# 2019 年 秋 季学期研究生课程考核

## （读书报告、研究报告）

考 核 科 目: 数 字 图 像 处 理

学生所在院（系）: 仪 器 科 学 与 工 程 学 院

学 生 所 在 学 科: 仪 器 科 学 与 技 术

学 生 姓 名: 姬 森 展　肖 博 文　高 文 佳

闫 宏 亮　郑 品 军

学 号: 姬 森 展　19S001050

肖 博 文　19S101100

高 文 佳　19S001030

闫 宏 亮　19S001020

郑 品 军　19S001041

学 生 类 别: 必 修

考 核 结 果　　阅 卷 人

# 数字图像处理大作业报告

## 彩色图像的锐化处理

姬森展 肖博文 高文佳 闫宏亮 郑品军

哈尔滨工业大学

**2019 年 12 月**

# 第一章 基础知识

## 1.1 课题研究背景和内容

视觉是人类最高级的感知器官，因此，图像在人类信息获取中占着非常重要的地位。图像作为一种信息传递的载体，与人类的学习生活息息相关，而其应用更是渗透到人类活动的各个领域。但由于各种不可控因素的存在，人们所得到的图像总会出现色彩偏暗，细节不突出等各种缺陷，而为了使图像达到人们的应用需求，则要对这些图像进行相应的处理。在一系列数字图象处理中，图像增强一直以来都是引人注目的研究点[1]。

图像增强就是有目的性地突出图像中有意义的区域，使图像中不同特征之间的差别加大，降低或去除不需要的图像信息，改善图像质量、丰富信息量，使之更适合人或机器进行理解和分析[1]。

### 1.1.1 研究背景

1. 图像锐化方法

在图像增强过程中，通常需要利用各类图像平滑算法消除噪声。一般来说，图像的能量主要集中在其低频部分，噪声所在的频段主要在高频段，同时图像边缘信息也主要集中在其高频部分。这将导致原始图像在平滑处理之后，图像边缘和图像轮廓模糊的情况出现。为了减少这类不利效果的影响，就需要利用图像锐化技术，使图像的边缘变得清晰。

图像锐化处理的目的是为了使图像的边缘、轮廓线以及图像的细节变得清晰，经过平滑的图像变得模糊的根本原因是因为图像受到了平均或积分运算，因此可以对其进行逆运算(如微分运算)就可以使图像变得清晰。微分运算是求信号的变化率，由傅立叶变换的微分性质可知，微分运算具有较强高频分量作用。从频率域来考虑，图像模糊的实质是因为其高频分量被衰减，因此可以用高通滤波器来使图像清晰。但要注意能够进行锐化处理的图像必须有较高的信噪比，否则锐化后图像信噪比反而更低，从而使得噪声增加的比信号还要多，因此一般是先去除或减轻噪声后再进行锐化处理[2]。

目前常用的图像锐化方法主要有两种：高通滤波法和空域微分法，下面将分别介绍两种方法[3]。

1. 空域微分法

梯度锐化法是空域微分法中的一个常用方法。对于图像 f(x,y)在点(x,y)处的梯度 G[f(x,y)]定义为一个二维列向量：

$$G[f(x,y)] = \left[\frac{\partial f}{\partial x}\frac{\partial f}{\partial y}\right]^{T}$$

梯度的幅值即向量的模值，为：

$$|G[f(x,y)]| = \left[\left(\frac{\partial f}{\partial x}\right)^2 + \left(\frac{\partial f}{\partial y}\right)^2\right]^{\frac{1}{2}}$$

梯度的方向在最大变化率方向上，方向角可表示为：

$$\theta = \arctan\left(\frac{\partial f}{\partial y} / \frac{\partial f}{\partial x}\right)$$

对于离散函数 也有相应的概念和公式，只是用差分代替微分。差分可取为后向差分，前向差分。在 x, y 方向上的一阶向后差分分别定义为：

$$\nabla_x f(i,j) = f(i,j) - f(i-1,j)$$

$$\nabla_y f(i,j) = f(i,j) - f(i,j-1)$$

梯度定义为：

$$G[f(i,j)] = \left[\nabla_x f(i,j) \quad \nabla_y f(i,j)\right]^T$$

其模和方向分别为：

$$|G[f(x,y)]| = \left[\left(\nabla_x f(i,j)\right)^2 + \left(\nabla_y f(i,j)\right)^2\right]^{\frac{1}{2}}$$

$$\alpha = \arctan\left[\frac{\nabla_x f(i,j)}{\nabla_y f(i,j)}\right]$$

梯度的模（幅值）就是 g(i,j) = G[f(i,j)]最大变化率方向的单位距离所增加的量。由梯度的计算可知，在图像灰度变化较大的边沿区域其梯度值大，在灰度变化平缓的区域梯度值较小，而在灰度均匀的区域其梯度值为零。我们根据得到的梯度值来返回像素的值，如将梯度值大的像素设置成白色，梯度值小的设置为黑色，这样就可以将边缘提取出来了，或者是加强梯度值大的像素灰度值就可以突出细节了达到了锐化的目的。

除梯度算子以外，还可采用 Roberts 算子，Sobel 算子，Prewitt 算子，Laplacian 算子计算梯度，来增强边缘实现图像锐化。由于篇幅有限，本文重点介绍 Laplacian 算子。Roberts 算子，Sobel 算子和 Prewitt 算子模板见附录 1。

Laplacian 锐化方法是一种二阶微分算法。首先记$\nabla^2 f$ 是 Laplacian 算子，定义：

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

然后针对离散数字图像 f(i,j)分别求一阶偏导和二阶偏导，可以计算得出它的 Laplacian 算子

$$\nabla^2 f = 4f(i,j) - f(i+1,j) - f(i-1,j) - f(i,j+1) - f(i,j-1)$$

然后将其写成模板系数的形式，就是图像锐化处理中使用的 Laplacian 算子。

Laplacian 算子利用二阶导数信息，具有各向同性，即与坐标轴方向无关，坐标轴旋转后梯度结果不变。使得图像经过二阶微分后，在边缘处产生一个陡峭的零交叉点，根据这个对零交叉点判断边缘。

一种 Laplacian 算子模板如下：

$$H_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

为了改善锐化效果，可以脱离微分的计算原理，在原有的算子基础上，对模板系数进行改变，获得 Laplacian 变形算子：

$$H_2 = \begin{bmatrix} -1 & -1 & -1 \\ -1 & 8 & -1 \\ -1 & -1 & -1 \end{bmatrix} \quad H_3 = \begin{bmatrix} 1 & -2 & 1 \\ -2 & 4 & -2 \\ 1 & -2 & 1 \end{bmatrix} \quad H_4 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 5 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

其中 $H_2$ 是在 $H_1$ 的基础上在考虑 45°和 135°方向的结果

2. 高通滤波法：

图像锐化的另一种常用方法是频域高通滤波法。图像的边缘、细节主要位于高频部分，而图像的模糊是由于高频成分比较弱产生的。高通滤波器的原理是在频域对图像的频率成分进行滤波，让高频成分通过，使低频成分削弱，再经逆傅立叶变换得到边缘锐化的图像。常用的高通滤波器有：

(1) 理想高通滤波器

二维理想高通滤波器的传递函数为：

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

(2) 巴特沃斯高通滤波器

n 阶巴特沃斯高通滤波器的传递函数的定义如下：

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

(3) 高斯高通滤波器

高斯高通滤波器的传递函数为：

$$H(u,v) = e^{-\left|\frac{D_0}{D(u,v)}\right|^n}$$

实际操作时，先将原图的傅里叶谱保留下来，然后叠加上高通滤波器的滤波结果，所得到的图像就是锐化后的图像。常用的计算公式为：

$$g(x,y) = F^{-1}\left[\left(k_1 + k_2 H_{HP}(u,v)\right) \cdot F(u,v)\right]$$

这里为了调整锐化程度，引入了两个变量$k_1$与$k_2$。$k_1$可以调整直流分量的衰减程

度，$k_2$可以调整高频分量的增幅程度。

## 1.2 主要研究内容

空域微分法是图像锐化中的最常用的有效方法，因此本文在 matlab 环境下使用不同的算子对彩色图像进行锐化处理，并将锐化处理之后的图像与锐化前的图像进行比较，以验证不同算子的锐化效果。

# 第二章 实验结果

## 2.1 程序流程设计

1. 实验一：不同算子锐化效果对比

由上节的图像锐化的原理，采用对比实验的方式，一组直接对图像进行锐化处理，另一组先进行均值滤波，再进行锐化处理，对比实验结果。其中，锐化过程分别采用了 Sobel 算子、Roberts 算子、Prewitt 算子和 Laplacian 算子。本实验所设计的 matlab 程序的流程图如下图 2-1 所示：

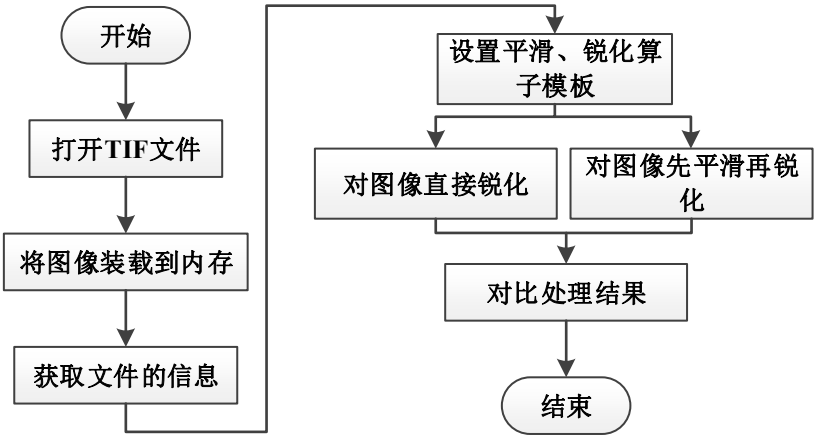


图 2-1 实验一的程序流程图

2. 实验二：Laplacian 算子和高斯-拉普拉斯算子的处理效果对比

锐化在增强边缘和细节的同时往往也会增强噪声，基于二阶微分的 Laplacian 算子对于细节〈细线和孤立点〉能产生更强的响应， 所以相较于一阶的梯度算子使用更为广泛，但是它对噪声点的响应也更强，锐化之后的噪声也更明显。因此为了在取得更好的锐化效果的同时把噪声的干扰降到最低，可以先对带有噪声的原始图像进图像进行平滑滤波，再进行锐化增强边缘和细节。基于以上原则，将在用于图像平滑的高斯平滑算子和用于图像锐化的拉普拉斯锐化算子结合起来，得到高斯-拉普拉斯(LoG)算子，使用这一个算子，既能去噪又能起到锐化的效果。高斯-拉普拉斯算子模板由如下公式(1)和公式(2)给出。

$$h_g\left(n_1,n_2\right)=e^{\frac{-\left(n_1^2+n_2^2\right)}{2\sigma^2}} \qquad (1)$$

$$h(n_1,n_2)=\frac{\left(n_1^2+n_2^2-2a^2\right)_n\left(n_1n_2\right)}{2\pi\sigma^6\sum_{n_1}\sum_{n_2}h_g} \qquad (2)$$

本实验中将分别使用先均值滤波再使用 Laplacian 算子和使用 LoG 算子对原图进行锐化，比较两者的锐化效果，以及 LoG 算子高斯函数的标准差取不同值时的锐化效果。设计的 matlab 程序的流程图，如下图 2-2 所示：

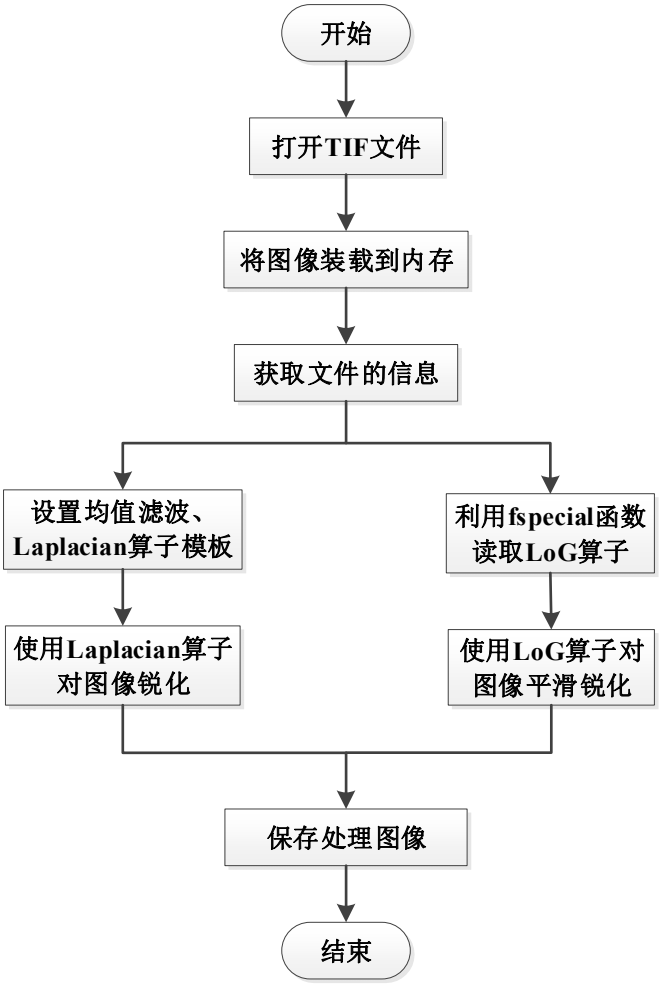


图 2-2 实验二的程序流程图

3. 实验三：高通滤波法不同滤波器的处理效果对比

本实验使用高通滤波法，将原图像进行离散傅里叶变换，在分别使用理想高通滤波器、巴特沃斯滤波器和高斯高通滤波器进行频域滤波。对滤波后的的图像锐化效果进行对比，以比较不同高通滤波器的锐化能力。图像尺寸为 512×512×3，设置 $D_0=80$，$k_1=1.0$，$k_2=1.1$。实验的程序流程框图如下图 2-3 所示。

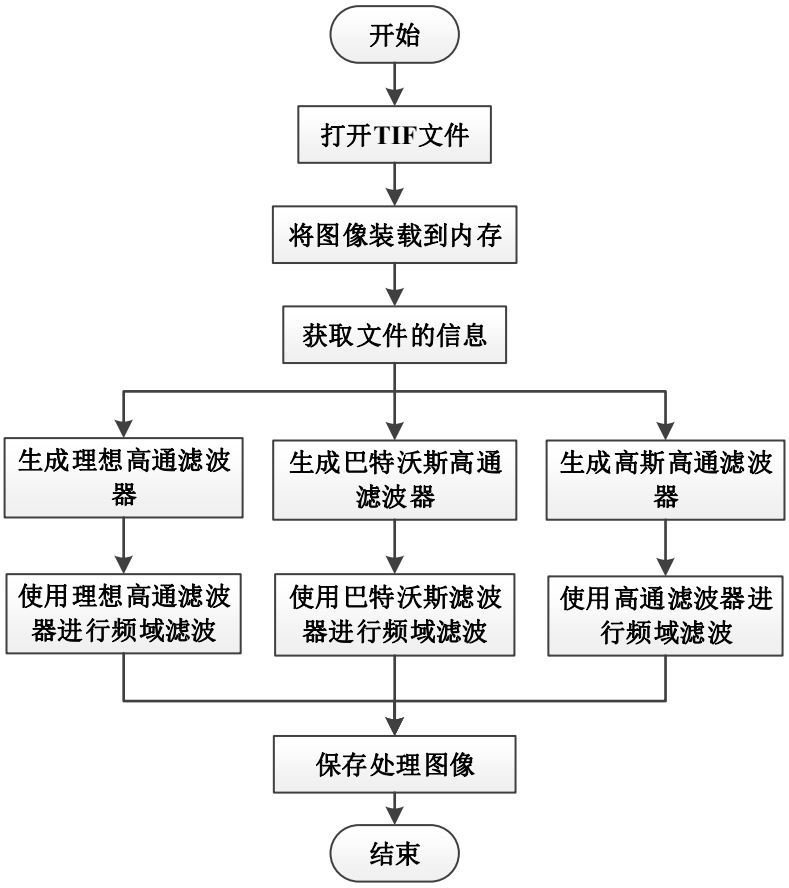


图 2-3 实验三的程序流程图

## 2.2 实验结果

1. 实验一

实验结果如下图2-4到2-11所示，左侧是输入原图像，中间是使用不同种类的算子模板提取的边缘特征，右侧是使用不同种类算子锐化之后的图像。在本实验中一共选取了Sobel算子、Roberts算子、Prewitt算子和Laplacian算子共四种算子。

（1）Sobel 算子


图 2-4 Sobel 算子锐化结果


图 2-5 Sobel 算子锐化结果

（2）Roberts 算子


图 2-6 Roberts 算子锐化结果


图 2-7 Roberts 算子锐化结果

（3）Prewitt 算子


图 2-8 Prewitt 算子锐化结果


图 2-9 Prewitt 算子锐化结果

（4）Laplacian 算子

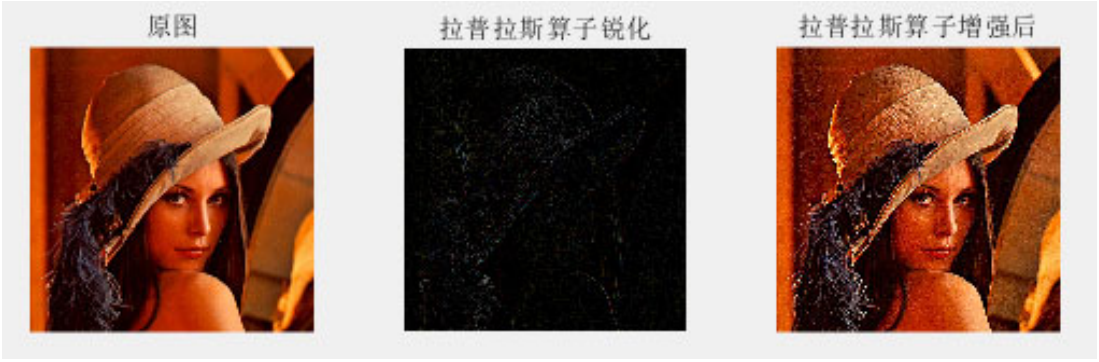


图 2-10 Laplacian 算子锐化结果


图 2-11 Laplacian 算子锐化结果

2. **实验二**

实验结果如下图2-12到2-16所示，其中左上方是原图，右上方是Laplacian锐化后的图像，图2-13到2-16分别是高斯函数的标准差σ＝0.5、σ＝0.6、σ＝2和σ＝4的时候，利用LoG算子平滑锐化的图像。

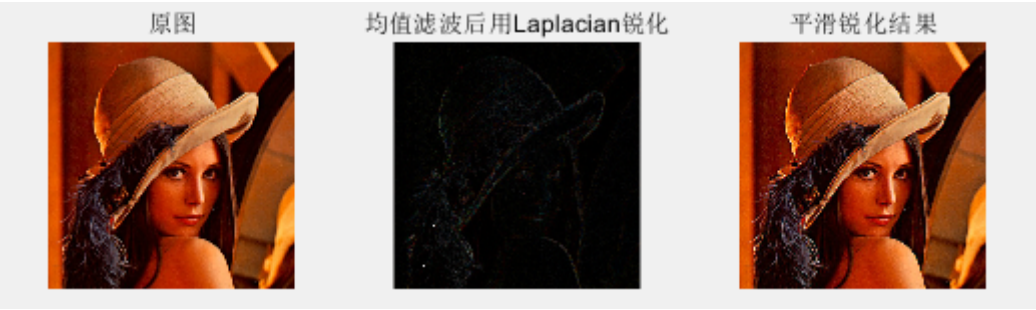

图 2-12 均值滤波后使用 Laplacian 算子锐化结果

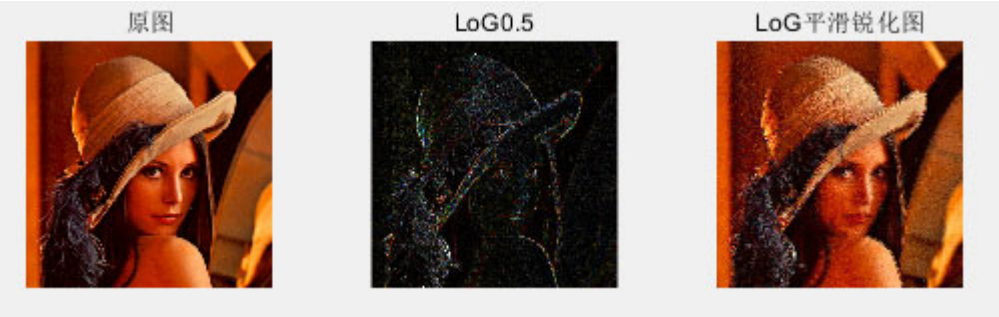


图 2-13 σ=0.5 LoG 算子平滑锐化结果


图 2-14 σ=0.6 LoG 算子平滑锐化结果


图 2-15 σ=2 LoG 算子平滑锐化结果

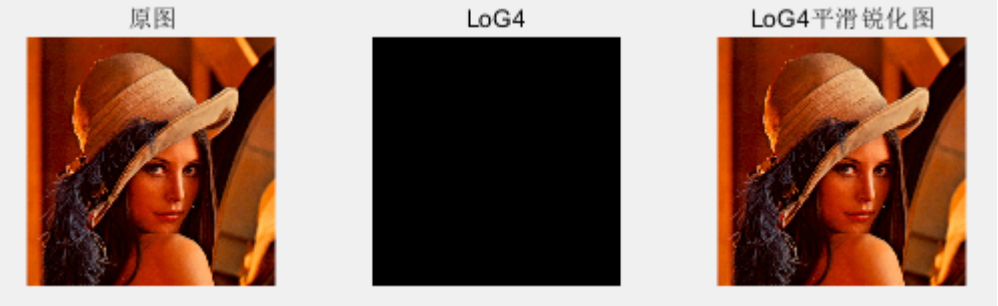


图 2-16 σ=4 LoG 算子平滑锐化结果

3. **实验三**

实验结果如下图2-17、图2-18和图2-19所示。每个实验结果图中，左图为原始图像，中间图为边缘特征，右图是锐化增强之后的图像。


图 2-17 理想高通滤波器锐化结果


图 2-18 巴特沃斯高通滤波器锐化结果


图 2-19 高斯高通滤波器锐化结果

## 2.4 实验结论

从上述实验结果我们可以得出以下结论：

实验一：

由图 2-4、2-6、2-8、2-10 可知，直接对原图像进行锐化处理时，锐化效果并不理想，图像边缘在变清晰的同时，原图像中的噪声也被放大了，从而图像整体变得模糊，从直接对图像进行锐化的效果来看，Prewitt 算子和 Roberts 算子的锐化效果比较好，而 Laplacian 算子和 Sobel 算子的锐化效果较差，这两种算子对噪声更敏感，其中 Sobel 算子由于只采用了两个方向的模板，只能检测水平方向

和垂直方向的边缘，对于图像中纹理较为复杂的部分，例如帽子上的翎毛其锐化效果较差，Laplacian 算子对噪声最为敏感，图像中增加了很多明显的噪点；

由图 2-5、2-7、2-9、2-11，对比未平滑直接锐化的图像可知，在对图像进行均值滤波后再进行锐化的效果明显好于未经平滑直接锐化的效果，因原图像中存在着一些细小的噪声，而各种锐化算子均对噪声比较敏感，在进行均值滤波即对图像进行平滑处理后，再进行锐化时，锐化效果有了明显改善，其中 Laplacian 算子改善效果明显，整体锐化效果 Laplacian 算子略好于 Prewitt 算子和 Roberts 算子，Sobel 算子锐化效果较差。

实验二：

由图 2-12、2-14、2-15、2-16 对比可知，采用高斯-拉普拉斯进行锐化的效果好于均值滤波后再进行锐化的效果；对比 2-12、2-13、2-14、2-15 的效果可知，$\sigma \leqslant 0.5$ 时，使用 LoG 算子进行锐化会使图像变得模糊，在 $\sigma=0.6$ 时，LoG 算子与 Laplacian 算子效果相近，当 $\sigma$ 继续增大时，对图像的平滑效果变好，但对细节的增强效果逐渐下降，整体效果好于 Laplacian 算子。

实验三：

由图 2-17、2-18、2-19 对比可知，使用理想高通滤波器和巴特沃斯滤波器，虽然能够起到锐化的作用，但均会在一定程度上产生振铃效应从而引入误差。而高斯高通滤波器在锐化的同时几乎不产生振铃效应，相比之下锐化效果最好。

# 参考文献

[1] 李达. 彩色图像增强算法研究[D]. 北方工业大学，2018.

[2] 刘三国. 图像锐化的研究[D]. 曲阜师范大学，2011.

[3] 朱玉欣. 图像增强算法综述[J]. 信息与电脑(理论版)，2017(16)：104-106.

# 附录一

附录一为实验中使用的算子模板

1. Laplacian 算子

$$H_1 = \begin{bmatrix} 0 & -1 & 0 \\ -1 & 4 & -1 \\ 0 & -1 & 0 \end{bmatrix}$$

2. Sobel 算子

$$H_1 = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix}$$

3. Roberts 算子

$$H_1 = \begin{bmatrix} -1 & 0 \\ 0 & 1 \end{bmatrix} \quad H_2 = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix}$$

4. Prewitt 算子

$$H_1 = \begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

5. Log 算子

$$H_1(\sigma = 0.5) = \begin{bmatrix} 0.044 & 0.047 & 0.056 & 0.047 & 0.044 \\ 0.047 & 0.032 & 0.715 & 0.317 & 0.047 \\ 0.056 & 0.715 & -4.905 & 0.715 & 0.056 \\ 0.047 & 0.032 & 0.715 & 0.317 & 0.047 \\ 0.044 & 0.047 & 0.056 & 0.047 & 0.044 \end{bmatrix}$$

$$H_2(\sigma = 0.6) = \begin{bmatrix} 0.006 & 0.020 & 0.048 & 0.020 & 0.006 \\ 0.020 & 0.276 & 0.243 & 0.276 & 0.020 \\ 0.048 & 0.243 & -2.4429 & 0.243 & 0.048 \\ 0.047 & 0.276 & 0.243 & 0.317 & 0.020 \\ 0.006 & 0.020 & 0.048 & 0.020 & 0.006 \end{bmatrix}$$

$$H_3(\sigma = 2) = \begin{bmatrix} 0.012 & 0.005 & 0.002 & 0.005 & 0.012 \\ 0.005 & -0.007 & -0.127 & -0.007 & 0.005 \\ 0.002 & -0.007 & -0.020 & -0.007 & 0.002 \\ 0.005 & -0.007 & -0.127 & -0.007 & 0.005 \\ 0.012 & 0.005 & 0.002 & 0.005 & 0.012 \end{bmatrix}$$

# 附录二

附录二为部分实验程序

实验一：

```
I=imread('0.tif');%提取图像
Ib=tofloat(I);                %将图像转化为浮点型
rgb = im2double(imread('0.tif'));

rgb_R=rgb(:,:,1);
rgb_G= rgb(:,:,2);
rgb_B= rgb(:,:,3);

lapMatrix=[0 -1 0;-1 5 -1;0 -1 0];  %Laplacian 增强算子
f_R=imfilter(rgb_R,lapMatrix,'replicate');
f_G=imfilter(rgb_G,lapMatrix,'replicate');
f_B=imfilter(rgb_B,lapMatrix,'replicate');

f_RGB = imfilter(rgb,lapMatrix,'replicate');

rgb_tmp=cat(3,rgb_R-f_R,rgb_G-f_G,rgb_B-f_B);

Average = [0 0.2 0;0.2 0.2 0.2;0 0.2 0]; %均值滤波

f_R_Aver=imfilter(rgb_R,Average,'replicate');
f_G_Aver=imfilter(rgb_G,Average,'replicate');
f_B_Aver=imfilter(rgb_B,Average,'replicate');

rgb_tmp_aver=cat(3,f_R_Aver,f_G_Aver,f_B_Aver);%原图像均值滤波后

f_R_aver_filter=imfilter(f_R_Aver,lapMatrix,'replicate');
f_G_aver_filter=imfilter(f_G_Aver,lapMatrix,'replicate');
f_B_aver_filter=imfilter(f_B_Aver,lapMatrix,'replicate');

f_RGB_aver = imfilter(rgb_tmp_aver,lapMatrix,'replicate');
rgb_tmp_aver_filter=cat(3,f_R_Aver-f_R_aver_filter,f_G_Aver-
f_G_aver_filter,f_B_Aver-f_B_aver_filter);

w1 = fspecial('sobel');%Sobel 算子
w2 = w1';
G1 = imfilter(Ib,w1);
G2 = imfilter(Ib,w2);
G = abs(G1)+abs(G2);
I1 = Ib+G;
G1_aver = imfilter(rgb_tmp_aver,w1);
```

```
G2_aver = imfilter(rgb_tmp_aver,w2);
G_aver = abs(G1_aver)+abs(G2_aver);
I1_aver = rgb_tmp_aver + G_aver ;

w3 = [-1 0; 0 1];%Roberts 算子
w4 = [0 -1;1 0];
G3 = imfilter(rgb,w3,'corr','replicate');
G4 = imfilter(rgb,w4,'corr','replicate');
GG = abs(G3)+abs(G4);
I2 = rgb+GG;
G3_aver = imfilter(rgb_tmp_aver,w3,'corr','replicate');
G4_aver = imfilter(rgb_tmp_aver,w4,'corr','replicate');
GG_aver = abs(G3)+abs(G4);
I2_aver = rgb_tmp_aver+GG_aver;

Prewitt1 = [-1 -1 -1;0 0 0;1 1 1]; %prewitt 算子
GGG = imfilter(rgb,Prewitt1,'corr','replicate');
I3 = rgb + GGG;
GGG_aver = imfilter(rgb_tmp_aver,Prewitt1,'corr','replicate');
I3_aver = rgb_tmp_aver + GGG_aver;

subplot(4,6,1), imshow(I);
title('原图');
subplot(4,6,2), imshow(rgb_tmp);
title('拉普拉斯算子锐化');
subplot(4,6,3), imshow(f_RGB);
title('拉普拉斯算子增强后');
subplot(4,6,4), imshow(rgb_tmp_aver);
title('对原图平滑');
subplot(4,6,5), imshow(rgb_tmp_aver_filter);
title('拉普拉斯算子锐化');
subplot(4,6,6), imshow(f_RGB_aver);
title('拉普拉斯算子增强后');

subplot(4,6,7), imshow(I);
title('原图');
subplot(4,6,8), imshow(G);
title('Sobel 算子锐化');
subplot(4,6,9), imshow(I1);
title('Sobel 算子增强后');
subplot(4,6,10), imshow(rgb_tmp_aver);
title('对原图平滑');
subplot(4,6,11), imshow(G_aver);
title('Sobel 算子锐化');
```

```
subplot(4,6,12), imshow(I1_aver);
title('Sobel 算子增强后');

subplot(4,6,13), imshow(I);
title('原图');
subplot(4,6,14), imshow(GG);
title('Roberts 算子锐化');
subplot(4,6,15), imshow(I2);
title('Roberts 算子增强后');
subplot(4,6,16), imshow(rgb_tmp_aver);
title('对原图平滑');
subplot(4,6,17), imshow(GG_aver);
title('Roberts 算子锐化');
subplot(4,6,18), imshow(I2_aver);
title('Roberts 算子增强后');

subplot(4,6,19), imshow(I);
title('原图');
subplot(4,6,20), imshow(GGG);
title('prewitt 算子锐化');
subplot(4,6,21), imshow(I3);
title('prewitt 算子增强后');
subplot(4,6,22), imshow(rgb_tmp_aver);
title('对原图平滑');
subplot(4,6,23), imshow(GGG_aver);
title('prewitt 算子锐化');
subplot(4,6,24), imshow(I3_aver);
title('prewitt 算子增强后');
```

实验二：

```
I = imread('0.tif');
Id = double(I);
h_lap = [ 0 -1 0;-1 5 -1;0 -1 0];
I_lap = imfilter(Id,h_lap,'corr','replicate');
I1 = Id - I_lap;

h_log = fspecial('log',5,0.6);
I_log = imfilter(Id,h_log,'corr','replicate');
I2 = Id + I_log;

h_log2 = fspecial('log',5,2);
I_log2 = imfilter(Id,h_log2,'corr','replicate');
I3 = Id + I_log2;

h_log3 = fspecial('log',5,4);
I_log3 = imfilter(Id,h_log3,'corr','replicate');
I4 = Id + I_log3;

h_log4 = fspecial('log',3,4);
I_log4 = imfilter(Id,h_log4,'corr','replicate');
I5 = Id + I_log4;

Average = [0 0.2 0;0.2 0.2 0.2;0 0.2 0]; %均值滤波后再用Laplacian锐化
I_aver = imfilter(Id,Average,'corr','replicate');
I_lap_aver = imfilter(I_aver,h_lap,'corr','replicate');
I6 = I_aver - I_lap_aver;

figure;
subplot(4,3,1);
imshow(I),title('原图');
subplot(4,3,2);
imshow(uint8(abs(I6)),[]),title('均值滤波后用Laplacian锐化');
subplot(4,3,3);
imshow(uint8(abs(I_lap_aver)),[]),title('平滑锐化结果');

subplot(4,3,4);
imshow(I),title('原图');
subplot(4,3,5);
imshow(uint8(abs(I_log)),[]),title('LoG0.6');
subplot(4,3,6);
imshow(uint8(abs(I2)),[]),title('LoG平滑锐化图');

subplot(4,3,7);
```

```
imshow(I),title('原图');
subplot(4,3,8);
imshow(uint8(abs(I_log2)),[]),title('LoG2');
subplot(4,3,9);
imshow(uint8(abs(I3)),[]),title('LoG2平滑锐化图');

subplot(4,3,10);
imshow(I),title('原图');
subplot(4,3,11);
imshow(uint8(abs(I_log3)),[]),title('LoG4');
subplot(4,3,12);
imshow(uint8(abs(I4)),[]),title('LoG4平滑锐化图');
```

实验三：

```
%--------------------------------------------------------------------
%  程序: Image_HPF.m
%  作者:  郑品军
%  时间: 2019-12-30
%  说明:  数字图像RGB彩色图锐化实验，高通滤波法
% Revision History:  版本  /  归档日期  /   修改人  /  修改内容
%                                    V1.0 / 2019-12-30 /  郑品军  /  创建程序
%--------------------------------------------------------------------

clc;
close all;
clear all;

disp('RGB图像高通滤波器法锐化实验，图像尺寸为512*512*3，D_0 = 80，k_1 = 1.0，k_2 = 1.1')
user_choose = input('选择滤波器:\n 1.  理想高通滤波器  \n 2.  巴特沃斯高通滤波器\n 3.  高斯高通滤波器\n 9.  结束\n') ;

%% ---------------------------  高通滤波
o_figure = imread('0.tif');
o_figure = mat2gray(o_figure,[0 255]);
g1 = zeros(size(o_figure));
g2 = zeros(size(o_figure));

for RGB = 1:3
    f = o_figure(:,:,RGB);

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

    %  频谱中心化并做傅氏变换
    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);

    %  根据用户输入来生成对应的滤波器
    switch user_choose
        case 1
```

```
        % 生成理想高通滤波器
        H_HP = zeros(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 = 80;           % 阻带半径设置为80，图像矩阵512*512*3
                if D <= D_0
                    H_HP(x+(P/2)+1,y+(Q/2)+1) = 0;
                else
                    H_HP(x+(P/2)+1,y+(Q/2)+1) = 1;
                end
            end
        end

    case 2
        % 生成巴特沃斯高通滤波器
        H_HP = zeros(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 = 80;           % 阻带半径设置为80，图像矩阵512*512*3
                H_HP(x+(P/2)+1,y+(Q/2)+1) = 1 / (1+(D_0/D)^(2*P));
            end
        end

    case 3
        % 生成高斯高通滤波器
        H_HP = zeros(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 = 80;           % 阻带半径设置为80，图像矩阵512*512*3
                H_HP(x+(P/2)+1,y+(Q/2)+1) = 1 - exp(-(D*D)/(2*D_0*D_0));
            end
        end

    case 9
        return
    otherwise
        error('----------！！！无效输入！！！----------');
end

% 高斯滤波器滤波（得到边缘）
G_HP = H_HP .* F;
```

```
        % 锐化滤波（锐化增强）
        G_HFE = (1 + 1.1 * H_HP) .* F;

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

        g_2 = real(ifft2(G_HFE));
        g_2 = g_2(1:1:M,1:1:N);

        % 反频谱中心化
        for x = 1:1:M
            for y = 1:1:N
                g_1(x,y) = g_1(x,y) * (-1)^(x+y);
                g_2(x,y) = g_2(x,y) * (-1)^(x+y);
            end
        end

% g(:,:,RGB) = histeq(g_2);   % 直方图均衡化
        g1(:,:,RGB) = g_1;
        g2(:,:,RGB) = g_2;
end

%% ---------------------------- 作图
figure(1)
imshow(o_figure);
xlabel('(a). Original Image');

figure(2)
imshow(g1);
xlabel('(b). Result image of High-pass Filter');

figure(3)
imshow(g2);
xlabel('(c). Result image of High-Fre-Emphasis Filter');
```