专业：计算机科学与技术
姓名：武思羽
学号：3220105846
日期：2023/12/22

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：双边滤波的实现

一、实验目的和要求

本实验的目的在于熟练掌握图像双边滤波的方法，能够熟练运用高斯滤波与双边滤波的公式对图像进行对应处理。

本实验要求利用代码实现对给定图片的双边滤波，输出图像双边滤波后的图像。

二、实验内容和原理

图像双边滤波的原理：

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。具有简单、非迭代、局部的特点。双边滤波器的好处是可以做边缘保存，一般用高斯滤波去降噪，会较明显地模糊边缘，对于高频细节的保护效果并不明显。

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，是由于其滤波器的核由两个函数生成：空间域核和值域核。

（1）空间域核：

由像素位置欧式距离决定的模板权值 Wd。

$$\omega_d(i,j,k,l) = \exp\left(-\frac{(i-k)^2-(j-l)^2}{2\sigma_d^2}\right)$$

其中 q(i, j)为模板窗口的其他系数的坐标；p(k, l)为模板窗口的中心坐标点；σd 是空间域上的高斯核函数的标准差，用于控制像素位置的权值。

（2）值域核：

由像素值的差值决定的模板权值 Wr。

$$\omega_r(i,j,k,l) = \exp\left(-\frac{\|f(i,j)-f(k,l)\|}{2\sigma_r^2}\right)$$

其中，q(i, j)为模板窗口的其他系数的坐标，f(i, j)表示图像在点 q(i, j)处的像素值；p(k ,l)为模板窗口的中心坐标点，对应的像素值为 f(k, l)；σr 是像素值域上的高斯核函数的标准差，用于控制像素值的权值。

最后，将上述两个模板相乘就得到了双边滤波器的模板权值：

$$\omega_d(i,j,k,l) = \exp(-\frac{(i-k)^2-(j-l)^2}{2\sigma_d^2} - \frac{\|f(i,j)-f(k,l)\|}{2\sigma_r^2})$$

因此，双边滤波器的数据公式可以表示如下:

$$g(i,j) = \frac{\sum_{kl} f(k,l)\omega(i,j,k,l)}{\sum_{kl} \omega(i,j,k,l)}$$

三、实验步骤与分析

图像双边滤波的函数实现：

（1）空间域核和值域核的实现：

```
double WS (int i, int j, int k, int l, double s_div){
        double index = (pow(i-k, 2)+pow(j-l, 2)) / (2*pow(s_div, 2)) * (-1.0);
        double res = exp(index);
        return res;
}

double WR (int pix_src, int pix_near, double r_div){
        double index = pow((pix_src - pix_near), 2) / (2*pow(r_div, 2)) * (-1.0);
        double res = exp(index);
        return res;
}
```

我将空间域核和值域核计算公式分母上的标准差作为函数传入的参数，依据实验原理中的公式来计算权值。

（2）综合函数实现：

```
Image* Bilateral_Filter (Image* bmpImg, int N, double r_div){
        int i, j, k, m, n;
        Image* newImg;
        newImg = (Image*)malloc(sizeof(Image));
        newImg->channels = 3;
        newImg->height = bmpImg->height;
        newImg->width = bmpImg->width;
```

```
        newImg->imageData = (unsigned char**)malloc(sizeof(unsigned char*) * newImg->height);
        for(i=0; i<newImg->height; i++) newImg->imageData[i] = (unsigned
char*)malloc(sizeof(unsigned char) * newImg->width * 3);

        double s_div = 0.02 * sqrt(pow(newImg->height, 2) + pow(newImg->width, 2));

        for(i=0; i<bmpImg->height; i++)
        {
                for(j=0; j<bmpImg->width; j++)
                {
                        double sum_W = 0, sum_r = 0, sum_g = 0, sum_b = 0;
                        int pix_src = bmpImg->imageData[i][j*3] + bmpImg->imageData[i][j*3+1] +
bmpImg->imageData[i][j*3+2];
                        for(m=mymax(i-N, 0); m<=mymin(i+N, bmpImg->height-1); m++)
                        {
                                for(n=mymax(j-N, 0); n<=mymin(j+N, bmpImg->width-1); n++)
                                {
                                        int pix_near = bmpImg->imageData[m][n*3] +
bmpImg->imageData[m][n*3+1] + bmpImg->imageData[m][n*3+2];
                                        double W = WS(i, j, m, n, s_div) * WR(pix_src, pix_near,
r_div);

                                        sum_W += W;
                                        sum_b += bmpImg->imageData[m][n*3] * W;
                                        sum_g += bmpImg->imageData[m][n*3+1] * W;
                                        sum_r += bmpImg->imageData[m][n*3+2] * W;
                                }
                        }
                        newImg->imageData[i][j*3] = sum_b / sum_W;
                        newImg->imageData[i][j*3+1] = sum_g / sum_W;
                        newImg->imageData[i][j*3+2] = sum_r / sum_W;
                }
        }

        return newImg;
}
```

函数参数传入原图像指针，卷积核的大小参数 N 以及像素值域上的高斯核函数的标准差 r_div。对于原图像中的每一个像素，函数通过原理中的卷积公式对卷积核内所有像素进行计算，得出新像素的 RGB 值，并输出到新图像指针中。

四、实验环境及运行方法

该程序使用 Dev-C++软件编写。在“Assignment 6”文件夹中我创建了 compiled、src、

project、test 四个子文件夹。其中 compiled 文件夹存储可执行文件以及输出的测试图片；src 文件夹存储源代码的.c、.h 文件；project 文件夹存储项目文件。同时，在 test 文件夹中我放入了待测试的图片，您也可以选择任意 bmp 格式的图片。

您可使用 Dev-C++直接打开 project 文件夹中的项目文件“Assignment 6”查看所有源代码。如需运行检验，需要按以下步骤执行：

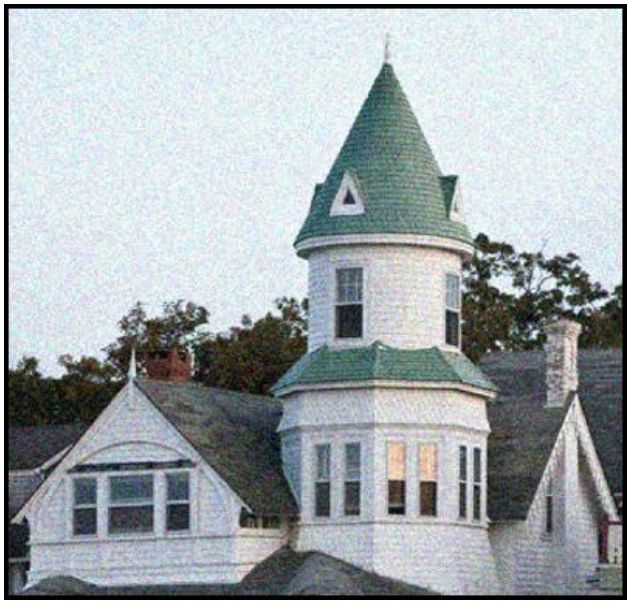
1. 选择您需要的输入图片（可以是 test 文件夹中提供的图片，也可以是您自己准备的图片），将其复制到 compiled 文件夹中，并命名为“testimage.bmp”，代表输入图片。

2. 使用 Dev-C++打开 project 文件夹中的“Assignment 6”项目文件，全部重新编译后运行，等待片刻（视图像大小与卷积核边长参数而定，一般需要 2-3 分钟，如果参数过大可能需要等待更长时间）。

3. 查看 compiled 文件夹中的输出图片。其中“filterimage.bmp”代表目标图像双边滤波后的图像，其中参数 N = 20，r_div = 32。当然，您也可以依据实验报告第三部分对各函数参数的介绍，自行修改参数并查看结果。

五、实验结果展示

原图像如下所示：



双边滤波后的图像如下所示：

可以看出，双边滤波后的图像很好地去除了图像背景中的噪声，同时也保留了图像像素值变化较大的边缘部分的信息，说明实验成功。

六、心得体会

本次实验我通过理解双边滤波相较于低通的高斯滤波的优点，结合双边滤波实现公式，明白了双边滤波是如何在去除噪声的同时保留信息的。同时，我能够熟练掌握利用代码实现双边滤波的方法，获益颇丰。