C语言程序设计大作业

基于清晰度的多聚焦图像融合算法的实现

单 位： 网络空间安全学院

班 级： 2021网络空间安全二班

学 号： 202105800610

姓 名： 刘继佳

任课教师： 李勃东

湘 潭 大 学

**2022**年5月

## 文档信息

|  |  |  |  |
| --- | --- | --- | --- |
| 源代码 | 可执行文件 | 自评成绩 | 答辩PPT |
| 有 | 有 | 优 | 有 |

## 学术诚信承诺书

刘继佳，学号202105800610，系网络空间安全学院21级网络空间安全专业二班学生。本人承诺本课程设计提交的所有源代码和文档不存在抄袭、剽窃行为。如果被他人发现存在抄袭、剽窃、雷同等学术不端行为，愿意承担所带来的一切后果。

签名、时间照片图片

## IMG_20220531_140618__01

## 三、题目概述

1. 概述

将两张焦点不同的图像融合成一张清晰的图片。

1. 所要完成的任务说明

基于清晰度的多聚焦图像融合算法的实现

|  |  |  |
| --- | --- | --- |
| 序号 | 功能分类 | 任务 |
| 1 | 必做 | 边界严拓预处理 |
| 2 | 必做 | 将bmp图片分块 |
| 3 | 必做 | 计算各个块的清晰度 |
| 4 | 必做 | 一致性校验 |
| 5 | 选做 | 能够自由设置分块大小 |
| 6 | 选做 | 能够保存原始图像和融合图像的分块图 |

1. 完成情况

|  |  |
| --- | --- |
| 功能序号 | 完成情况说明 |
| 1 | 在图像长宽不能除尽所选的分块数量时，以上边界和右边界为轴将像素数据对称填充 |
| 2 | 可将原图与融合图像的分块图输出保存 |
| 3 | 根据公式成功计算出各个块的清晰度 |
| 4 | 比较一个块周围的块来自哪个图片的块多，若来自另一张图片的块更多则修改这一个块的来源为另一张图的相应位置 |
| 5 | 可以在程序开始后输入想要把图片分成多少乘多少块。例如要分成8\*8，输入8即可 |
| 6 | 可以选择输出原图片的分块图和融合后的图片的分块图 |

1. 开发环境说明

Dev-C++ 5.11

## 四、程序分析与设计

1. 系统分析

要求1：

void print (BMPFi \* bmpf,BMPIn \* bmpi)

功能：打印bmp图片基本信息。

参数意义：

BMPFi \* bmpf：从图片中读取出来的bmp文件头地址

BMPIn \* bmpi：从图片中读取出来的位图信息头地址

返回值类型：空。

U8 \* read\_image (BMPFi \* bmpf, BMPIn \* bmpi,U8 \*pallet,U8 \* date)

功能：读取bmp图像。

参数意义：

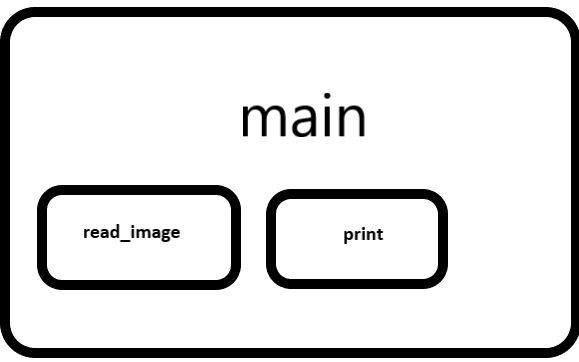
BMPFi \* bmpf：bmp文件头地址

BMPIn \* bmpi：位图信息头地址

U8 \*pallet：调色板地址

返回值类型：U8\*

要求2：函数之间的调用关系图：



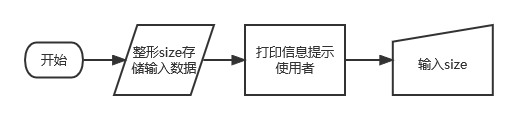
1. 函数设计

（1）输入分为多少块

int size; //size为要分成多少乘多少的块

printf ("你想要把图片分成多少乘多少的？例如你想分成8\*8共64块，输入8即可。\n请输入一个数字：\n");

scanf ("%d",&size);



（2）读取数据

//图一

BMPFi bmpf1; //位图文件头

BMPIn bmpi1; //位图信息头

U8 pallet1[1024]; //调色板

U8 \* date1; //位图数据

date1=read\_image (&bmpf1,&bmpi1,pallet1,date1);

//读取数据调用read\_image函数

print(&bmpf1,&bmpi1); //打印信息

//图二

BMPFi bmpf2; //位图文件头

BMPIn bmpi2; //位图信息头

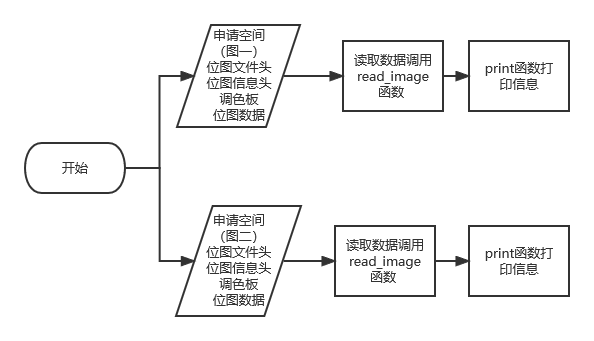
U8 pallet2[1024]; //调色板

U8 \* date2; //位图数据

date2=read\_image (&bmpf2,&bmpi2,pallet2,date2);

//读取数据用read\_image函数

print(&bmpf2,&bmpi2); //打印信息



（3）数据转二维并分块

int width=(int)bmpi1.bWidth; //图像宽度

int hight=(int)bmpi1.bHeight; //图像高度

int i,j,k=0;

U8\*\* date\_1; //二级指针用来申请空间存储转为二维的位图数据

U8\*\* date\_2;

date\_1=(U8\*\*)malloc(sizeof(U8\*)\*hight); //二级指针申请一级指针

date\_2=(U8\*\*)malloc(sizeof(U8\*)\*hight);

for (i=0;i<hight;i++){ //申请存储空间

date\_1[i]=(U8\*)malloc(sizeof(U8)\*width);

date\_2[i]=(U8\*)malloc(sizeof(U8)\*width);

}

for (i=0;i<hight;i++){

for (j=0;j<width;j++){

date\_1[i][j]=date1[k]; //将一维数据转为二维，方便后面的处理

date\_2[i][j]=date2[k];

if(i==0||j==0||i%(hight/size)==0||j%(hight/size)==0) {

date1[k]=(U8)255; //将原来的数据分块

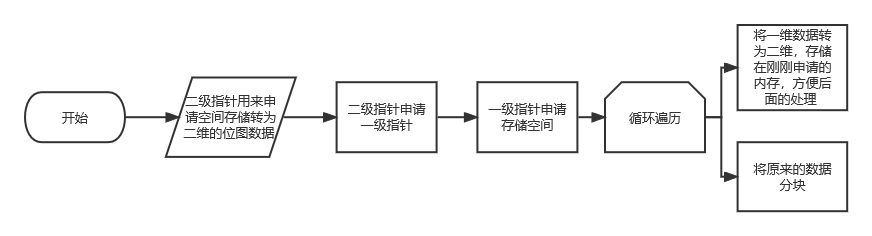
date2[k]=(U8)255;

}

k++;

}

}



（4）输出原图分块图

printf("\n请问是否要输出原图分块图\n请输入1或0 1为输出，0为不输出:\n");

int d;

scanf ("%d",&d);

if (d){

FILE\* fp=NULL; //文件指针

FILE\* fq=NULL;

char name1[20]="Block\_1.bmp"; //输出分块图的名字

char name2[20]="Block\_2.bmp";

fp=fopen (name1,"wb"); //打开文件

fq=fopen (name2,"wb");

if (fp==NULL){

printf ("生成失败\n");

}

fwrite(&bmpf1,sizeof (BMPFi),1,fp); //写入数据

fwrite(&bmpi1,sizeof (BMPIn),1,fp);

fwrite(&bmpf2,sizeof (BMPFi),1,fq);

fwrite(&bmpi2,sizeof (BMPIn),1,fq);

fwrite(pallet1,sizeof(U8)\*1024,1,fp);

fwrite(pallet2,sizeof(U8)\*1024,1,fq);

fwrite(date1,sizeof(U8)\*hight\*width,1,fp);

fwrite(date2,sizeof(U8)\*hight\*width,1,fq);

printf ("分块图输出完成\n");

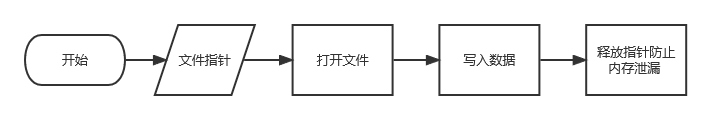
fclose(fp);

fclose(fq);

}

free(date1); //释放指针防止内存泄漏

free(date2);



（5）严拓

U8 date\_11[(hight/size+1)\*size][(width/size+1)\*size];

// 申请存储严拓之后的数据的空间

U8 date\_22[(hight/size+1)\*size][(width/size+1)\*size];

if (hight%size!=0&&width%size!=0){ // 宽和高都除不尽size的情况

for (i=0;i<(hight/size+1)\*size;i++){

//遍历图片 (hight/size+1)\*size为延拓之后的高

for (j=0;j<(width/size+1)\*size;j++){

// (width/size+1)\*size为延拓之后的宽

if (i<hight&&j<width){ // 遍历区域未超过原图

date\_11[i][j]=date\_1[i][j]; // 填入数据

date\_22[i][j]=date\_2[i][j];

}

else if (i>=hight&&j<width){

//遍历区域超过原图区域高但未超过宽（对超过部分开始严拓）

date\_11[i][j]=date\_1[2\*hight-i-1][j];

date\_22[i][j]=date\_2[2\*hight-i-1][j];

//超出部分的数组下标为 2\*hight-i-1

}

else if (i<hight&&j>=width){

// 遍历区域超过原图区域宽但未超过高（对超过部分开始严拓）

date\_11[i][j]=date\_1[i][2\*width-j-1];

date\_22[i][j]=date\_2[i][2\*width-j-1];

}

else if (i>=hight&&j>=width){

// 遍历区域超过原图区域宽和高（对超过部分开始严拓）

date\_11[i][j]=date\_1[2\*hight-i-1][2\*width-j-1];

date\_22[i][j]=date\_2[2\*hight-i-1][2\*width-j-1];

}

}

}

hight=(hight/size+1)\*size; //因为延拓，数据的高改变

width=(width/size+1)\*size; //因为延拓，数据的宽改变

}

else if (hight%size!=0&&width%size==0){ //高除不尽size的情况

for (i=0;i<(hight/size+1)\*size;i++){

for (j=0;j<width;j++){

if (i<hight&&j<width){

date\_11[i][j]=date\_1[i][j];

date\_22[i][j]=date\_2[i][j];

}

else if (i>=hight&&j<width){

date\_11[i][j]=date\_1[2\*hight-i-1][j];

date\_22[i][j]=date\_2[2\*hight-i-1][j];

}

}

}

hight=(hight/size+1)\*size;

}

else if (hight%size==0&&width%size!=0){ //宽除不尽size的情况

for (i=0;i<hight;i++){

for (j=0;j<(width/size+1)\*size;j++){

if (i<hight&&j<width){

date\_11[i][j]=date\_1[i][j];

date\_22[i][j]=date\_2[i][j];

}

else if (i<hight&&j>=width){

date\_11[i][j]=date\_1[i][2\*width-j-1];

date\_22[i][j]=date\_1[i][2\*width-j-1];

}

}

}

width=(width/size+1)\*size;

}

else{ //都能除尽的情况

for (i=0;i<hight;i++){

for (j=0;j<width;j++){

date\_11[i][j]=date\_1[i][j];

date\_22[i][j]=date\_2[i][j];

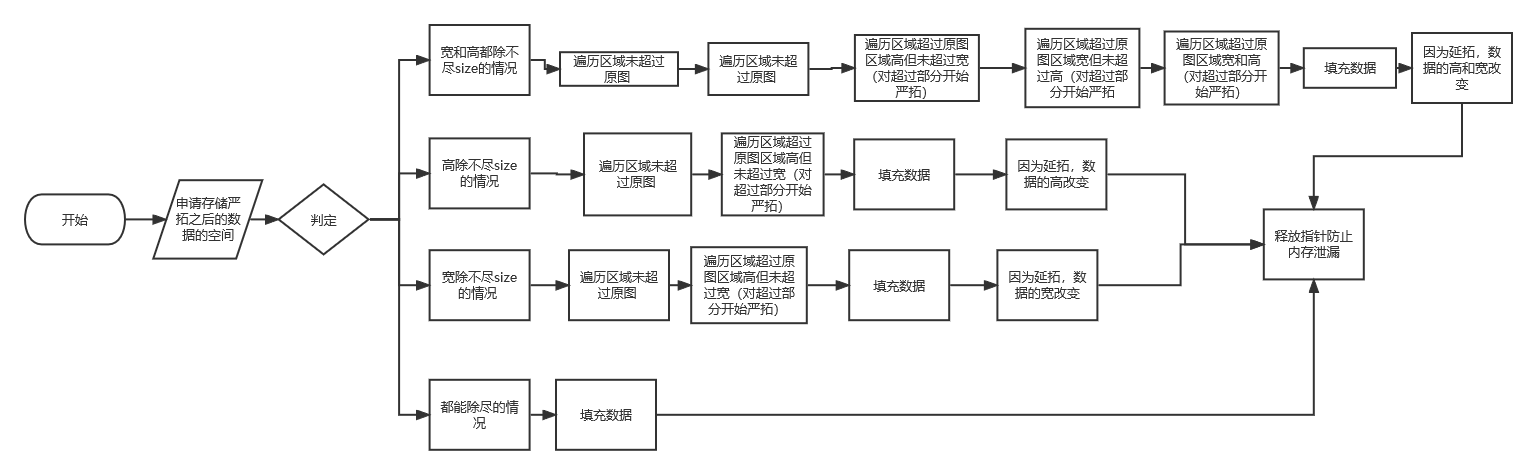
}

}

}

free(date\_1);

free(date\_2);



（5）清晰度计算

double definition1[size][size]; //储存各个块清晰度的二维数组

double definition2[size][size];

for (i=0;i<size;i++){ //遍历

for (j=0;j<size;j++){

int x,y;

//水平

double HD1=0;

//HD1为第一个块的水平清晰度 HD2为第二个块的水平清晰度

double HD2=0; //分块遍历

for (y=i\*hight/size;y<(i+1)\*(hight/size);y++){

// i\*hight/size为块开始的高 (i+1)\*(hight/size)块结束的高

for (x=j\*width/size;x<(j+1)\*(width/size);x++){

// j\*width/size为块开始的宽 (j+1)\*(width/size)块结束的宽

HD1+=(double)pow(date\_11[y][x]-date\_11[y][x+1],2);

//计算水平清晰度

HD2+=(double)pow(date\_22[y][x]-date\_22[y][x+1],2);

}

}

HD1/=(double)(width\*hight/size);

HD2/=(double)(width\*hight/size);

//垂直

double VD1=0;

//VD1为第一个块的垂直清晰度 VD2为第二个块的垂直清晰度

double VD2=0;

for (x=j\*width/size;x<(j+1)\*(width/size);x++){

for (y=i\*hight/size;y<(i+1)\*(hight/size);y++){

VD1+=(double)pow(date\_11[y][x]-date\_11[y+1][x],2);

//计算垂直清晰度

VD2+=(double)pow(date\_22[y][x]-date\_22[y+1][x],2);

}

}

VD1/=(double)(hight\*width/size);

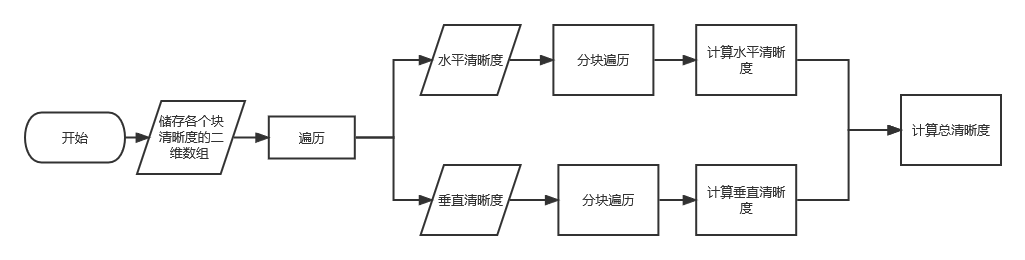
VD2/=(double)(hight\*width/size);

definition1[i][j]=sqrt(HD1+VD1); //计算总清晰度

definition2[i][j]=sqrt(HD2+VD2);

}

}



（6）比较清晰度并初始化计数器

cut def[size][size];

//其中的值决定最终的块是来自哪张图片，值为1则该块来自第一张图片，值为2则该块来自第二张图片，下标为块的位置

for (i=0;i<size;i++){

for (j=0;j<size;j++){

if (definition1[i][j]>definition2[i][j]){ //比较来自两个块的清晰度

def[i][j].da=1;

//图一清晰度大则新图相应的块来自图一

def[i][j].cnt1=0;

//def[i][j].cnt1为计数器，记录块def[i][j]周围来自第一张图的块的数量，在这里是把他初始化为0

def[i][j].cnt2=0;

//def[i][j].cnt2为计数器，记录块def[i][j]周围来自第二张图的块的数量，在这里是把他初始化为0

}

else{ //图二清晰度大则新图相应的块来自图二

def[i][j].da=2;

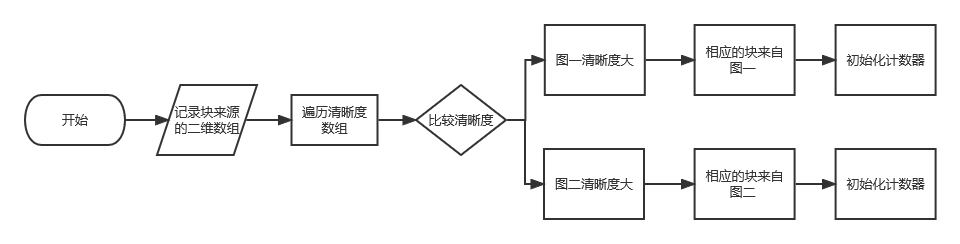
def[i][j].cnt1=0;

def[i][j].cnt2=0;

}

}

}



（7）一致性校验

for (i=0;i<size;i++){

for (j=0;j<size;j++){

if (i!=size-1&&def[i+1][j].da==1) def[i][j].cnt1++;

//分别记录下一个块周围来自两张图的块的数量

else if (i!=size-1&&def[i+1][j].da==2) def[i][j].cnt2++;

if (j!=size-1&&def[i][j+1].da==1) def[i][j].cnt1++;

else if (j!=size-1&&def[i][j+1].da==2) def[i][j].cnt2++;

if (j!=0&&def[i][j-1].da==1) def[i][j].cnt1++;

else if (j!=0&&def[i][j-1].da==2) def[i][j].cnt2++;

if (i!=0&&def[i-1][j].da==1) def[i][j].cnt1++;

else if (i!=0&&def[i-1][j].da==2) def[i][j].cnt2++;

if (i!=size-1&&j!=size-1&&def[i+1][j+1].da==1) def[i][j].cnt1++;

else if (i!=size-1&&j!=size-1&&def[i+1][j+1].da==2) def[i][j].cnt2++;

if (i!=0&&j!=0&&def[i-1][j-1].da==1) def[i][j].cnt1++;

else if (i!=0&&j!=0&&def[i-1][j-1].da==2) def[i][j].cnt2++;

if (i!=0&&j!=size-1&&def[i-1][j+1].da==1) def[i][j].cnt1++;

else if (i!=0&&j!=size-1&&def[i-1][j+1].da==2) def[i][j].cnt2++;

if (i!=size-1&&j!=0&&def[i+1][j-1].da==1) def[i][j].cnt1++;

else if (i!=size-1&&j!=0&&def[i+1][j-1].da==2) def[i][j].cnt2++;

}

}

for (i=0;i<size;i++){ //遍历比较一个块周围来自两张图的块的数量

for (j=0;j<size;j++){//若来自另外一张图块的数量多则更改来自块的来源

if (def[i][j].cnt1>=def[i][j].cnt2){

def[i][j].da=1;

}

if (def[i][j].cnt1<def[i][j].cnt2){

def[i][j].da=2;

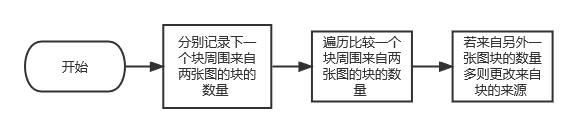
}

def[i][j].cnt1=0;

def[i][j].cnt2=0;

}

}



（8）填充新图片数据

U8 date[hight][width]; //开辟新数组存储新图片数据

int x=0,y=0;

for (i=0;i<size;i++){

for (j=0;j<size;j++){

if (def[i][j].da==1){ //判断块的来源，并填入来源块的数据

for (y=i\*(hight/size);y<(i+1)\*(hight/size);y++){

for (x=j\*(width/size);x<(j+1)\*(width/size);x++){

date[y][x]=date\_11[y][x];

}

}

}

else if (def[i][j].da==2){

for (y=i\*(hight/size);y<(i+1)\*(hight/size);y++){

for (x=j\*(width/size);x<(j+1)\*(width/size);x++){

date[y][x]=date\_22[y][x];

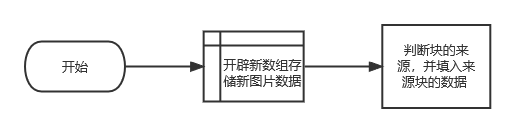
}

}

}

}

}



（9）数据转一维并去除延拓数据

hight=(int)bmpi1.bHeight;

width=(int)bmpi1.bWidth;

U8 finaldate[hight\*width];

k=0;

for (i=0;i<hight;i++){

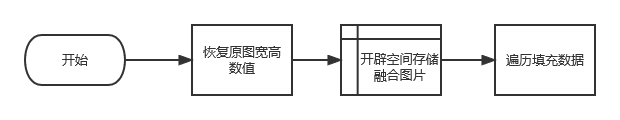
for (j=0;j<width;j++){

finaldate[k]=date[i][j];

k++;

}

}



（10）输出融合图像分块图

printf("\n请问是否要输出融合图像分块图\n请输入1或0 1为输出，0为不输出:\n");

int s;

scanf ("%d",&s);

if (s){

U8 finaldate\_block[hight\*width]; //分块图图像数据

k=0;

for (i=0;i<hight;i++){

for (j=0;j<width;j++){

if(i==0||j==0||i%(hight/size)==0||j%(hight/size)==0) {

date[i][j]=255;

}

finaldate\_block[k]=date[i][j];

k++;

}

}

FILE\* fp=NULL;

char u[30]="OUTPUT\_block.bmp";

fp=fopen (u,"wb");

if (fp==NULL){

printf ("生成失败\n");

}

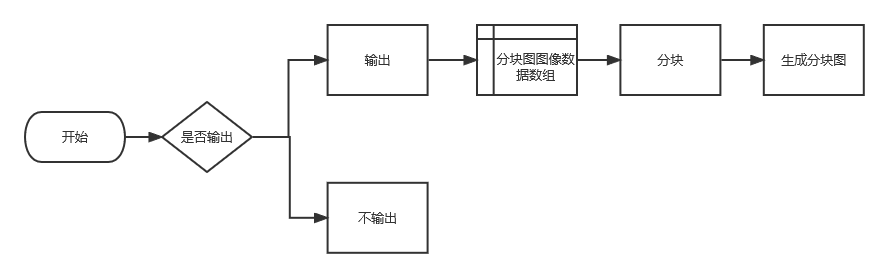
fwrite(&bmpf1,sizeof (BMPFi),1,fp);

fwrite(&bmpi1,sizeof (BMPIn),1,fp);

fwrite(pallet1,sizeof(U8)\*1024,1,fp);

fwrite(finaldate\_block,sizeof(U8)\*hight\*width,1,fp);

}



（11）生成图片

FILE\* fp=NULL;

char name[100];

printf ("请输入新图片名字\n");

scanf ("%s",name);

fp=fopen (name,"wb");

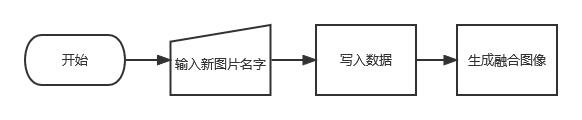
fwrite(&bmpf1,sizeof (BMPFi),1,fp); //写入数据

fwrite(&bmpi1,sizeof (BMPIn),1,fp);

fwrite(pallet1,sizeof(U8)\*1024,1,fp);

fwrite(finaldate,sizeof(U8)\*hight\*width,1,fp);

printf ("OK\n");

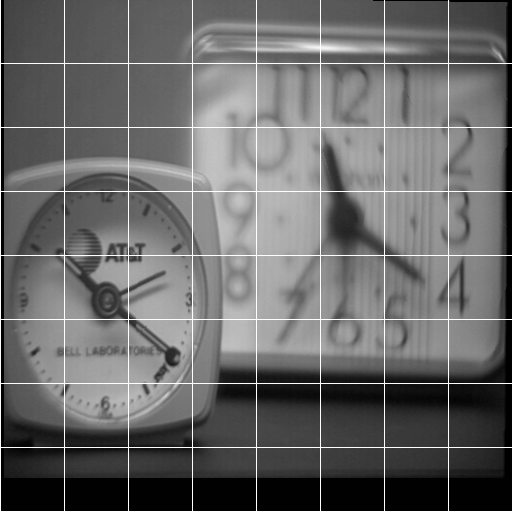


## 五、测试分析

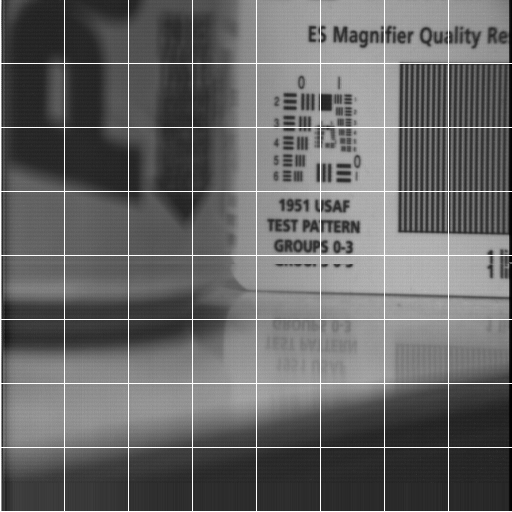
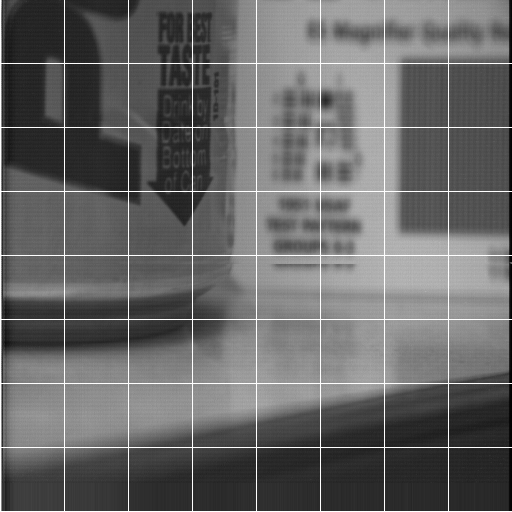
1. 测试与调试

分块为8\*8时

（1）分块图：



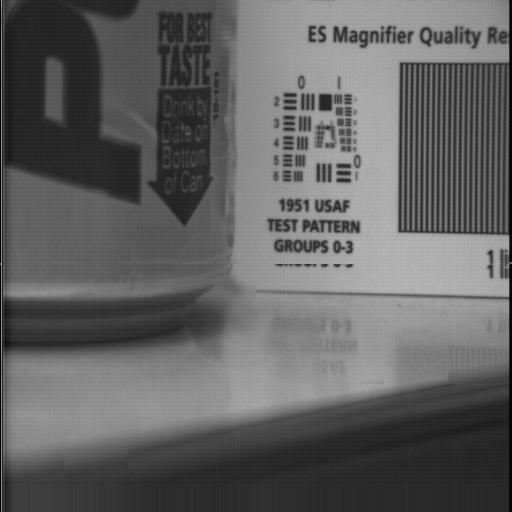






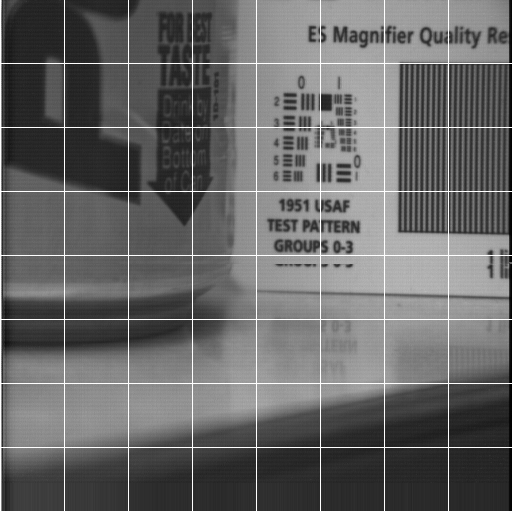
1. 融合图像：





1. 融合图像分块图：





1. 测试结果分析

可以输出两张原图融合后的清晰图像和相应原图和融合图像的分块图，但在有些地方还是会出现模糊的块。

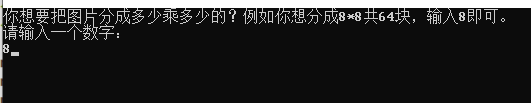
1. 结论

可以实现两张焦点不同的图像的融合并且输出。

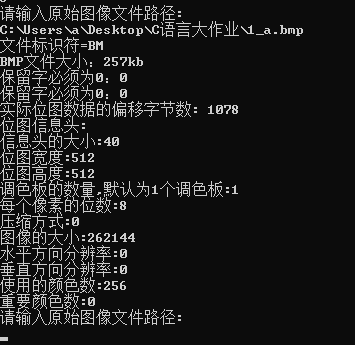
## 六、系统演示

1. 系统个功能运行截图与说明

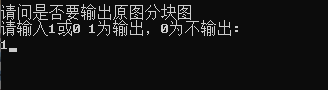
（1）输入要把图像分成多少乘多少的块



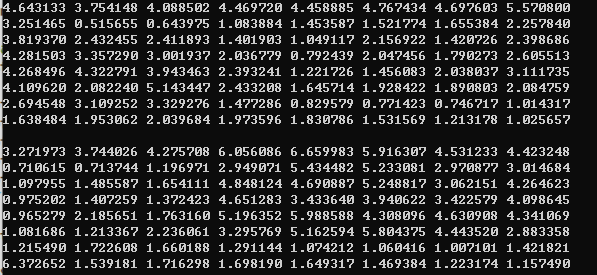
（2）打印图像基本信息



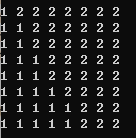
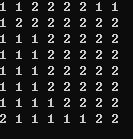
（3）是否输出原图像分块图



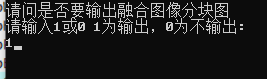
（4）打印两张图片的相应块的清晰度



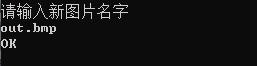
（5）打印融合图像对应位置块来自哪张原图，1代表来自输入的第一张图片，2代表来自输入的第二张图片。



（6）输出融合图像分块图



（7）输入新图片的名字，并生成新图片



## 七、设计体会

1. 设计过程中的问题及其解决方案
2. 问题：在读取数据后数据是一维的，不太方便后面的处理。

解决方案：将数据转为二维的。

1. 问题：如果在计算完两张图片的清晰度后，直接根据清晰度填入数据，则后面的一致性校验将不好进行。

解决方案：设置一个记录块来源的数组，然后进行一致性校验，再填充数据。

1. 问题：再main函数里面申请数组过多导致栈溢出，使程序崩溃。

解决方案：动态申请内存，及时释放不用了的内存空间。

1. 设计过程中的错误及其原因分析

错误1：在数据转二维时数组越界，程序崩溃。

分析：在设置存储数据的数组时没有先读取信息头的宽和高，导致在设置存储数据的数组过小。

错误2：两张图计算出来的清晰度差别过大。

分析：只有一张图片的清晰度除了(width\*hight/size)。

错误3：输出的图像是一张黑图，只有几个白点点。

分析：存储数据时要用一字节的unsigned char，但是首先用的是两个字节的unsigned short。

错误4：输出的融合图像有黑线。

分析：填入数据时的循环条件没有设置好。

3、体会和收获

（1）了解了bmp图片的储存结构，并且可以对其进行修改。

（2）对结构体，指针等C语言的知识理解更加透彻。

（3）熟悉了文件操作，知道读取文件和生成文件的操作和方法。

（4）对自己的编程能力提升很大。