

哈爾濱工業大學

数字媒体技术
实验报告

题	目	BMP 格式数据处理
学	院	计算机科学与技术
专	业	大数据
学	号	L170300901
学	生	卢兑琬
任	课 教 师	

哈尔滨工业大学计算机科学与技术学院

2021. 3

实验 X:PCM 音频数据、位图数据读取及显示

注意：请按照大家阅读文献的格式进行撰写，确保文档格式的规范性！

一、实验内容或者文献情况介绍

- 熟悉编程环境
- 熟悉 BMP 图像的结构，编程实现 BMP 图像的阅读和显示
 - # 能获取图像任意一点的像素值
 - # 能将图像分成任意块大小，并置乱块的位置；能指定区域内的图像分块并置乱块的大小
- 能阅读 wav 音频文件，并将原始的 PCM 音频数据显示出来，并画出其大小示意图（画出波形图）
- 能调用 DFT，DCT 对图像和音频进行变换处理
 - # 对图像进行二维 DCT 和 DFT 正变换和反变换，并显示正变换后的图像，注意如何才能获得更好的显示效果
 - # 对图像进行分块 8*8DCT 变换后，将其中的 64 个 DCT 系数按照 Zigzag 扫描排序，设定保留的 DCT 系数作为函数参数，然后逆变换，并显示逆变换后的图像，比较原始图像和该图像的 PSNR 值。
- 有以上知识的同学尝试下列任务
 - # 熟悉 JPEG 压缩的流程，对上述 BMP 图像按照 8*8 块分块加密后进行压缩
 - # 对 JPEG 图像进行加密处理

位图格式

- # 每行字节数必须是 4 的整数倍
- # 8 比特及其以下图像都带有调色板，采用调色板的索引值来表示图像的像素值，因此可以是彩色的，例如 GIF 是 8 比特图像
- # 8 比特以上的图像一般没有调色板，直接将图像的 RGB 值放在相应的位置上
- # 位图文件：14 字节的文件头+40 字节的信息头+[调色板]+像素数值

二、算法简介及其实现细节

首先简单的了解并分析 BMP 文件格式，BMP 文件格式的数据可以分四个部分，按照先后顺序分别为 BMP 文件头，位图信息头，调色板，位图数据。其中 BMP 文件头占 14 个字节，位图信息头占 40 个字节，其他部分按照头部信息来决定。

BITMAPFILE 信息(FILE HEAD)

■ BITMAPFILEHEADER(14 bytes)

```
typedef struct tagBITMAPFILEHEADER {  
    WORD bfType;  
    DWORD bfSize;  
    WORD bfReserved1;  
    WORD bfReserved2;  
    DWORD bfOffBits;  
} BITMAPFILEHEADER, *PBITMAPFILEHEADER;
```

UINT16 bfType; //2Bytes, 必须为“BM”, 即 0x424D 才是 Windows 位图文件
DWORD bfSize; //4Bytes, 整个 BMP 文件的大小
UINT16 bfReserved1; //2Bytes, 保留, 为 0
UINT16 bfReserved2; //2Bytes, 保留, 为 0
DWORD bfOffBits; //4Bytes, 文件起始位置到图像像素数据的字节偏移量

变量名	地址偏移	大小	作用说明
bfType	0000h	2Bytes	文件标识符, 必须为“BM”, 即0x424D 才是 Windows位图文件 'BM': Windows 3.1x, 95, NT,... 'BA': OS/2 Bitmap Array 'CI': OS/2 Color Icon 'CP': OS/2 Color Pointer 'IC': OS/2 Icon 'PT': OS/2 Pointer 因为OS/2系统并没有被普及开, 所以在编程时, 你只需判断第一个标识“BM”就行
bfSize	0002h	4Bytes	整个BMP文件的大小 (以位B为单位)
bfReserved1	0006h	2Bytes	保留, 必须设置为0
bfReserved2	0008h	2Bytes	保留, 必须设置为0
bfOffBits	000Ah	4Bytes	说明从文件头0000h开始到图像像素数据的字节偏移量 (以字节Bytes为单位), 以为位图的调色板长度根据位图格式不同而变化, 可以用这个偏移量快速从文件中读取图像数据

确认是否为 BITMAPFILE 的变数就是 BfType。BMP 文件的前两个字节常存有“BM”字。bfOffBits 从文件起始部分到实际视频数据存在的位置, 显示字节单位的距离。WORD 被定义为 unsigned short, DWORD 被定义为 unsigned long。

关于影像本身的信息 (影像头条)

■ BITMAPINFOHEADER(40 Bytes)



```
typedef struct tagBITMAPINFOHEADER{  
    DWORD biSize;  
    LONG biWidth;  
    LONG biHeight;  
    WORD biPlanes;(1)  
    WORD biBitCount;  
    DWORD biCompression;  
    DWORD biSizeImage;  
    LONG biXPelsPerMeter;  
    LONG biYPelsPerMeter;  
    DWORD biClrUsed;  
    DWORD biClrImportant;  
} BITMAPINFOHEADER, *PBITMAPINFOHEADER;
```

DWORD biSize; //4Bytes, INFOHEADER 结构体大小, 存在其他版本 INFOHEADER, 用作区分

LONG biWidth; //4Bytes, 图像宽度 (以像素为单位)

LONG biHeight; //4Bytes, 图像高度, +: 图像存储顺序为 Bottom2Top, -: Top2Bottom

WORD biPlanes; //2Bytes, 图像数据平面, BMP 存储 RGB 数据, 因此总为 1

WORD biBitCount; //2Bytes, 图像像素位数

DWORD biCompression; //4Bytes, 0: 不压缩, 1: RLE8, 2: RLE4

DWORD biSizeImage; //4Bytes, 4 字节对齐的图像数据大小

LONG biXPelsPerMeter; //4 Bytes, 用像素/米表示的水平分辨率

LONG biYPelsPerMeter; //4 Bytes, 用像素/米表示的垂直分辨率

DWORD biClrUsed; //4 Bytes, 实际使用的调色板索引数, 0: 使用所有的调色板索引

DWORD biClrImportant; //4 Bytes, 重要的调色板索引数, 0: 所有的调色板索引都重要

变量名	地址偏移	大小	作用说明
biSize	000Eh	4Bytes	BNP信息头即BMP_INFOHEADER结构体所需要的字节数（以字节为单位）
biWidth	0012h	4Bytes	说明图像的宽度（以像素为单位）
biHeight	0016h	4Bytes	说明图像的高度（以像素为单位）。这个值还有一个用处，指明图像是正向的位置还是倒向的位置，该值是正数说明图像是倒向的即图像存储是由下到上；该值是负数说明图像是倒向的即图像存储是由上到下。大多数BMP位图是倒向的位置，所以此值是正值。
biPlanes	001Ah	2Bytes	为目标设备说明位面数，其值总设置为1
biBitCount	001Ch	2Bytes	说明一个像素点占几位（以比特位/像素位单位），其值可为1,4,8,16,24或32
biCompression	001Eh	4Bytes	说明图像数据的压缩类型，取值范围为： 0 BI_RGB 不压缩（最常用） 1 BI_RLE8 8比特游程编码（BLE），只用于8位位图 2 BI_RLE4 4比特游程编码（BLE），只用于4位位图 3 BI_BITFIELDS比特域（BLE），只用于16/32位位图 4
biSizeImage	0022h	4Bytes	说明图像的大小，以字节为单位。当用BI_RGB格式时，总设置为0
biXPelsPerMeter	0026h	4Bytes	说明水平分辨率，用像素/米表示。有符号整数
biYPelsPerMeter	002Ah	4Bytes	说明垂直分辨率，用像素/米表示。有符号整数
biClrUsed	002Eh	4Bytes	说明位图实际使用的调色板索引数，0：使用所有的调色板索引
biClrImportant	0032h	4Bytes	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。

调色板

调色板是一种用于存储因时间标刻而产生色值的结构。使用该结构体，按调色板数量分配和储存排列。256 彩色模式的调色板尺寸为256，而16 位彩色视频的调色板尺寸为 2^{16} 。

■ RGBQUAD

```
typedef struct tagRGBQUAD {
    BYTE rgbBlue;
    BYTE rgbGreen;
    BYTE rgbRed;
    BYTE rgbReserved;
} RGBQUAD;
```

```
BYTE rgbBlue; //指定蓝色强度
```

```
BYTE  rgbGreen;      //指定绿色强度
BYTE  rgbRed;        //指定红色强度
BYTE  rgbReserved;   //保留，设置为 0
```

1, 4, 8 位图像才会使用调色板数据, 16, 24, 32 位图像不需要调色板数据, 即调色板最多只需要 256 项 (索引 0 - 255)。

颜色表的大小根据所使用的颜色模式而定: 2 色图像为 8 字节; 16 色图像位 64 字节; 256 色图像为 1024 字节。其中, 每 4 字节表示一种颜色, 并以 B (蓝色)、G (绿色)、R (红色)、alpha (32 位位图的透明度值, 一般不需要)。即首先 4 字节表示颜色号 1 的颜色, 接下来表示颜色号 2 的颜色, 依此类推。

颜色表中 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=1 时, 为 2 色图像, BMP 位图中有 2 个数据结构 RGBQUAD, 一个调色板占用 4 字节数据, 所以 2 色图像的调色板长度为 2×4 为 8 字节。

当 biBitCount=4 时, 为 16 色图像, BMP 位图中有 16 个数据结构 RGBQUAD, 一个调色板占用 4 字节数据, 所以 16 色的调色板长度为 16×4 为 64 字节。

当 biBitCount=8 时, 为 256 色图像, BMP 位图中有 256 个数据结构 RGBQUAD, 一个调色板占用 4 字节数据, 所以 256 色图像的调色板长度为 256×4 为 1024 字节。

当 biBitCount=16, 24 或 32 时, 没有颜色表。

DIB 和 DDB

DDB 是 Device Dependent Bitmat 的缩写, 是设备从属的位图;

DIB 是 Device Independent Bitmat 的缩写, 是设备独立的位图。

换句话说, DIB 是指无论在哪个机种上看到, 无论是 PC 还是 MAC, 还是用手机看, 或者通过 PDA 看, 都会出现相同的图片和颜色。

BMP 中使用 DIB 时的注意事项

在保存位图影像时, 图像会被反过来保存。即, 在位图中将视频数据重新保存为用于影像处理的排列时, 要反过来保存。

```

#include <stdio.h>
#include <windows.h>
#define WIDTHBYTES(bytes) (((bytes)+3)/4*4)
int main(int argc, char *argv[])
{
    FILE *file;
    BITMAPFILEHEADER hf;
    BITMAPINFOHEADER hInfo;
    RGBQUAD rgb[256];
    int widthStep;
    BYTE *lpImg;
    BYTE *lpOutImg;
    int x, y;
    if (argc < 3) {
        printf("Insufficient Input Arguments \n");
        printf("  invertImage input_file ouput_file \n");
        return -1;
    }
}

```

该程序通过输入 8bit gray-scale 及 true color 影像进行反转后, 以相同格式的影像进行保存。

```

fread(&hf, sizeof(BITMAPFILEHEADER), 1, file);
if (hf.bfType != 0x4D42) {
    printf("Not a BMP file \n");
    return -1;
}
fread(&hInfo, sizeof(BITMAPINFOHEADER), 1, file);
printf("Size: (%3dx%3d) \n", hInfo.biWidth, hInfo.biHeight);
if ((hInfo.biBitCount != 8 || hInfo.biClrUsed != 0) && hInfo.biBitCount != 24) {
    printf("It's not an 8BIT GRAY-SCALE image \n");
    return -1;
}
if (hInfo.biBitCount == 8) {
    fread(rgb, sizeof(RGBQUAD), 256, file);
}
widthStep = WIDTHBYTES((hInfo.biBitCount / 8) * hInfo.biWidth);
fseek(file, hf.bfOffBits, SEEK_SET);
lpImg = (BYTE *)malloc(widthStep * hInfo.biHeight);
fread(lpImg, sizeof(BYTE), widthStep*hInfo.biHeight, file);
fclose(file);
lpOutImg = (BYTE *)malloc(widthStep * hInfo.biHeight);

```

为输入数据计算每条生产线的字节数。bit 向数据开始的位置移动。存储输入数据的存储器分配。在输入影像中读取影像数据。存储结果数据的内存分配。

```

if (hInfo.biBitCount == 24) {
    for (y = 0; y < hInfo.biHeight; y++) {
        for (x = 0; x < hInfo.biWidth; x++) {
            lpOutImg[y*widthStep + 3 * x + 2] = 255 - lpImg[y*widthStep + 3 * x + 2]; /* R */
            lpOutImg[y*widthStep + 3 * x + 1] = 255 - lpImg[y*widthStep + 3 * x + 1]; /* G */
            lpOutImg[y*widthStep + 3 * x + 0] = 255 - lpImg[y*widthStep + 3 * x + 0]; /* B */
        }
    }
}
else if (hInfo.biBitCount == 8) {
    for (y = 0; y < hInfo.biHeight; y++) {
        for (x = 0; x < hInfo.biWidth; x++) {
            lpOutImg[y*widthStep + x] = 255 - lpImg[y*widthStep + x];
        }
    }
}
file = fopen(argv[2], "wb");

fwrite(&hf, sizeof(char), sizeof(BITMAPFILEHEADER), file);
fwrite(&hInfo, sizeof(char), sizeof(BITMAPINFOHEADER), file);
if (hInfo.biBitCount == 8) {
    fwrite(rgb, sizeof(RGBQUAD), 256, file);
}

```

影像反转运算

```

    fseek(file, hf.bfOffBits, SEEK_SET);
    fwrite(lpOutImg, sizeof(BYTE), widthStep*hInfo.biHeight, file);
    fclose(file);
    free(lpOutImg);
    free(lpImg);
    return 0;
}

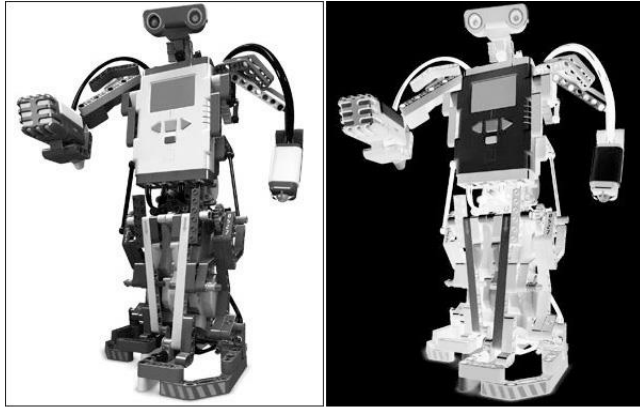
```

向 bitmap 数据开始的位置移动内存解除

三、 实验设置及结果分析（包括实验数据集）

Inverting images

out = 255 - in



四、 结论

通过该实验，我们理解了位图格式结构，基本上设备独立成像文件存储标准规格存在 JPEG、GIF、BMP、TIFF、PCX、PGM 等多个规格，以及常用的成像格式使用压缩算法转换成小尺寸。

五、 参考文献

- [1] James D. Murray; William vanRyper (April 1996). "Encyclopedia of Graphics File Formats" (Second ed.). O'Reilly. bmp. ISBN 1-56592-161-5. Retrieved 2014-03-07.
- [2] 심계은, “디지털 시각 요소를 이용한 써피스 디자인 의 표현 방법 연구,” 건국대학교 대학원, 2011.
- [3] 김영준, 최화재, 김휘강, “24Bit BMP 이미지를 이용 한 셀코드 은닉 기법,” 정보보호학회논문지 22(3), pp.691-705, 2012.6.