**实验报告**

专业：计算机科学与技术

姓名：薛柔

学号：3220104854

日期：2023/10/29

课程名称： 图像信息处理 指导老师： 宋明黎 成绩：

实验名称： 暴力法实现双边滤波

**一、实验目的和要求**

1. 理解双边滤波的实现原理

2. 能够用暴力法实现双边滤波操作

**二、实验内容和原理**

实验内容：

暴力法实现双边滤波

实验原理：

双边滤波

1. 基本思想

图像有两个主要特征：

* 空间域 S，它是图像中可能位置的集合。这与分辨率有关，即图像中的行数和列数。
* 强度域 R，它是可能的像素值的集合。用于表示像素值的位数可能会有所不同。常见的像素表示形式是无符号字节（0 到 255）和浮点数。

每个样本都被其相邻样本的加权平均值所取代，这些权重反映了两方面的影响：

* 相邻样本和中心样本的距离有多近，因此对较近样本的权重越大。
* 相邻样本和中心样本的相似程度，相似样本的权重更大。

所有权重都应归一化以保持局部均值。（例如保持在0-255之内）

2. 具体形式

回顾高斯滤波，相当于用下面的函数和滤波器对图像进行卷积。它使图像平滑

但也会使边缘模糊。

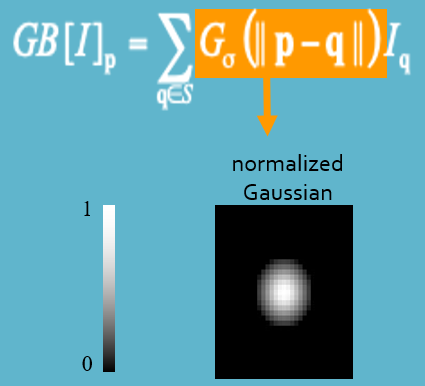
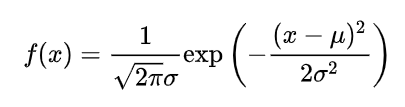


图2.1 高斯滤波

其中，G(x)为高斯函数：



把强度域纳入考虑，即可得双边滤波的计算方式：

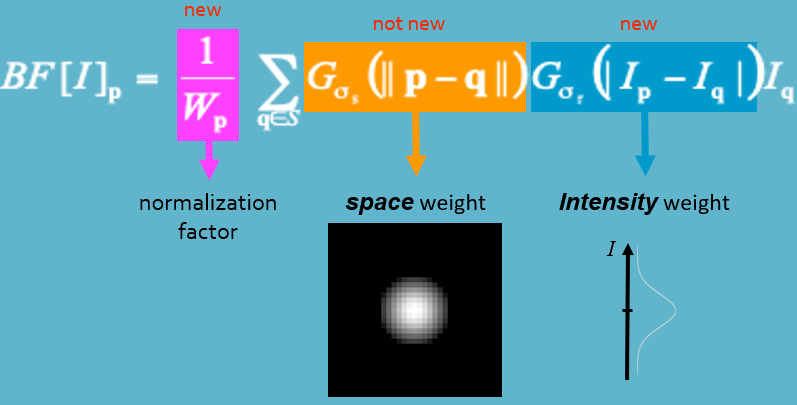


图2.2 双边滤波

其中ss内是核的空间范围，考虑邻域的大小。sr是边的振幅范围。这样与中心像素差异大的点的权重就会很小，边得到了保留。

下面用示例图像展示参数的影响：

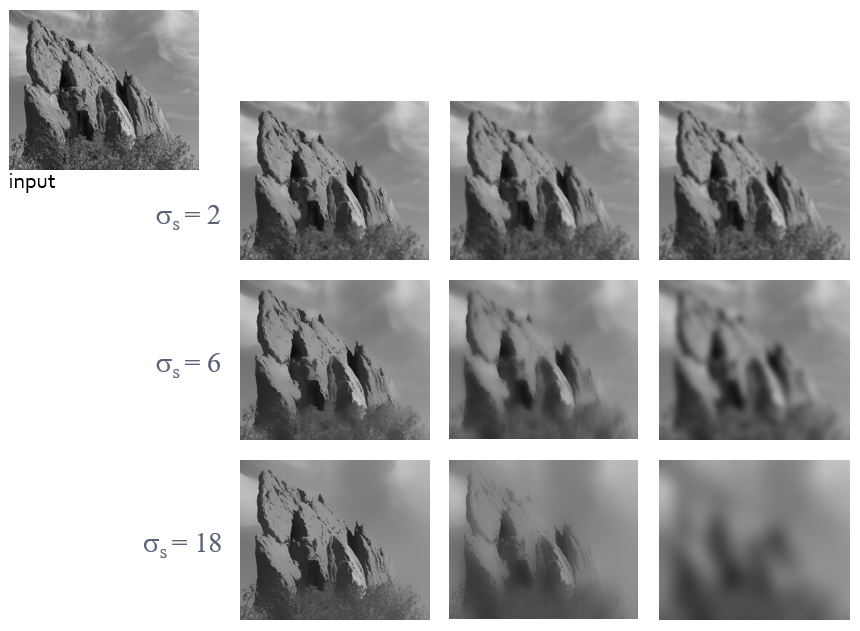


图2.2 双边滤波参数的影响

参数的选择上，取决于应用。例如：

* 空间参数：与图像大小成正比（例如，图像对角线的 2%）
* 强度参数：与边缘振幅成正比（例如，图像梯度的平均值或中位数）

而这与分辨率和曝光无关。

对于彩色图像，强度可以用色彩空间的欧氏距离替代。

**三、实验步骤与分析**

1. 灰度图像的双边滤波

下面是计算权重的高斯函数，分别根据距离和强度，用原理中提到的高斯函数算出权重并返回。这在对灰度图像和彩色图像的操作中能通用。

1. **double** Guass\_p(**int** i1,**int** j1,**int** i2,**int** j2,**double** sigma)
2. {
3. //根据距离求高斯函数值
4. //x:高斯函数的距离,这里为简便计算x^2
5. **double** x2=(i1-i2)\*(i1-i2)+(j1-j2)\*(j1-j2);
6. **return** 1/sigma/2.5\*exp(-x2/(2\*sigma\*sigma));//高斯函数的值
7. }
8. **double** Guass\_i(PIXEL p1,PIXEL p2,**double** sigma)
9. {
10. //根据灰度/颜色求高斯函数值
11. //x:高斯函数的距离,这里为简便计算x^2
12. //色彩空间的欧氏距离的平方
13. **double** x2= (p1.blue-p2.blue)\*(p1.blue-p2.blue)+
14. (p1.green-p2.green)\*(p1.green-p2.green)+(p1.red-p2.red)\*(p1.red-p2.red);
15. **return** 1/sigma/2.5\*exp(-x2/(2\*sigma\*sigma));//高斯函数的值
16. }

下面是对灰度图像进行双边滤波的函数，a[][]中是原始图像数据。前半部分先设定好空间系数的值以及窗口大小：对角线的2%除此之外。并对每一个像素周边的小窗口计算强度系数：梯度值的平均（注意取绝对值）。最终可能由于高斯函数系数的不同，需要调整一下强度系数的大小，将其乘以5。

1. **void** bilateral\_gray(PIXEL \*\*a)
2. {
3. PIXEL \*new\_pixel[bmih.biHeight];
4. **for** (**int** i=0;i<bmih.biHeight;i++){
5. new\_pixel[i] = (PIXEL\*)malloc(**sizeof**(PIXEL)\*(bmih.biWidth));
6. }
7. **double** sigma\_s,sigma\_r;
8. //sigma\_s为对角线距离的2%
9. sigma\_s=0.02\*sqrt(bmih.biHeight\*bmih.biHeight+bmih.biWidth\*bmih.biWidth);
10. //窗口大小
11. **int** size=sigma\_s;
12. **for**(**int** i=0;i<bmih.biHeight;i++){
13. **for**(**int** j=0;j<bmih.biWidth;j++){
14. //对每一个窗口，计算sigma\_r
15. //经验公式：sigma\_r为窗口梯度的均值
16. **double** sum\_gradient=0;
17. **int** cnt=0;//有效像素数
18. **for**(**int** m=i-size/2;m<=i+size/2;m++){
19. **for**(**int** n=j-size/2;n<=j+size/2;n++){
20. //梯度等于灰度值变化比距离
21. **if**(m>=0&&n>=0&&m<bmih.biHeight&&n<bmih.biWidth&&(m!=i||n!=j)){
22. **double** y=fabs((a[m][n].blue-a[i][j].blue));
23. **double** x=sqrt((m-i)\*(m-i)+(n-j)\*(n-j));
24. sum\_gradient+=y/x;
25. cnt++;
26. }
27. }
28. }
29. sigma\_r=sum\_gradient/cnt\*5;//sigma\_r为窗口梯度的均值

下面再根据高斯函数返回的权值，对当前像素周围像素的灰度进行加权平均，并赋给新图像。其中，对于强度系数为0的像素，周围梯度均为0，即周围像素强度相同，将原来的灰度值赋给新像素即可，避免在高斯函数中用0作为除数。

1. //计算窗口中心新的像素值
2. **double** temp\_i=0;
3. **double** sum\_weight=0;//归一化需要的权重值
4. **if**(sigma\_r==0){
5. //防止0作为除数
6. //梯度总为0，周围像素相同
7. temp\_i=a[i][j].blue;
8. }**else**{
9. **for**(**int** m=i-size/2;m<=i+size/2;m++){
10. **for**(**int** n=j-size/2;n<=j+size/2;n++){
11. //计算加权和
12. **if**(m>=0&&n>=0&&m<bmih.biHeight&&n<bmih.biWidth){
13. **double** weight\_s=Guass\_p(i,j,m,n,sigma\_s);
14. **double** weight\_i=Guass\_i(a[i][j],a[m][n],sigma\_r);
15. sum\_weight+=weight\_i\*weight\_s;
16. temp\_i+=a[m][n].blue\*weight\_i\*weight\_s;
17. }
18. }
19. }
20. temp\_i=temp\_i/sum\_weight;
21. }
23. new\_pixel[i][j].red=new\_pixel[i][j].blue=new\_pixel[i][j].green=temp\_i;
24. }
25. }
26. //将图输出
27. //释放内存
28. }

2. 彩色图像的双边滤波

与灰度图像大同小异：只需将用到灰度值的地方改为色彩空间的欧氏距离。高斯函数使用的强度差即为两个色彩向量差的模。

1. **void** bilateral\_color(PIXEL \*\*a)
2. {
3. PIXEL \*new\_pixel[bmih.biHeight];
4. **for** (**int** i=0;i<bmih.biHeight;i++){
5. new\_pixel[i] = (PIXEL\*)malloc(**sizeof**(PIXEL)\*(bmih.biWidth));
6. }
7. **double** sigma\_s,sigma\_r;
8. //sigma\_s为对角线距离的2%
9. sigma\_s=0.02\*sqrt(bmih.biHeight\*bmih.biHeight+bmih.biWidth\*bmih.biWidth);
10. //窗口大小
11. **int** size=sigma\_s;
12. **for**(**int** i=0;i<bmih.biHeight;i++){
13. **for**(**int** j=0;j<bmih.biWidth;j++){
14. //对每一个窗口，计算sigma\_r
15. //经验公式：sigma\_r为窗口梯度的均值
16. **double** sum\_gradient=0;
17. **int** cnt=0;//有效像素数
18. **for**(**int** m=i-size/2;m<=i+size/2;m++){
19. **for**(**int** n=j-size/2;n<=j+size/2;n++){
20. //梯度等于色彩空间距离变化比距离
21. **if**(m>=0&&n>=0&&m<bmih.biHeight&&n<bmih.biWidth&&(m!=i||n!=j)){
22. **double** y=sqrt((a[m][n].blue-a[i][j].blue)\*(a[m][n].blue-a[i][j].blue)+(a[m][n].red-a[i][j].red)\*
23. (a[m][n].red-a[i][j].red)+(a[m][n].green-a[i][j].green)\*(a[m][n].green-a[i][j].green));
24. **double** x=sqrt((m-i)\*(m-i)+(n-j)\*(n-j));
25. sum\_gradient+=y/x;
26. cnt++;
27. }
28. }
29. }
30. sigma\_r=sum\_gradient/cnt\*2.5;//sigma\_r为窗口梯度的均值
32. //计算窗口中心新的像素值
33. **double** temp\_r=0;
34. **double** temp\_g=0;
35. **double** temp\_b=0;
36. **double** sum\_weight=0;//归一化需要的权重值
37. **if**(sigma\_r==0){
38. //防止0作为除数
39. //梯度总为0，周围像素相同
40. temp\_r=a[i][j].red;
41. temp\_g=a[i][j].green;
42. temp\_b=a[i][j].blue;
43. }**else**{
44. **for**(**int** m=i-size/2;m<=i+size/2;m++){
45. **for**(**int** n=j-size/2;n<=j+size/2;n++){
46. //计算加权和
47. **if**(m>=0&&n>=0&&m<bmih.biHeight&&n<bmih.biWidth){
48. **double** weight\_s=Guass\_p(i,j,m,n,sigma\_s);
49. **double** weight\_i=Guass\_i(a[i][j],a[m][n],sigma\_r);
50. sum\_weight+=weight\_i\*weight\_s;
51. temp\_r+=a[m][n].red\*weight\_i\*weight\_s;
52. temp\_g+=a[m][n].green\*weight\_i\*weight\_s;
53. temp\_b+=a[m][n].blue\*weight\_i\*weight\_s;
54. }
55. }
56. }
57. temp\_r=temp\_r/sum\_weight;
58. temp\_g=temp\_g/sum\_weight;
59. temp\_b=temp\_b/sum\_weight;
60. }
62. new\_pixel[i][j].red=temp\_r;
63. new\_pixel[i][j].blue=temp\_b;
64. new\_pixel[i][j].green=temp\_g;
65. }
66. }
67. //将图输出
68. //释放内存
69. }

**四、实验环境及运行方法**

* 实验环境：Windows 11 系统

gcc 10.3.0 (tdm64-1) x86\_64-w64-mingw32

* 运行方法：先将dip\_hw6.exe文件和42.bmp放在同一个目录下，运行dip\_hw6.exe文件，等待30s左右会得到2张新的图像。它们的命名分别Bilateral\_color.bmp和Bilateral\_gray.bmp，分别是对原始图像进行双边滤波和对灰度化后的图像进行双边的结果。当终端出现 “Successfully open the image”时说明我们成功打开了原始图像，否则会输出“BMP Image Not Found!”。

如果运行不成功，可以将文件夹中的.c源文件重新编译运行。

**五、实验结果展示**



图5.1 输入图像42.bmp

下面的九张图，从左到右强度参数依次增大，从上到下空间参数依次增大，同一行列的另一参数相同。

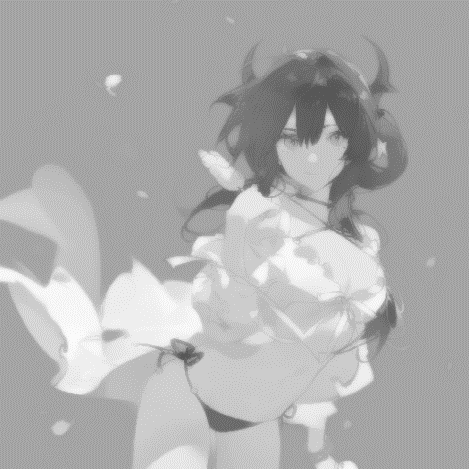
  

图5.2 彩色图像结果

下图为拉普拉斯增强的过程图，对于彩色图，我在三个色彩通道中分别增强。

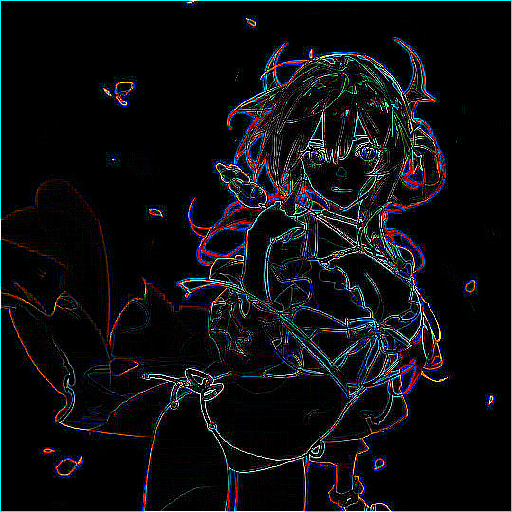
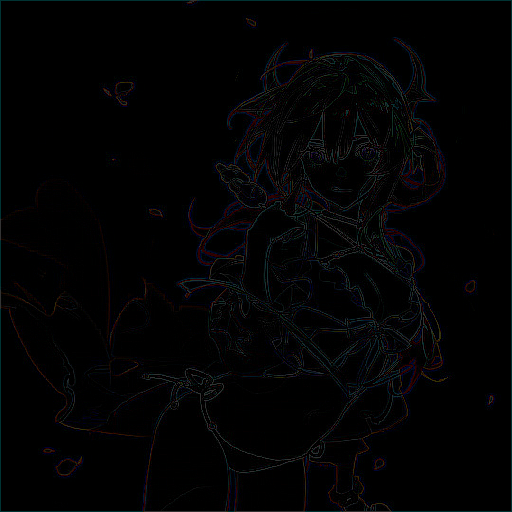
 

图5.3 计算拉普拉斯算子后的中间图Laplacian\_result.bmp（左）

和Rearrange后的rearranged\_Laplacian\_result.bmp（右）

将输入图像与增强结果放在一起作对比。可以看到，结果图像明显锐化，线条更加明显，达到了我们想要的效果。

图5.4 输入图像42.bmp（左）和拉普拉斯增强后的图像Laplacian\_enhance.bmp（右）

图5.5 灰度图像计算拉普拉斯算子后的中间图Laplacian\_result.bmp（左）

和Rearrange后的rearranged\_Laplacian\_result.bmp（右）

图5.6 灰度输入图像gray.bmp（左）和拉普拉斯增强后的图像Laplacian\_enhance.bmp（右）

**六、心得体会**

本次还有很多可以额外探索的地方：例如如何找到合适的遮罩大小、改变系数得到更好滤波效果。可以看到由于我的原始图像相较而言不算很小，适当增大滤波器的大小可以获得更好的模糊和锐化效果。值得一提的是，尽管展示图片在报告pdf中的对比不强烈，但用图片查看器看我的锐化结果图像，效果是较为明显的。

只要掌握基本原理，无论是均值滤波，还是非线性的中值滤波，实现都是较为容易的。因此它们在图像处理中得到了广泛的应用。