图像缩放

双线性插值法

双线性插值，又称为双线性内插。在数学上，双线性插值是有两个变量的插值函数的线性插值扩展，其核心思想是在两个方向分别进行一次线性插值。

在这里插入图片描述

在这里插入图片描述

假设一个图像的大小是485x647，放大分别放大1.3倍和1.7倍，即485x1.3=630.5,647x1.7=1099.9，根据四舍五入的原则确定放大后的图像为631x1100，接下来就是计算放大后图像各个位置的像素值，例如计算放大后图像位于（136，345）位置的像素值，则136/1.3=104.615,345/1.7=202.941,这里由于示例的原因取小数点后三位，则原图像中相邻的四个位置分别是（104,203），（104,204），（105,203），（105,204）这四个点。

在这里插入图片描述

如图是这个点对应的周围的四个点，203.941-203=0.941

所以f(R1)=(1-0.941)xf(104,203)+0.941xf(104,204),

f(R2)=(1-0.941)xf(105,203)+0.941xf(105,204),

104.615-104=0.615

所以f（P）=(1-0.615)xf(R1)+0.615xf(R2),

其中f表示的那一点的像素值，这样就计算出了f（P）,实际上就是放大后图像(136,345)处对应的像素值。

代码如下：

#include <string.h>

#include <math.h>

#include <stdio.h>

#include <stdlib.h>

#include <malloc.h>

#include<time.h>//时间相关头文件，可用其中函数计算图像处理速度

#define WIDTHBYTES(bits) (((bits)+31)/32\*4)//用于使图像宽度所占字节数为4byte的倍数

//#define MYDRAW\_HEIGHT 986 //目标图像高度

//define MYDRAW\_WIDTH 1572 //目标图像宽度

#define MYDRAW\_HEIGHT 246 //目标图像高度

#define MYDRAW\_WIDTH 393 //目标图像

typedef unsigned char BYTE;

typedef unsigned short WORD;

typedef unsigned long DWORD;

typedef long LONG;

//位图文件头信息结构定义

//其中不包含文件类型信息（由于结构体的内存结构决定，要是加了的话将不能正确读取文件信息）

typedef struct tagBITMAPFILEHEADER {

DWORD bfSize; //文件大小

WORD bfReserved1; //保留字，不考虑

WORD bfReserved2; //保留字，同上

DWORD bfOffBits; //实际位图数据的偏移字节数，即前三个部分长度之和

} BITMAPFILEHEADER;

//信息头BITMAPINFOHEADER，也是一个结构，其定义如下：

typedef struct tagBITMAPINFOHEADER {

//public:

DWORD biSize; //指定此结构体的长度，为40

LONG biWidth; //位图宽

LONG biHeight; //位图高

WORD biPlanes; //平面数，为1

WORD biBitCount; //采用颜色位数，可以是1，2，4，8，16，24，新的可以是32

DWORD biCompression; //压缩方式，可以是0，1，2，其中0表示不压缩

DWORD biSizeImage; //实际位图数据占用的字节数

LONG biXPelsPerMeter; //X方向分辨率

LONG biYPelsPerMeter; //Y方向分辨率

DWORD biClrUsed; //使用的颜色数，如果为0，则表示默认值(2^颜色位数)

DWORD biClrImportant; //重要颜色数，如果为0，则表示所有颜色都是重要的

} BITMAPINFOHEADER;

void main()

{

long now = 0;

now = clock();//存储图像处理开始时间

BITMAPFILEHEADER bitHead, writebitHead;

BITMAPINFOHEADER bitInfoHead, writebitInfoHead;

FILE\* pfile;//输入文件

FILE\* wfile;//输出文件

char strFile[50] = "C:\\testpicture\\1.bmp";//打开图像路径，BMP图像必须为24位真彩色格式

char strFilesave[50] = "C:\\testpicture\\3.bmp";//处理后图像存储路径

fopen\_s(&pfile, strFile, "rb");//文件打开图像

fopen\_s(&wfile, strFilesave, "wb");//打开文件为存储修改后图像做准备

//读取位图文件头信息

WORD fileType;

fread(&fileType, 1, sizeof(WORD), pfile);

fwrite(&fileType, 1, sizeof(WORD), wfile);

if (fileType != 0x4d42)

{

printf("file is not .bmp file!");

return;

}

//读取位图文件头信息

fread(&bitHead, 1, sizeof(tagBITMAPFILEHEADER), pfile);

writebitHead = bitHead;//由于截取图像头和源文件头相似，所以先将源文件头数据赋予截取文件头

//读取位图信息头信息

fread(&bitInfoHead, 1, sizeof(BITMAPINFOHEADER), pfile);

writebitInfoHead = bitInfoHead;//同位图文件头相似

writebitInfoHead.biHeight = MYDRAW\_HEIGHT;//为截取文件重写位图高度

writebitInfoHead.biWidth = MYDRAW\_WIDTH;//为截取文件重写位图宽度

int mywritewidth = WIDTHBYTES(writebitInfoHead.biWidth\*writebitInfoHead.biBitCount);//BMP图像实际位图数据区的宽度为4byte的倍数，在此计算实际数据区宽度

writebitInfoHead.biSizeImage = mywritewidth\*writebitInfoHead.biHeight;//计算位图实际数据区大小

writebitHead.bfSize = 54 + writebitInfoHead.biSizeImage;//位图文件头大小为位图数据区大小加上54byte

fwrite(&writebitHead, 1, sizeof(tagBITMAPFILEHEADER), wfile);//写回位图文件头信息到输出文件

fwrite(&writebitInfoHead, 1, sizeof(BITMAPINFOHEADER), wfile);//写回位图信息头信息到输出文件

int width = bitInfoHead.biWidth;

int height = bitInfoHead.biHeight;

//分配内存空间把源图存入内存

int l\_width = WIDTHBYTES(width\*bitInfoHead.biBitCount);//计算位图的实际宽度并确保它为4byte的倍数

int write\_width = WIDTHBYTES(writebitInfoHead.biWidth\*writebitInfoHead.biBitCount);//计算写位图的实际宽度并确保它为4byte的倍数

BYTE \*pColorData = (BYTE \*)malloc(height\*l\_width);//开辟内存空间存储图像数据

memset(pColorData, 0, height\*l\_width);

BYTE \*pColorDataMid = (BYTE \*)malloc(mywritewidth\*MYDRAW\_HEIGHT);//开辟内存空间存储图像处理之后数据

memset(pColorDataMid, 0, mywritewidth\*MYDRAW\_HEIGHT);

long nData = height\*l\_width;

long write\_nData = mywritewidth\*MYDRAW\_HEIGHT;//截取的位图数据区长度定义

//把位图数据信息读到数组里

fread(pColorData, 1, nData, pfile);//图像处理可通过操作这部分数据加以实现

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*图像处理部分\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*双线性插值\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

for (int hnum = 0; hnum < MYDRAW\_HEIGHT; hnum++)

for (int wnum = 0; wnum < MYDRAW\_WIDTH; wnum++)

{

double d\_original\_img\_hnum = hnum\*height / (double)MYDRAW\_HEIGHT;

double d\_original\_img\_wnum = wnum\*width / (double)MYDRAW\_WIDTH;

int i\_original\_img\_hnum = d\_original\_img\_hnum;

int i\_original\_img\_wnum = d\_original\_img\_wnum;

double distance\_to\_a\_x = d\_original\_img\_wnum - i\_original\_img\_wnum;//在原图像中与a点的水平距离

double distance\_to\_a\_y = d\_original\_img\_hnum - i\_original\_img\_hnum;//在原图像中与a点的垂直距离

int original\_point\_a = i\_original\_img\_hnum\*l\_width + i\_original\_img\_wnum \* 3;//数组位置偏移量，对应于图像的各像素点RGB的起点,相当于点A

int original\_point\_b = i\_original\_img\_hnum\*l\_width + (i\_original\_img\_wnum + 1) \* 3;//数组位置偏移量，对应于图像的各像素点RGB的起点,相当于点B

int original\_point\_c = (i\_original\_img\_hnum + 1)\*l\_width + i\_original\_img\_wnum \* 3;//数组位置偏移量，对应于图像的各像素点RGB的起点,相当于点C

int original\_point\_d = (i\_original\_img\_hnum + 1)\*l\_width + (i\_original\_img\_wnum + 1) \* 3;//数组位置偏移量，对应于图像的各像素点RGB的起点,相当于点D

if (i\_original\_img\_hnum +1== MYDRAW\_HEIGHT - 1)

{

original\_point\_c = original\_point\_a;

original\_point\_d = original\_point\_b;

}

if (i\_original\_img\_wnum +1== MYDRAW\_WIDTH - 1)

{

original\_point\_b = original\_point\_a;

original\_point\_d = original\_point\_c;

}

int pixel\_point = hnum\*write\_width + wnum \* 3;//映射尺度变换图像数组位置偏移量

pColorDataMid[pixel\_point] =

pColorData[original\_point\_a] \* (1 - distance\_to\_a\_x)\*(1 - distance\_to\_a\_y) +

pColorData[original\_point\_b] \* distance\_to\_a\_x\*(1 - distance\_to\_a\_y) +

pColorData[original\_point\_c] \* distance\_to\_a\_y\*(1 - distance\_to\_a\_x) +

pColorData[original\_point\_c] \* distance\_to\_a\_y\*distance\_to\_a\_x;

pColorDataMid[pixel\_point + 1] =

pColorData[original\_point\_a + 1] \* (1 - distance\_to\_a\_x)\*(1 - distance\_to\_a\_y) +

pColorData[original\_point\_b + 1] \* distance\_to\_a\_x\*(1 - distance\_to\_a\_y) +

pColorData[original\_point\_c + 1] \* distance\_to\_a\_y\*(1 - distance\_to\_a\_x) +

pColorData[original\_point\_c + 1] \* distance\_to\_a\_y\*distance\_to\_a\_x;

pColorDataMid[pixel\_point + 2] =

pColorData[original\_point\_a + 2] \* (1 - distance\_to\_a\_x)\*(1 - distance\_to\_a\_y) +

pColorData[original\_point\_b + 2] \* distance\_to\_a\_x\*(1 - distance\_to\_a\_y) +

pColorData[original\_point\_c + 2] \* distance\_to\_a\_y\*(1 - distance\_to\_a\_x) +

pColorData[original\_point\_c + 2] \* distance\_to\_a\_y\*distance\_to\_a\_x;

}

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*双线性插值\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

/\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*图像处理部分\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*\*/

fwrite(pColorDataMid, 1, write\_nData, wfile); //将处理完图像数据区写回文件

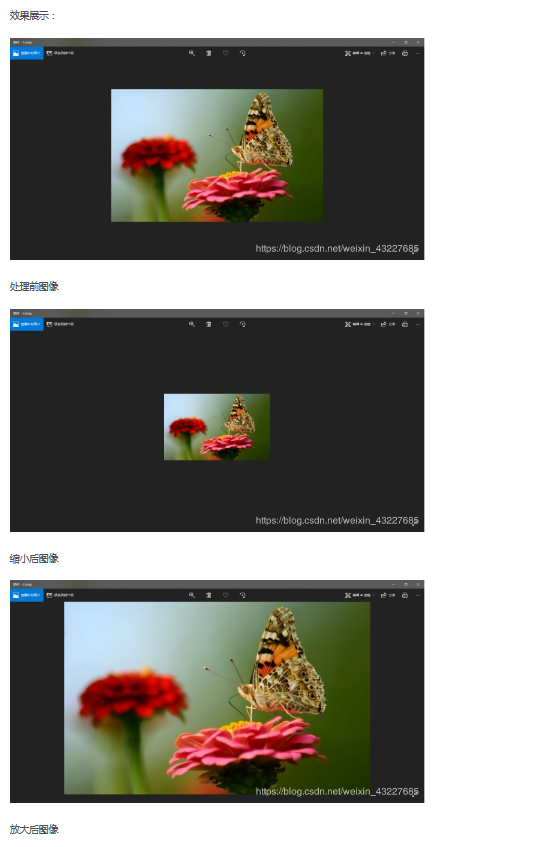
fclose(pfile);

fclose(wfile);

printf("图像处理完成\n");

printf("运行时间为：%dms\n", int(((double)(clock() - now)) / CLOCKS\_PER\_SEC \* 1000));//输出图像处理花费时间信息

}



---------------------

作者：遥远星球

来源：CSDN

原文：https://blog.csdn.net/weixin\_43227685/article/details/88970457

版权声明：本文为博主原创文章，转载请附上博文链接！