专业：计算机科学与技术
姓名：
学号：
日期：2021/10/29

浙江大学实验报告

课程名称：图像信息处理 指导老师：宋明黎 成绩：

实验名称：均值滤波与拉普拉斯锐化

一、实验目的和要求

1. 均值滤波
2. 拉普拉斯锐化

二、实验内容和原理

1. 卷积

连续函数一维卷积公式如下：

The convolution $g(x)$ of two 1D functions $f(x)$ and $h(x)$

$$g(x)=f(x)*h(x)=\int_{-\infty}^{\infty}f(t)h(x-t)dt$$

而对于离散型函数的卷积公式如下：

$$g(x)=f(x)*h(x)=\frac{1}{M}\sum_{t=0}^{M-1}f(t)h(x-t)$$

在图像信息处理中，我们使用一个固定大小$M*M$的“卷积核”，将点(i,j)周围$M*M$大小的区域和卷积核对应元素相乘并相加得到的值填入(i,j)处，这就是图像的卷积操作。

2. 均值滤波

均值滤波的操作目的是去噪，它的方法就是将每一点用它周围点像素值的平均值来代替。均值滤波基于上述卷积操作，使用的卷积核为全是 1 的矩阵，为归一化，再全部除以矩阵元素的个数，即$\frac{1}{M*M}J$（其中M是矩阵的边长，J是全为 1 的矩阵）。当$M=3$的时候，这个卷积核如下：

$$\frac{1}{9}\times\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix}$$

3. 拉普拉斯锐化

拉普拉斯算子是 n 维欧几里得空间中的一个二阶微分算子，它的定义为：

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

离散形式下的拉普拉斯算子如下：

$$\nabla^2 f = \left[f(x+1,y) + f(x-1,y) + f(x,y+1) + f(x,y-1) \right] - 4f(x,y)$$

拉普拉斯算子体现了图像中灰度的突变，最后可以用矩阵表示出拉普拉斯算子：

0	1	0
1	-4	1
0	1	0

另外也可以用如下的形式：

1	1	1
1	-8	1
1	1	1

我们将得到的拉普拉斯图像与原图像对应的像素值相加即可得到最后生成的图像。

注意到：当邻域内像素灰度相同时，卷积结果为 0；当中心像素值高于邻域内其他像素平均灰度时，卷积结果为正；当中心像素灰度低于邻域其他像素平均灰度时候，卷积结果为负数，最后把卷积结果加到原中心像素，这样可以加剧中心像素与邻域像素灰度值的差异，因而达到锐化目的。

三、实验步骤与分析

1. 卷积

注意到这两个个图像几何操作都使用了卷积操作，所以笔者先实现了统一的卷积操作，该函数定义为 Convolution，需要的三个参数是原图像 bmpImage、卷积核 matrix、卷积核的 size。

该函数的主体部分如下：

```
for (i = 0; i < bmpFilter→bmpInfo→biHeight; i++) {
    for (j = 0; j < bmpFilter→bmpInfo→biWidth; j++) {
        startX = j - (size - 1) / 2;
        endX = j + size / 2;
        startY = i - (size - 1) / 2;
        endY = i + size / 2;
        for (k = 0; k < bmpFilter→bmpInfo→biBitCount / 8; k++) {
            result = 0;
            for (m = startY; m ≤ endY; m++) {
                for (n = startX; n ≤ endX; n++) {
                    _m = min(max(0, m), bmpFilter→bmpInfo→biHeight - 1);
                    _n = min(max(0, n), bmpFilter→bmpInfo→biWidth - 1);
                    result += bmpFilter→bmpData[_m * dataPerLine + _n * bmpFilter→bmpInfo→biBitCount / 8 + k]
                              * matrix[(m - startY) * size + n - startX];
                }
            }
            if (result > 255)
                result = 255;
            else if (result < 0)
                result = 0;
            bmpDataNew[i * dataPerLine + j * bmpFilter→bmpInfo→biBitCount / 8 + k] = result;
        }
    }
}
```

实现的方式就是将(i,j)周围的点和卷积核对应点相乘再相加。注意对边界的处理，笔者的框选的范围是$\left[i-\frac{(size-1)}{2},i+\frac{size}{2}\right]\times\left[j-\frac{size-1}{2},j+\frac{size}{2}\right]$，但是有可能会超出图像的范围，笔者希望实现 pad 的效果，最后就使用$_m$和$_n$两个变量，当超出范围的时候就用边界上的像素值来代替，相乘得到的结果加到$result$中，最后判断$result$是否超出了范围，截断后记录到$bmpDataNew$中，当所有都计算完后更新图片的图像数据。

2. 均值滤波

实现了上述卷积操作之后均值滤波简单的，只要将原理部分的矩阵代入即可。

```
Image *MeanFilter(Image *bmpImage, int size)
{
    int i;
    double *matrix = (double*)malloc(sizeof(double) * size * size); // mean matrix
    for (i = 0; i < size * size; i++) {
        matrix[i] = 1.0 / (size * size);
    }
    Image *bmpImageMF = Convolution(bmpImage, matrix, size);
    free(matrix);
    return bmpImageMF;
}
```

3. 拉普拉斯锐化

首先像上述均值滤波操作也对图像卷积，其中卷积核笔者最后选用了中心为 4 的拉普拉斯算子。

```
int i, j, k;
// double matrix[9] = {-1, -1, -1, -1, 8, -1, -1, -1, -1};
double matrix[9] = {0, -1, 0, -1, 4, -1, 0, -1, 0}; // laplace matrix
Image *bmpImageL = Convolution(bmpImage, matrix, 3);
```

接下来将原图像中的像素值加到拉普拉斯图像上，注意也要截断超过 255 和低于 0 的像素值。

```
// fuse the laplacian graph and the original graph
for (i = 0; i < bmpImage->bmpInfo->biHeight; i++) {
    for (j = 0; j < bmpImage->bmpInfo->biWidth; j++) {
        for (k = 0; k < bmpImage->bmpInfo->biBitCount / 8; k++) {
            result = bmpImage->bmpData[i * dataPerLine + j * bmpImageL->bmpInfo->biBitCount / 8 + k]
                   + bmpImageL->bmpData[i * dataPerLine + j * bmpImageL->bmpInfo->biBitCount / 8 + k];
            if (result > 255)
                result = 255;
            else if (result < 0)
                result = 0;
            bmpImageL->bmpData[i * dataPerLine + j * bmpImageL->bmpInfo->biBitCount / 8 + k] = result;
        }
        }
    }
return bmpImageL;
```

四、实验环境及运行方法

编译环境：

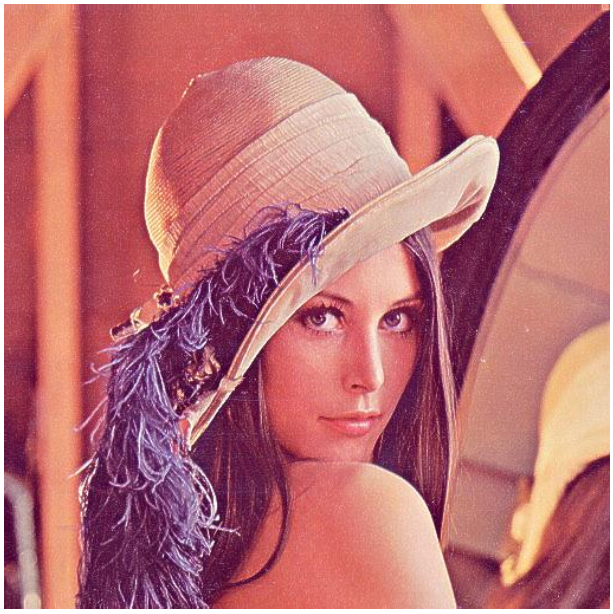
gcc 6.3.0、Windows 11 Insider Preview 22483.1011

测试方法：

在命令行中输入 gcc image.c main.c，然后运行.\a.exe。输入和输出图像都放在了 output 文件夹，输入为 24 位 BMP 文件，文件名为 input.bmp，输出的图片有均值滤波图片 outputMF.bmp 和拉普拉斯锐化图像 outputL.bmp。

五、实验结果展示

24 位 BMP 图	
均值滤波	

拉普拉斯锐化	

六、心得体会

本次实验相对比较顺利，但也遇到了一些 BUG，特别是卷积核奇偶性的问题，笔者一开始没有比较好地处理点附近窗口的大小，导致偶数 size 的卷积核会导致与原图像窗口 size 不匹配的情况，在一步步地调试中最终发现了问题。在本次实验中我对图片的卷积有了更深入的了解，之前也学习和应用过卷积，但通过 C 的实现让我有更直观的感受。