第 5 章

CHAPTER 5

图 像 复 原

本章结构如图 5-1 所示，具体要求如下：

（1）掌握图像的退化/复原模型。

（2）掌握均值滤波器、统计排序滤波器的图像复原方法。

（3）掌握带阻滤波器、带通滤波器的图像复原方法。

（4）了解自适应滤波器、陷波滤波器的图像复原方法。

（5）了解因退化函数 H 而退化的图像复原方法。

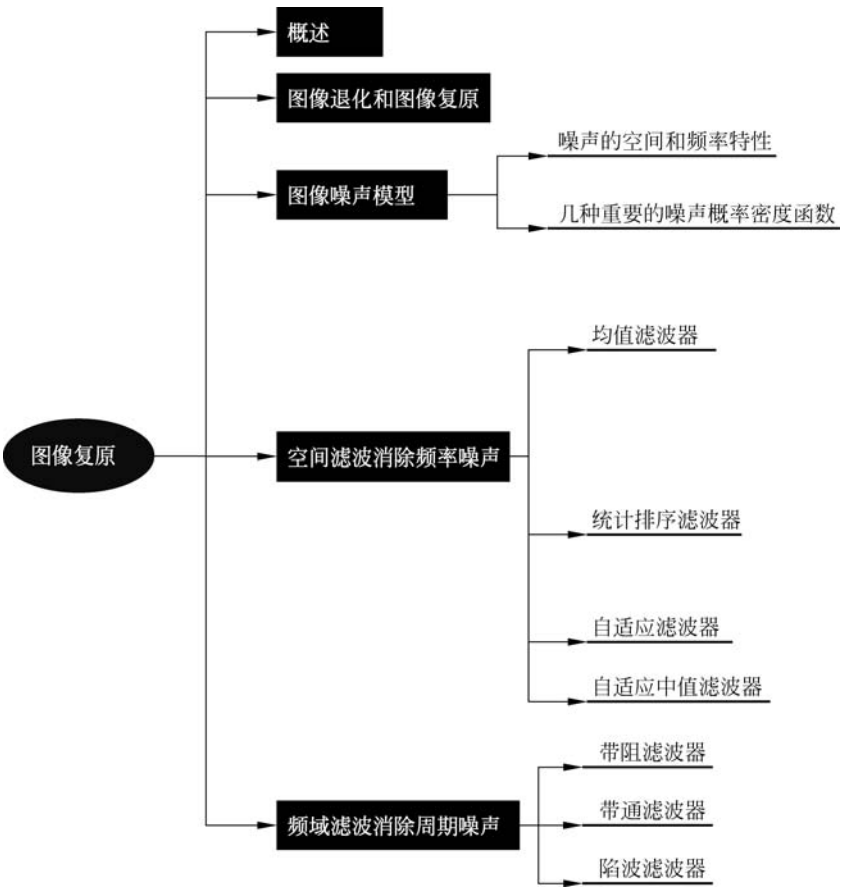


图 5-1　本章结构

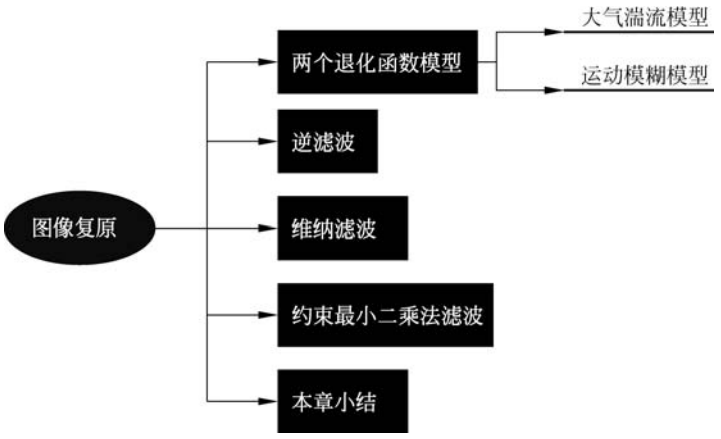


图 5-1 （续）

5.1 概述

图像在采集、传输和转换过程中，会产生一些噪声，表现为图像模糊、失真和有噪点等。图 5-2 展示了两种退化图像，退化原因分别是镜头聚焦不好和汽车运动，具体表现为图像模糊。在实际应用中需要清晰、高质量的图像，图像复原就是要尽可能准确地恢复退化图像，这个过程也称为图像退化的逆过程。典型的图像复原技术是根据图像退化的先验知识建立一个退化模型，并以此模型为基础，采用各种逆退化处理方法进行图像恢复，从而得到改善质量的复原图像。本章将详细地介绍图像复原技术，主要包括图像噪声模型、图像滤波，以及常用的图像复原方法等。

(a) 因镜头聚焦不好而引起的图像模糊　　(b) 因汽车运动而引起的图像模糊

图 5-2　图像退化示例

图像复原和图像增强都是为了改善图像的质量，但是两者是有区别的，主要在于：图像增强不考虑图像是如何退化的，而是试图采用各种技术来增强图像的视觉效果；图像复原需要知道图像退化的机制和过程等先验知识，并据此找到一种相应的逆退化处理方法，从而得到恢复的图像。

图像复原通常都会设立一个最佳准则，该准则将产生期望结果的最佳估计。相比之下，图像增强基本上是一个探索性过程，即根据人眼视觉系统的特性来改善图像。例如，对比度拉伸被认为是一种图像增强方法，去除图像模糊则被认为是一种图像复原方法。

一些图像复原方法适用于空域，如逆谐波均值滤波器适用于处理脉冲噪声。一些图像

复原方法则更适用于频域，如当退化仅仅是加性噪声时，空域处理就非常合适；但如果图像模糊这样的退化，在空域处理就很困难，而频域处理却很容易。基于不同优化准则的频域滤波是可选择的方法。

图像复原的目的是使质量降低或失真的图像恢复其原质量或内容。在图像的成像过程中，由于成像系统各种因素的影响，使图像质量降低，这种现象称为图像退化。图像复原可以看作是图像退化的逆过程，是对图像退化的过程进行估计，建立退化过程数学模型，并补偿退化过程所造成的图像失真。图像复原试图利用退化过程的某种先验知识来复原退化的图像，因而，图像复原方法是面向图像退化模型的。

引起图像退化的原因有以下几种。

(1) 成像系统的带宽有限、畸变、像差等造成图像失真。

(2) 扫描非线性及成像器件拍摄姿势造成图像几何失真。

(3) 事物与成像传感器之间存在相对运动，造成图像模糊。

(4) 由于成像传感器和光学系统的不均匀特性，使物体亮度相同而成像灰度不同，造成图像失真。

(5) 在场景能量传输通道中的大气成分变化和湍流效应等介质特性使得图像出现辐射失真。

(6) 在图像的采集、数字化、成像及进行图像处理时被噪声污染。

5.2 图像退化和图像复原

图像复原在图像处理中有非常重要的研究意义。图像复原最基本的任务是在去除图像噪声的同时，不丢失图像的细节信息。然而去除噪声和保持细节往往是矛盾的，也是图像处理中至今尚未得到很好解决的一个问题。

图像退化和复原模型如图 5-3 所示，图像退化过程被建模为一个退化函数 H 和一个加性噪声 $\eta(x,y)$，其中，$f(x,y)$为原图像，$g(x,y)$为退化图像，$\hat{f}(x,y)$为复原图像。给定 $g(x,y)$、退化函数 H 和加性噪声项 $\eta(x,y)$后，图像复原的目的就是获得原图像 $f(x,y)$的一个估计子(x,y)。通常，我们希望这一估计尽可能地接近原图像，并且 H 和 η 的信息知道得越多，复原图像 $\hat{f}(x,y)$就会越接近原图像 $f(x,y)$。本章复原方法都是以不同类型的图像复原滤波器为基础。

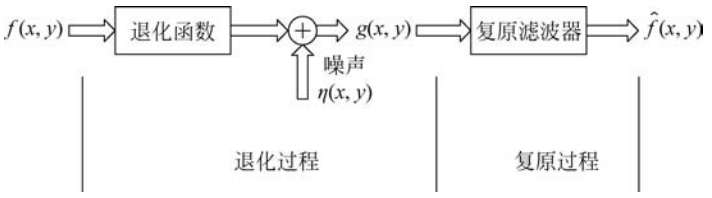


图 5-3 图像退化和复原模型

如果 H 是一个线性的、位置不变的过程，那么空域中的退化图像可由式(5-1)给出。

$$g(x,y)=f(x,y)\otimes h(x,y)+\eta(x,y) \tag{5-1}$$

其中，$h(x,y)$表示空域的退化函数，“$\otimes$”表示卷积运算。由于空域中的卷积运算等同于频

域中的乘法运算，因此式(5-1)在频域的等价表示如式(5-2)所示。

$$G(u,v)=H(u,v)F(u,v)+N(u,v) \tag{5-2}$$

其中，$G(u,v)$表示频域的退化图像，$H(u,v)$表示频域的退化函数，$N(u,v)$表示频域的噪声。

图像复原是在已知 $g(x,y)$、$h(x,y)$、$\eta(x,y)$等先验知识的条件下，来求解 $f(x,y)$的过程。

图像复原是根据图像退化的原因，从退化的图像中提取所需要的信息，建立相应的数学模型，对图像进行逆退化处理，以恢复图像的过程。图像复原是设计一个复原滤波器，从退化图像 $g(x,y)$中计算得到真实图像的估计值 $\hat{f}(x,y)$，最大程度地接近原图像 $f(x,y)$。

5.3 图像噪声模型

数字图像的噪声主要来自图像的获取和传输过程。成像传感器的性能可能受各种因素的影响，如图像获取时的环境条件和传感元器件质量，使图像在获取时受到干扰。图像在传输中被污染的主要原因是传输信道中的干扰，例如，通过无线网络传输的图像会因为光照或其他大气因素而受到污染。有的噪声具有一定的规律，有的噪声是随机的。这些噪声可以通过概率密度函数来表示。

5.3.1 噪声的空间和频率特性

在图像复原中，需要定义噪声空间特性参数，以及噪声与图像的相关性。噪声的频率特性是指傅里叶域中噪声的频率内容(即相对于电磁波谱的频率)。例如，当噪声的傅里叶谱是常量时，噪声通常称为白噪声。这个术语是从白光的物理特性派生出来的。白光以相等的比例包含可见光谱中的所有频率。

本章假设噪声独立于空间坐标，并且噪声与图像不相关，即像素值与噪声分量的值不相关。虽然这种假设在某些应用中(例如 X 射线成像和核医学成像)是无效的，但这超出了本章的范围，故不进行讨论。

5.3.2 几种重要的噪声概率密度函数

对噪声特性和影响进行模拟是图像复原的核心。本章介绍两种基本的噪声模型：空域中的噪声(由噪声概率密度函数描述)和频域中的噪声(由噪声的傅里叶性质描述)。同时，本章中假设噪声与图像坐标无关。图 5-4 所示是几种重要的概率密度函数曲线，其中，z 表示灰度值，$P(z)$表示灰度值出现的概率。

1. 高斯噪声概率密度函数

在空域和频域中，由于高斯噪声(也称为正态噪声)在数学上具有易处理性，故应用中常用这种噪声模型。

高斯噪声随机变量 z 的概率密度函数由式(5-3)给出。

$$P(z)=\frac{1}{\sqrt{2\pi}\sigma}\mathrm{e}^{-(z-\mu)^2/2\sigma^2} \tag{5-3}$$

其中，μ 表示均值，σ 表示标准差。

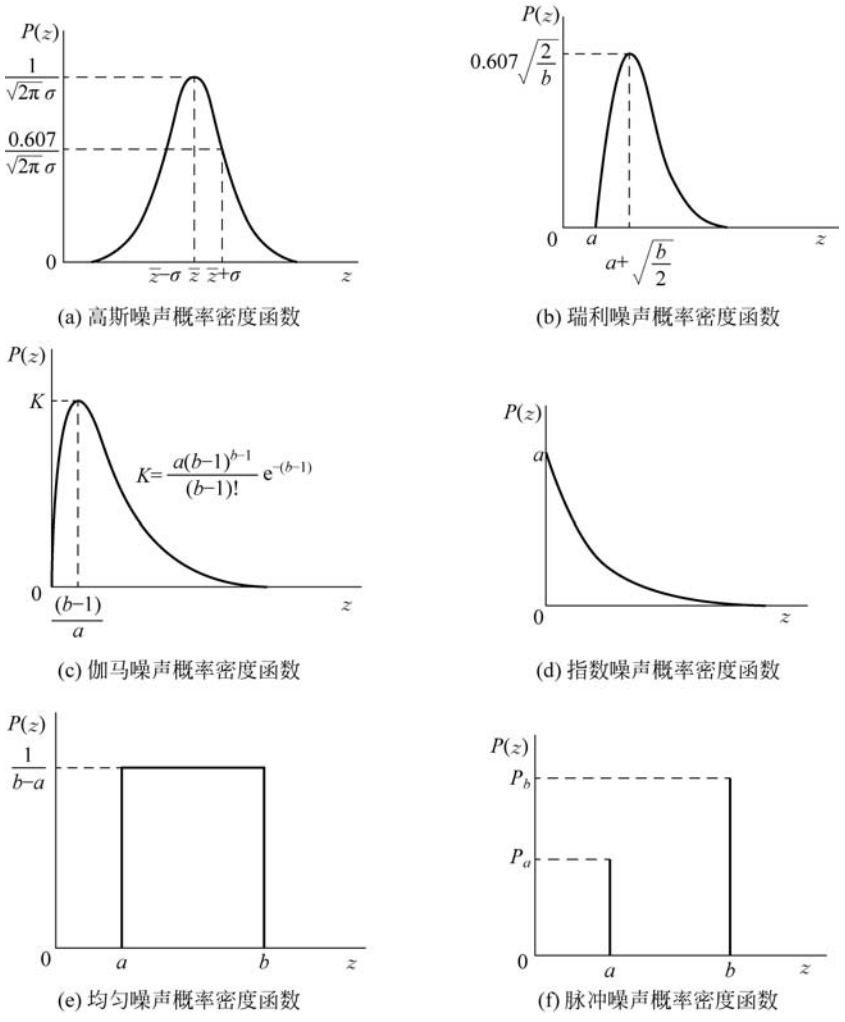


图 5-4　几种重要的噪声概率密度函数曲线

在式(5-3)中，z 值有 70%的概率落在区间$[(\mu-\sigma),(\mu+\sigma)]$，有 95%的概率落在区间$[(\mu-2\sigma),(\mu+2\sigma)]$。

2. 瑞利噪声概率密度函数

瑞利噪声概率密度函数由式(5-4)给出。

$$P(z)=\begin{cases}\dfrac{2}{b}(z-a)\mathrm{e}^{-(z-a)^2/b}, & z\geqslant a\\ 0, & z<a\end{cases} \tag{5-4}$$

均值 $\mu=a+\sqrt{\pi b/4}$，方差 $\sigma^2=\dfrac{b(4-\mu)}{4}$。

注意：由图 5-4(b)可以看出，距离原点的位移 a 和函数曲线的基本形状向右变形。瑞利噪声概率密度函数十分适用于近似歪斜的直方图。

3. 伽马噪声概率密度函数

伽马噪声概率密度函数由式(5-5)给出。

$$P(z)=\begin{cases}\dfrac{a^b z^{b-1}}{(b-1)!}\mathrm{e}^{-az}, & z\geqslant a\\ 0, & z<a\end{cases} \tag{5-5}$$

其中，$a>0$；b 为正整数；“!”表示阶乘运算。均值 $\mu=\dfrac{b}{a}$，方差 $\sigma^2=\dfrac{b}{a^2}$。

4. 指数噪声概率密度函数

指数噪声概率密度函数由式(5-6)给出。

$$P(z)=\begin{cases}a\mathrm{e}^{-az}, & z\geqslant a\\ 0, & z<a\end{cases} \tag{5-6}$$

均值 $\mu=\dfrac{1}{a}$，方差 $\sigma^2=\dfrac{1}{a^2}$。

当 $b=1$ 时，伽马噪声概率密度函数就成为指数噪声概率密度函数。

5. 均匀噪声概率密度函数

均匀噪声的概率密度函数由式(5-7)给出。

$$P(z)=\begin{cases}\dfrac{1}{b-a}, & a\leqslant z\leqslant b\\ 0, & \text{其他}\end{cases} \tag{5-7}$$

均值 $\mu=\dfrac{a+b}{2}$，方差 $\sigma^2=\dfrac{(b-a)^2}{12}$。

6. 脉冲噪声概率密度函数

双极脉冲噪声概率密度函数由式(5-8)给出。

$$P(z)=\begin{cases}P_a, & z=a\\ P_b, & z=b\\ 0, & \text{其他}\end{cases} \tag{5-8}$$

如果 $b>a$，则灰度级 b 在图像中将显示为一个亮点；反之，则灰度级 a 在图像中将显示为一个暗点。若 P_a 或 P_b 为 0，则双极脉冲噪声概率密度函数变为单极脉冲噪声概率密度函数。如果 P_a 和 P_b 均不为 0，尤其是近似相等时，则脉冲噪声将类似于图像上随机分布的胡椒和盐粉微粒。由于这个原因，双极脉冲噪声又称为椒盐噪声、散粒噪声或尖峰噪声。本章将使用脉冲噪声和椒盐噪声这两个术语。

脉冲噪声可以为正(正脉冲)也可以为负。与图像信号的强度相比，脉冲污染通常较大，在图像中，脉冲噪声通常被数字化为最小值(或最大值)。因此，假设 a 和 b 是饱和值，从某种意义上看，在数字化图像中，它们等于所允许的最大值和最小值。基于这一假设，负脉冲以黑点(胡椒点)出现在图像中，正脉冲以白点(盐粒点)出现在图像中。对于一幅 8bit 图像而言，可以看到 $a=0$(黑点)和 $b=255$(白点)。

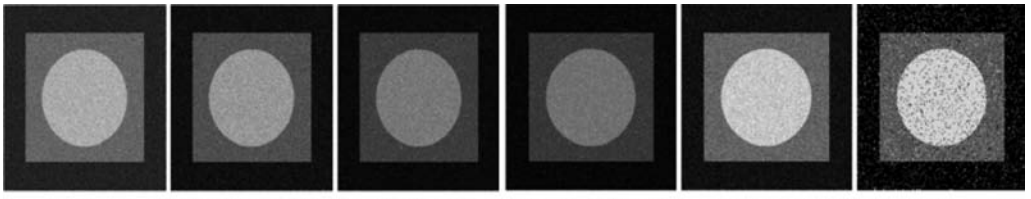
图 5-5 测试图像

图 5-5 显示了一幅非常适合于说明上述几种噪声概率密度函数的测试图案(不含噪声)。这幅图案由简单的、恒定的区域组成，且从纯黑到接近纯白仅仅有 3 个灰度级。这方便对图

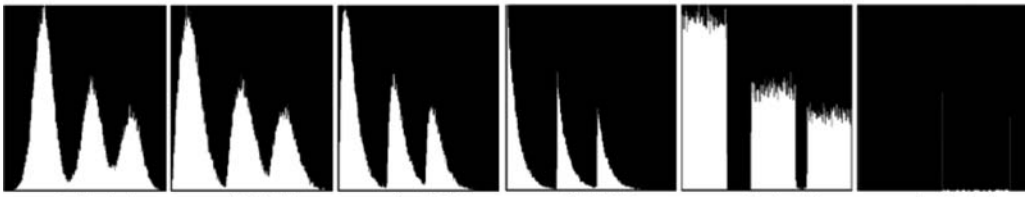
像的各种噪声进行分析。

叠加了噪声的测试图像及其直方图如图 5-6 所示。每种情况选择合适的噪声参数，使测试图案的 3 种灰度级的直方图便于观察。

由图 5-6(l)可以看出，椒盐噪声的直方图为双极脉冲。不同于其他噪声，椒盐噪声分量为纯黑或纯白。图 5-6(a)～图 5-6(e)除了少许亮度不同外，并没有太大不同，即使它们的直方图有明显的区别。因此，椒盐噪声是唯一一种使图像退化、视觉上可区分的噪声类型。



(a) 叠加高斯噪声 (b) 叠加瑞利噪声 (c) 叠加伽马噪声 (d) 叠加指数噪声 (e) 叠加均匀噪声 (f) 叠加椒盐噪声

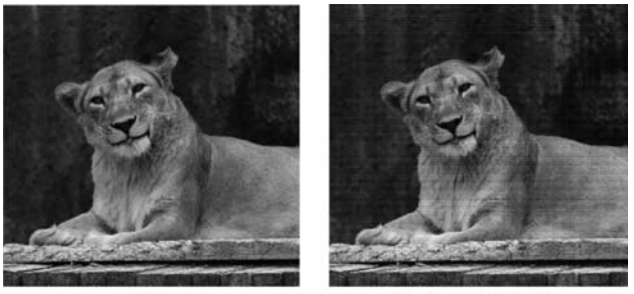


(g) 高斯噪声测试图像直方图 (h) 瑞利噪声测试图像直方图 (i) 伽马噪声测试图像直方图 (j) 指数噪声测试图像直方图 (k) 均匀噪声测试图像直方图 (l) 椒盐噪声测试图像直方图

图 5-6 叠加噪声的测试图像及其直方图

7. 周期噪声

周期噪声是图像在获取过程中受电力或机电干扰所产生的空间相关噪声，可以通过频域滤波来减少。周期噪声更趋向于产生频率尖峰。被周期噪声污染的图像如图 5-7 所示。



(a) 原图像 (b) 含周期噪声图像

图 5-7 被周期噪声污染的图像

5.4 空间滤波消除频率噪声

当图像中唯一导致图像退化的因素是噪声时，式(5-1)变为式(5-9)的形式。

$$g(x,y)=f(x,y)+\eta(x,y) \tag{5-9}$$

噪声项是未知的，故从 $g(x,y)$ 中减去噪声项不是一个现实的选择。在仅存在加性噪

声的情况下，可以选择空间滤波的方法来去除噪声。

5.4.1 均值滤波器

均值滤波器包括算术均值滤波器、几何均值滤波器、谐波均值滤波器和逆谐波均值滤波器。

1. 算术均值滤波器

算术均值滤波器的数学表达式如式(5-10)所示。

$$\hat{f}(x,y)=\frac{1}{mn}\sum_{(s,t)\in S_{xy}} g(s,t) \tag{5-10}$$

其中，S_{xy} 表示中心在(x,y)，尺寸为 $m\times n$ 的矩形窗口。算术均值滤波器能使图像的局部变化更平滑，在减少噪声的同时也会产生模糊效果，适用于高斯噪声和均匀噪声。

2. 几何均值滤波器

几何均值滤波器的数学表达式如式(5-11)所示。

$$\hat{f}(x,y)=\prod_{(s,t)\in S_{xy}} g(s,t) \tag{5-11}$$

几何均值滤波器能做到的平滑度和算术均值滤波器相差不大，但丢失的图像细节更少，即相对锐化。

图 5-8 为算术均值滤波器与几何均值滤波器处理含高斯噪声图像的结果对比。图 5-8(a)为原图像，图 5-8(b)为含均值为 0、方差为 400 的加性高斯噪声图像。图 5-8(c)和图 5-8(d)分别为大小为 3×3 的算术均值滤波器和同样大小的几何均值滤波器滤除噪声后的结果。可以看出，这两种噪声滤波器都起到了减少噪声的作用，但几何均值滤波器的处理结果图中，文字边缘更加锐化，猫的身体边缘更加清晰，即几何均值滤波器并未像算术均值滤波器那样，使图像变得模糊。

(a) 原图像

(b) 含高斯噪声图像

(c) 算术均值滤波器处理结果图像

(d) 几何均值滤波器处理结果图像

图 5-8　算术均值滤波器与几何均值滤波器处理结果对比

3. 谐波均值滤波器

谐波均值滤波器数学表达式如式(5-12)所示。

$$\hat{f}(x,y)=\frac{mn}{\sum\limits_{(s,t)\in S_{xy}}\frac{1}{g(s,t)}} \tag{5-12}$$

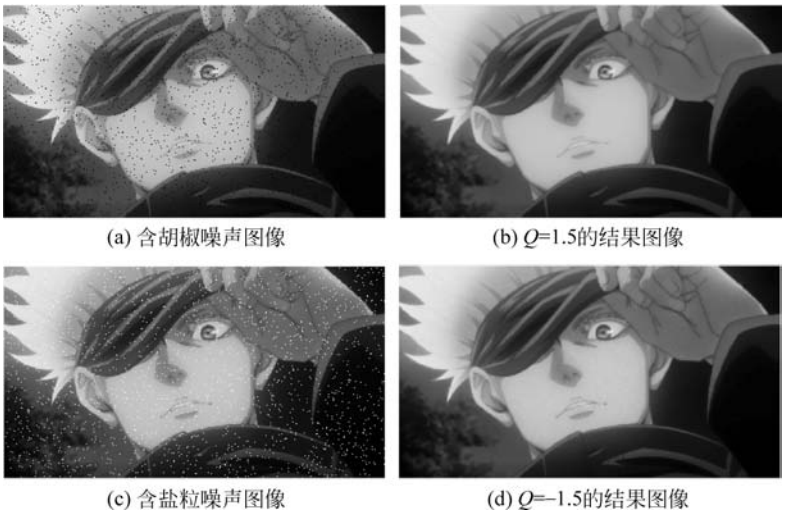
谐波均值滤波器对于盐粒噪声效果好且善于处理高斯噪声,但不适用于胡椒噪声。

4. 逆谐波均值滤波器

逆谐波均值滤波器数学表达式如式(5-13)所示。

$$\hat{f}(x,y)=\frac{\sum\limits_{(s,t)\in S_{xy}}g(s,t)^{Q+1}}{\sum\limits_{(s,t)\in S_{xy}}g(s,t)^{Q}} \tag{5-13}$$

其中,Q 表示滤波器的阶数。逆谐波均值滤波器适合减少或消除椒盐噪声。当 Q 符号为正时,该滤波器可以消除胡椒噪声;当 Q 符号为负时,该滤波器可以消除盐粒噪声。但是,逆谐波均值滤波器不能同时消除这两种噪声。当 $Q=0$ 时,逆谐波均值滤波器变为算术均值滤波器;当 $Q=-1$ 时,则为谐波均值滤波器。图 5-9 为 Q 为不同符号时,逆谐波均值滤波器分别对含胡椒噪声和盐粒噪声的图像进行处理的结果对比。



(a) 含胡椒噪声图像　(b) Q=1.5的结果图像

(c) 含盐粒噪声图像　(d) Q=-1.5的结果图像

图 5-9　Q 为不同符号时逆谐波均值滤波器处理结果对比

图 5-9(a)为含概率为 0.1 的胡椒噪声图像,图 5-9(c)为含相同概率的盐粒噪声图像。图 5-9(b)为 $Q=1.5$ 的逆谐波均值滤波器胡椒噪声的去噪结果。图 5-9(d)为 $Q=-1.5$ 的逆谐波均值滤波器对盐粒噪声的去噪结果。可以看出,两种阶数的逆谐波均值滤波器都有很好的去除噪声效果,其中,正阶数逆谐波均值滤波器除了使暗区变得稍微有些淡化和模糊外,还使背景变得更为清晰;而负阶数逆谐波均值滤波器的作用正好相反。

图 5-10 为当 Q 符号错误时,逆谐波均值滤波器的处理结果。可以看出,这种情况下的逆谐波均值滤波器不仅没有去除噪声的作用,反而使噪声更加明显。

图 5-10(d)为图 5-10(a)使用 $Q=-1.5$ 的逆谐波均值滤波器的去噪结果,图 5-10(e)为

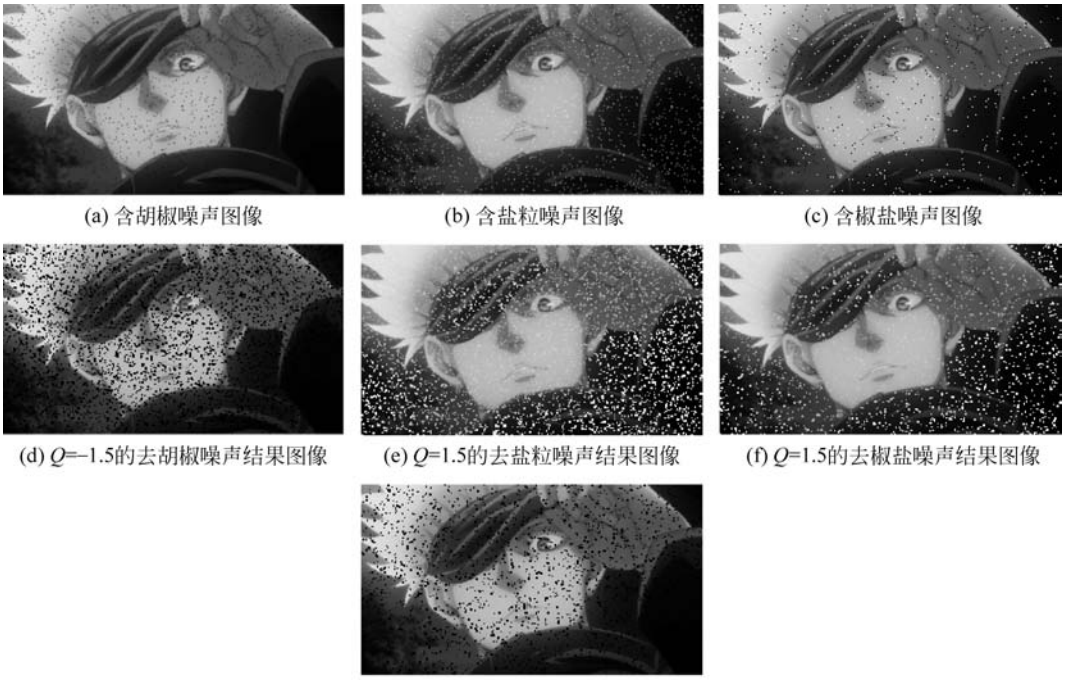


图 5-10　Q 符号错误时逆谐波均值滤波器的处理结果对比

图 5-10(b)使用 $Q=1.5$ 的逆谐波均值滤波器的去噪结果。可以看出,两个结果都是噪声污染变得更严重。图 5-10(f)和图 5-10(g)为分别使用上述两种阶数的逆谐波均值滤波器对图 5-10(c)的去噪结果。可以看出,被椒盐噪声污染的图像无法使用逆谐波均值滤波器来进行去噪处理。由此可见,逆谐波均值滤波器更适合处理脉冲噪声,但有一个缺点:即必须知道噪声是暗噪声还是亮噪声,以便于选择正确 Q 符号。

5.4.2　统计排序滤波器

统计排序滤波器是空域滤波器,空域滤波器的响应由该滤波器范围内图像像素值的排序结果来决定。

1. 中值滤波器

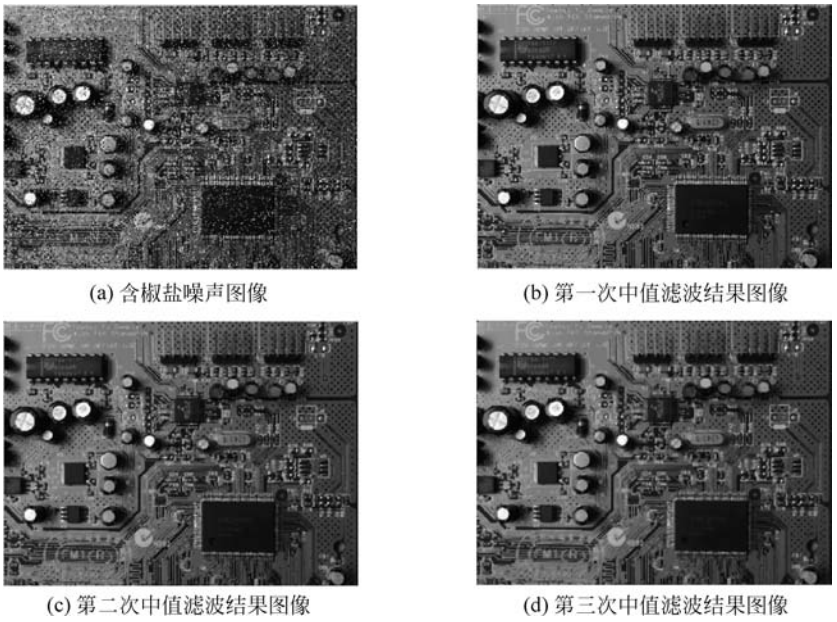
中值滤波器使用一个像素邻域中的像素灰度值的中值来作为该像素的值,即

$$\hat{f}(x,y)=\underset{(s,t)\in S_{xy}}{\text{median}}\{g(s,t)\} \tag{5-14}$$

中值滤波器的应用非常普遍,这是因为对于某些类型的随机噪声,它们可提供良好的去噪能力,且比相同尺寸的线性平滑滤波器造成的图像模糊更少。在存在单极或双极脉冲噪声的情况下,中值滤波器尤其有效。

中值滤波器对含椒盐噪声图像处理 3 次的结果如图 5-11 所示。

图 5-11(a)为含概率为 $P_a=P_b=0.1$ 的椒盐噪声图像。图 5-11(b)为用大小为 3×3 的中值滤波器第一次滤波的结果。可以看出,本次滤波对图 5-11(a)的改进是显而易见的,但还有一些噪声点仍然可见。图 5-11(c)为使用中值滤波器对图 5-11(b)进行第二次滤波



(a) 含椒盐噪声图像　(b) 第一次中值滤波结果图像

(c) 第二次中值滤波结果图像　(d) 第三次中值滤波结果图像

图 5-11　中值滤波器对含椒盐噪声图像的 3 次处理结果

处理结果。可以发现，第二次滤波去掉了大部分噪声点，仅剩下非常少的可见噪声点。这些噪声点在经过第三次中值滤波处理后全部消除了，如图 5-11(d)所示。这些结果说明中值滤波器具有很好的脉冲噪声处理能力。但是，可以看出，图 5-11(d)的锐化程度明显低于图 5-11(b)，图中右下角的引脚部分明显变得模糊不清。这是因为中值滤波器虽然能够减少噪声，但同时会使图像变得模糊，因此，应该控制其使用次数。

2. 最大值和最小值滤波器

中值相当于由小到大排列的数组中间的那个数，统计学认为也可以使用序列中的最后一个数值，这便得到了最大值滤波器，由式(5-15)给出。

$$\hat{f}(x,y)=\max_{(s,t)\in S_{xy}}\{g(s,t)\} \tag{5-15}$$

最大值滤波器对发现图像中的最亮点非常有用。同样，因为“胡椒”噪声的亮度很低，所以，可以使用最大值滤波器进行处理。

同理，使用起始值的滤波器称为最小值滤波器，由式(5-16)给出。

$$\hat{f}(x,y)=\min_{(s,t)\in S_{xy}}\{g(s,t)\} \tag{5-16}$$

最小值滤波器对发现图像中的最暗点非常有用。同样，最小值滤波器可以去除盐粒噪声。最大值和最小值滤波器对噪声的处理结果如图 5-12 所示。

图 5-12(b)为最大值滤波器处理“胡椒”噪声的结果图像。可以看出，这种滤波器对去除图像中的“胡椒”噪声的确很合适，但同时也从黑色物体的边缘去除了一些黑色像素(将这些像素设置为亮灰度级)。图 5-12(d)为最小值滤波器处理“盐粒”噪声的结果图像。可以发现，最小值滤波器的确比最大值滤波器能更好地处理“盐粒”噪声，但同时也从明亮物体的边缘去除了一些白色像素，从而使亮物体变小，使暗物体变大，这是因为围绕这些亮物体的白点被设置成了暗灰度级。

(a) 含胡椒噪声图像　　(b) 使用最大值滤波器处理的结果图像

(c) 含盐粒噪声图像　　(d) 使用最小值滤波器处理的结果图像

图 5-12　最大值和最小值滤波器对噪声的处理结果

3. 中点滤波器

中点滤波器为简单地计算滤波器区域中像素灰度值的最大值和最小值之间的均值，即

$$\hat{f}(x,y)=\frac{1}{2}\left[\max_{(s,t)\in S_{xy}}\{g(s,t)\}+\min_{(s,t)\in S_{xy}}\{g(s,t)\}\right] \tag{5-17}$$

它对于去除随机分布噪声的效果最好，如高斯噪声或均匀分布噪声。

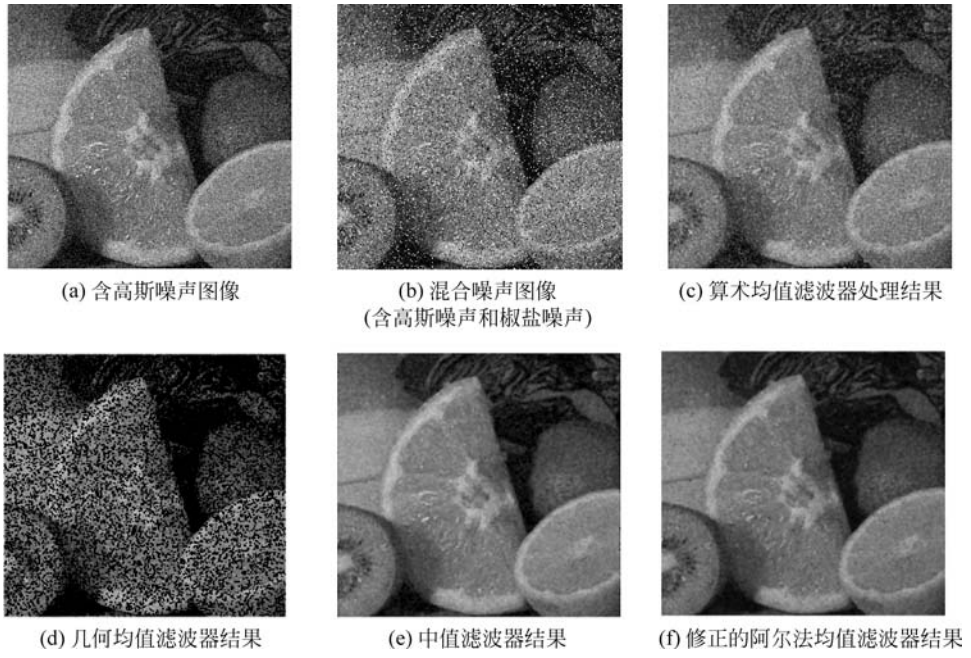
4. 修正的阿尔法均值滤波器

假设在 S_{xy} 邻域内，将灰度值按从小到大的顺序进行排序，并去掉前 $d/2$ 和后 $d/2$ 的数据。令 $g_r(s,t)$ 表示剩下的 $d=mn-1$ 个像素。由这些剩余像素平均值形成的滤波器称为修正的阿尔法均值滤波器，如式(5-18)所示。

$$\hat{f}(x,y)=\frac{1}{mn-d}\sum_{(s,t)\in S_{xy}}g_r(s,t) \tag{5-18}$$

其中，d 的取值范围为 0～$(mn-1)$。当 $d=0$ 时，修正的阿尔法均值滤波器为算术均值滤波器。当 $d=mn-1$ 时，则修正的阿尔法均值滤波器为中值滤波器。当 d 取其他值时，修正的阿尔法均值滤波器适用于多种混合噪声，例如高斯噪声和椒盐噪声的混合噪声。几种滤波器对由高斯噪声和椒盐噪声组成的混合噪声的处理结果如图 5-13 所示。

图 5-13(a)为含均值为 0、方差为 800 的高斯噪声图像。图 5-13(a)又叠加了 $P_a=P_b=0.1$ 的椒盐噪声，使图像质量进一步退化了，如图 5-13(b)所示。图 5-13(c)～图 5-13(f)依次显示了使用大小为 5×5 的算术均值滤波器、几何均值滤波器、中值滤波器和修正的阿尔法均值滤波器($d=5$)处理后的图像。由于椒盐噪声的存在，算术均值滤波器和几何均值滤波器(尤其是后者)并没有产生良好的去噪效果，而中值滤波器和修正的阿尔法均值滤波器的去噪则好得多。由此可见，在降噪方面修正的阿尔法均值滤波器的效果更好一些。d 越高，修正的阿尔法均值滤波器越接近中值滤波器的性能，但仍然保留了一些平滑能力。



(a) 含高斯噪声图像　(b) 混合噪声图像（含高斯噪声和椒盐噪声）　(c) 算术均值滤波器处理结果

(d) 几何均值滤波器结果　(e) 中值滤波器结果　(f) 修正的阿尔法均值滤波器结果

图 5-13　几种滤波器对混合噪声的处理结果

5.4.3　自适应滤波器

前面几种滤波器在应用时，并未考虑图像的不同位置会具有不同特性。有些图像应用可以通过使用能够根据被滤波区域的图像特性而自适应的滤波器来改进处理结果，但是，目前讨论的用于图像处理的滤波器并没有考虑图像中的一点对于其他点的特征变化。本节将考虑两个简单的自适应滤波器，它们的特性变化以大小为 $m \times n$ 的矩形窗口 S_{xy} 所定义的滤波器区域内的图像统计特性为基础。自适应滤波器的性能要优于本书已讨论的所有滤波器的性能，当然，改善滤波器性能的代价是滤波器的复杂度变高了。

对于随机变量而言，最简单的统计度量是均值和方差。这些参数也是自适应滤波器的基础，因为它们与图像质量紧密相关。均值给出了计算区域中平均灰度的度量，方差则给出了该区域的对比度的度量。

自适应滤波器行为变化基于矩形窗口 S_{xy} 所定义图像区域内的统计特性，其响应基于以下 4 个参数，具体计算式如式(5-19)所示。

(1) $g(x,y)$：表示噪声图像在点(x,y)的灰度值。

(2) σ_{η}^2：表示 $g(x,y)$的噪声灰度值方差。

(3) m_L：表示在 S_{xy} 内像素的灰度值均值。

(4) σ_{η}^2：表示在 S_{xy} 内像素的灰度值方差。

$$\hat{f}(x,y)=g(x,y)-\frac{\sigma_{\eta}^2}{\sigma_L^2}[g(x,y)-m_L] \tag{5-19}$$

在式(5-19)中，唯一未知的是方差 σ_{η}^2，其他参数通过计算(x,y)处 S_{xy} 中的像素获得，其中，(x,y)是滤波器窗口的中心。图像模型的噪声是加性和位置独立的。

自适应滤波器与均值滤波器对图像的处理结果对比如图 5-14 所示。

图 5-14　自适应滤波器与均值滤波器对图像的处理结果对比

图 5-14(a)为原图像，图 5-14(b)为被均值为 0、方差为 1000 的加性高斯噪声污染的图像，其噪声污染比较严重。图 5-14(c)为使用大小为 7×7 的算术均值滤波器处理噪声后的结果图像。可以看出，图 5-14(c)的噪声被平滑掉了，但其代价是图像变得模糊了。图 5-14(e)与图 5-14(c)类似，它为使用大小为 7×7 的几何均值滤波器处理噪声后的结果图像。可以发现，图 5-14(e)所示图像出现了一些黑点，这是因为滤波器窗口过大，使像素灰度值为 0 的点被放大。当如图 5-14(d)选择较小的窗口如 3×3 时，黑点明显减少，但此时图像噪声依然明显。

图 5-14(f)为使用自适应滤波器处理噪声的结果图像，其中，$\sigma_\eta^2=1000$。与图 5-14(c)和图 5-14(e)相比，可以看出，图 5-14(f)的改进是很大的。从噪声减少的情况来看，自适应滤波器所达到的效果与算术均值滤波器和几何均值滤波器相似。但是，自适应滤波器滤波处理后的图像更清晰，例如，在图 5-14(f)中，帽子的边缘部分更清晰。这是自适应滤波器处理图像的典型结果。

在图 5-14(f)中，自适应滤波器的方差 $\sigma_\eta^2=1000$，该值准确地与噪声方差相匹配。若该值未知，且使用的估计值太小，自适应滤波器则会因校正量比应有的小，返回与原图像质量非常接近的结果图像。若估计值太高，则会造成方差的比例在 1.0 处被削平，并且自适应滤波器会比正确校正量下更频繁地从图像中减去平均值。如果允许灰度值为负值，且允许图像在处理后被重新标定，则如前所述，自适应滤波器处理后的结果图像将损失动态范围。

5.4.4　自适应中值滤波器

对于中值滤波器而言，只要脉冲噪声的空间密度不大，其性能就会很好（P_a 和 P_b 小于 0.2）。而自适应中值滤波可以处理具有更大概率的脉冲噪声，并且在处理平滑非脉冲噪声时能够保留更多的图像细节。而后者是传统中值滤波器所做不到的。自适应中值滤波器也工作于矩形窗口 S_{xy} 内，然而，与传统滤波器不同的是在进行滤波处理时会根据某些条件

来改变 S_{xy} 的尺寸。

自适应滤波器的输出是一个单值，该值用于代替(x,y)处的像素值，其中，(x,y)是 S_{xy} 的中心。自适应中值滤波器考虑的参数如下。

(1) $Z_{\min}$：表示 S_{xy} 中最小灰度值。

(2) $Z_{\max}$：表示 S_{xy} 中最大灰度值。

(3) Z_{med}：表示 S_{xy} 中灰度值的中值。

(4) Z_{xy}：表示(x,y)处的灰度值。

(5) $S_{\max}$：表示 S_{xy} 允许的最大矩形窗口尺寸。

自适应中值滤波器工作进程包括两部分，具体如下。

进程 A：

步骤 1　$A_1=Z_{\text{med}}-Z_{\min}$；

步骤 2　$A_2=Z_{\text{med}}-Z_{\max}$；

步骤 3　如果 $A_1>0$ 且 $A_2<0$，转到进程 B 的步骤 1，否则增大矩形窗口尺寸；

步骤 4　如果矩形窗口尺寸$\leqslant S_{\max}$，转到进程 A 的步骤 1，否则输出 Z_{xy}。

进程 B：

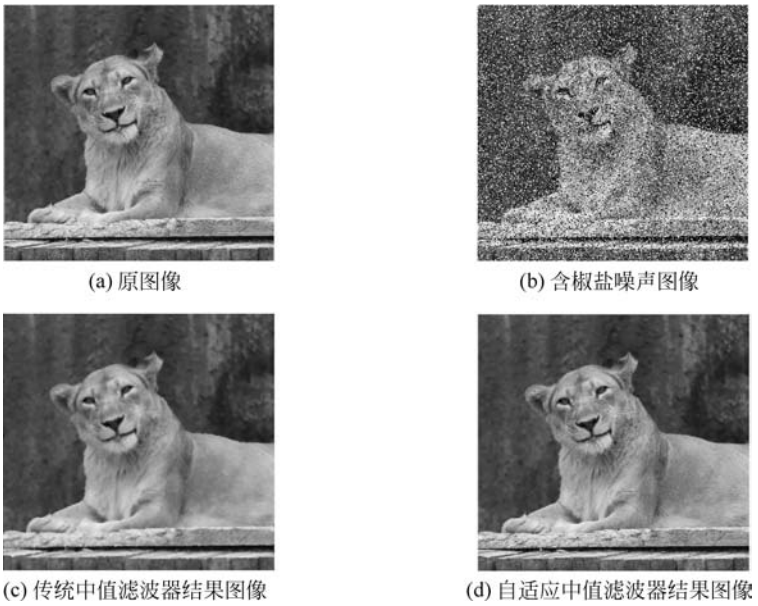
步骤 1　$B_1=Z_{xy}-Z_{\min}$；

步骤 2　$B_2=Z_{xy}-Z_{\max}$；

步骤 3　如果 $B_1>0$ 且 $B_2<0$，输出 Z_{xy}，否则输出 Z_{med}。

理解自适应滤波器原理的关键在于记住它有 3 个主要目的：去除椒盐噪声、平滑非脉冲噪声、减少失真(如物体边界细化或粗化等)。$Z_{\min}$ 和 $Z_{\max}$ 在统计上被认为是类脉冲噪声分量，即使它们在图像中可能并不是最低和最高的像素灰度值。自适应滤波器每输出一个值，S_{xy} 就被移到图像中的下一个位置。然后，自适应滤波器重新初始化并应用到新位置的像素。

自适应滤波器仅使用新像素就可以反复更新 S_{xy} 内的像素灰度值中值，因而减少了计算开销。自适应中值滤波器和传统中值滤波器对图像的处理结果对比如图 5-15 所示。



(a) 原图像　(b) 含椒盐噪声图像

(c) 传统中值滤波器结果图像　(d) 自适应中值滤波器结果图像

图 5-15　自适应中值滤波器与传统中值滤波器对图像的处理结果对比

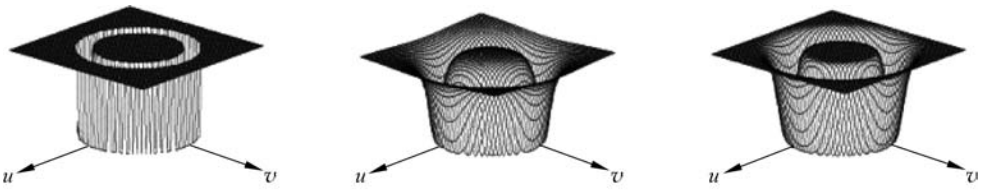
图 5-15(b)为被概率为 $P_a=P_b=0.25$ 的椒盐噪声污染的图像，图中的噪声污染非常严重，以致模糊了图像的大部分细节。作为比较，图 5-15(b)首先使用最小中值滤波器进行滤波，消除大部分可见的椒盐噪声。该滤波器的大小为 7×7，其处理结果如图 5-15(c)所示。可以看出，虽然图 5-15(c)的噪声被有效地消除了，但图像细节受到了损失。自适应滤波器处理噪声的结果图像如图 5-15(d)所示。

5.5 频域滤波消除周期噪声

频域滤波技术可以有效地分析并去除周期噪声，其基本原理是在傅里叶变换中，周期噪声在对应于周期干扰的频率处，以集中的能量脉冲形式出现，通过选择性滤波器来分离噪声。周期噪声的表现类似于冲激响应脉冲，这在傅里叶频谱中很常见，其去除噪声的主要滤波器是陷波带阻滤波器。

5.5.1 带阻滤波器

带阻滤波器能够阻止一定频率范围内的信号通过，允许其他频率范围内的信号通过，消除或衰减傅里叶变换原点处的频段。带阻滤波器一般有 3 种：理想带阻滤波器、巴特沃斯带阻滤波器和高斯带阻滤波器。图 5-16 为 3 种滤波器的透视图。



(a) 理想带阻滤波器　(b) 巴特沃斯带阻滤波器　(c) 高斯带阻滤波器

图 5-16　3 种滤波器的透视图

(1) 理想带阻滤波器的表达式如式(5-20)所示。

$$H(u,v)=\begin{cases}1, & D(u,v)<D_0-\dfrac{W}{2}\\ 0, & D_0-\dfrac{W}{2}\leqslant D(u,v)\leqslant D_0\\ 1, & D(u,v)>D_0+\dfrac{W}{2}\end{cases}\qquad(5\text{-}20)$$

$$H(u,v)=\left[\left(u-\frac{M}{2}\right)^2+\left(v-\frac{N}{2}\right)^2\right]^{\frac{1}{2}}$$

其中，W 是频带的宽度，D_0 是频带的中心半径。

(2) n 阶巴特沃思带阻滤波器的表达式如式(5-21)所示。

$$H(u,v)=\frac{1}{1+\left[\dfrac{D(u,v)W}{D^2(u,v)-D_0^2}\right]^{2n}}\qquad(5\text{-}21)$$

(3) 高斯带阻滤波器的表达式如式(5-22)所示。

$$H(u,v)=1-\mathrm{e}^{-\frac{1}{2}\left[\frac{D^2(u,v)-D_0^2}{D(u,v)W}\right]^2} \tag{5-22}$$

带阻滤波器的主要应用之一是在频域噪声分量的一般位置近似已知的应用中消除噪声。一个典型的例子就是一幅被加性周期噪声污染的图像,其噪声可近似为二维正弦函数,如图 5-17 所示。不难看出,一个正弦波的傅里叶变换由两个脉冲组成,它们是关于变换域坐标原点互为镜像。

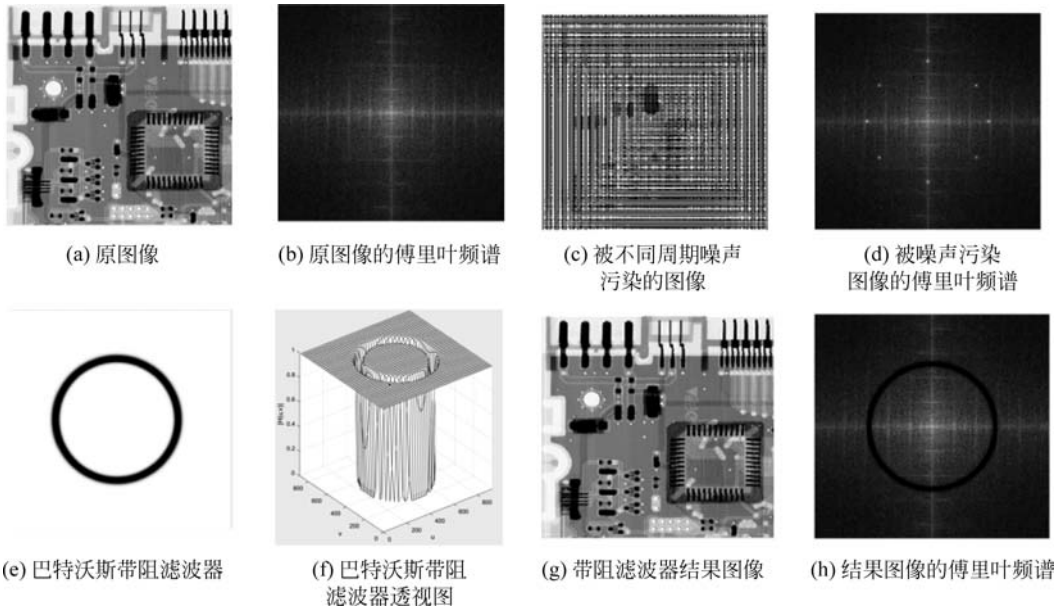


图 5-17　带阻滤波器的使用

由图 5-17(c)可以看出,噪声分量很容易被看成图 5-17(d)显示的傅里叶频谱中对称的亮点对。在本例中,噪声分量位于关于变换原点的近似圆上,因此使用圆对称带阻滤波器进行去噪是一个正确的选择。图 5-17(e)和图 5-17(f)为巴特沃斯带阻滤波器,它设置了适当的半径和宽度,完全包围了噪声脉冲。由于在变换中要尽可能小地损失细节,带阻滤波器通常为尖锐的窄带滤波器。图 5-17(g)和图 5-17(h)为带阻滤波器去噪后的结果图像及其傅里叶频谱。可以看出,图 5-17(c)所示的图像质量的改进是非常明显的,即使细小的纹理也被有效地修复了。

5.5.2　带通滤波器

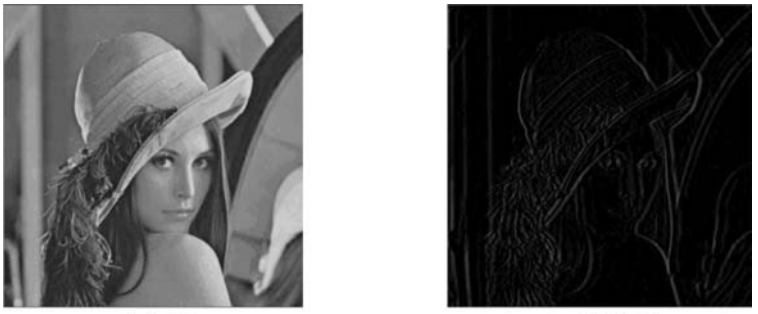
带通滤波器能够允许一定频率范围内的信号通过,阻止其他频率范围内的信号通过。其表达式如式(5-23)所示。

$$H_{bp}(u,v)=1-H_{br}(u,v) \tag{5-23}$$

其中,$H_{bp}(u,v)$表示带通滤波器,$H_{br}(u,v)$表示相应的带阻滤波器。

一幅图像通常不会直接进行带通滤波,因为这样会消除太多的图像细节。但是,带通滤波器在一幅图像中屏蔽选中频段导致的效果时非常有用(帮助提取了噪声模式,简化了噪声

分析，而且与图像内容无关）。如图 5-18 为使用带通滤波器提取原图像中衣服边缘的示例。

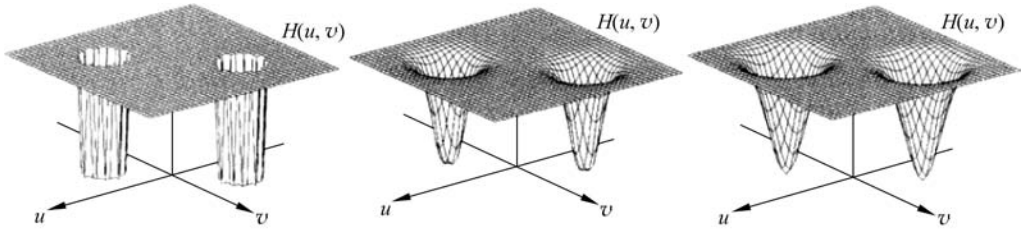


(a) 原图像　　(b) 结果图像

图 5-18　使用带通滤波器提取边缘

5.5.3　陷波滤波器

陷波滤波器阻止（或通过）事先定义的中心频率邻域内的频率。图 5-19 分别显示了理想陷波滤波器、巴特沃斯陷波滤波器和高斯陷波滤波器的透视图。由于傅里叶变换的对称性，要获得有效的结果，陷波滤波器必须以关于原点对称的形式出现。这个原则的特例是：如果陷波滤波器位于原点处，那么在这种情况下，陷波滤波器的对称是其本身。为了便于说明，本章只列举了一对陷波滤波器，但是，陷波滤波器可实现的对数是任意的，陷波区域的形状也可以是任意的（如矩形）。



(a) 理想陷波滤波器　(b) 巴特沃斯陷波滤波器(2阶)　(c) 高斯陷波滤波器

图 5-19　3 种陷波滤波器透视图

（1）理想陷波滤波器，其表达式如式（5-24）所示。

$$\begin{cases} H_{bp}(u,v)=\begin{cases}0, & D_1(u,v)\leqslant D_0 \text{ 或 } D_2(u,v)\leqslant D_0\\ 1, & \text{其他}\end{cases} \\ D_1(u,v)=\left[\left(u-\dfrac{M}{2}-u_0\right)^2+\left(v-\dfrac{M}{2}-v_0\right)^2\right]^{\frac{1}{2}} \\ D_2(u,v)=\left[\left(u-\dfrac{M}{2}+u_0\right)^2+\left(v-\dfrac{M}{2}+v_0\right)^2\right]^{\frac{1}{2}} \end{cases} \tag{5-24}$$

其中，中心在(u_0,v_0)且与$(-u_0,-v_0)$对称。

（2）巴特沃斯陷波滤波器，其表达式如式（5-25）所示。

$$H(u,v)=\frac{1}{1+\left[\dfrac{D_0^2}{D_1(u,v)D_2(u,v)}\right]^n} \tag{5-25}$$

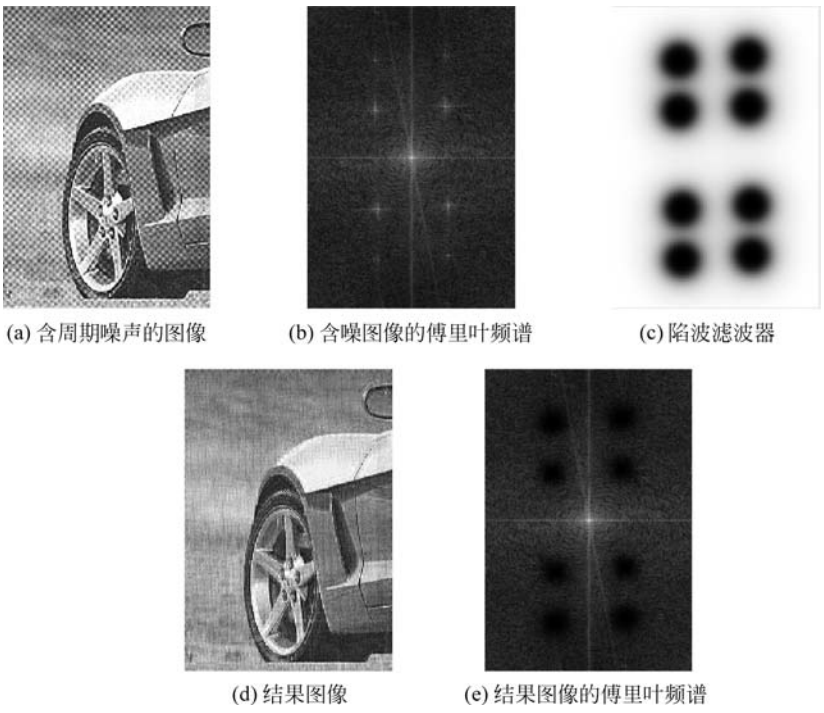
(3) 高斯陷波滤波器,其表达式如式(5-26)所示。

$$H(u,v)=1-\mathrm{e}^{-\frac{1}{2}\left[\frac{D_1(u,v)D_2(u,v)}{D_0^2}\right]} \tag{5-26}$$

当 $u_0=v_0=0$ 时,上述 3 个滤波器变为高通滤波器。

陷波滤波器也可以用于去除周期噪声。虽然带阻滤波器也可以用于去除周期噪声,但是也会使噪声以外的成分衰减。而陷波滤波器主要对某个点进行衰减,对其余的成分则无影响。

陷波滤波器对图像的处理结果如图 5-20 所示。



(a) 含周期噪声的图像　(b) 含噪图像的傅里叶频谱　(c) 陷波滤波器

(d) 结果图像　(e) 结果图像的傅里叶频谱

图 5-20　陷波滤波器对图像的处理

5.6　两个退化函数模型

假设有一幅退化图像,但没有关于退化函数 H 的任何知识。基于图像被线性、位置不变的过程退化的假设,一种估计 H 的方法就是从图像本身收集信息。例如,如果图像已被模糊,那么可以观察图像中包含样本结构的一个小矩形区域,如某一物体和背景的一部分。

空域内的卷积即频域内的乘积,如式(5-27)所示。那么,频域内的逆滤波运算其实就是除法。

$$G(u,v)=H(u,v)F(u,v)+N(u,v) \tag{5-27}$$

由式(5-27)可知,该表达式没有卷积运算,是很简单的四则运算。那么,所谓的去卷积或者逆滤波,就是去除退化函数 $H(u,v)$的过程。因此,式(5-27)直接做除法运算就可以了,如式(5-28)所示。

$$\hat{F}(u,v)=\frac{G(u,v)}{H(u,v)}=F(u,v)+\frac{N(u,v)}{H(u,v)} \tag{5-28}$$

对于式(5-28)而言,在值为 0 或非常小的点上,$F(u,v)$将变得无穷大或非常大。而 $H(u,v)$有许多零点,即奇异条件下求解。当 $H(u,v)\neq 0$ 但非常小时,$N(u,v)/H(u,v)$将变得很大,从而影响恢复的数据结果,即所谓的病态条件。

上述问题的解决要点是限制滤波频率,使其接近原点。

综上所述,逆滤波要解决的问题有两个:

(1) 退化函数 $H(u,v)$的推测;

(2) 尽可能地降低噪声项 $N(u,v)$对图像质量的影响。

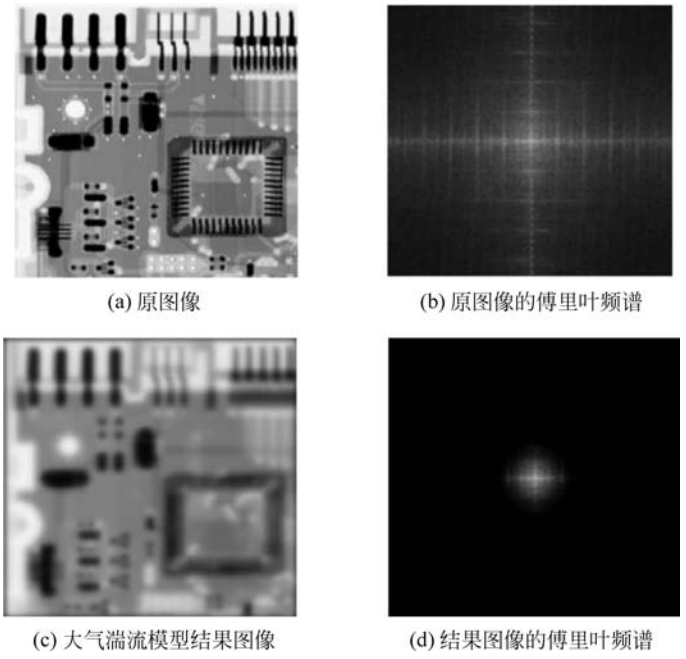
5.6.1 大气湍流模型

大气湍流模型的表达式如式(5-29)所示。

$$H(u,v)=\mathrm{e}^{-k(u^2+v^2)^{\frac{5}{6}}} \tag{5-29}$$

由(式 5-29)可以看出,大气湍流模型没有 0 值。与高斯低通滤波器相似,阻带的值都极小。随着 k 越来越大,大气湍流模型得到的图像越来越模糊,这会使图像的直接逆滤波失败。

大气湍流模型对图像的处理结果如图 5-21 所示。



(a) 原图像　(b) 原图像的傅里叶频谱

(c) 大气湍流模型结果图像　(d) 结果图像的傅里叶频谱

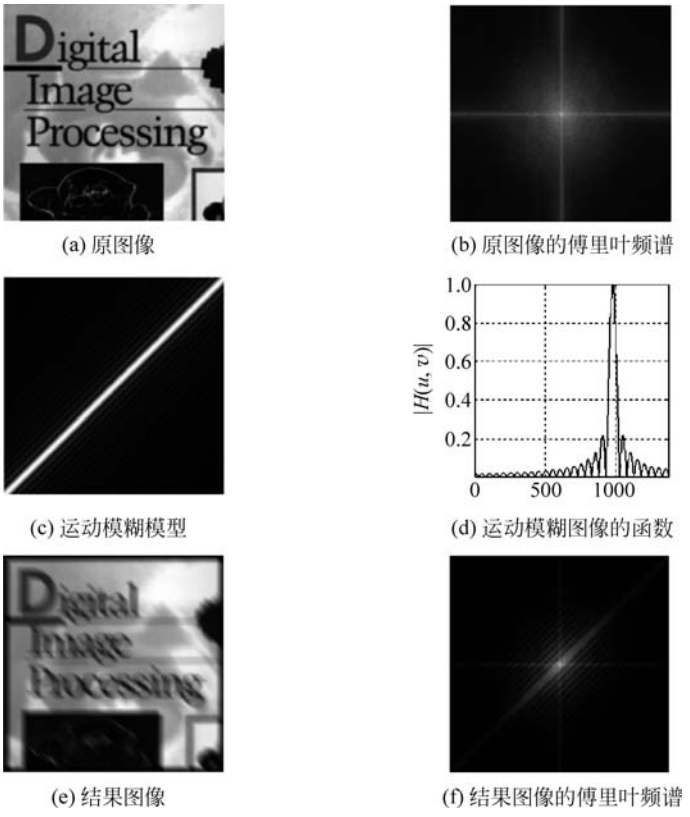
图 5-21　大气湍流模型对图像的处理结果

5.6.2 运动模糊模型

$$H(u,v)=\frac{T\sin[\pi(ua+vb)]}{\pi(ua+vb)}e^{-j\pi(ua+vb)} \tag{5-30}$$

其中，T 表示曝光时间，a 与 b 分别表示水平移动量与垂直移动量。

运动模糊图像的尺寸会变化。这时，如果截取操作还是按照原图像进行，会造成图像成分的损失，使复原图像时效果不太好。此外，导致复原效果不好的原因不好确定，到底是因为成分的缺失，还是因为噪声的干扰。因此，运动模糊模型适当地扩展了图像的尺寸，以保留图像的所有成分，其对图像的处理结果如图 5-22 所示。



(a) 原图像　(b) 原图像的傅里叶频谱

(c) 运动模糊模型　(d) 运动模糊图像的函数

(e) 结果图像　(f) 结果图像的傅里叶频谱

图 5-22　运动模糊模型对图像的处理结果

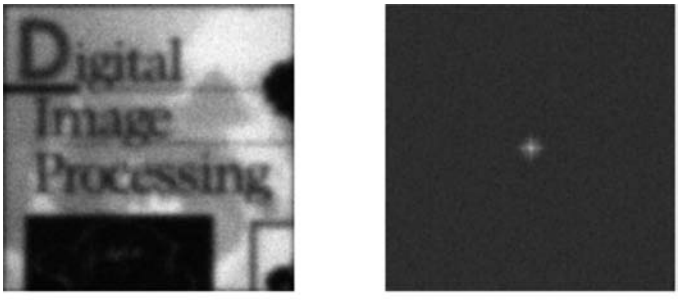
5.7 逆滤波

逆滤波是研究退化函数 H 以进行图像复原的方法。最简单的复原方法是对图像直接进行逆滤波，即用退化函数除退化图像的傅里叶变换 $G(u,v)$，获得原图像傅里叶变换的估计值 $\hat{F}(x,y)$，即

$$\hat{F}(u,v)=\frac{G(u,v)}{H(u,v)} \tag{5-31}$$

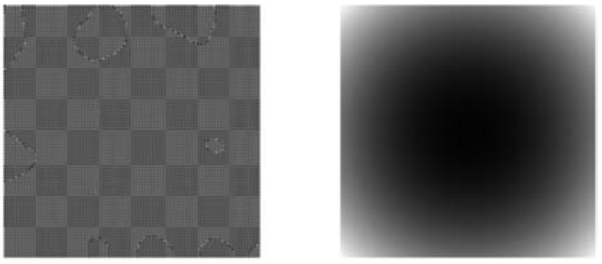
将式(5-2)代入式(5-31),可以得到式(5-28)。由于式(5-28)的 $N(u,v)$未知,即使知道退化函数 $H(u,v)$,也不能准确地复原未退化的图像。如果退化函数 $H(u,v)$是 0 或非常小的值,则 $N(u,v)/H(u,v)$很容易找到估计值 $\hat{F}(u,v)$。解决退化函数 $H(u,v)$为 0 或为非常小的值的一种方法是限制滤波的频率,使其接近原点。在频域中,由于 $H(0,0)$通常是 $H(u,v)$的最高值,因此,通过将频率限制在原点附近分析,能够降低退化函数 $H(u,v)$值为 0 的概率。

逆滤波对图像的处理实验如图 5-23 和图 5-24 所示。实验使用退化函数 $H(u,v)$来处理原图像,再加上均值为 0、方差为 0.08 的高斯噪声 $N(u,v)$,并以该含噪图像进行逆滤波实验。退化函数 $H(u,v)$则选用先前叙述的两种:①大气湍流模型;②运动模糊。所谓直接逆滤波,就是不考虑噪声对图像的影响,直接进行逆滤波。以大气湍流模型为例,对于大气湍流模型而言,直接逆滤波会得到很不理想的结果,通过大气湍流模型得到的模糊图像及其傅里叶频谱如图 5-23 所示。



(a) 模糊图像　　(b) 模糊图像的傅里叶频谱

图 5-23　大气湍流模型得到的处理图像

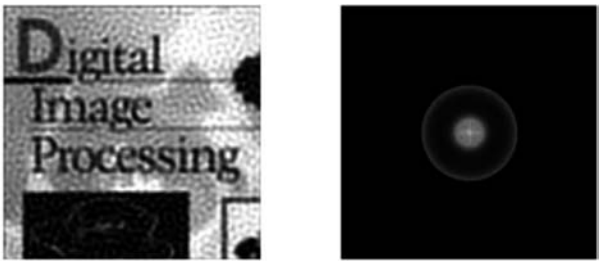


(a) 模糊图像直接逆滤波结果图像　　(b) 结果图像的傅里叶频谱

图 5-24　对大气湍流模型模糊图像的直接逆滤波处理结果

由图 5-24 可知,直接逆滤波处理结果没有任何价值。观察其频谱发现,频谱的 4 个角很亮,而原频谱最亮的直流分量却看不到了。因此,这里做一个限制处理,也就是说,仅仅处理靠近直流分量的部分,其他的则不做处理。然后处理结果通过一个 10 阶巴特沃斯低通滤波器进行滤波,得到如图 5-25 所示的结果。

由上可知,只需要调整限制半径,便可以得到一个较好的结果。但是对于运动模糊的图像来说,这种方法的处理结果并不理想。



(a) 只处理靠近直流分量部分的结果图像　(b) 结果图像的傅里叶频谱

图 5-25　进行限制处理后得到的结果

5.8　维纳滤波器

维纳滤波由 Wiener 首先提出，并应用于一维信号，取得了很好的效果。后来维纳滤波又被引入二维信号处理中，也取得相当满意的效果。尤其是在图像复原领域，由于维纳滤波器的复原效果好，计算量较低，并且抗噪性能优，因而得到了广泛应用。逆滤波并没有清楚地说明如何处理噪声，因而本节将讨论一种综合了退化函数和噪声统计特征的图像复原处理方法。该方法建立在图像和噪声都是随机变量的基础上，其目标是找到未污染图像 f 的一个估计子 $\hat{f}$，使它们之间的均方误差最小。这种误差度量如式(5-32)所示。

$$e^2 = E\{(f-\hat{f})^2\} \tag{5-32}$$

许多高效的图像复原算法都是以维纳滤波为基础。维纳滤波的表达式如式(5-33)所示。

$$\hat{F}(u,v) = \left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2+K}\right]G(u,v) \tag{5-33}$$

其中，K 为常数。对于合适的常数 K，有如下结论。

(1) 对于退化函数很小的点，相对而言常数 K 的值很大，其倒数 $\frac{1}{H(u,v)}$ 不会太大。

(2) 对于退化函数很小的点，相对而言常数 K 的值很小，其倒数 $\frac{1}{H(u,v)}$ 基本保持不变。

维纳滤波对两个退化函数模型图像的处理结果如图 5-26 所示。

(a) 大气湍流模型图像处理结果　(b) 运动模糊模型图像处理结果

图 5-26　维纳滤波对两个退化函数模型图像模型图像的处理结果

维纳滤波是一种基于最小均方误差准则，适用于平稳过程的最优估计器。这种滤波器的输出与期望输出之间的均方误差为最小，因此，是一个最佳滤波系统，可用于提取被平稳噪声所污染的信号。但维纳滤波的使用前提是知道信号和噪声的功率谱，但这在实际应用中较难得到，因而只能根据先验知识进行估计。

5.9 约束最小二乘法滤波

对于维纳滤波来说，未退化图像和噪声的功率谱必须是已知的。然而，功率谱比的常数估计并不总是合适的值。

本节讨论的约束最小二乘法滤波仅要求噪声方差和均值的相关知识。这些参数通常可从一幅给定的退化图像得到，因而这是一个很重要的优点。另外，维纳滤波建立在最小化统计准则的基础之上，因此在平均意义上是最优的。约束最小二乘法滤波对其所应用的每一幅图像都能产生最优结果，而这些理论上满足的最优准则与动态的视觉感知并没有关系，因此算法的选择往往由结果图像的视觉感知质量决定。约束最小二乘法滤波将图像的能量作为评价图像平滑程度的度量，并尽可能地使其平滑。设噪声的能量是一个定值，使用拉普拉斯未定系数法，将其进行迭代，然后求解。

根据卷积的定义，式(5-1)可以写成如式(5-34)所示的向量-矩阵的形式。

$$\boldsymbol{g}=\boldsymbol{H}\boldsymbol{f}+\boldsymbol{\eta} \tag{5-34}$$

假设 $\boldsymbol{g}$ 的大小为 $M\times N$ 维，结果向量的大小将为 $MN\times 1$ 维。$\boldsymbol{f}$ 和 $\boldsymbol{\eta}$ 的维数也是 $MN\times 1$ 维，那么，矩阵 $\boldsymbol{H}$ 的大小为 $MN\times MN$ 维。约束最小二乘法滤波期望找到标准函数 C 的最小值，C 定义如式(5-35)所示。

$$C=\sum_{x=0}^{M-1}\sum_{y=0}^{N-1}[\nabla^2 f(x,y)]^2 \tag{5-35}$$

其约束为

$$\|\boldsymbol{g}-\boldsymbol{H}\hat{\boldsymbol{f}}\|^2=\|\boldsymbol{\eta}\|^2 \tag{5-36}$$

使用约束最小二乘法滤波的目的是减少噪声对于逆滤波的影响。约束最小二乘法滤波最优化问题在频域的解决表达式为

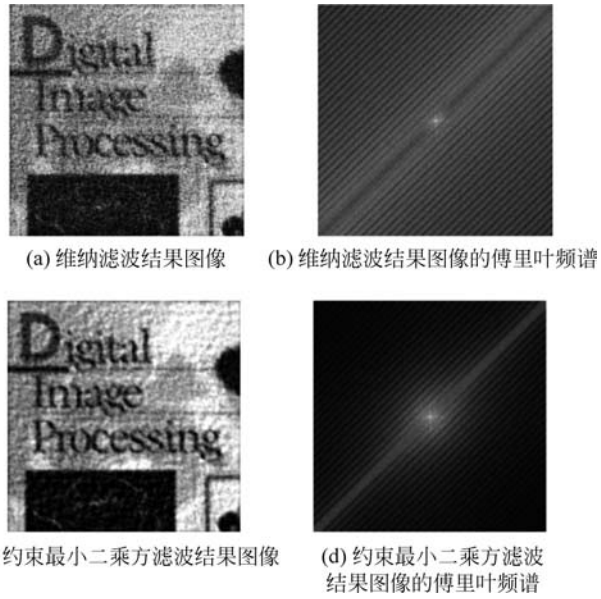
$$\hat{F}(u,v)=\left[\frac{1}{H(u,v)}\frac{|H(u,v)|^2}{|H(u,v)|^2+\gamma|P(u,v)|^2}\right]G(u,v) \tag{5-37}$$

其中，γ 表示参数，满足式(5-36)，若不满足，进行调整以满足；$P(u,v)$表示式(5-38)的傅里叶变换。

$$P(x,y)=\begin{bmatrix}0&-1&0\\-1&4&-1\\0&-1&0\end{bmatrix} \tag{5-38}$$

约束最小二乘法滤波可以消除很严重的噪声，得到复原图像。将上一节实验图像的噪声的方差提高为 0.2，再使用约束最小二乘法滤波进行滤波，结果如图 5-27 所示。

图 5-27(c)和图 5-27(d)为使用约束最小二乘法滤波对实验图像的处理结果，其中，为了产生最好的视觉效果，γ 的值是手工选择的。比较约束最小二乘法滤波和维纳滤波的处理结果可以发现，前者对于高等噪声和中等噪声的情况的处理结果要稍好一些。当手工选择



(a) 维纳滤波结果图像　(b) 维纳滤波结果图像的傅里叶频谱

(c) 约束最小二乘方滤波结果图像　(d) 约束最小二乘方滤波结果图像的傅里叶频谱

图 5-27　不同滤波器处理结果比较

参数 γ，以取得更好的视觉效果时，约束最小二乘法滤波的处理结果可能比维纳滤波的结果更好。式(5-37)的参数 γ 是一个标量；而式(5-33)的参数 K 是两个未知频域函数的近似比，其值很少是常数，因此，手工选择 γ 将得到未退化图像的更准确估计。

5.10　本章小结

本章中的复原结果基于这样一个假设：图像退化可建模为一个线性的、位置不变的过程，以及与图像灰度值不相关的加性噪声。本章推导的某些复原技术基于各种最佳准则。“最佳”一词的使用涉及严格的数学概念，而不是指人类视觉感知系统的最佳响应。事实上，视觉感知知识的缺乏妨碍了图像复原问题的一般表达，而图像复原问题要考虑观察者的偏好与能力。在这些限制条件下，本章所介绍图像复原相关概念的优势是基本方法的推导，使读者具有合理的预测能力和坚实的基础知识。一定的复原任务，譬如降低随机噪声，是在空域中使用卷积模板来执行的。频域对于降低周期性噪声和某些重要的退化建模是很理想的，如在图像获取期间因为运动导致的模糊。对于表达复原滤波器而言，频域也是很有用的工具，如维纳滤波器和约束最小二乘法滤波器。频域为实验提供了直观且可靠的基础。一旦对于给定的应用找到了令人满意的一种方法(滤波器)，其实现通常是通过数字滤波器的设计在频域近似解决来做到的。

课后作业

(1) 人为地在图 5-28(或自行采集的图像)中加入均值为 0、方差为 800 的高斯噪声，再添加 $P_a = P_b = 0.25$ 的脉冲噪声，并显示添加噪声后的图像。然后，分别利用以下几种方

法进行图像复原，人为调整参数（窗口大小等），并比较不同滤波器的处理效果，以及解释其原因。

① 算术均值滤波器。

② 逆谐波均值滤波器。

③ 中值滤波器。

④ 最大值和最小值滤波器。

⑤ 修正的阿尔法均值滤波器。

⑥（可选）自适应中值滤波器。

（2）人为地在图 5-28 中加入正弦周期噪声，并显示添加噪声后的图像及其傅里叶频谱。然后分别利用①～③这 3 种方法进行图像复原，显示处理后的图像及其傅里叶频谱。手动调整参数（阶数等）并比较处理结果。

图 5-28 待处理的图像

① 带阻滤波器。

② 带通滤波器。

③ 陷波滤波器。

④（可选）最佳陷波滤波器可以很好地处理一个及以上的干扰分量或者多个周期性的噪声。相比于其他的滤波器，最佳陷波滤波可以最小化复原估计值 $\hat{f}(x,y)$ 的局部方差。

最佳陷波滤波可以分为两步：屏蔽干扰分量的主要成分；从被污染的图像中减去该模式一个可变的加权部分。具体如下。

首先提取干扰分量中的主频率分量。空域中噪声的相对模式可由式(5-39)获得。

$$N(u,v)=H_{\mathrm{NP}}(u,v)G(u,v) \tag{5-39}$$

$$\eta(x,y)=F^{-1}\{H_{\mathrm{NP}}(u,v)G(u,v)\} \tag{5-40}$$

$H_{\mathrm{NP}}(u,v)$ 的形式需要进行多方面判断，以确定哪些是干扰分量的主要成分。

然后，从被污染的图像中减去该模式的一个可变的加权部分，其表达式为

$$\hat{f}(x,y)=g(x,y)-w(x,y)\eta(x,y) \tag{5-41}$$

其中，$w(x,y)$ 为最佳陷波滤波的权重函数，由式(5-42)得到。

图 5-29 “水手 6 号”飞船拍摄的火星地形图像

$$w(x,y)=\frac{\overline{g(x,y)\eta(x,y)}-\bar{g}(x,y)\bar{\eta}(x,y)}{\overline{\eta^2}(x,y)-\bar{\eta}^2(x,y)} \tag{5-42}$$

最佳陷波滤波器的实现步骤为：①读取图像；②进行快速傅里叶变换；③将频域图像居中显示；④通过观察频域图像，确定需要去除的频率成分，制作最佳陷波带通滤波器；⑤对滤波后图像进行傅里叶反变换，在空域显示噪声干扰模式；⑥受污染图像减去噪声干扰模式的加权部分，得到结果图像，并保存。

图 5-29 为“水手 6 号”飞船拍摄的火星地形图像，请编写代码显示该图像的傅里叶频谱，再使用最佳陷波

滤波器，选择 $a=b=15$ 的邻域进行处理，获得 $N(u,v)$ 傅里叶频谱和相应的噪声干扰模式 $\eta(x,y)$，最后显示处理后的图像。

(3) 盲解卷积复原是利用模糊图像，以某种方式提取退化信息，获得清晰图像的一种图像复原方法。盲解卷积复原有一个很好的优点：在对失真(噪声和模糊)毫无先验知识的情况下，仍然能够实现对模糊图像的复原。基于露西-理查德森算法的盲解卷积算法属于图像复原中的非线性算法，与维纳滤波这种较为直接的算法不同，该算法使用非线性迭代技术，在计算量和性能方面都有了一定提升。露西-理查德森算法由贝叶斯公式推导出，因为使用了条件概率(即算法考虑了信号的固有波动)，因此具有复原噪声图像的能力。贝叶斯公式如式(5-43)所示。

$$P(x \mid y)=\frac{P(y \mid x)P(x)}{\int P(y \mid x)P(x)\mathrm{d}x} \tag{5-43}$$

结合图像退化/复原模型，可以得到迭代函数，如式(5-44)所示。

$$f_{i+1}(x)=\int \frac{g(y,x)c(y)\mathrm{d}y}{\int g(y,z)f_i(z)\mathrm{d}z}f_i(x) \tag{5-44}$$

其中，f_i 表示第 i 轮迭代复原图像，对应贝叶斯公式的 $P(x)$；g 表示退化函数，对应贝叶斯公式的 $p(y|x)$；c 表示退化图像($c(y)\mathrm{d}y$ 意为在退化图像上积分)。如果满足图像各区域的模糊函数相同的条件，则式(5-44)所示的迭代函数可简化为

$$f_{i+1}(x)=\left\{\left[\frac{c(x)}{f_i(x)\otimes g(x)}\right]\otimes g(-x)\right\}f_i(x) \tag{5-45}$$

这就是露西-理查德森迭代公式，其中 c 表示退化图像，g 表示退化函数，f 表示第 k 轮复原图像。如果系统的退化函数 g 已知，只要有一个初始估计 f 就可以进行迭代求解了。在开始迭代后，由于算法的形式，估计值与真实值的差距会迅速减小，从而后续迭代过程 f 的更新速度会逐渐变慢，直至收敛。露西-理查德森算法的另一优点就是初始值 $f>0$，后续迭代值均保持非负性，并且能量不会发散。

在第 k 轮迭代，假设原图像已知，即 $k-1$ 轮得到的 f_{k-1}，通过露西-理查德森公式求解 g_k；然后，用 g_k 求解 f_k，如此反复迭代，最后求得 f 和 g。因此，求解最初需要同时假设一个复原图像 f_0 和一个退化函数 g_0。迭代式为

$$g_{i+1}^{k}(x)=\left\{\left[\frac{c(x)}{g_i^k(x)\otimes f^{k-1}(x)}\right]\otimes f^{k-1}(-x)\right\}g_i^k(x) \tag{5-46}$$

$$f_{i+1}^{k}(x)=\left\{\left[\frac{c(x)}{f_i^k(x)\otimes g^{k}(x)}\right]\otimes g^{k}(-x)\right\}f_i^k(x) \tag{5-47}$$

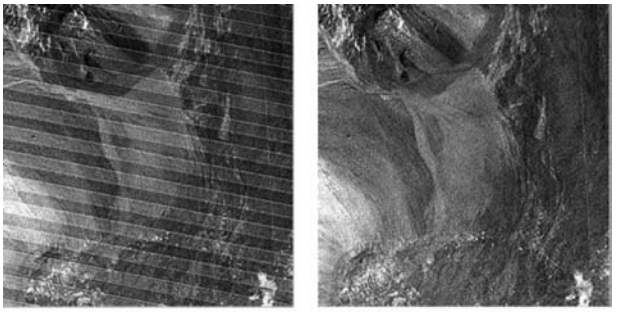
将一幅原图像进行模糊处理(模拟大气湍流)，分别使用维纳滤波和盲解卷积复原、进行图像复原，并对比结果图像。

(4) 随着经济的发展，人们生活水平得到提高，视频监控在日常生活中的应用越来越普遍。尤其是交通管理中用到了大量监控视频，如超速车辆抓拍。但有时由于车辆的运动会导致照片中产生运动模糊而使细节不够清晰。请采用本章图像复原方法，对产生运动模糊的图像进行复原，以便能看清车辆细节(如车牌号码)，并使用 MATLAB 完成该实验。

(5) 如今遥感技术变得越来越重要。但由于传感器的因素,一些获取的遥感图像会出现周期性的噪声,因此必须对其进行消除或减弱,以进行使用。

① 去除周期性噪声和尖锐性噪声。周期性噪声一般重叠在图像上,周期性地干扰图形。这些噪声具有不同的幅度、频率、和相位,形成一系列的尖峰或者亮斑,代表在某些空间频率位置最为突出。一般可以用带通或者槽形滤波的方法来消除。消除尖峰噪声,特别是与扫描方向不平行的,一般用傅里叶变换进行滤波处理的方法比较方便。

② 去除坏线和条带。遥感图像中通常会出现与扫描方向平行的条带,还有一些与辐射信号无关的条带噪声,一般称为坏线。这种情况一般采用傅里叶变换和低通滤波进行消除或减弱(如图 5-30 为坏线消除前后对比图)。



(a) 含条纹图像　　(b) 去条纹后图像

图 5-30　去除坏线图像对比

③ 薄云处理。由于天气原因,有些遥感图像会出现薄云,因此对其可以进行减弱处理。

④ 阴影处理。由于太阳高度角的原因,有些图像会出现山体阴影,可以采用比值法对阴影进行消除。

附录 MATLAB 实现代码

(1) 算术均值滤波器和几何均值滤波器处理图像的 MATLAB 实现代码。

```
clear all; close all; clc;
I = imread('biaoqingb.jpg');
I = rgb2gray(I);
I = im2double(I);
J = imnoise(I,'gaussian',0,0.00615);
PSF = fspecial('average',3);                    %生成滤波器
K1 = imfilter(J,PSF);                           %算术均值滤波器
K2 = exp(imfilter(log(J),PSF));                 %几何均值滤波器
figure;
subplot(2,2,1); imshow(I);
subplot(2,2,2); imshow(J);
subplot(2,2,3); imshow(K1);
subplot(2,2,4); imshow(K2);
```

(2) 逆谐波均值滤波器处理图像的 MATLAB 实现代码。

```
close all; clear all; clc;
I = imread('641.png');
I = rgb2gray(I);
I = im2double(I);
R = rand(size(I));
J1 = I;
J1(R<=0.02) = 0;                %添加胡椒噪声
J2 = I;
J2(R<=0.02) = 1;                %添加盐粒噪声
PSF = fspecial('average',3);
Q1 = 1.5;
Q2 = -1.5;
k1 = imfilter(J1.^(Q1+1),PSF);
k2 = imfilter(J1.^Q1,PSF);
K = k1./k2;                     %Q=1.5的逆谐波均值滤波
m1 = imfilter(J2.^(Q2+1),PSF);
m2 = imfilter(J2.^Q2,PSF);
M = m1./m2;                     %Q=-1.5的逆谐波均值滤波
figure;
```

(3) 中值滤波器处理图像的 MATLAB 实现代码。

```
close all; clear all; clc;
I = imread('board.jpg');
I = rgb2gray(I);
I = im2double(I);
```

```
J = imnoise(I,'salt & pepper',0.1);
K1 = medfilt2(J,[3,3]);
K2 = medfilt2(K1,[3,3]);
K3 = medfilt2(K2,[3,3]);

figure;
subplot(2,3,1); imshow(J);
subplot(2,3,2); imshow(K1);
subplot(2,3,3); imshow(I);
subplot(2,3,4); imshow(K2);
subplot(2,3,5); imshow(K3);
```

（4）最大值滤波器和最小值滤波器处理图像的 MATLAB 实现代码。

```
close all; clear all; clc;
I = imread('duodushu.jpg');
I = rgb2gray(I);
I = im2double(I);
R = rand(size(I));
J1 = I;
J1(R<=0.02) = 0;          %添加胡椒噪声
J2 = I;
J2(R<=0.02) = 1;          %添加盐粒噪声
K1 = ordfilt2(J1,9,ones(3,3));
K2 = ordfilt2(J2,1,ones(3,3));

figure;
subplot(2,3,1); imshow(J1);
subplot(2,3,2); imshow(J2);
subplot(2,3,3); imshow(I);
subplot(2,3,4); imshow(K1);
subplot(2,3,5); imshow(K2);
```

（5）修正的阿尔法均值滤波器处理图像的 MATLAB 实现代码。

```
close all; clear all; clc;
I = imread('fruits.jpg');
I = rgb2gray(I);
I = im2double(I);
J1 = imnoise(I,'gaussian',0,0.01);
J2 = imnoise(J1,'salt & pepper',0.1);
PSF = fspecial('average',3);                    %生成滤波器
K1 = imfilter(J2,PSF);                          %算术均值滤波器
K2 = exp(imfilter(log(J2),PSF));                %几何均值滤波器
K3 = medfilt2(J2,[5,5]);
[MM,NN] = size(J2);

%定义子窗口的尺寸
m = 5;
n = 5;
```

```
%确定要扩展的行列数
len_m = floor(m/2);
len_n = floor(n/2);
%将原图像进行扩展,此处采用镜像扩展,以计算图像边缘
I_D_pad = padarray(J2,[len_m,len_n],'symmetric');
%获得扩展后的图像尺寸
[M,N] = size(I_D_pad);
d = 5;
K4 = zeros(MM,NN);
%逐点计算子窗口的谐波平均
for i = 1 + len_m:M - len_m
    for j = 1 + len_n:N - len_n
        %从子窗口中取出子图像
        Block = I_D_pad(i - len_m:i + len_m,j - len_n:j + len_n);
        Block = sort(Block(:));
        Block = Block(floor(d/2 + 1):floor(m * n - 4));
        %计算矩阵的阿尔法均值
        K4(i - len_m,j - len_n) = sum(sum(Block))/(m * n - d);
    end
end

figure;
subplot(2,4,1); imshow(I);
subplot(2,4,2); imshow(J1);
subplot(2,4,3); imshow(J2);
subplot(2,4,5); imshow(K1);
subplot(2,4,6); imshow(K2);
subplot(2,4,7); imshow(K3);
subplot(2,4,8); imshow(K4);
```

(6) 自适应中值滤波器处理图像的 MATLAB 实现代码。

```
close all; clear all; clc;
f = imread('20210208_175419.jpg');
image_gray = rgb2gray(f);
ff = image_gray;
ff(:) = 0;
alreadyProcessed = false(size(image_gray));        %生成逻辑非矩阵
%迭代
Smax = 7;
for k = 3:2:Smax
   zmin = ordfilt2(image_gray,1,ones(k,k),'symmetric');
   zmax = ordfilt2(image_gray,k * k,ones(k,k),'symmetric');
   zmed = medfilt2(image_gray,[k k],'symmetric');

   processUsingLevelB = (zmed > zmin) & (zmax > zmed) & ~alreadyProcessed;
   zB = (image_gray > zmin) & (zmax > image_gray);
   outputZxy = processUsingLevelB & zB;
   outputZmed = processUsingLevelB & ~zB;
   ff(outputZxy) = image_gray(outputZxy);
```

```
    ff(outputZmed) = zmed(outputZmed);

    alreadyProcessed = alreadyProcessed | processUsingLevelB;
    if all(alreadyProcessed(:))
        break;
    end
end
ff(~alreadyProcessed) = zmed(~alreadyProcessed);
f1 = imnoise(image_gray,'salt & pepper',0.25);      % 添加椒盐噪声
f2 = medfilt2(f1,[7,7]);                            % 使用中值滤波

subplot(2,2,1); imshow(image_gray);
subplot(2,2,2); imshow(f1);
subplot(2,2,3); imshow(f2);
subplot(2,2,4); imshow(ff);
```

（7）带阻滤波器处理图像的 MATLAB 实现代码。

```
H_1 = ones(P,Q);

for x = (-P/2):1:(P/2)-1
    for y = (-Q/2):1:(Q/2)-1
        D = (x^2 + y^2)^(0.5);
        D_0 = 250;
        W = 30;
        H_1(x+(P/2)+1,y+(Q/2)+1) = 1/(1+((D*W)/((D*D) - (D_0*D_0)))^6);
    end
end

G_1 = H_1 .* G_Noise;

g_1 = real(ifft2(G_1));
g_1 = g_1(1:1:M,1:1:N);

for x = 1:1:M
    for y = 1:1:N
        g_1(x,y) = g_1(x,y) * (-1)^(x+y);
    end
end
```

（8）陷波带阻滤波器处理图像的 MATLAB 实现代码。

```
f = imread('Fig0507(a)(ckt-board-orig).tif');
f = mat2gray(f,[0 255]);

[M,N] = size(f);
P = 2*M;
Q = 2*N;
fc = zeros(M,N);
```

```
for x = 1:1:M
    for y = 1:1:N
        fc(x,y) = f(x,y) * (-1)^(x+y);
    end
end

F = fft2(fc,P,Q);

H_NP = ones(P,Q);

for x = (-P/2):1:(P/2)-1
     for y = (-Q/2):1:(Q/2)-1
        D = 2;

        v_k = -200; u_k = 150;
        D_k = ((x+u_k)^2 + (y+v_k)^2)^(0.5);
        H_NP(x+(P/2)+1,y+(Q/2)+1) = H_NP(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D/D_k)^4);
        D_k = ((x-u_k)^2 + (y-v_k)^2)^(0.5);
        H_NP(x+(P/2)+1,y+(Q/2)+1) = H_NP(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D/D_k)^4);
        v_k = 200; u_k = 150;
        D_k = ((x+u_k)^2 + (y+v_k)^2)^(0.5);
        H_NP(x+(P/2)+1,y+(Q/2)+1) = H_NP(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D/D_k)^4);
        D_k = ((x-u_k)^2 + (y-v_k)^2)^(0.5);
        H_NP(x+(P/2)+1,y+(Q/2)+1) = H_NP(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D/D_k)^4);
        v_k = 0; u_k = 250;
        D_k = ((x+u_k)^2 + (y+v_k)^2)^(0.5);
        H_NP(x+(P/2)+1,y+(Q/2)+1) = H_NP(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D/D_k)^4);
        D_k = ((x-u_k)^2 + (y-v_k)^2)^(0.5);
        H_NP(x+(P/2)+1,y+(Q/2)+1) = H_NP(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D/D_k)^4);
        v_k = 250; u_k = 0;
        D_k = ((x+u_k)^2 + (y+v_k)^2)^(0.5);
        H_NP(x+(P/2)+1,y+(Q/2)+1) = H_NP(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D/D_k)^4);
        D_k = ((x-u_k)^2 + (y-v_k)^2)^(0.5);
        H_NP(x+(P/2)+1,y+(Q/2)+1) = H_NP(x+(P/2)+1,y+(Q/2)+1) * 1/(1+(D/D_k)^4);
        H_NP(x+(P/2)+1,y+(Q/2)+1) = 1 + 700*(1 - H_NP(x+(P/2)+1,y+(Q/2)+1));
     end
end

G_Noise = F .* H_NP;

g_noise = real(ifft2(G_Noise));
g_noise = g_noise(1:1:M,1:1:N);

for x = 1:1:M
    for y = 1:1:N
        g_noise(x,y) = g_noise(x,y) * (-1)^(x+y);
    end
end
```

（9）去除乘性噪声的 MATLAB 实现代码。

```
close all;
clear all;
clc;
%初始化
f = imread('original_DIP.tif');
f = mat2gray(f,[0 255]);

f_original = f;

[M,N] = size(f);

P = 2*M;
Q = 2*N;
fc = zeros(M,N);

for x = 1:1:M
    for y = 1:1:N
        fc(x,y) = f(x,y) * (-1)^(x+y);
    end
end

F_I = fft2(fc,P,Q);

figure();
subplot(1,2,1);
imshow(f,[0 1]);
xlabel('a)原图像');

subplot(1,2,2);
imshow(log(1 + abs(F_I)),[]);
xlabel('b)原图像的傅里叶频谱');
%运动模糊
H = zeros(P,Q);
a = 0.02;
b = 0.02;
T = 1;
for x = (-P/2):1:(P/2)-1
     for y = (-Q/2):1:(Q/2)-1
        R = (x*a + y*b)*pi;
        if(R = = 0)
            H(x+(P/2)+1,y+(Q/2)+1) = T;
        else H(x+(P/2)+1,y+(Q/2)+1) = (T/R)*(sin(R))*exp(-1i*R);
        end
     end
end

%大气湍流模型
H_1 = zeros(P,Q);
```

```
k = 0.0025;
for x = (-P/2):1:(P/2)-1
    for y = (-Q/2):1:(Q/2)-1
        D = (x^2 + y^2)^(5/6);
        D_0 = 60;
        H_1(x+(P/2)+1,y+(Q/2)+1) = exp(-k*D);
    end
end
%噪声
a=0;
b=0.2;
n_gaussian=a + b .* randn(M,N);
Noise=fft2(n_gaussian,P,Q);
figure();
subplot(1,2,1);
imshow(n_gaussian,[-1 1]);
xlabel('a)高斯噪声');
subplot(1,2,2);
imshow(log(1+ abs(Noise)),[]);
xlabel('b)高斯噪声的傅里叶频谱');
G = H .* F_I + Noise;
%G = H_1 .* F_I + Noise;
gc = ifft2(G);

gc = gc(1:1:M+27,1:1:N+27);
for x = 1:1:(M+27)
    for y = 1:1:(N+27)
        g(x,y) = gc(x,y) .* (-1)^(x+y);
    end
end

gc = gc(1:1:M,1:1:N);
for x = 1:1:(M)
    for y = 1:1:(N)
        g(x,y) = gc(x,y) .* (-1)^(x+y);
    end
end

figure();
subplot(1,2,1);
imshow(f,[0 1]);
xlabel('a)含噪图像');

subplot(1,2,2);
imshow(log(1+ abs(F_I)),[]);
xlabel('b)含噪图像的傅里叶频谱');

figure();
subplot(1,2,1);
imshow(abs(H),[]);
xlabel('c)运动模型 H(u,v)(a=0.02,b=0.02,T=1)');
```

```
subplot(1,2,2);
n = 1:1:P;
plot(n,abs(H(400,:)));
axis([0 P 0 1]);grid;
xlabel('H(n,400)');
ylabel('|H(u,v)|');

figure();
subplot(1,2,1);
imshow(real(g),[0 1]);
xlabel('d)结果图像');

subplot(1,2,2);
imshow(log(1 + abs(G)),[ ]);
xlabel('e)结果图像的傅里叶频谱');

%逆滤波
%F = G./H;
%F = G./H_1;

for x = (-P/2):1:(P/2)-1
     for y = (-Q/2):1:(Q/2)-1
        D = (x^2 + y^2)^(0.5);
        if (D<258)
            F(x+(P/2)+1,y+(Q/2)+1) = G(x+(P/2)+1,y+(Q/2)+1)./H_1(x+(P/2)+1,
y+(Q/2)+1);
         %不含噪声 D<188
         %噪声 D<56
        else F(x+(P/2)+1,y+(Q/2)+1) = G(x+(P/2)+1,y+(Q/2)+1);
        end
     end
end

%巴特沃斯低通滤波器
H_B = zeros(P,Q);
D_0 = 70;
for x = (-P/2):1:(P/2)-1
     for y = (-Q/2):1:(Q/2)-1
        D = (x^2 + y^2)^(0.5);
         %if(D<200) H_B(x+(P/2)+1,y+(Q/2)+1) = 1/(1+(D/D_0)^100);
        H_B(x+(P/2)+1,y+(Q/2)+1) = 1/(1+(D/D_0)^20);
     end
end

F = F.*H_B;

f = real(ifft2(F));
f = f(1:1:M,1:1:N);

for x = 1:1:M
```

```
    for y = 1:1:N
        f(x,y) = f(x,y) * (-1)^(x+y);
    end
end
%滤波结果
figure();
subplot(1,2,1);
imshow(f,[0 1]);
xlabel('a)结果图像');

subplot(1,2,2);
imshow(log(1 + abs(F)),[ ]);
xlabel('b)结果图像的傅里叶频谱');
figure();
n = 1:1:P;
plot(n,abs(F(400,:)),'r-',n,abs(F(400,:)),'b-');
axis([0 P 0 1000]);grid;
xlabel('行数(第 400 列)');
ylabel('傅里叶幅度谱');
legend('F_{limit}(u,v)','F(u,v)');
figure();
n = 1:1:P;
plot(n,abs(H(400,:)),'g-');
axis([0 P 0 1]);grid;
xlabel('H''_{s}(n,400)');
ylabel('|H''_{s}(u,v)|');

%维纳滤波
%K = 0.000014;
K = 0.02;
%H_Wiener = ((abs(H_1).^2)./((abs(H_1).^2) + K)).*(1./H_1);
H_Wiener = ((abs(H).^2)./((abs(H).^2) + K)).*(1./H);

F_Wiener = H_Wiener .* G;
f_Wiener = real(ifft2(F_Wiener));
f_Wiener = f_Wiener(1:1:M,1:1:N);

for x = 1:1:(M)
    for y = 1:1:(N)
        f_Wiener(x,y) = f_Wiener(x,y) * (-1)^(x+y);
    end
end
[SSIM_Wiener mssim] = ssim_index(f_Wiener,f_original,[0.01 0.03],ones(8),1);
SSIM_Wiener
%滤波结果
figure();
subplot(1,2,1);
%imshow(f_Wiener(1:128,1:128),[0 1]);
imshow(f_Wiener,[0 1]);
xlabel('d)维纳滤波结果图像');
```

```
subplot(1,2,2);
imshow(log(1 + abs(F_Wiener)),[ ]);
xlabel('c)维纳滤波结果图像的傅里叶频谱');
% subplot(1,2,2);
% % imshow(f(1:128,1:128),[0 1]);
% imshow(f,[0 1]);
% xlabel('e)逆滤波结果图像');
figure();
n = 1:1:P;
plot(n,abs(F(400,:)),'r - ',n,abs(F_Wiener(400,:)),'b - ');
axis([0 P 0 500]);grid;
xlabel('行数(第 400 行)');
ylabel('傅里叶幅度谱');
legend('F(u,v)','F_{Wiener}(u,v)');

figure();
subplot(1,2,1);
imshow(log(1 + abs(H_Wiener)),[]);
xlabel('a).F_{Wiener}(u,v).');

subplot(1,2,2);
n = 1:1:P;
plot(n,abs(H_Wiener(400,:)));
axis([0 P 0 80]);grid;
xlabel('行数(第 400 行)');
ylabel('傅里叶幅度谱');
% 约束最小二乘法滤波 %
p_laplacian = zeros(M,N);
Laplacian = [ 0 - 1  0;
            - 1  4 - 1;
              0 - 1  0];
p_laplacian(1:3,1:3) = Laplacian;

P = 2 * M;
Q = 2 * N;
for x = 1:1:M
    for y = 1:1:N
        p_laplacian(x,y) = p_laplacian(x,y) * ( - 1)^(x + y);
    end
end
P_laplacian = fft2(p_laplacian,P,Q);
F_C = zeros(P,Q);
r = 0.2;
H_clsf = ((H')./((abs(H).^2) + r. * P_laplacian));
F_C = H_clsf . * G;
f_c = real(ifft2(F_C));
f_c = f_c(1:1:M,1:1:N);
for x = 1:1:(M)
   for y = 1:1:(N)
       f_c(x,y) = f_c(x,y) * ( - 1)^(x + y);
    end
```

```
end

figure();
subplot(1,2,1);
imshow(f_c,[0 1]);
xlabel('e)约束最小二乘滤波器结果图像 (r = 0.2)');
subplot(1,2,2);
imshow(log(1 + abs(F_C)),[]);
xlabel('f)结果图像的傅里叶频谱');
[SSIM_CLSF mssim] = ssim_index(f_c,f_original,[0.01 0.03],ones(8),1);
figure();
subplot(1,2,1);
imshow(log(1 + abs(H_clsf)),[]);
xlabel('a).F_{clsf}(u,v).');

subplot(1,2,2);
n = 1:1:P;
plot(n,abs(H_clsf(400,:)));
axis([0 P 0 80]);grid;
xlabel('行数(第 400 行)');
ylabel('傅里叶幅度谱');
```

第 6 章

CHAPTER 6

图 像 压 缩


图像压缩

本章的结构如图 6-1 所示，具体要求如下：

（1）掌握图像压缩的基础知识。

（2）掌握无损压缩方法。

（3）掌握有损压缩方法。

（4）了解并掌握图像压缩标准。

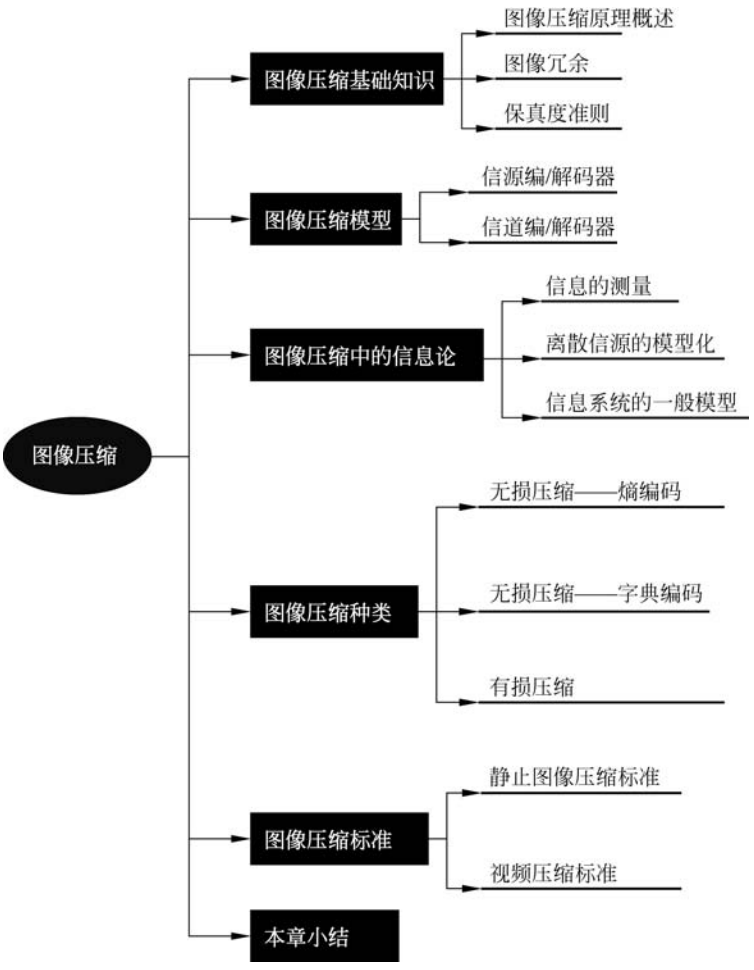


图 6-1　本章结构