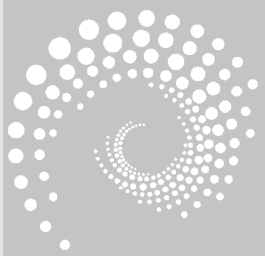
第5章

# 图像滤波运算

CHAPTER 5



# 5.1 空间滤波

## 5.1.1 空间滤波基础

在信号处理中，将信号中特定频率的波段滤除的操作称为滤波，在数字信号处理中通常采用傅里叶变换及其逆变换实现。而在数字图像处理中存在着一种操作，其和通过傅里叶变换实现的频域下的滤波是等效的，故而也称为滤波。一幅数字图像可以看成一个二维函数 $f(x,y)$, 而 $(x,y)$ 平面表明了空间位置信息，称为空间域。基于 $(x,y)$ 空间邻域的滤波操作称为空间滤波，即主要直接基于邻域（空间域）对图像中像素执行计算。

空间滤波或者邻域处理的过程包括以下四步：

（1）定义中心点 $(x,y)$；

（2）仅对预先定义的以 $(x,y)$ 为中心点的邻域内的像素进行运算；

（3）令运算结果为该点处处理的响应；

（4）对图像中的每点重复此步骤。

如果邻域中的像素计算为线性运算，则称为线性空间滤波，否则称为非线性空间滤波。常见的线性运算为邻域中每像素与对应的系数相乘，然后结果进行累加，从而得到点 $(x,y)$ 处的响应。若邻域的大小为 $m\times n$，则总共需要 $m\times n$ 个系数。这些系数排列为一个矩阵，称为滤波器掩模、滤波掩模、核、模板或窗口。线性空间滤波的过程就是在图像中逐点移动滤波器掩模 $w$ 的中心。在每个点 $(x,y)$ 处，滤波器在该点的响应是滤波器掩模所限定的相应邻域像素与滤波器系数的乘积结果的累加。因为具有唯一中心点的特性，掩模的大小应均为奇数，所以有意义的掩模的最小尺寸是 $3\times 3$。图 5.1 所示为 $3\times 3$ 的掩模在某图像中点 $(x,y)$ 处做线性滤波，其响应 $g(x,y)$ 为

$$\begin{aligned} g(x,y) =& w(-1,-1)f(x-1,y-1)+w(-1,0)f(x-1,y)+\cdots \\ & w(0,0)f(x,y)+w(0,1)f(x,y+1)+\cdots \\ & w(1,0)f(x+1,y)+w(1,1)f(x+1,y+1) \end{aligned} \tag{5.1}$$

一般来说，在 $M\times N$ 像素的图像 $f(x,y)$ 上，用 $m\times n$ 大小的滤波器掩模进行线性滤波的结果由下式给出：

$$g(x,y)=\sum_{s=-a}^{a}\sum_{t=-b}^{b}w(s,t)f(x+s,y+t) \tag{5.2}$$

其中，$a=(m-1)/2$，$b=(n-1)/2$，$x$ 和 $y$ 是可变的，以便 $w$ 中的每个元素可访问 $f$ 中的每像素。

非线性空间滤波处理也是基于邻域处理，且掩模滑过一幅图像的机理与刚刚论述的一样。例如，中值计算是非线性操作，基于中值计算的非线性滤波器的基本函数是计算滤波器掩模所作用邻域的灰度中值。

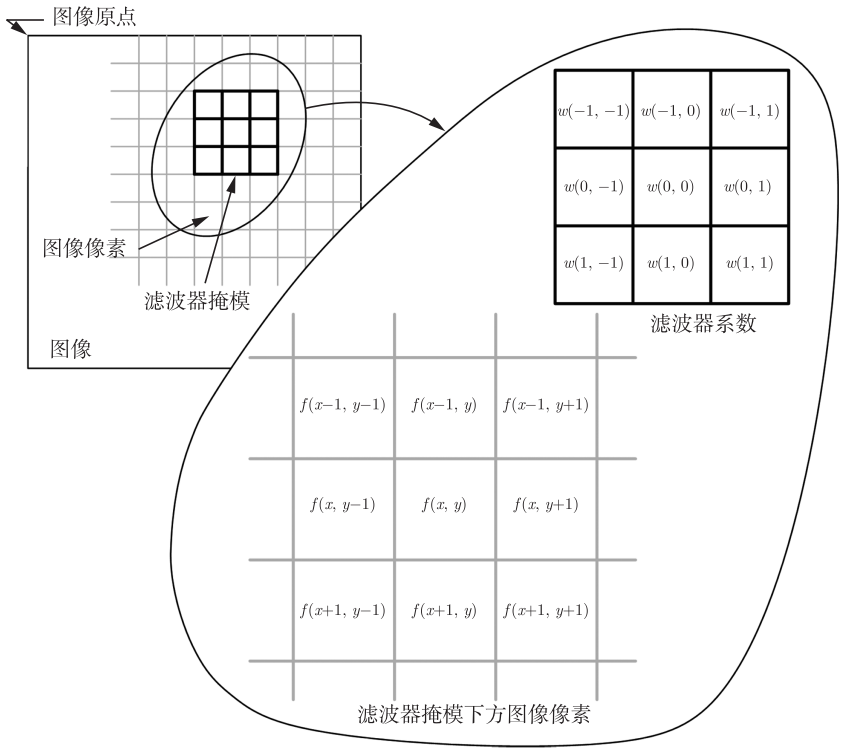



**图 5.1　线性空间滤波**

实现空间滤波邻域处理时要重点考虑滤波中心靠近图像边缘的情况。给定一个简单的 $n \times n$ 大小的方形滤波器掩模，当掩模中心距离图像边缘为 $(n-1)/2$ 像素时，该掩模至少有一条边与图像边缘相重合。如果掩模的中心继续向图像边缘靠近，那么掩模的行或列就会处于图像平面之外。有很多方法可以处理这种问题，最简单的方法是将掩模中心点的移动范围限制在距离图像边缘不小于 $(n-1)/2$ 像素处，这种做法将使处理后的图像比原始图像稍小，但滤波后的图像中的所有像素点都由整个掩模处理。如果要求处理后的输出图像与原始图像一样大，则仅用包含于图像中的掩模来滤波所有像素，通过这种方法，图像靠近边缘部分的像素带将用部分滤波掩模来处理。另一种方法是在图像边缘以外再补上若干行和若干列灰度为零的像素点，或者将边缘复制补在图像之外，补上的那部分经过滤波处理后去除，这种方法保持了处理后的图像与原始图像尺寸大小相等，但是补在靠近图像边缘的部分会带来不良影响，这种影响随着掩模尺寸的增加而增大。

### 5.1.2　空间滤波器

#### 1. 均值滤波器

均值滤波器是用滤波掩模确定的邻域内像素的平均灰度值代替图像中每个像素点的值，此操作能够减小图像灰度的“尖锐”变化，常被用于模糊处理和减小噪声，属于平滑线性空间滤波器。然而，由于图像中物体边缘也是图像灰度尖锐变化的一种表现，所以均值滤波处理还存在着边缘模糊的负面效应。均值滤波器的主要应用是去除图像中的不相干

细节，“不相干”是指与滤波掩模尺寸相比较小的像素区域。式 (5.3) 中给出了两个 $3 \times 3$ 均值滤波器掩模。

$$\boldsymbol{w}_1 = \frac{1}{9} \times \begin{bmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \boldsymbol{w}_2 = \frac{1}{16} \times \begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} \tag{5.3}$$

第一个滤波器计算掩模下方的像素平均值，把掩模系数代入式 (5.3) 中即可清楚地看出这一点。

$$R = \frac{1}{9} \sum_{i=1}^{9} z_i \tag{5.4}$$

滤波器的系数全为“1”，即 $z_i = 1$，所以 $R$ 是由掩模定义的 $3 \times 3$ 邻域内像素灰度的平均值。可见，一个系数全为“1”的 $m \times n$ 掩模应有等于 $1/mn$ 的归一化常数。

式 (5.3) 中所示的第二种掩模也称为加权平均，使用这一术语是指用不同的系数乘以像素，因此从权值上看，一些像素比另一些更重要。对于 $\boldsymbol{w}_2$ 中的 $3 \times 3$ 掩模，处于掩模中心位置的像素比其他任何像素的权值都要大，因此在均值计算中给定的这一像素显得更重要，而距离掩模中心较远的其他像素就显得不太重要。由于对角项离中心比离正交方向相邻的像素更远，所以它的重要性比与中心直接相邻的四像素低。把中心点系数设为最高，而随着距中心点距离的增加减小系数值，是为了减小均值滤波处理产生的模糊。在实践中，由于这些掩模在一幅图像中所占的区域很小，通常很难看出使用图中的各种掩模或用其他类似手段平滑处理后的图像之间的区别。

均值滤波原理程序如下，滤波结果图 5.2 所示。

```
close all;
clear;
clc;

img = imread('chepai.png');
img = rgb2gray(img);
[m, n] = size(img);
img = double(img);
fz = 3;                                %滤波器窗口大小
img_pad = zeros(m+fz-1, n+fz-1);       %拓展图像边界
half = floor(fz/2);                    %滤波器窗口中点的index-1
img_pad(half+1:m+half, half+1:n+half) = img;
filter1 = 1/fz^2 * ones(fz);
G1 = zeros(m, n);
for i = 1:m
    for j = 1:n
        L = img_pad(i:i+fz-1, j:j+fz-1).*filter1;
        G1(i, j) = sum(sum(L));
```

```
    end
end
figure,imshow(mat2gray(img));          %显示原图
figure,imshow(mat2gray(G1));           %显示均值滤波后的图片
```

（a）灰度图像　（b）3×3 像素掩模均值滤波　（c）7×7 像素掩模均值滤波　（d）11×11 像素掩模均值滤波

**图 5.2　均值滤波**

### 2. 中值滤波器

中值滤波器是用滤波掩模确定的邻域内所有像素的中值代替图像中每个像素点的值，其中的中值计算是一种非线性处理技术，故中值滤波器属于平滑非线性空间滤波器。例如，采用 $3\times3$ 像素中值滤波器，某点 $(i,j)$ 的 8 个邻域的一系列像素值为：[17, 25, 13, 11, 19, 81, 9, 28, 27]，统计排序结果为：[9, 11, 13, 17, 19, 25, 27, 28, 81]。排在中间位置（第 5 位）的 19 即作为 $(i,j)$ 点中值滤波的响应 $g(i,j)$。在一定的条件下，中值滤波可以克服均值滤波器等线性滤波器所带来的图像细节模糊问题，但是对一些细节多，特别是点、线、尖顶细节较多的图像则不宜采用中值滤波的方法。

中值滤波器掩模的形状可以是方形、矩形和十字形等，但不管哪种形状，随窗口的增大有效信号的损失也将明显增加。另外，随着窗口的移动，一个像素要重复参与多次计算，处理时间变长，且窗口越大，处理时间越长。在实际操作中，掩模的尺寸一般先用 $3\times3$ 再取 $5\times5$，然后逐渐增大，直到其滤波效果满意为止。对于有缓变的较长轮廓线物体的图像，采用方形或圆形掩模为宜，对于包含尖顶角物体的图像，适宜用十字形掩模。与均值滤波器相比，从总体上来说，中值滤波器能够较好地保留原图像中的跃变部分。

中值滤波原理程序如下，滤波结果如图 5.3 所示。

```
close all;
clear;
clc;

img = imread('chepai.png');
img = rgb2gray(img);
[m, n] = size(img);
img = double(img);
fz = 3;                                %滤波器窗口大小
```

```
img_pad = zeros(m+fz-1, n+fz-1);     %拓展图像边界
half = floor(fz/2);                  %滤波器窗口中点的index-1
img_pad(half+1:m+half, half+1:n+half) = img;
G2 = zeros(m, n);
for i = 1:m
    for j = 1:n
        area = img_pad(i:i+fz-1, j:j+fz-1);
        area = area(:);
        med = median(area);
        G2(i, j) = med;
    end
end

figure,imshow(mat2gray(img));            %显示原图
figure,imshow(mat2gray(G2));             %显示中值滤波后的图片
```

（a）灰度图像　（b）3×3 掩模中值滤波　（c）7×7 掩模中值滤波　（d）11×11 掩模中值滤波

**图 5.3　中值滤波**

### 3. 锐化滤波器

锐化滤波器主要用于增强图像的灰度跳变部分，这一点与均值滤波器等平滑线性滤波器对灰度跳变的抑制正好相反。此处主要讨论基于一阶和二阶微分的锐化滤波器，其核心在于考查恒定灰度区域中突变的开始点与结束点（台阶和斜坡突变）及沿着灰度斜坡处的微分性质。

对于一阶微分的定义必须保证以下几点：

（1）在恒定灰度区域的微分值为零；

（2）在灰度台阶或斜坡处微分值非零；

（3）沿着斜坡的微分值非零。

类似地，任何二阶微分的定义必须保证以下几点：

（1）在恒定灰度区域的微分值为零；

（2）在灰度台阶或斜坡的起点处微分值非零；

（3）沿着斜坡的微分值非零。

因为处理的是数字量，其值是有限的，故最大灰度级的变化也是有限的，并且变化发生的最短距离是在两相邻像素之间。对于一元函数 $f(x)$，表达一阶微分的定义是一个差值：

$$\frac{\partial f}{\partial x} = f(x+1) - f(x) \tag{5.5}$$

这里，为了与对二维图像函数 $f(x,y)$ 求微分时的表达式保持一致，使用了偏导数符号。类似地，用如下差分定义二阶微分：

$$\frac{\partial^2 f}{\partial x^2} = f(x+1) + f(x-1) - 2f(x) \tag{5.6}$$

在此基础上，二维函数 $f(x,y)$ 的二阶微分（拉普拉斯算子）定义为

$$\nabla^2 f(x,y) = \frac{\partial^2 f}{\partial x^2} + \frac{\partial^2 f}{\partial y^2} \tag{5.7}$$

对于离散的二维图像，二阶偏导数与二阶差分近似，由此可推导出拉普拉斯算子表达式为

$$\nabla^2 f(x,y) = f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) - 4f(x,y) \tag{5.8}$$

其对应的滤波模板如下：

$$\boldsymbol{w}_1 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & -4 & 1 \\ 0 & 1 & 0 \end{bmatrix} \tag{5.9}$$

因为在锐化过程中，绝对值相同的正值和负值实际上表示相同的响应，故也等同于使用如下模板：

$$\boldsymbol{w}_2 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix} \tag{5.10}$$

分析拉普拉斯模板的结构，可知这种模板对于 90° 的旋转是各向同性的。所谓对于某角度各向同性是指把原图像旋转该角度后再进行滤波与先对原图像滤波再旋转该角度的结果相同。这说明拉普拉斯算子对于接近水平和接近竖直方向的边缘都有很好的响应，从而也就避免在锐化滤波时要进行两次滤波的麻烦。更进一步，还可以得到如下对于 45° 旋转各向同性的滤波器：

$$\boldsymbol{w}_3 = \begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix} \qquad \boldsymbol{w}_4 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \tag{5.11}$$

以 $\boldsymbol{w}_1$ 滤波模板为例，锐化滤波程序代码如下，其滤波结果如图 5.4 所示。

```
close all;
clear all;
clc;

img = imread('chepai.png');
img = rgb2gray(img);
img = im2double(img);
figure, imshow(img);
f = padarray(img,[1,1],'symmetric','both');
[m,n] = size(f);
M = zeros(size(f));              %提前定义梯度图像M，有利于提高运算速度
for x=2:m-1
    for y=2:n-1
        M(x,y)=f(x+1,y)+f(x-1,y)+f(x,y+1)+f(x,y-1)-4*f(x,y);
    end
end
M = M(2:m-1,2:n-1);              %去掉扩充的行列
figure, imshow(M);
```

（a）灰度图像　　（b）锐化滤波结果

**图 5.4　锐化空间滤波**

## 5.2　频域滤波

### 5.2.1　傅里叶变换基本概念

1807 年，傅里叶提出了傅里叶级数的概念，即任一周期信号均可分解为多个正弦信号的叠加。1822 年，傅里叶又提出了傅里叶变换，目前已经成为一种常用的正交变换，在数字图像处理领域也起着非常重要的作用。傅里叶变换主要分为连续傅里叶变换和离散傅里叶变换，在数字图像处理中经常用到的是二维离散傅里叶变换。

对于有限长序列 $f(x)(x=0,1,\cdots,N-1)$，定义一维离散傅里叶变换对如下：

$$\begin{aligned} F(u) &= \mathrm{DFT}[f(x)] = \sum_{x=0}^{N-1} f(x)W^{ux} \quad u=0,1,\cdots,N-1 \\ f(x) &= \mathrm{IDFT}[F(u)] = \frac{1}{N}\sum_{u=0}^{N-1} F(u)W^{-ux} \quad x=0,1,\cdots,N-1 \end{aligned} \tag{5.12}$$

式中，$W=\mathrm{e}^{-\mathrm{j}\frac{2\pi}{N}}$ 称为变换核。由式 (5.12) 可见，给定序列可以求出其傅里叶谱 $F(u)$，反之亦然。因此离散傅里叶变换对可以简记为 $f(x)\leftrightarrow F(u)$。$F(u)$ 一般可以写成复数形式：

$$F(u)=|F(u)|\mathrm{e}^{\mathrm{j}\varphi(u)} \tag{5.13}$$

其中，$|F(u)|$ 为傅里叶幅度谱；$\varphi(u)$ 为相位谱。

将一维离散傅里叶变换推广到二维，变换对定义为

$$\begin{aligned} F(u,v) &= \frac{1}{MN}\sum_{x=0}^{M-1}\sum_{y=0}^{N-1} f(x,y)\mathrm{e}^{-\mathrm{j}2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \\ f(x,y) &= \sum_{u=0}^{M-1}\sum_{v=0}^{N-1} F(u,v)\mathrm{e}^{\mathrm{j}2\pi\left(\frac{ux}{M}+\frac{vy}{N}\right)} \end{aligned} \tag{5.14}$$

其中，$u,x=0,1,2,\cdots,M-1$；$v,y=0,1,\cdots,N-1$；$x$、$y$ 为时域变量；$u$、$v$ 为频域变量。

### 5.2.2　频域滤波基础

傅里叶变换可以将图像从空间域变换到频域，而傅里叶反变换则可以将图像的频谱逆变换为空间域图像，即人眼可以直接识别的图像。这样一来，可以利用空间域图像与频谱之间的对应关系，尝试将空间域卷积滤波变换为频域滤波，经过频域滤波处理后，再将图像反变换回空间域。根据卷积定理，两个二维连续函数在空间域中的卷积可由其相应的两个傅里叶变换乘积的反变换而得；反之，在频域中的卷积可由在空间域中乘积的傅里叶变换而得。即

$$\begin{aligned} f(x,y)*h(x,y) &\Leftrightarrow F(u,v)H(u,v) \\ f(x,y)h(x,y) &\Leftrightarrow F(u,v)*H(u,v) \end{aligned} \tag{5.15}$$

其中，$F(u,v)$ 和 $H(u,v)$ 分别表示 $f(x,y)$ 和 $h(x,y)$ 的傅里叶变换，而符号“$\Leftrightarrow$”表示傅里叶变换对，即左侧的表达式可通过傅里叶正变换得到右侧的表达式，而右侧的表达式可通过傅里叶反变换得到左侧的表达式。式 (5.15) 构成了整个频域滤波的基础，式中的乘积实际上就是两个二维矩阵对应元素之间的乘积。

频域滤波的基本步骤如下：

（1）对原始图像 $f(x,y)$ 做傅里叶变换，得到 $F(u,v)$；

（2）计算滤波器函数 $H(u,v)$ 与 $F(u,v)$ 的乘积 $G(u,v)$；

（3）对频谱 $G(u,v)$ 做傅里叶反变换得出时域 $g(x,y)$；

（4）取 $g(x,y)$ 的实部作为最终滤波结果图像。

由上述基本步骤可以看出，滤波函数 $H(u,v)$ 对于滤波操作能否取得理想结果起着关键作用，即在滤波过程中抑制或滤除频谱中某些频率的分量，而保留其他的一些频率不受影响。因此，多种滤波器，也称滤波器传递函数，被设计以应对各种情况。

### 5.2.3 频域滤波器

#### 1. 频域低通滤波器

在频谱中，低频主要对应图像在平滑区域的总体灰度级分布，而高频对应图像的细节部分。因此，图像平滑可以通过衰减图像频谱中的高频部分来实现，这就建立了均值滤波、中值滤波等空间域图像平滑滤波与频域低通滤波之间对应关系。

（1）理想低通滤波器。

设傅里叶平面上理想低通滤波器离开原点的截止频率为 $D_0$，则理想低通滤波器的传递函数为

$$H(u,v)=\begin{cases}1, & D(u,v)\leqslant D_0\\ 0, & D(u,v)>D_0\end{cases} \tag{5.16}$$

其中，$D(u,v)=\sqrt{u^2+v^2}$。理想低通滤波器的函数如图 5.5(a)、图 5.6(a) 所示，$F(u,v)$ 在 $D_0$ 内的频率分量无损通过，而大于 $D_0$ 的分量则被过滤掉。

由于图像高频成分中除了噪声外还包含大量的边缘信息，因此采用该滤波器在去除噪声得到平滑图像的同时将会导致边缘信息的损失而使图像边缘模糊，并且会产生振铃效应。

（2）巴特沃思低通滤波器。

$n$ 阶巴特沃思（Butterworth）滤波器的传递函数如下所示：

$$H(u,v)=\frac{1}{1+\left[\dfrac{D(u,v)}{D_0}\right]^{2n}} \tag{5.17}$$

图 5.5（b）、图 5.6（b）所示为巴特沃思低通滤波器函数，可以看出，巴特沃思低通滤波器的特性是连续性衰减，而不像理想滤波器那样陡峭和明显不连续。因此采用该滤波器滤波在抑制图像噪声的同时，图像边缘的模糊程度大大减小，没有振铃效应产生，但计算量大于理想低通滤波器。

（3）高斯低通滤波器。

由于高斯函数的傅里叶变换和反变换均为高斯函数，并常常用来帮助寻找空间域与频率域之间的联系，所以基于高斯函数的滤波具有特殊的重要意义。一个二维的高斯低通滤波器的转移函数定义为

$$H(u,v)=\mathrm{e}^{-D(u,v)^2/2\sigma^2} \tag{5.18}$$