图像文件的读写和转换

1）BMP文件格式



Typedef struct tagBITMAPFILEHEADER {

WORD bfType; /\* 说明文件的类型 \*/

DWORD bfSize; /\* 说明文件的大小，用字节为单位 \*/

/\*注意此处的字节序问题

WORD bfReserved1; /\* 保留，设置为0 \*/

WORD bfReserved2; /\* 保留，设置为0 \*/

DWORD bfOffBits; /\* 说明从BITMAPFILEHEADER结构

开始到实际的图像数据之间的字 节

偏移量 \*/

} BITMAPFILEHEADER;

typedef struct tagBITMAPINFOHEADER {

DWORD biSize; /\* 说明结构体所需字节数 \*/

LONG biWidth; /\* 以像素为单位说明图像的宽度 \*/

LONG biHeight; /\* 以像素为单位说明图像的高速 \*/

WORD biPlanes; /\* 说明位面数，必须为1 \*/

WORD biBitCount; /\* 说明位数/像素，1、2、4、8、24 \*/

DWORD biCompression; /\* 说明图像是否压缩及压缩类型

BI\_RGB，BI\_RLE8，BI\_RLE4，BI\_BITFIELDS \*/

DWORD biSizeImage; /\* 以字节为单位说明图像大小 ，必须是4 的整数倍\*/

LONG biXPelsPerMeter; /\* 目标设备的水平分辨率，像素/米 \*/

LONG biYPelsPerMeter; /\*目标设备的垂直分辨率，像素/米 \*/

DWORD biClrUsed; /\* 说明图像实际用到的颜色数，如果为0

则颜色数为2的biBitCount次方 \*/

DWORD biClrImportant; /\*说明对图像显示有重要影响的颜色

索引的数目，如果是0，表示都重要。\*/

} BITMAPINFOHEADER;

typedef struct tagRGBQUAD {

BYTE rgbBlue; /\*指定蓝色分量\*/

BYTE rgbGreen; /\*指定绿色分量\*/

BYTE rgbRed; /\*指定红色分量\*/

BYTE rgbReserved; /\*保留，指定为0\*/

} RGBQUAD;

调用文件头信息头

#include<windows.h>

BITMAPFILEHEADER File\_header;

BITMAPINFOHEADER Info\_header;

// read file & info header

if(fread(&File\_header,sizeof(BITMAPFILEHEADER),1,bmpFile) != 1)

{

printf("read file header error!");

exit(0);

}

if (File\_header.bfType != 0x4D42)

{

printf("Not bmp file!");

exit(0);

}

else

{

printf("this is a %s\n",File\_header.bfType);

}

if(fread(&Info\_header,sizeof(BITMAPINFOHEADER),1,bmpFile) != 1)

{

printf("read info header error!");

exit(0);

}

// end read header

调用调色板

RGBQUAD \*pRGB=(RGBQUAD\*)malloc(sizeof(RGBQUAD)\*(unsigned char)pow(2,info\_h.biBitCount));

if(!MakePalette(pFile,file\_h,info\_h,pRGB))

printf("No palette!");

bool MakePalette(FILE \* pFile,BITMAPFILEHEADER &file\_h,BITMAPINFOHEADER & info\_h,RGBQUAD \*pRGB\_out)

{

if ((file\_h.bfOffBits - sizeof(BITMAPFILEHEADER) - info\_h.biSize) == sizeof(RGBQUAD)\*pow(2,info\_h.biBitCount))

{

fseek(pFile,sizeof(BITMAPFILEHEADER)+info\_h.biSize,0);

fread(pRGB\_out,sizeof(RGBQUAD),(unsigned int)pow(2,info\_h.biBitCount),pFile);

return true;

}

else

return false;

}

2）BMP to YUV文件转换

①转换流程

程序初始化（打开两个文件、定义变量和缓冲区等）

读取BMP文件，抽取或生成RGB数据写入缓冲区

读位图文件头（判断是否可读出、判断是否是BMP文件）

读位图信息头（判断是否读出）

判断像素的实际点阵数

开辟缓冲区，读数据，倒序存放

根据每像素位数的不同，执行不同的操作

（8bit以下，构造调色板，位与移位取像素数据查调色板，写RGB缓冲区

16bit，位与移位取像素数据转换为8bit/彩色分量，写RGB缓冲区

24/32bit，直接取像素数据，写RGB缓冲区）

16bitBMP：

for (Loop = 0;Loop < height \* width;Loop +=2)

{

\*rgbDataOut = (Data[Loop]&0x1F)<<3;

\*(rgbDataOut + 1) = ((Data[Loop]&0xE0)>>2) + ((Data[Loop+1]&0x03)<<6);

\*(rgbDataOut + 2) = (Data[Loop+1]&0x7C)<<1;

rgbDataOut +=3;

}

1~8bitBMP：

int shiftCnt = 1;

while (mask)

{

unsigned char index=mask

==0xFF?Data[Loop]:((Data[Loop]&mask)>>(8-shiftCnt\* info\_h.biBitCount));

\* rgbDataOut = pRGB[index].rgbBlue;

\* (rgbDataOut+1) = pRGB[index].rgbGreen;

\* (rgbDataOut+2) = pRGB[index].rgbRed;

if(info\_h.biBitCount == 8)

mask = 0;

Else

mask >>= info\_h.biBitCount;

rgbDataOut+=3;

shiftCnt ++;

}

调用RGB2YUV的函数实现RGB到YUV数据的转换

采用部分查找表法，提高运行效率

写YUV文件

程序收尾工作（关闭文件，释放缓冲区）

②代码分析





