

浙江大学

图像双边滤波实验报告



姓名：欧翌昕
专业：软件工程
学号：3190104783
课程名称：图像信息处理
指导老师：宋明黎

2020~2021秋冬学期 2020 年 12 月 30 日

1 实验目的和要求

- 图像双边滤波

2 实验内容和原理

双边滤波是一种非线性滤波器，它可以达到保持边缘、降噪平滑的效果。和其他滤波原理一样，双边滤波也是采用加权平均的方法，用周边像素亮度值的加权平均代表某个像素的强度，所用的加权平均基于高斯分布。

最重要的是，双边滤波的权重不仅考虑了像素的欧氏距离（如普通的高斯低通滤波，只考虑了位置对中心像素的影响），还考虑了像素范围域中的辐射差异（例如卷积核中像素与中心像素之间相似程度、颜色强度，深度距离等），在计算中心像素的时候同时考虑这两个权重。

双边滤波的公式：

$$\bar{I}(p) = \frac{1}{W_p} \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I(p) - I(q)|) I(q)$$

其中 W_p 表示为：

$$W_p = \sum_{q \in S} G_{\sigma_s}(\|p - q\|) G_{\sigma_r}(|I(p) - I(q)|)$$

双边滤波中有两个衡量图像信息的核心变量，如上式中的 G_{σ_s} 为空间域核， G_{σ_r} 为图像像素域核。空间域核是二维高斯函数，可以把它视作高斯滤波，像素域核就是衡量像素变化剧烈程度的量。将这两个变量相乘，就可以得到他们俩共同作用的结果：在图像的平坦区域，像素值变化很小，对应的像素范围域权重接近于1，此时空间域权重起主要作用，相当于进行高斯模糊；在图像的边缘区域，像素值变化很大，像素范围域权重变大，从而保持了边缘的信息。

空间域核的计算方法为：

$$G_{\sigma_s}(\|p - q\|) = e^{-\frac{(i-m)^2 + (j-n)^2}{2\sigma_s^2}}$$

像素域核的计算方法为：

$$G_{\sigma_r}(|I(p) - I(q)|) = e^{-\frac{[I(i,j) - I(m,n)]^2}{2\sigma_r^2}}$$

在上面两个式子中， σ_s 与 σ_r 都是已知的，或者说是自己输入的预设值，而其他的*i, j, m, n*都是需要在遍历中确定的值。其中(*i, j*)代表是窗口中心值，(*m, n*)代表的是滑动窗口中的某个值。

3 实验步骤和分析

图双边滤波的过程如下：

读取原始图像。

```
1 BITMAPFILEHEADER fileHeader; //位图文件头
2 BITMAPINFOHEADER infoHeader; //位图信息头
3 FILE* pic1= fopen(input, "rb"); //打开输入文件
4 if (pic1 == NULL)
5 {
6     printf("Open file failed!\n");
7     exit(-1);
8 }
9
10 fread(&fileHeader, sizeof(BITMAPFILEHEADER), 1, pic1); //读取
   文件头
11 fread(&infoHeader, sizeof(BITMAPINFOHEADER), 1, pic1); //读取
   信息头
12 WORD bitCount = infoHeader.biBitCount; //颜色位数
13 if (bitCount != 24)
14 {
15     printf("Only 24 bit color map is allowed!\n");
16     exit(-1);
17 }
18
19 LONG w = infoHeader.biWidth; //图像数据的宽度
20 LONG h = infoHeader.biHeight; //图像数据的高度
21
22 int bytesPerLine = ((w * bitCount+31)>>5)<<2; //行数据大小
23 int imageSize = bytesPerLine*h; //图像数据大小
24 int skip = 4-((w * bitCount)>>3)&3; //需跳过的字节
25
26 BYTE* data = (BYTE*)malloc(imageSize); //申请存储图像数据的空间
27 fread(data, imageSize, 1, pic1); //记录图像数据
```

设定半径大小，设定空间函数方差和相似函数方差，分别计算空间权重以及相似权重。

```
1 int r = 7;
2
3 double sigma_s = 1000; //空间函数方差
4 double sigma_r = 10; //相似函数方差
5 double s_coef = 0.5/(sigma_s*sigma_s);
6 double r_coef = 0.5/(sigma_r*sigma_r);
7
8 double s_w[200][200]; //空间权重
9 double r_w[256]; //相似权重
10
11 int i, j;
12
13 //计算空间权重
14 for(i = -r; i <= r; i++)
15 {
16     for(j = -r; j <= r; j++)
17     {
18         int x = i+r;
19         int y = j+r;
20         s_w[x][y] = exp(-(i*i+j*j)*s_coef);
21     }
22 }
23
24 //计算相似权重
25 for(i = 0; i < 256; i++)
26 {
27     r_w[i] = exp(-i*i*r_coef);
28 }
```

根据空间权重以及相似权重计算出复合权重作为特征值。将每一点的特征值与该点像素值乘积相加，除以所有特征值的和，得到的结果即作为该点的像素值。

```
1 //双边滤波
2 int k, m, n;
3 for (i = 0; i < h; i++)
4 {
5     for (j = 0; j < w; j++)
6     {
7         for (k = 0; k < 3; k++)
8         {
```

```
9         double weightSum = 0, dataSum = 0;
10
11     for (m = -r; m <= r; m++)
12     {
13         for (n = -r; n <= r; n++)
14         {
15             if (m * m + n * n > r * r) continue;
16
17             int H = i + m;
18             int W = j + n;
19
20             if (H < 0) H = 0;
21             if (H > h-1) H = h-1;
22             if (W < 0) W = 0;
23             if (W > w-1) W = w-1;
24
25             int d = abs(data[H*3*W+W*3+k] -
26             data[i*3*W+j*3+k]);
27
28             double weight = s_w[m+r][n+r]*r_w[d];
//复合权重
29             dataSum += data[H*3*W+W*3+k]*weight;
30             weightSum += weight;
31         }
32         double final = dataSum/weightSum;
33         data[i*3*W+j*3+k] = (BYTE)final;
34     }
35 }
36 }
```

4 实验结果

4.1 原始图像



4.2 灰度图像



4.3 双边滤波结果图像



5 心得体会

通过比较灰度图像与双边滤波结果图像，我们可以发现，双边滤波结果图像中的噪点减少了许多，整幅图像看起来更加平滑细腻。但本次实验中依然存在一些问题，比如未在RGB三个通道上均进行双边滤波操作，而只是实现了灰度图像的双边滤波。除此之外，有关于系数 σ_s 与 σ_r 的选取，也会对结果图像效果造成较大影响，若将 σ_s 与 σ_r 设置大些，结果图像会比较模糊，丢失细节信息。总体来说，通过本次实验，我对图像双边滤波有了更清晰的认识，在对权重的处理上也有了自己的思考，收获很大。