基于Lagrange插值的图像缩放

## 概述

1. 概述

利用Lagrange插值法，对图像进行指定大小、样本点数的缩放

1. 完成情况

|  |  |
| --- | --- |
| 功能序号 | 完成情况说明 |
| 1 | 对灰度图像的缩放：  能够指定待缩放图像  能够指定目标图像尺寸  能够指定插值所需样本点数  能够保存缩放图像 |
| 2 | 对彩色图像的缩放：  能够指定待缩放图像  能够指定目标图像尺寸  能够指定插值所需样本点数  能够保存缩放图像 |

1. 开发环境说明

Windows10下的Dev-C++

## 程序分析与设计

1. 程序设计
2. 设计思路如下：

输入

缩放素材名，读取图像信息

输入

缩放后的宽度、高度、样本点数

创建新文件

将缩放后的图像的文件头、信息头内容写进去

灰度或彩色图像？

否

退出

无法处理

是

灰度图像？

是

否

按宽度缩放

找各点附近所需的插值节点

读入调色板信息

计算Lagrange插值对图像的宽度进行缩放

按高度缩放

找各点附近所需的插值节点

计算Lagrange插值对图像的高度进行缩放

写入缩放后图像的数据部分的信息，关闭文件

结束

代码与相关注释如下：

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

{

int fd;

int i,j,k;

int width,height,compression,bpp,sizeImage;

int stride;

char fName0[256],fName[256];

BMP\_HEADER header;

U8 data[512\*512\*3];

U8 clrPal[256\*4];

U8 p;

double Q[256];

int width0,height0,n;

U8 data0[512\*512\*3];

**/\*输入素材名，读取图像信息\*/**

if(argc==1)// 处理图片后的命名

{

printf("File name of the image to be processed: ");

scanf("%s", fName0);

strcpy(fName,"");

strncat(fName,fName0,strlen(fName0)-4);

strcat(fName,"\_eq.bmp");

}

fd = open(fName0, O\_RDONLY | O\_BINARY);

if(-1 == fd)

perror("open bmp file fail");

read(fd, &header, sizeof(BMP\_HEADER));

printf("原图片的信息如下所示：\n") ;

printf(" bfType : %c%c\n", header.bfType%256,header.bfType/256 );

printf(" bfSize : %d \n", header.bfSize );

printf(" bfReserved1 : %d \n", header.bfReserved1 );

printf(" bfReserved2 : %d \n", header.bfReserved2 );

printf(" bfOffBits : %d\n\n", header.bfOffBits );

printf(" biSize : %d \n", header.biSize );

printf(" biWidth : %d \n", header.biWidth );

printf(" biHeight : %d \n", header.biHeight );

printf(" biPlanes : %d \n", header.biPlanes );

printf(" biBitCount : %d \n", header.biBitCount );

printf(" biCompression : %d \n", header.biCompression );

printf(" biSizeImage : %d \n", header.biSizeImage );

printf(" biXPelsPerMeter : %d \n", header.biXPelsPerMeter );

printf(" biYPelsPerMeter : %d \n", header.biYPelsPerMeter );

printf(" biClrUsed : %d \n", header.biClrUsed );

printf(" biClrImportant : %d \n", header.biClrImportant );

lseek(fd, header.bfOffBits, SEEK\_SET);

read(fd, data, header.biSizeImage);

close(fd);

**/\*输入缩放后的宽度、高度、样本点数\*/**

printf("宽度大小（4的倍数）：");

scanf("%d",&width0);

printf("高度大小（4的倍数）：");

scanf("%d",&height0);

printf("插值样本点数：");

scanf("%d",&n);

**/\*创建新文件将缩放后的图像的文件头、信息头内容写进去\*/**

fd = open(fName, O\_WRONLY | O\_CREAT | O\_TRUNC | O\_BINARY);

if(fd < 0 )

perror("open x.bmp fail");

width = header.biWidth;

height = header.biHeight;

bpp = header.biBitCount;

compression = header.biCompression;

sizeImage = header.biSizeImage;

stride = sizeImage/height;

memset(&header, 0, sizeof(BMP\_HEADER));

header.bfType = 'M'\*256 + 'B';

header.bfOffBits = sizeof(BMP\_HEADER) + ((bpp==8)?1024:0);

header.bfSize = header.bfOffBits + height0\*(width0/4)\*4\*((bpp==8)?1:3) ;//缩放后图片总大小

header.biSize = 40;

header.biWidth = width0;//缩放后的宽度

header.biHeight = height0;//缩放后的高度度

header.biPlanes = 1;

header.biBitCount = bpp;

header.biCompression = compression;

header.biSizeImage =header.biHeight\*(header.biWidth/4)\*4\*((bpp==8)?1:3) ;//缩放后图片数据大小

header.biClrUsed = (bpp==8)?256:0;

write(fd, &header, sizeof(BMP\_HEADER));

**/\*若为灰度或彩色图像，进行缩放\*/**

if (bpp==8||bpp==24)

{

**/\*若为灰度图像，写入调色板\*/**

if(bpp==8) //灰度图片调色板

{

for(i=0; i<256; i++)

{

clrPal[4\*i] = clrPal[4\*i+1] = clrPal[4\*i+2] = i;

clrPal[4\*i+3] = 0;

}

write(fd, clrPal, 256\*4);

}

**/\*先进行宽度缩放查找插值点\*/**

int k,n1,n2,a[512\*3][20],b[512\*3][20];

double temp,wid,hei,g[512\*4];

g[0]=0;

temp=(width-1);

wid=temp/(width0-1);

//printf("%lf",wid);

for(i=0;i<width0;i++)

{

temp=i\*wid;

g[i]=g[0]+temp;

//printf("%lf\n",g[i]);

}

if(n%2==0) // 所找的该点的附近点左右各有几个

{

n1=n\*0.5;

n2=n1;

}

else

{

n1=n\*0.5;

n2=n1+1;

}

for(i=0;i<width0;i++)//找每个点附近的插值点（按宽度缩放得出的点）

{

k=(int)(g[i]);

if((g[i]-k)>=0.5)

{

for(j=0;j<n2;j++)

{

b[i][j]=k+1+j;

if((k+1+j)>(width-1)) a[i][j]=k+1+j-(1+j)\*2+1;

else a[i][j]=k+1+j;

}

for(j=n2;j<n;j++)

{

b[i][j]=k-(j-n2);

if(k-(j-n2)<0) a[i][j]=0-(k-(j-n2));

else a[i][j]=k-(j-n2);

}

}

else

{

for(j=0;j<n1;j++)

{

b[i][j]=k+1+j;

if((k+1+j)>(width-1)) a[i][j]=k+1+j-(1+j)\*2;

else a[i][j]=k+1+j;

}

for(j=n1;j<n;j++)

{

b[i][j]=k-(j-n1);

if(k-(j-n1)<0) a[i][j]=0-(k-(j-n1));

else a[i][j]=k-(j-n1);

}

}

}

**/\*用插值点对宽度进行缩放\*/**

if(bpp==8)//灰度图像，缩放宽度

{

for(i=0;i<height;i++)

{

for(j=0;j<width0;j++)

{

int t=-1,m1,m2;

double temp2=0.0;

for(m1=0;m1<n;m1++)//计算各点的Lagrange插值

{

t++;

double temp1=1.0;

for(m2=0;m2<n;m2++)

{

if(m2!=t) temp1=(g[j]-b[j][m2])/(b[j][t]-b[j][m2])\*temp1;

}

temp2=temp2+data[i\*stride+a[j][t]]\*temp1;

}

//temp2=temp2+0.5;

data0[i\*width0+j]=(U8)temp2;

}

}

}

else//彩色图片，缩放宽度

{

for(i=0;i<height;i++)

{

for(j=0;j<width0;j++)

{

for(k=0;k<3;k++)

{

int t=-1,m1,m2;

double temp2=0.0;

for(m1=0;m1<n;m1++)

{

t++;

double temp1=1.0;

for(m2=0;m2<n;m2++)

{

if(m2!=t) temp1=(g[j]-b[j][m2])/(b[j][t]-b[j][m2])\*temp1;

}

temp2=temp2+data[i\*stride+a[j][t]\*3+k]\*temp1;

}

//temp2=temp2+0.5;

data0[i\*width0\*3+3\*j+k]=(U8)temp2;

}

}

}

}

**/\*同理，按高度缩放\*/**

temp=height-1;

hei=temp/(height0-1);

for(i=0;i<height0;i++)

{

temp=i\*hei;

g[i]=g[0]+temp;

//printf("%lf\n",g[i]);

}

if(n%2==0)

{

n1=n\*0.5;

n2=n1;

}

else

{

n1=n\*0.5;

n2=n1+1;

}

for(i=0;i<height0;i++)

{

k=(int)(g[i]);

if((g[i]-k)>=0.5)

{

for(j=0;j<n2;j++)

{

b[i][j]=k+1+j;

if((k+1+j)>(height-1)) a[i][j]=k+1+j-(1+j)\*2+1;

else a[i][j]=k+1+j;

}

for(j=n2;j<n;j++)

{

b[i][j]=k-(j-n2);

if(k-(j-n2)<0) a[i][j]=0-(k-(j-n2));

else a[i][j]=k-(j-n2);

}

}

else

{

for(j=0;j<n1;j++)

{

b[i][j]=k+1+j;

if((k+1+j)>(height-1)) a[i][j]=k+1+j-(1+j)\*2;

else a[i][j]=k+1+j;

}

for(j=n1;j<n;j++)

{

b[i][j]=k-(j-n1);

if(k-(j-n1)<0) a[i][j]=0-(k-(j-n1));

else a[i][j]=k-(j-n1);

}

}

}

if(bpp==8)//灰色图片，缩放高度

{

for(i=0;i<width0;i++)

{

for(j=0;j<height0;j++)

{

int t=-1,m1,m2;

double temp2=0.0;

for(m1=0;m1<n;m1++)

{

t++;

double temp1=1.0;

for(m2=0;m2<n;m2++)

{

if(m2!=t) temp1=(g[j]-b[j][m2])/(b[j][t]-b[j][m2])\*temp1;

}

temp2=temp2+data0[i+a[j][t]\*width0]\*temp1;

}

//temp2=temp2+0.5;

data[i+j\*width0]=(U8)temp2;

}

}

}

else//彩色图片，缩放高度

{

for(i=0;i<width0;i++)

{

for(j=0;j<height0;j++)

{

for(k=0;k<3;k++)

{

int t=-1,m1,m2;

double temp2=0.0;

for(m1=0;m1<n;m1++)

{

t++;

double temp1=1.0;

for(m2=0;m2<n;m2++)

{

if(m2!=t) temp1=(g[j]-b[j][m2])/(b[j][t]-b[j][m2])\*temp1;

}

temp2=temp2+data0[3\*i+a[j][t]\*width0\*3+k]\*temp1;

}

//temp2=temp2+0.5;

data[3\*i+j\*width0\*3+k]=(U8)temp2;

}

}

}

}

}

**/\*不是灰度或彩色图片，退出\*/**

else

perror("Wrong type of image.");

**/\*写入数据部分的信息，并关闭文件\*/**

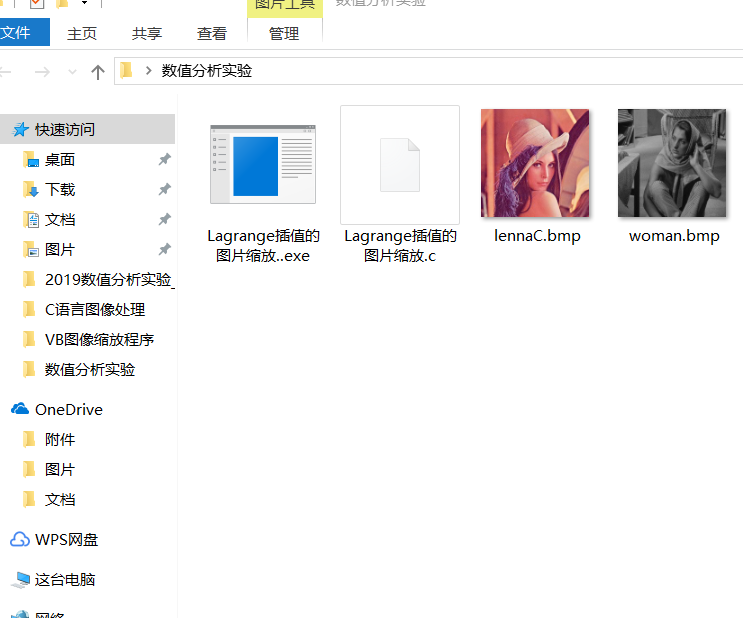
write(fd,data, header.biSizeImage);

close(fd);

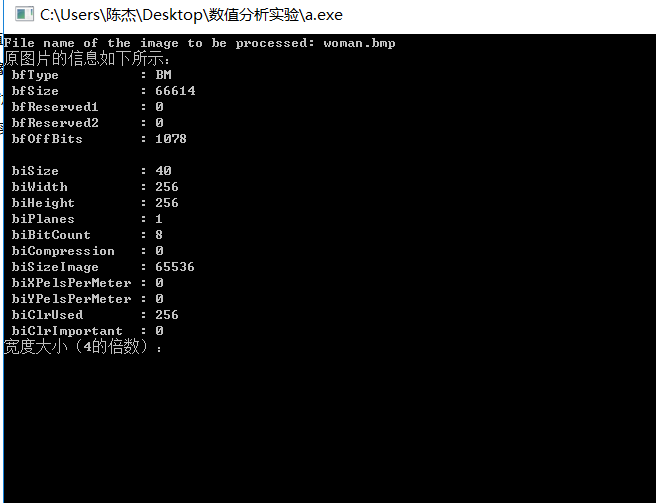
}**//结束**

## 系统演示说明

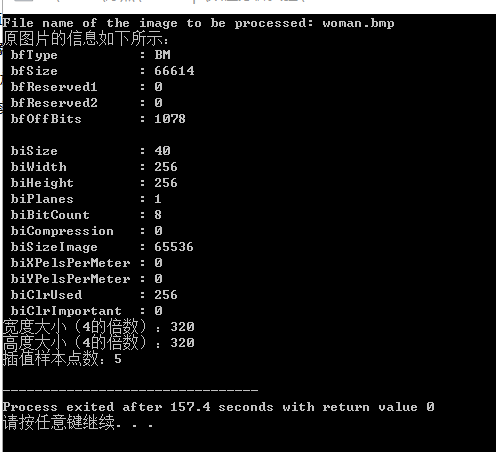
1. 系统功能运行截图与说明
2. 将素材文件放在代码可执行文件的文件夹下，如下图所示：



1. 运行可执行文件，先按要求输入原图像名字，输出原图像信息，如下图所示：



1. 再按要求输入所要缩放的宽度与高度的大小，和Lagrange插值样本点数，如下图所示：



1. 最后在放素材的文件夹下，找到所缩放的图片，如下图所示：

