



# 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**

## 实验六

|  |  |  |  |  |  |
| --- | --- | --- | --- | --- | --- |
| **实验名称：** | 作业6 | **指导老师：** |  | **成绩：** |  |

### 一、实验目的和要求（必填）

1、双边过滤的强力实现(Brute-force Implementation of Bilateral Filtering)

### 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、图像双边滤波

双边滤波（Bilateral filter）是一种非线性的滤波方法，是结合图像的空间邻近度和像素值相似度的一种折中处理，同时考虑空域信息和灰度相似性，达到保边去噪的目的。

双边滤波器之所以能够做到在平滑去噪的同时还能够很好的保存边缘（Edge Preserve），是由于其滤波器的核由两个函数生成：

·一个函数由像素欧氏距离决定滤波器模板的系数（距离模板系数d）

其中,(k,l)为模板窗口的中心坐标；(i,j)为模板窗口的其他系数的坐标；σ为高斯函数的标准差。

·另一个函数由像素的灰度差值决定滤波器的系数（值域模板系数r）

其中，函数f(x,y)表示要处理的图像，表示图像在点(x,y)处的像素值；(k,l)为模板窗口的中心坐标；(i,j)为模板窗口的其他系数的坐标；σ为高斯函数的标准差。

其综合了高斯滤波器（Gaussian Filter）和α-截尾均值滤波器（Alpha-Trimmed mean Filter）的特点。高斯滤波器只考虑像素间的欧式距离，其使用的模板系数随着和窗口中心的距离增大而减小；Alpha截尾均值滤波器则只考虑了像素灰度值之间的差值，去掉α%的最小值和最大值后再计算均值。

### 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("bilateral\_timg.bmp");     //打开原图片

    fpout = writeFile("bilateral\_timg2.bmp");   //分别输出双边滤波结果

    //实验发现多次进行双边滤波，效果较佳

    readBMP(fpbmp);

    addHeader(fpbmp, fpout);

    //RGBtoYUV();

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

    Bilateral\_Filtering();//进行双边滤波

    //YUVtoRGB();//本次实验中不需要进行yuv转换

    outBMP(fpout);

    fclose(fpbmp);

    fclose(fpout);

}

#### 2、图像双变滤波（bilateral filtering）

void  Bilateral\_Filtering() {

    float wr,wg,wb; //权重系数w

    float d;    //定义域核d

    float rr,rg,rb; //值域核r

    int i, j;   //目标像素位置

    int k, l;   //mask各个像素位置

    int num\_pixs;   //mask边长

    num\_pixs = 3;

    for (j = 0; j < height; j++) {

        for (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 {

                float fw\_r, fw\_g, fw\_b;

                float totalw\_r, totalw\_g, totalw\_b;

                fw\_r = 0; fw\_g = 0; fw\_b = 0;

                totalw\_r = 0; totalw\_g = 0; totalw\_b = 0;

                for (l = j - num\_pixs / 2; l <= j + num\_pixs / 2; l++) {

                    for (k = i - num\_pixs / 2; k <= i - num\_pixs / 2; k++) {

                        d = exp(-((l - j) ^ 2 + (k - i) ^ 2)\*1.0/ (2 \* num\_pixs\*num\_pixs));

                        rr = exp(-((R[j][i] - R[l][k]) ^ 2)\*1.0 / (2 \* num\_pixs\*num\_pixs));

                        rg = exp(-((G[j][i] - G[l][k]) ^ 2)\*1.0 / (2 \* num\_pixs\*num\_pixs));

                        rb = exp(-((B[j][i] - B[l][k]) ^ 2)\*1.0 / (2 \* num\_pixs\*num\_pixs));

                        wr = d \* rr;

                        wg = d \* rg;

                        wb = d \* rb;

                        fw\_r = fw\_r + R[l][k] \* wr;

                        fw\_g = fw\_g + G[l][k] \* wg;

                        fw\_b = fw\_b + B[l][k] \* wb;

                        totalw\_r = totalw\_r + wr;

                        totalw\_g = totalw\_g + wg;

                        totalw\_b = totalw\_b + wb;

                    }

                }

                out\_R[j][i] = (int)(fw\_r / totalw\_r);

                out\_G[j][i] = (int)(fw\_g / totalw\_g);

                out\_B[j][i] = (int)(fw\_b / totalw\_b);

            }

        }

    }

}

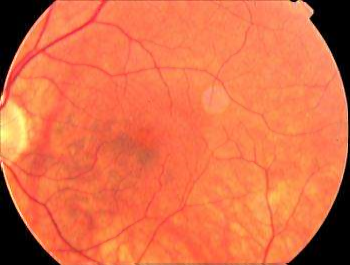
#### 3、结果

【原图】

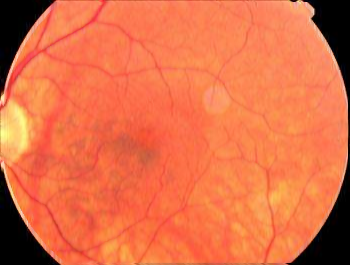


【bilateral filtering】num\_pixs=3

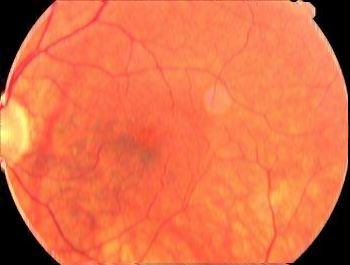
【第一次bilateral filtering】



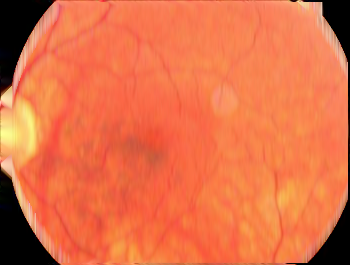
【第二次bilateral filtering】



【第三次bilateral filtering】

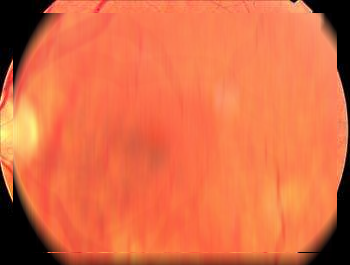


【第十次bilateral filtering】



### 四、心得体悟

1、双边滤波可以将图像保边的同时模糊化，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；若mask过大，可能会导致图像边缘的波动，造成失真，如图：



产生原因可能是在进行双边滤波时对原图像四周进行了保边操作，与滤波后图像无法平滑连结。

2、对于图像重复进行多次实验，会得到较为完美的结果图。