



# 图像信息处理实验报告

**Digital Image Processing (Experiments)**

|  |  |
| --- | --- |
| 姓 名 |  |
| 指导老师 |  |
| 学 号 |  |
| 专业班级 |  |

**二〇一九年**

**秋冬学期**

## 实验五

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称：** | 作业5 | **指导老师：** |  | **成绩：** |  |

### 一、实验目的和要求（必填）

1、图像均值滤波（Image mean filtering）

2、拉普拉斯边缘检测与图像增强（Laplacian image enhancement）

### 二、实验内容和原理（必填）

#### 1、空间滤波原理

对于图像中的每个像素（x, y），根据过滤器中元素之间的预定义关系计算其对过滤器的响应值。对于空间线性滤波，响应值是通过将系数与其对应像素的乘积相加来计算的。

滤波原理：

#### 2、图像线性平滑滤波

滤波器w的大小为（2a+1）×（2b+1），f为输入图像，g为输出图像。w的大小对最终结果非常重要。当遮罩很小时，模糊效果非常微妙，反之亦然。

#### 3、拉普拉斯边缘检测

对于函数f(x,y)，拉普拉斯算子定义为：

拉普拉斯算子在图像像素上的表现：

x轴：

y轴：

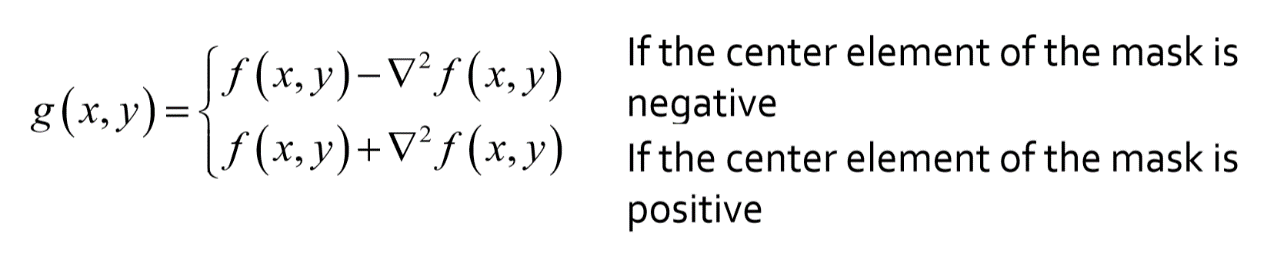
离散化：

|  |  |  |
| --- | --- | --- |
| 0 | 1 | 0 |
| 1 | -4 | 1 |
| 0 | 1 | 0 |

滤波器的拓展：

|  |  |  |
| --- | --- | --- |
| 1 | 1 | 1 |
| 1 | -8 | 1 |
| 1 | 1 | 1 |

#### 4、拉普拉斯图像增强

融合原始图像和拉普拉斯结果可以保持锐化效果，恢复原始视觉信息。

### 三、源代码与分析

#### 1、主函数（main）

int main(int argc, char\*argv[]) {

    unsigned char\* file\_temp;

    FILE\* fpbmp;

    FILE\* fpout;

    fpbmp = openFile("input.bmp");      //打开原图片

    fpout = writeFile("meanfiltering.bmp"); //分别输出meanfiltering，laplacian,laplacian\_enhancement

    readBMP(fpbmp);

    addHeader(fpbmp, fpout);

    //RGBtoYUV();

    //\*-----------------my functions--------------\*//

    //MeanFiltering();  //均值滤波函数

    //Laplacian();      //拉普拉斯变换获得边缘图像

    //Laplacian\_enhencement();  //拉普拉斯图像增强

    //YUVtoRGB();

    outBMP(fpout);

    fclose(fpbmp);

    fclose(fpout);

}

#### 2、图像均值滤波（Image mean filtering）

void MeanFiltering() {

    int num\_pixs = 16;      //定义mask大小，num\_pix为mask像素

    for (int j = 0; j < height; j++) {

        for (int i = 0; i < width; i++) {

            if (i<=num\_pixs/2 || j <=num\_pixs/2 || i >= width-1-num\_pixs/2 || j == height-1-num\_pixs/2) {

                out\_R[j][i] = R[j][i];

                out\_G[j][i] = G[j][i];

                out\_B[j][i] = B[j][i];  //输出边缘保持原图像不变

            }

            else {

                int totalR, totalG, totalB,totalY;

                totalR = 0; totalG = 0; totalB = 0; totalY = 0;

                int k = 0;

                for (int ki = 0; ki < num\_pixs; ki++) {

                    for (int kj = 0; kj < num\_pixs; kj++) {

                        totalR = totalR + R[j + kj-num\_pixs/2][i + ki-num\_pixs/2];

                        totalG = totalG + G[j + kj-num\_pixs/2][i + ki-num\_pixs/2];

                        totalB = totalB + B[j + kj-num\_pixs/2][i + ki-num\_pixs/2];  //计算mask大小内原图像rgb各值之和

                        k++;

                    }

                }

                out\_R[j][i] = totalR / (num\_pixs\*num\_pixs);

                out\_G[j][i] = totalG / (num\_pixs\*num\_pixs);

                out\_B[j][i] = totalB / (num\_pixs\*num\_pixs);     //图像均值滤波变化

            }

        }

    }

}

#### 3、拉普拉斯边缘检测与图像增强（Laplacian image enhancement）

void Laplacian() {

    for (int j = 0; j < height; j++) {

        for (int i = 0; i < width; i++) {

            if (i <= 0 || j <= 0 || i == width - 1 || j == height ) {

                out\_R[j][i] = R[j][i];

                out\_G[j][i] = G[j][i];

                out\_B[j][i] = B[j][i];

            }

            else {

                out\_R[j][i] = abs(R[j-1][i-1]+ R[j - 1][i + 1] + R[j + 1][i - 1] + R[j + 1][i + 1] + R[j - 1][i] + R[j + 1][i] + R[j][i - 1] + R[j][i + 1] - 8 \* R[j][i]);

                out\_G[j][i] = abs(G[j - 1][i - 1] + G[j - 1][i + 1] + G[j + 1][i - 1] + G[j + 1][i + 1]+G[j - 1][i] + G[j + 1][i] + G[j][i - 1] + G[j][i + 1] - 8 \* G[j][i]);

                out\_B[j][i] = abs(B[j - 1][i - 1] + B[j - 1][i + 1] + B[j + 1][i - 1] + B[j + 1][i + 1]+B[j - 1][i] + B[j + 1][i] + B[j][i - 1] + B[j][i + 1] - 8 \* B[j][i]);

            }

        }

    }

}

void Laplacian\_enhencement() {

    int ddr, ddg, ddb,judge;

    judge = 0;

    float index;//拉普拉斯图像增强算子系数

    index = 0.2;

    for (int j = 0; j < height; j++) {

        for (int i = 0; i < width; i++) {

            if (i <= 0 || j <= 0 || i == width - 1 || j == height) {

                out\_R[j][i] = R[j][i];

                out\_G[j][i] = G[j][i];

                out\_B[j][i] = B[j][i];

            }

            else {

                ddr = abs(R[j - 1][i - 1] + R[j - 1][i + 1] + R[j + 1][i - 1] + R[j + 1][i + 1] + R[j - 1][i] + R[j + 1][i] + R[j][i - 1] + R[j][i + 1] - 8 \* R[j][i]);

                ddg = abs(G[j - 1][i - 1] + G[j - 1][i + 1] + G[j + 1][i - 1] + G[j + 1][i + 1] + G[j - 1][i] + G[j + 1][i] + G[j][i - 1] + G[j][i + 1] - 8 \* G[j][i]);

                ddb = abs(B[j - 1][i - 1] + B[j - 1][i + 1] + B[j + 1][i - 1] + B[j + 1][i + 1] + B[j - 1][i] + B[j + 1][i] + B[j][i - 1] + B[j][i + 1] - 8 \* B[j][i]);

                if (ddr != 0)

                    judge = 1;

                else

                    judge = 0;

                out\_R[j][i] =( R[j][i] + index\*abs(ddr))>255?255: (R[j][i] + index\*abs(ddr));

                out\_G[j][i] =( G[j][i] + index\*abs(ddg))>255?255: (G[j][i] + index\*abs(ddr));

                out\_B[j][i] =( B[j][i] + index\*abs(ddb))>255?255: (B[j][i] + index\*abs(ddr));

                //存在输出值计算所得大于255，避免出现溢出情况

            }

        }

    }

}

#### 4、结果

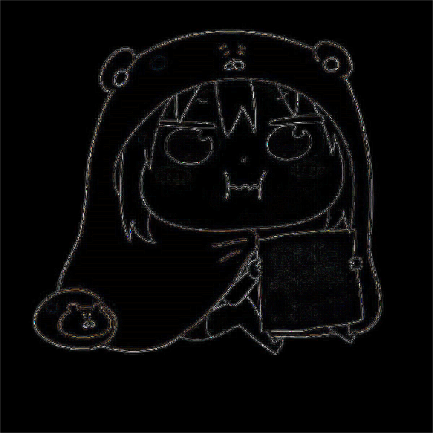
【原图】



【meanfiltering】num\_pixs=9 / 16 /25



【laplacian】



【laplacian\_enhancement】



【原图】

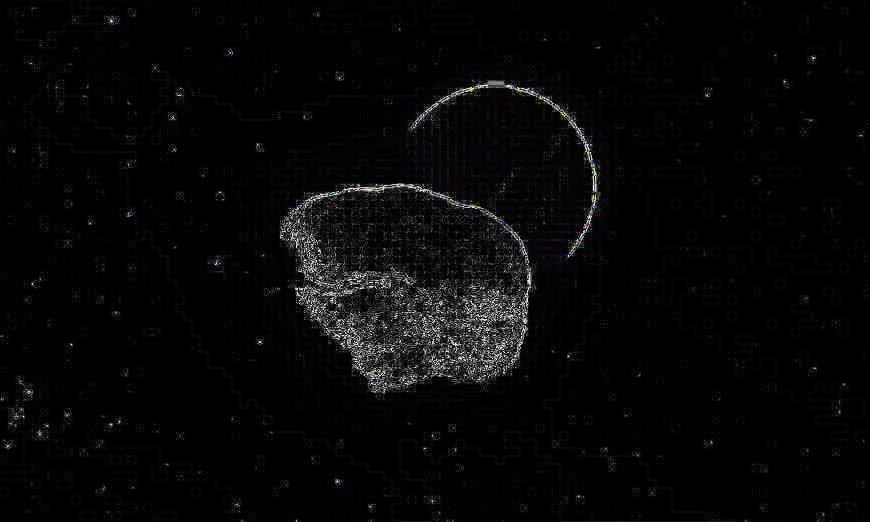


【meanfiltering】



【laplacian】

操作后发现背景中的星点明显增强，暗部细节突出



【laplacian\_enhancement】



### 四、心得体悟

1、均值滤波可以将图像模糊化，mask大小的不同会导致图片模糊程度改变，运用在人像上可以帮助皮肤平滑、减少斑点等细节

2、拉普拉斯图像增强是基于拉普拉斯边缘检测函数的基础之上的，通过滤波可以将图像暗部极小处的细节增强，图像边缘得到锐化。