ 1 1  2 1 1; 1  2 2 2  1; 2 2 8  2 2; 1 2  2 2 1; 1  1 2 1  1] ;
for i = 1: x
Xe = zeros( M + 4, N + 4) ;
        Xe( 3: M + 2, 3: N + 2)  = B;
        Xe( 1, 3: N + 2)  = B( 1, 1: N) ;
        Xe( 2, 3: N + 2)  = B( 1, 1: N) ;
        Xe( M + 3, 3: N + 2)  = B( M, 1: N) ;
        Xe( M + 4, 3: N + 2)  = B( M, 1: N) ;
        Xe( 3: M + 2, 1)  = B( 1: M, 1) ;
        Xe( 3: M + 2, 2)  = B( 1: M, 1) ;
        Xe( 3: M + 2, N + 3)  = B( 1: M, N) ;
        Xe( 3: M + 2, N + 4)  = B( 1: M, N) ;
        Xe( 1: 2, 1: 2)  = B( 1, 1) ;
        Xe( M + 3: M + 4, N + 3: N + 4)  = B( M, N) ;
        Xe( M + 3: M + 4, 1: 2)  = B( M, 1) ;
        Xe( 1: 2, N + 3: N + 4)  = B( 1, N) ;
    C = conv2( Xe, blur, 'valid') ;
    B = C;
    end
    N = N/3;
    image_opd = reshape( C, M, N, 3) ;
    % 下面是对 8 位图的处理方式
    %  imwrite(  image_opd, imagegoal) ;
    % 下面是对 16 位图的处理方式
    imwrite( image_opd, 'temp2. png', 'BitDepth', 16) ;
    imshow( 'temp2. png') ;
```

下面我们来说明“模糊处理—检测相关性图”(W-SVD 抗模糊处理攻击曲线) 的绘制方法。

“模糊处理—检测相关性图”能刻画模糊处理次数对嵌入水印图像的影响, 其中的 x **轴表示模糊的次数,** y **轴表示检测器检测出的相关性值。编写函数** plotblurring. m **完成实验, 函数代码如下**:

```
% 文件名: plotblurring. m
```

```
% 程序员: 李鹏
% 编写时间: 2004.4.1
% 函数功能: 本函数用于绘制对加有水印的图像进行模糊处理(对 RGB 图像)后,检测相关性值与模糊程度的关系曲线
% 输入格式举例: plotblurring('test.png',20,'lenna.jpg',10,'db6',2,0.1,0.99);
% 函数说明:
% 横坐标表示模糊的次数,纵坐标为相关性值
% 参数说明:
% test 为已经加入水印的待检测图像
% original 为输入原始图像
% x 为模糊处理的最大次数
% seed 为随机数种子
% wavelet 为使用的小波函数
% level 为小波分解的尺度
% alpha 为水印强度
% ratio 为算法中 d/n 的比例
function plotblurring(test, x, original, seed, wavelet, level, alpha, ratio)
quality = 1:1:x;
corr_coef = zeros(max(size(quality)),1);
count = 0;
for q = quality
    count = count + 1;
    image_opd = blurringL16('test.png',q);
[corr_coef(count), corr_DCTcoef(count)] = wavedetect('temp2.png', original, seed, wavelet, level, alpha, ratio);
end
plot(quality, abs(corr_DCTcoef));
xlabel('模糊次数');
ylabel('相关性值');
```

模糊处理对加有水印的图像的破坏是很大的,而且随着模糊次数的增加,这种破坏还会更大,从图 9.29 中可以看出这两点,曲线的趋势也是符合我们直观理解的。

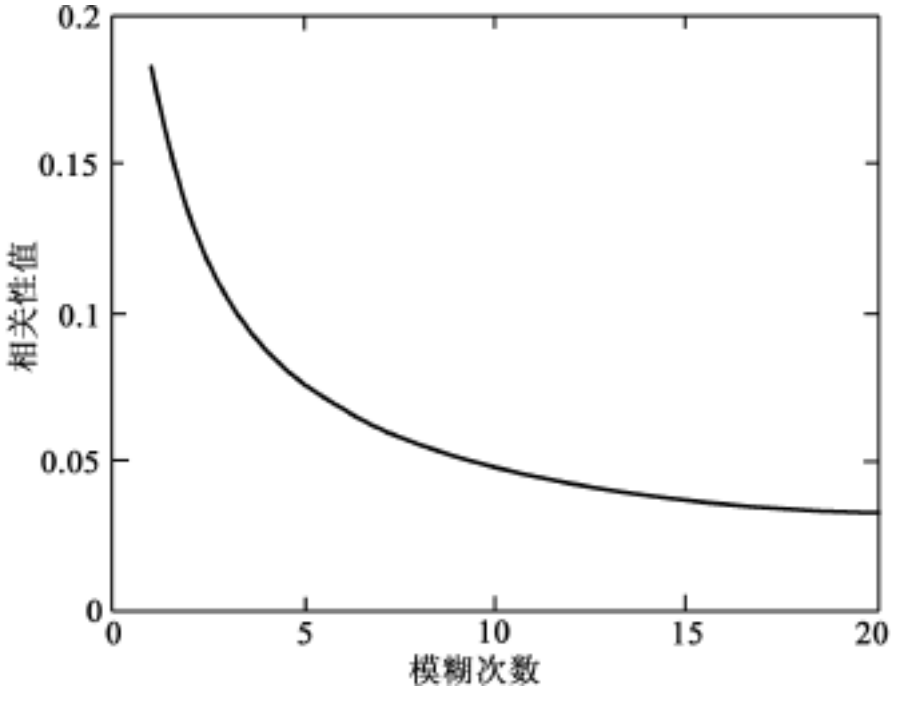


图 9.29　模糊处理—检测相关性值图

9.3　水印攻击者

有这样一些水印系统的攻击者是我们必须认识的。

(1) 无知的攻击者

所谓“无知的攻击者”是指试图非法获取某作品版权的人(组织)完全不知道该作品所含的水印是何种水印系统嵌入的,并且也不具有任何具体的与水印系统相关的工具(如水印检测器)。在这种情况下,攻击者只能依照对水印的一般认识来对作品进行攻击。使用的方法更多的也就是“无意攻击”。

(2) 拥有多个水印作品的攻击者

这一类攻击者的危险性是比较大的。在一般情况下,他们会有意识地收集大量的含有某水印系统所嵌入水印的作品。这样一来,即使他们不知道任何水印的具体算法,也可以利用手中拥有的大量样本的优势来去除水印。这一攻击的典型代表就是“合谋攻击”。

(3) 知道算法的攻击者

根据 Kerckhoff 准则,水印系统的算法往往是应该公开的,所以,这一类攻击者是最多的。完全掌握水印算法的他们可以通过分析算法的方式找到算法中的弱点,简单的话就可以通过穷举密钥来攻击,当穷举不现实时,也能结合其他方法攻击。

这一类攻击者另一个重点攻击的方向是水印检测器。一旦他们获得检测器无法校正的一个特定攻击手段时,就可以利用此手段成功实施对水印的削弱。在极端情况下,攻击者还可以获得水印检测器。使用检测器能给他们在攻击水印方面带来极大的好处。将检测器看做一个“黑盒”,攻击者可以将一个水印作品输出而得到相应

的检测区域,从而也能找到一些检测器的盲点进行作品伪造。

9.4 有意攻击

这一节我们介绍对水印攻击的另外一类方式,即有意攻击。顾名思义,这种攻击是攻击者有目的、有计划地对水印进行的攻击,以达到破坏水印、伪造水印和抽取水印等非法目的。

有意攻击可分为以下几类:

①伪造水印的抽取。攻击者从嵌有水印的图像中用自己的密钥产生一个并不存在的水印,用以说明对作品的所有权。

②伪造肯定检测。攻击者用一定的办法使水印检测器产生一个肯定的结论,用以说明自己是向作品嵌入了水印的。

③统计学上的水印抽取。

④多重水印。攻击者也在水印作品中加入自己的水印,并能够检测出来。

9.4.1 多重水印与解释攻击示例

对于可逆且非盲水印,IBM 攻击是一种典型的解释攻击。其基本原理为:设原始图像为 I,加入水印 W_A 得到加有水印的图像 $I_A = I + W_A$。攻击者首先生成自己的伪造水印 W_B,然后创建一个伪造的水印图像 $I_B = I_A - W_B$,即 $I_A = I_B + W_B$。此后攻击者声称自己拥有作品 I_A 的版权。由于在伪造作品 I_B 中既可以检测出 W_A 又可以检测出 W_B,从而给水印认证造成很难解释的困境。

类似 IBM 攻击,我们以 W-SVD 为例进行一次多重水印攻击。

已经在前面讲过,对有意攻击的防止和解决不仅仅需要技术手段,还要求有一套协议来防止这种攻击,而这种协议比水印嵌入的方法更加重要。

我们还是使用 lenna 图像为原始载体,在这里我们理解为需要保护版权的作品,我们向其中加入水印,使用到的参数为:a) $\alpha = 0.1$; b) $d/n = 0.99$; c) db6 小波; d) 2 尺度分解; e) seed = 10。图 9.30 显示的是水印图像和加入水印后的图像。

对该水印进行检测,得到检测相关性值为 0.83212。

对于没有任何保护协议的水印系统来说,攻击者的攻击行为就变得十分简单,他们获取传播的图像(即加有水印的图像)后,无需知道加入水印的种子,他们只需自己随便取一个种子(如 8),就可以再生成自己的水印,把这个水印再次加入到传播的图像中,他们可以声称自己是作品的制作者,也可以从水印作品中检测到自己的水印,使真正的所有者很难反驳。图 9.31(a)是第二重水印(种子为 8)图 9.31(b)是双重水印图像。可以看到,加一次水印的图像和加两次水印的图像是用肉眼区分不出来的。

对伪造的水印进行检测,得到相关性值为 0.76215,这也足够证明第二重水印的

水印形态图

加入水印后的图像

图 9.30　第一重水印和图像

（a）第二重水印形态图

（b）双重水印图像

图 9.31　第二重水印和双重水印图像

存在。

正如前面所提到的，防止多重水印攻击的方法是一个系统工程，涉及到仲裁协议、实施框架等多方面的问题。一般来说，原始作品的制作者再加入自己的水印后，是有必要将原始作品提供给可信任的第三方的，在提交作品给仲裁机构时还应该引入诸如 Hash 时戳等技术，这些都在一定程度上可以对抗解释攻击。

9.4.2　合谋攻击

所谓合谋攻击(collusion attack) 是指攻击者在拥有多个水印作品的情况下进行的一种攻击，在这种情况下，攻击者即使不知道算法，也常常可以利用这种优势来去除水印，或者使水印嵌入者很难检测出自己的水印。

合谋攻击有两种类型。

第一类合谋攻击是指攻击者设法获得包含同一水印的不同作品，并且将这些作品结合起来进行研究，用以了解算法是如何进行的。这一类攻击最简单的例子是攻击者对几个不同标志作品进行平均，如果加到所有作品的水印模板是相同的，这一平

均就可以得到与此模板非常接近的结果，然后攻击者只需从作品中将此模板减去，就可以去除水印了。

第二类合谋攻击是指攻击者获得了同一作品含有不同水印的副本，在这种情况下，攻击者可以通过结合几个独立的副本从而得到原始作品的精密近似。最简单的结合方法是平均所有副本，从而将不同的水印混合在一起并且减小它们各自的能量，使各自嵌入者很难检测到自己的水印。

下面就第二种类型的合谋攻击做出实验。

有五个嵌入者分别以 100，200，300，400，500 为种子在原始图像中嵌入自己的水印，得到的结果为图 9.32 所示的前五幅图像，攻击者获取了这五幅图像，并且将它们做平均，得到平均后的结果如图 9.32 所示的第六幅图。

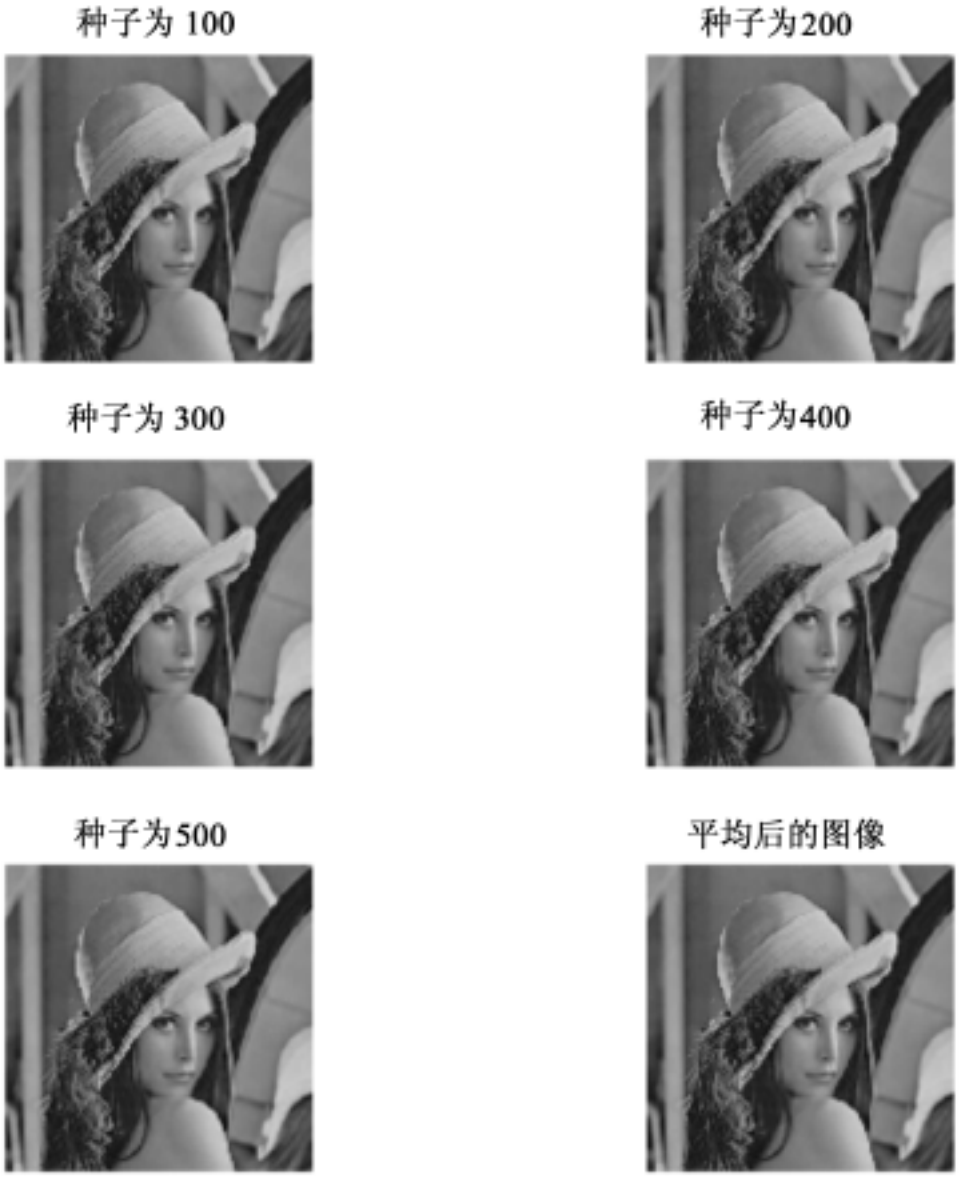


图 9.32　不同水印的图像与合谋攻击

表 9.3 是在各自的水印图像中使用各自的种子做 DCT 检测的相关性值。

表 9.3　　合谋攻击前相关性值表

种子	100	200	300	400	500
DCT 检测值	0.73716	0.76708	0.7682	0.81644	0.77644

表 9.4 是在平均后的水印图像中做检测的相关值。

表 9.4　　合谋攻击后相关性值表

种子	100	200	300	400	500
DCT 检测值	0.37884	0.3416	0.3429	0.35176	0.36618

可以看出，攻击过后的检测值明显降低，攻击者达到他的目的。

对于其他的有意攻击，有的需要采取的措施要复杂得多，所遵循的协议也很多，我们在此不再赘述，需要更多的了解可以参考有关资料。我们在这一章里简单地介绍了对水印系统的性能评价和对它的攻击。其实，评价水印系统是一个很复杂的技术，需要用到多方面的知识。对于水印的攻击，我们举例说明了一些，联系前面章节介绍的各种水印系统，可以实验验证一下它们各自抵抗不同攻击的优缺点，以方便实际应用中的选择。