**实验报告**

专业：计算机科学与技术

姓名：张三

学号：111111

日期：2022/12/5

课程名称： 图像信息处理 指导老师： 成绩：

实验名称： 双边滤波

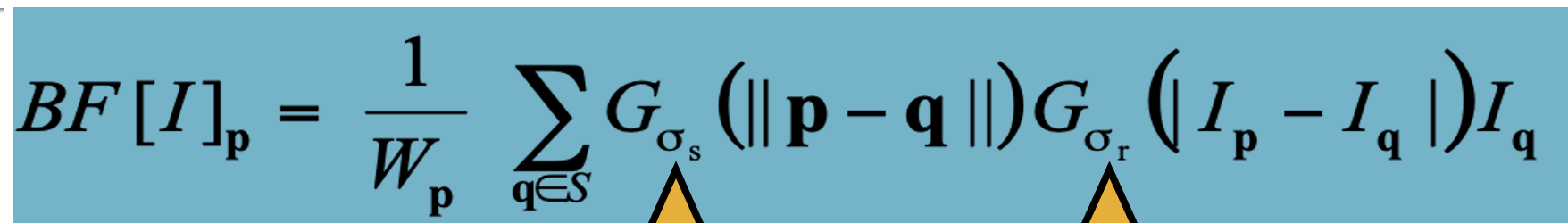
**一、实验目的和要求**

实验目的：对bmp图像实现双边滤波，体会不同参数对双边滤波的影响

实验要求：用C/C++实现，不允许调用相关的库

**二、实验内容和原理**

双边滤波是高斯滤波的加强版，为了应付高斯滤波对边和对内部细节一视同仁的平滑从而导致的边的模糊，双边滤波加入了以相邻点的灰度差值为输入的高斯函数作为权重的一项影响因子，因而权重的计算公式如下：



其中的Gs是以(0,s)为参数的正态密度函数，Gr是以(0,r)为参数的正态密度函数，前者保证了空间上距离越远的像素点对中心像素点的权重越低，后者保证了与中心点灰度值相差越大的点的权重越低，而这两项权重降低的速率由参数s、r决定，s、r越小，权重降低的越快，s、r越大，权重降低的越慢，直至所有点的权重都只有一个小值的差异，最后的效果就接近于平滑滤波。注意到Wp这一项适用于归一化的，Wp的值等于所有计算出来的权重之和，这保证了新的灰度值仍然落在0-255的区间之内。

这就意味着在设置了合适的参数的情况下，边会因为与两侧的点的灰度值相差很大而避免被模糊，因而具有保边的特性。增加r可以降低保边的能力。

本次实验会直接将边缘的点原封不动的复制到新图之中，而对于内部的点就会用卷积核处理一番之后再赋给新图。

**三、实验步骤与分析**

代码已详细注释

#include <stdio.h>

#include <stdlib.h>

#include <windows.h>

#include <math.h>

#define pi 3.1415926

typedef struct BMP{

    FILE\* file;

    BITMAPFILEHEADER bf;

    BITMAPINFOHEADER bi;

} BMP;

int write\_gray\_data(BMP bmp,double\* graydata);//write the gray intensity array into the file

int adjust\_gray\_metadata(BMP\*); // change the metadata of the graybmp

double\* find(double\*,int,int,int);//find the address of the wanted member of a 2d array

void fill\_gray\_palette(RGBQUAD\*);//fill the gray palette

double Gauss(double sigma,double mu,double x);//the Gauss function with the parameter of mu and sigma

void bilateral(double\*,double,double,BMP,int);//do the bilateral filter

int write\_data(BMP bmp,unsigned char\* black\_white);//write the prepared bmp data into file

int initial\_bmp(BMP);//write the file header into the file

BMP create\_bmp();//read the first bmp in D://test.bmp

int fill\_rgb\_yuv\_array(BMP,unsigned char\*,double\*);//read bmp data from the 24color file and transfer it into gray number array

int main(int argc,char\*\* argv){

    BMP rawbmp = create\_bmp();//read the test.bmp

    unsigned char \*rgbdata= malloc (sizeof(unsigned char)\*3\*rawbmp.bi.biWidth\*rawbmp.bi.biHeight);// to store the gray number for gray bmp

    double \*yuv= malloc (sizeof(double)\*3\*rawbmp.bi.biWidth\*rawbmp.bi.biHeight);

    fill\_rgb\_yuv\_array(rawbmp,rgbdata,yuv);//fill in the array

    FILE \* grayfile =fopen("D://gray.bmp","wb");//the new file of gray bmp

    BMP graybmp = {grayfile,rawbmp.bf,rawbmp.bi};

    adjust\_gray\_metadata(&graybmp);

    initial\_bmp(graybmp);

    RGBQUAD rgbquad[256];

    fill\_gray\_palette(rgbquad);

    fwrite(&rgbquad,sizeof(rgbquad),1,grayfile);// write in the palette

    write\_gray\_data(graybmp,yuv);//write the y component into the file

    fclose(grayfile);

    FILE\* bilfile = fopen("D://bilateral.bmp","wb");

    BMP bilbmp = {bilfile,graybmp.bf,graybmp.bi};

    initial\_bmp(bilbmp);

    fwrite(&rgbquad,sizeof(rgbquad),1,bilfile);//write in the palette

    double s = 0.0025\*sqrt((pow(bilbmp.bi.biHeight,2)+pow(bilbmp.bi.biWidth,2)));//the s parameter

    double r = 20;//the r parameter

    int window=13;

    if(argc>1)     s = strtod(argv[1],0);//if s is determined by the command line parameter

    if(argc>2)     r = strtod(argv[2],0);//if r is determined by the command line parameter

    bilateral(yuv,s,r,bilbmp,window);

write\_gray\_data(bilbmp,yuv+1);//remember that the processed data is stored in u component

    if(argc>3)      window = (int)strtod(argv[3],0);//if window is given by the command line parameter

    free(yuv);

    free(rgbdata);

}

//func

int fill\_rgb\_yuv\_array(BMP rawbmp,unsigned char\* rgbdata, double \* yuv){

    int width = rawbmp.bi.biWidth\*3;

    int adjusted\_width = (width+3)/4\*4;//if we need to write 0 to take up places

    unsigned char dustbin;

    int count=0;

    for(int i=0; i< rawbmp.bi.biWidth \* rawbmp.bi.biHeight; i++){

        fread(&rgbdata[3\*i],1,1,rawbmp.file);// read the  rgb number

        fread(&rgbdata[3\*i+1],1,1,rawbmp.file);

        fread(&rgbdata[3\*i+2],1,1,rawbmp.file);

        unsigned char y = 0.299\*rgbdata[3\*i+2] + 0.587\*rgbdata[3\*i+1] + 0.114\*rgbdata[3\*i];//calculate yuv

        yuv[3\*i] = 0.299\*rgbdata[3\*i+2] + 0.587\*rgbdata[3\*i+1] + 0.114\*rgbdata[3\*i] ;

        yuv[3\*i+1] = -0.147\*rgbdata[3\*i+2] + -0.289\*rgbdata[3\*i+1] + 0.435 \*rgbdata[3\*i];

        yuv[3\*i+2] = 0.615\*rgbdata[3\*i+2] + -0.515\*rgbdata[3\*i+1] + -0.1\*rgbdata[3\*i];

        count+=3;

        if (count == width){// read the 0s that just take up places

            while(count++ != adjusted\_width){

                fread(&dustbin,1,1,rawbmp.file);

            }

            count = 0;

        }

    }

}

BMP create\_bmp(){//read and initial from test.bmp

    BMP newbmp;

    FILE \* bmp;

    BITMAPFILEHEADER bf;

    BITMAPINFOHEADER bi;

    if( !(bmp = fopen("D:\\test.bmp","rb"))){

        printf("Not find file in D:\\test.bmp");

        return newbmp;

    }

    fread(&bf,sizeof(BITMAPFILEHEADER),1,bmp);

    fread(&bi,sizeof(BITMAPINFOHEADER),1,bmp);

    newbmp.bf = bf;

    newbmp.bi = bi;

    newbmp.file = bmp;

    return newbmp;

}

int initial\_bmp(BMP bmp){//initial the certain bmp

    fwrite(&bmp.bf,sizeof(bmp.bf),1,bmp.file);

    fwrite(&bmp.bi,sizeof(bmp.bi),1,bmp.file);

}

int write\_gray\_data(BMP bmp,double\* graydata){

    int width = bmp.bi.biWidth;

    int adjusted\_width = (width+3)/4\*4;//if we need to write 0 to take up places

    int count=0;

    unsigned char y;

    for(int i=0;i<bmp.bi.biHeight\*bmp.bi.biWidth;i++){

        y = (unsigned char)graydata[3\*i];

        fwrite(&y,sizeof(unsigned char),1,bmp.file);

        count+=1;

        if (count == width){// write the 0s that just take up places

            while(count++ != adjusted\_width){

                fwrite(graydata,1,1,bmp.file);

            }

            count = 0;

        }

    }

}

int adjust\_gray\_metadata(BMP\* Graybmp){// adjust the metadata to fit with the new gray bmp

    Graybmp->bf.bfOffBits += 256 \*sizeof(RGBQUAD);

    Graybmp->bf.bfSize = Graybmp->bf.bfOffBits + (Graybmp->bi.biWidth+3/4\*4) \*Graybmp->bi.biHeight;

    Graybmp->bi.biBitCount = 8;

    Graybmp->bi.biSizeImage =  (Graybmp->bi.biWidth+3/4\*4) \*Graybmp->bi.biHeight;

}

void fill\_gray\_palette(RGBQUAD\* rgbquad){//fill in the palette array

    for(int i=0; i<256; i++){

        rgbquad[i].rgbBlue = i;

        rgbquad[i].rgbGreen = i;

        rgbquad[i].rgbRed = i;

        rgbquad[i].rgbReserved = 0 ;

    }

}

void bilateral(double\* yuv,double s,double r,BMP bilbmp,int window){

    double Gauss\_s[window][window];

    double \*dest = yuv+1;

    double wid = window;

    double half = (wid-1)/2;

    for(int i=0;i<window;i++){//the space factor in the convolution kernal

        for(int j=0;j<window;j++){

            Gauss\_s[i][j] = Gauss(s,0,sqrt(pow(i-half,2)+pow(j-half,2)));

        }

    }

    for(int i=0;i<(window-1)/2;i++){// the edge where we cant do the convolution, we just copy the gray intensity

        for(int j=0;j<bilbmp.bi.biWidth;j++){

            \*(find(dest,bilbmp.bi.biWidth,j,i)) = \*(find(yuv,bilbmp.bi.biWidth,j,i));

            \*(find(dest,bilbmp.bi.biWidth,j,bilbmp.bi.biHeight-1-i)) = \*(find(yuv,bilbmp.bi.biWidth,j,bilbmp.bi.biHeight-1-i));

        }

    }

    for(int i=0;i<half;i++){//the edge where we cant do the convolution,we just copy the gray intensity

        for(int j=0;j<bilbmp.bi.biHeight;j++){

            \*(find(dest,bilbmp.bi.biWidth,i,j)) = \*(find(yuv,bilbmp.bi.biWidth,i,j));

            \*(find(dest,bilbmp.bi.biWidth,bilbmp.bi.biWidth-1-i,j)) = \*(find(yuv,bilbmp.bi.biWidth,bilbmp.bi.biWidth-1-i,j));

        }

    }

    for(int i= half;i<bilbmp.bi.biWidth-half;i++){//for every pixel to be processed

        for(int j =half;j<bilbmp.bi.biHeight-half;j++){

            int modify\_xy[window];//the offset of x and y

            double final\_Gauss[window][window];// the modified kernal take both intensity and distance into account

            double total = 0;//do the normallization

            for(int k=0;k<window;k++)   modify\_xy[k] = k - (window-1)/2;//fill in the offset such as -2 -1 0 1 2

            for(int i1=0;i1<window;i1++){

                for(int j1 =0;j1<window;j1++){//for every neighbour

                    int modified\_x = i+modify\_xy[i1];

                    int modified\_y = j+modify\_xy[j1];//the coordinate of the neighbour

                    final\_Gauss[i1][j1] = Gauss\_s[i1][j1]\*Gauss(r,0,\*(find(yuv,bilbmp.bi.biWidth,modified\_x,modified\_y))-\*(find(yuv,bilbmp.bi.biWidth,i,j)));//the final weight of this neighbour

                    total += final\_Gauss[i1][j1];

                }

            }

            double new\_intensity=0;

            for(int i1=0;i1<window;i1++){

                for(int j1=0;j1<window;j1++){

                    int modified\_x = i+modify\_xy[i1];

                    int modified\_y = j+modify\_xy[j1];

                    new\_intensity += \*(find(yuv,bilbmp.bi.biWidth,modified\_x,modified\_y))\*final\_Gauss[i1][j1];//do the convolution

                }

            }

            new\_intensity /= total;//normalization

            \*(find(dest,bilbmp.bi.biWidth,i,j)) = new\_intensity;//assignment

        }

    }

}

double Gauss(double sigma,double mu,double x){//calculate the Gauss function with given parameter

    return  1/sqrt(2\*pi)/sigma\*exp(-(pow(x-mu,2))/2/pow(sigma,2));

}

double\* find(double\* array,int width,int x,int y){//find the address of a member in a 2d array

    return array+y\*3\*width+3\*x;

}

**四、实验环境及运行方法**

直接用gcc编译运行就可以，但是要保证D:\\test.bmp是存在的。

另外 可以用命令行参数指定s 和 r 和window的边长 顺序不可变。

**五、实验结果展示**

（展示实验中的输入输出图像等）

原彩色图：



灰度图：  


双边滤波（默认参数）：



双边滤波：s=1 r=80 window=13：



双边滤波：s=30 r=80 window=13：



**六、心得体会**

本次是实现双边滤波，基本上没有以前所没有遇到过的新内容，只是在计算中多了用正态密度函数进行计算的过程，对于边缘点，我们没有对边缘进行padding处理，而是直接将原像素点搬过去，这可能会在我们选择的窗口比较大的时候导致边缘有一圈的清晰的反常的点，这么来看的话采取padding处理还算是一点提升。