BMP图像的结构及读写和灰度化

 (2012-04-07 21:53:35)

[[http://simg.sinajs.cn/blog7style/images/common/sg_trans.gif](javascript:;)转载▼](javascript:;)

|  |  |
| --- | --- |
| 标签：  [bmp](http://search.sina.com.cn/?c=blog&q=bmp&by=tag)    [灰度化](http://search.sina.com.cn/?c=blog&q=%BB%D2%B6%C8%BB%AF&by=tag)    [杂谈](http://search.sina.com.cn/?c=blog&q=%D4%D3%CC%B8&by=tag) | 分类： [学习](http://blog.sina.com.cn/s/articlelist_1736049747_4_1.html) |

**1.文档目的**

本文档主要给出24位真彩BMP图像的结构、读写和灰度化方法。

**2．一般BMP图像的结构**

一般的bmp文件的结结构主要包括文件头，BMP信息头，调色板，位图数据内容

(1)BMP文件头(14字节) ,文件的第0字节到第13字节为BMP图像的文件头。BMP文件头数据结构含有BMP文件的类型、文件大小和位图起始位置等信息。

　　其结构定义如下:

　　typedef struct tagBITMAPFILEHEADER

　　{

　　WORD bfType; // 位图文件的类型，必须为BM(0-1字节)

　　DWORD bfSize; // 位图文件的大小，以字节为单位(2-5字节)

　　WORD bfReserved1; // 位图文件保留字，必须为0(6-7字节)

　　WORD bfReserved2; // 位图文件保留字，必须为0(8-9字节)

　　DWORD bfOffBits; // 位图数据的起始位置，以相对于位图(10-13字节)

　　// 文件头的偏移量表示，以字节为单位

　　} BITMAPFILEHEADER；

(2) BMP信息头

位图信息头(40字节)，文件的第14个字节到第53个字节为BMP图像的信息头，位图信息头数据用于说明位图的尺寸等信息。

　　typedef struct tagBITMAPINFOHEADER{

　　DWORD biSize; // 本结构所占用字节数(14-17字节)

　　LONG biWidth; // 位图的宽度，以像素为单位(18-21字节)

　　LONG biHeight; // 位图的高度，以像素为单位(22-25字节)

　　WORD biPlanes; // 目标设备的级别，必须为1(26-27字节)

　　WORD biBitCount;// 每个像素所需的位数，必须是1(双色), 4(16色)，8(256色)或24(真彩色)之一(28-29字节)

　　DWORD biCompression; // 位图压缩类型，必须是 0(不压缩), 1(BI\_RLE8压缩类型)或2(BI\_RLE4压缩类型)之一(30-33字节)

　　DWORD biSizeImage; // 位图的大小，以字节为单位(34-37字节)

　　LONG biXPelsPerMeter; // 位图水平分辨率，每米像素数(38-41字节)

　　LONG biYPelsPerMeter; // 位图垂直分辨率，每米像素数(42-45字节)

　　DWORD biClrUsed;// 位图实际使用的颜色表中的颜色数(46-49字节)

　　DWORD biClrImportant;// 位图显示过程中重要的颜色数(50-53字节)

　　} BITMAPINFOHEADER;

(3) 调色板

调色板用于说明位图中的颜色，它有若干个表项，每一个表项是一个RGBQUAD类型的结构，定义一种颜色。RGBQUAD结构的定义如下:

　　typedef struct tagRGBQUAD {

　　BYTE rgbBlue;    // 蓝色的亮度(值范围为0-255)

　　BYTE rgbGreen;   // 绿色的亮度(值范围为0-255)

　　BYTE rgbRed;     // 红色的亮度(值范围为0-255)

　　BYTE rgbReserved;// 保留，必须为0

　　} RGBQUAD;

调色板中RGBQUAD结构数据的个数有biBitCount来确定:

当biBitCount=1,4,8时，分别有2,16,256个表项;

当biBitCount=24时，该BMP图像就是24Bit真彩图，没有调色板项。

　　(4)：位图数据内容

位图数据记录了位图的每一个像素值，记录顺序是在扫描行内是从左到右,扫描行之间是从下到上。位图的一个像素值所占的字节数由biBitCount来确定:

当biBitCount=1时，8个像素占1个字节;

　　当biBitCount=4时，2个像素占1个字节;

　　当biBitCount=8时，1个像素占1个字节;

当biBitCount=24时,1个像素占3个字节;

Windows规定一个扫描行所占的字节数必须是4的倍数(即以long为单位),不足的以0填充。

例如：

24Bit真彩图每一行占的实际字节数：

nBytesPerLine =[(bi.biWidth\*3+3)/4\*4   // bi.biWidth为图像宽度

灰度图每一行占的实际字节数：

nBytesPerLine = ((bi.biWidth+3)/4)\*4

**3. 24位真彩BMP图像的结构及灰度图的结构**

     (1)24位真彩BMP图像的结构主要由文件头，BMP信息头， BMP数据内容，没有调色板。

     (2)灰度图的结构主要包括文件头，BMP信息头，调色板，BMP数据内容四部分。灰度图的调色板共有256项RGBQUAD结构，存放0到255的灰度值，每一项rgbRed、rgbGreen、rgbBlue分量值相等。调色板空间的申请的具体设置如下：

     RGBQUAD \*ipRGB2 = (RGBQUAD \*)malloc(256\*sizeof(RGBQUAD));//调色板的大小为1024字节

      for ( i = 0; i < 256; i++ )

      ipRGB2[i].rgbRed = ipRGB2[i].rgbGreen = ipRGB2[i].rgbBlue = i;

**4.BMP图像的读写**

    BMP图像的读:

(1)首先定义BMP文件头和信息头变量

BITMAPFILEHEADER bf; //BMP文件头结构体

BITMAPINFOHEADER bi; //BMP信息头结构体

     (2)创建文件输入流 fp

 fp=fopen(fileName,"rb");   //fileName为BMP图像文件名

(3)读取信息头、文件头

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

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

经过这两条程序把BMP图像的信息头、文件头赋给bf和bi变量，可以根据bf和bi得到图像的各种属性。

(4) 读取BMP调色板

fread(ipRGB2,sizeof(RGBQUAD),256,fp);

(5)读取BMP位图数据

定义一个二维数组Imgdata来存取BMP位图数据

unsigned char \* \* Imgdata;

Imgdata=new unsigned char\*[bi.biHeight]; //声明一个指针数组

for ( i=0;i<bi.biHeight;i++)

        Imgdata[i]=new unsigned char[(bi.biWidth\*3+3)/4\*4]; //每个数组元素也是一个指针数组

       for ( i=0;i<bi.biHeight;i++ )

        for(j=0;j<(bi.biWidth\*3+3)/4\*4;j++)

            fread(&Imgdata[i][j],1,1,fp);//每次只读取一个字节，存入数组

BMP图像的写:

(1)创建一个输出流fp

 fp=fopen("mybmp.bmp","wb");

(2) 写BMP图像的信息头、文件头

fwrite(&bf2,sizeof(BITMAPFILEHEADER),1,fp);

fwrite(&bi2,sizeof(BITMAPINFOHEADER),1,fp);

(3) 写BMP调色板

fwrite(ipRGB2,sizeof(RGBQUAD),256,fp);

(4) 写BMP图像的位图数据部分

for (i=(bi.biHeight)-1 ;i>=0;i--)

        for (j=0 ;j<(bi.biWidth\*3+3)/4\*4;j++)

              fwrite(&Imgdata[i][j],1,1,fp);

**5. 24位真彩BMP图像的灰度化**

      把24位真彩BMP图像转变成256阶灰度图的具体步骤如下：

(1) 修改信息头

       信息头共有11部分，灰度化时需要修改两部分

bi2.biBitCount=8;

bi2.biSizeImage=( (bi.biWidth+3)/4 ) \* 4\*bi.biHeight；

(2)修改文件头

       文件头共有5部分，灰度化时需要修改两部分

          bf2.bfOffBits = sizeof(bf2)+sizeof(BITMAPINFOHEADER)+256\*sizeof(RGBQUAD);

bf2.bfSize = bf2.bfOffBits + bi2.biSizeImage;

(3)创建调色板

RGBQUAD \*ipRGB2 = (RGBQUAD \*)malloc(256\*sizeof(RGBQUAD));

for ( i = 0; i < 256; i++ )

ipRGB2[i].rgbRed = ipRGB2[i].rgbGreen = ipRGB2[i].rgbBlue = i;

      (4)修改位图数据部分

         这部分主要是由原真彩图的rgbRed、rgbGreen、rgbBlue分量值得到灰度图像的灰度值Y，可以用下面公式得到：

            Y=0.299\*rgbRed+0.587\* rgbGreen+0.114\*rgbBlue；

具体修改代码如下：

int nBytesPerLine2 = ( (bi.biWidth+3)/4 ) \* 4;

nLineStart2 = nBytesPerLine2 \* i;

for ( int j = 0; j<nBytesPerLine2;j++ )

ImgData2[nLineStart2+j]= int( (float)Imgdata[i][3 \* j] \* 0.114 + \

(float)Imgdata[i][3 \* j + 1] \* 0.587 + \

(float)Imgdata[i][3 \* j + 2] \* 0.299 );//用一个一维数组顺序存储灰度值

(5)按顺序写入BMP图像的各个部分

          fwrite(&bf2,sizeof(BITMAPFILEHEADER),1,fp);

fwrite(&bi2,sizeof(BITMAPINFOHEADER),1,fp);

fwrite(ipRGB2,sizeof(RGBQUAD),256,fp);

fwrite(ImgData2,nImageSize2,1,fp);

附：BMP的读取和灰度化程序代码：

#include <windows.h>

#include <stdio.h>

#include <stdlib.h>

#include <ctype.h>

#include <string.h>

#include <malloc.h>

int main()

{

    BITMAPFILEHEADER bf; //BMP文件头结构体

    BITMAPINFOHEADER bi; //BMP信息头结构体

    FILE\* fp;           //指向文件的指针

    RGBQUAD \*ipRGB; //

    DWORD LineByte,ImgSize;

    unsigned char \* \* Imgdata;

    int i,j;

    char fileName[256];

    //打开文件

    printf("please enter filename:");

    scanf("%s",fileName);

    fp=fopen(fileName,"rb");

    if(fp == NULL){

        printf("Open file error!");

        exit(0);

    }

    //读取信息头、文件头

    fread(&bf,sizeof(BITMAPFILEHEADER),1,fp); //把指针fp所指向的文件的头信息写入bf（地址）

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

    LineByte=bi.biSizeImage / bi.biHeight; //计算位图的实际宽度并确保它为的倍数

    ImgSize=(DWORD)LineByte\*bi.biHeight;

     Imgdata=new unsigned char\*[bi.biHeight]; //声明一个指针数组

     if(bi.biBitCount==24){

    for ( i=0;i<bi.biHeight;i++)

        Imgdata[i]=new unsigned char[(bi.biWidth\*3+3)/4\*4]; //每个数组元素也是一个指针数组

    for ( i=0;i<bi.biHeight;i++ )

        for(j=0;j<(bi.biWidth\*3+3)/4\*4;j++)

            fread(&Imgdata[i][j],1,1,fp);//每次只读取一个字节，存入数组

}

fclose(fp);

//写入另一个文件

fp=fopen("mybmp.bmp","wb");

//以下定义中带“2”的都是新图像信息。

BITMAPFILEHEADER bf2;

BITMAPINFOHEADER bi2;

int nBytesPerLine2 = ( (bi.biWidth+3)/4 ) \* 4;

int nImageSize2 = nBytesPerLine2 \* bi.biHeight;

  memcpy( &bi2, &bi, sizeof(BITMAPINFOHEADER) );

  bi2.biBitCount=8;

  bi2.biSizeImage=nImageSize2;

bf2.bfType = 0x4d42;

bf2.bfReserved1 = bf2.bfReserved2 = 0;

bf2.bfOffBits = sizeof(bf2)+sizeof(BITMAPINFOHEADER)+256\*sizeof(RGBQUAD);

bf2.bfSize = bf2.bfOffBits + nImageSize2;

fwrite(&bf2,sizeof(BITMAPFILEHEADER),1,fp);

fwrite(&bi2,sizeof(BITMAPINFOHEADER),1,fp);

RGBQUAD \*ipRGB2 = (RGBQUAD \*)malloc(256\*sizeof(RGBQUAD));

for ( i = 0; i < 256; i++ )

ipRGB2[i].rgbRed = ipRGB2[i].rgbGreen = ipRGB2[i].rgbBlue = i;

      fwrite(ipRGB2,sizeof(RGBQUAD),256,fp);

unsigned char \*ImgData2 = new unsigned char[nImageSize2];

int nLineStart2;

for ( i=0;i<bi.biHeight;i++ )

{

nLineStart2 = nBytesPerLine2 \* i;

for ( int j = 0; j<nBytesPerLine2;j++ )

ImgData2[nLineStart2+j]= int( (float)Imgdata[i][3 \* j] \* 0.114 + \

(float)Imgdata[i][3 \* j + 1] \* 0.587 + \

(float)Imgdata[i][3 \* j + 2] \* 0.299 );

}

    fwrite(ImgData2,nImageSize2,1,fp);

free(Imgdata);

fclose(fp);

return 0;

}