AVI 文件结构的实例分析

黄东军，贺宏遵
（中南大学 信息科学与工程学院，湖南 长沙 410083）

摘 要：AVI 文件是研究视频技术的重要工具。正确理解 AVI 文件结构并掌握 AVI 图像数据操作，是实现各种基于 AVI 文件的视频处理的必要条件。文章结合一个具体 AVI 文件的 debug 分析，展示了详尽而具体的 AVI 文件内容，这项研究对希望深入了解视频图像处理技术的人员有较大参考价值。

关键词：AVI 文件；格式分析；数据操作
中图分类号：TP37 **文献标识码**：A **文章编号**：1006-8937(2008)03-0003-04

Research on operating the image pixels of AVI file through specific example

HUANG Dong-jun, HE Hong-zun
(College of Information Science and Engineering, Central South University, Changsha, Hunan 410083, China)

Abstract: AVI is one of the most important format of video files which can be widely used by various researchers of video techniques. Understanding the format and knowing how to use an AVI file is necessary for any one who pursues video techniques. This paper shows the format of an AVI file in detail by a specific example. Finally, a general description is given. We believe that many learners can benefit from this study.

Keywords: AVI file; format analysis; data operation

AVI(Audio Video Interleaved)是 Windows 操作系统上最基本、最常用的一种视频格式文件，多用于音视频捕捉、编辑、回放等应用程序中。由于声音和图像数据在 AVI 文件中通常是未经压缩的，因此人们可以方便地操作这些数据，例如对图像数据实施二值化、淡入、淡出、缩放等各种处理，所以 AVI 文件是我们研究视频技术的重要工具[1]。

已经有不少文章讨论 AVI 文件的格式分析、播放程序设计、AVI 文件分解与合并等，但是这些文章多半是英文版 AVI 规范的翻译，对程序设计方法也多为抽象说明[2-3]。正确理解 AVI 文件结构并掌握图像数据操作，是完成各种基于 AVI 的视频开发的必备条件。本文结合一个具体 AVI 文件的 debug 分析，展示了详尽而具体的 AVI 文件内容。通过示例，读者容易建立对 AVI 文件的直观印象，然后再用一般结构进行归纳，一个完整而形象的认识就建立起来了。因此，本文的工作对希望深入了解 AVI 视频图像处理技术的人有较大参考价值。

1 AVI 文件格式的实例分析

通常情况下，一个 AVI 文件可以包含多个不同类型的媒体流（典型的情况下有一个音频流和一个视频流），含有单一音频流或单一视频流的 AVI 文件也是合法的。为简明起见，本文拟集中研究仅含图像序列的 AVI 文件。

AVI 文件是一种 RIFF(Resource Interchange File Format，资源交换文件格式)格式文件。RIFF 文件使用四字符码 FOURCC(four-character code)来表征数据类型，比如用‘RIFF’表示格式类型，用“AVI”表示具体的文件类型，用‘LIST’表示列表等。“AVI”四字符码中含有空格，这也是合法的。

RIFF 文件首先含有一个如图 1 所示的文件头结构。

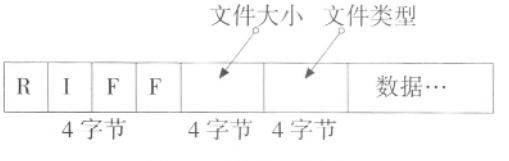

图 1 RIFF 文件结构

图 2 是用 debug 程序打开 AVI 文件后所显示的文件最开始部分的十六进制数据。可以看出前 4

收稿日期：2007-12-07
作者简介：黄东军(1966—)，男，湖南常德人，博士研究生，教授，主要研究方向：网络与多媒体技术。

个字符为 RIFF，随后是文件大小（F0 D2 01 00）。这个数的二进制表示应当是这样的：00000000 00000001 11010010 11110000，即 0x00 到了前面，而 0xF0 则放在了最后，这样才能正确换算为十进制数。文件大小后面又是一个四字符码，描述文件的具体类型（比如 AVI、WAVE 等），这里是“AVI ”，AVI 后面有一个空格。

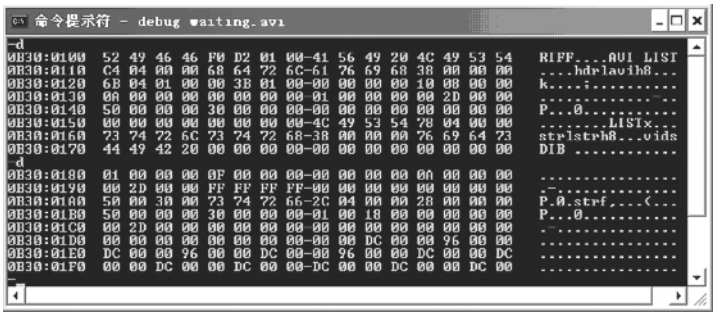

图 2 用 debug 程序打开 AVI 文件后所显示的文件最开始部分的十六进制数据

RIFF 文件的数据从 LIST 开始。为直观起见，下面按顺序依次说明各个部分的格式。

在图 2 中，我们可以看到“LIST”四个字符，其后为 LIST 的大小，是 0xC4040000；接着是“hdrl”，表示下面是头部列表的数据；然后是“avih”，表示 AVI 头部；接着是 AVI 头部的大小：0x38000000。AVI 头部大小的后面跟的是什么？在图 2 中不容易看出来，实际上是一个如下的数据结构：

```
typedef struct
{
 DWORD dwMicroSecPerFrame; //显示每桢所需的时间 ns，定义 avi 的显示速率
 DWORD dwMaxBytesPerSec; //最大的数据传输率
 DWORD dwPaddingGranularity; //记录块的长度需为此值的倍数，通常是 2048
 DWORD dwFlages; //AVI 文件的特殊属性，如是否包含索引块，音视频数据是否交叉存储
 DWORD dwTotalFrame; //文件中的总桢数
 DWORD dwInitialFrames; //说明在开始播放前需要多少桢
 DWORD dwStreams; //文件中包含的数据流种类
 DWORD dwSuggestedBufferSize; //建议使用的缓冲区的大小，
 //通常为存储一桢图像以及同步声音所需要的数据之和
 WORD dwWidth; //图像宽
 DWORD dwHeight; //图像高
 DWORD dwReserved[4]; //保留值
}MainAVIHeader;
```

这个数据结构占据 56 个字节，正是图 2 中从 k 开始到第二个 LIST 前的这一段。这个数据结构告诉我们很多关于这个 AVI 文件的信息，如显示速率等，这就为打开该 AVI 文件的程序提供了播放根据。

MainAVIHeader structure 后面是所谓的 AVI 流头部列表，也是以“LIST”开始，接着是这个 LIST 的大小：0x78040000；然后是“strl”，表示这个列表的数据是流列表类型。之后紧跟着的是“strh”块，表示流头部，后面是这个流头部的大小：0x38000000。接着是什么？是和“LIST”类似的标识符“vids”吗？不是。跟在流头部大小后面的又是一个如下的数据结构：

```
typedef struct
{
 FOURCC fccType; //4 字节，表示数据流的种类 vids 表示视频数据流
 //auds 音频数据流
 FOURCC fccHandler; //4 字节，表示数据流解压缩的驱动程序代号
 DWORD dwFlags; //数据流属性
 WORD wPriority; //此数据流的播放优先级
 WORD wLanguage; //音频的语言代号
 DWORD dwInitalFrames; //说明在开始播放前需要多少桢
 DWORD dwScale; //数据量，视频每桢的大小或者音频的采样大小
 DWORD dwRate; //dwScale /dwRate=每秒的采样数
 DWORD dwStart; //数据流开始播放的位置，以 dwScale 为单位
 DWORD dwLength; //数据流的数据量，以 dwScale 为单位
 DWORD dwSuggestedBufferSize; //建议缓冲区的大小
 DWORD dwQuality; //解压缩质量参数，值越大，质量越好
 DWORD dwSampleSize; //音频的采样大小
 RECT rcFrame; //视频图像所占的矩形
```

}AVIStreamHeader;

显然,这是 AVI 流头部结构,它描述了这个流的类型、压缩方式等属性。这个结构也占据 56 字节,从"vids"到"strf"之前。现在来看看"vids"是什么。原来它是 AVI 流头部结构内部的一个四字符码,表示这个流是视频数据,不是音频数据。"vids"后面是"DIB ",注意后面有一个空格;它表示这个视频流的图像是没有压缩的 DIB 位图,即设备无关位图。

AVI 流头部结构后面是流格式块,以"strf"标识。"strf"标识后是这个块的大小:0x2C040000。接着是BITMAPINFO 结构:

```
typedef struct tagBITMAPINFO
{
 BITMAPINFOHEADER bmiHeader;
 RGBQUAD bmiColors[1]; //颜色表
}BITMAPINFO;
```

其中的 BITMAPINFOHEADER 是位图信息头结构,如下所示:

```
typedef struct tagBITMAPINFOHEADER
{
 DWORD biSize; //位图信息头结构的大小
 LONG biWidth; //图像宽度
 LONG biHeight; //图像高度
 WORD biPlanes; //目标设备位面数,设为 1
 WORD biBitCount; //单位像素的位数,即图像的位深度
 DWORD biCompression; //图像的压缩类型
 DWORD biSizeImage; //图像的大小, 以字节为单位
 LONG biXPelsPerMeter; //水平方向每米像素数
 LONG biYPelsPerMeter; //垂直方向每米像素数
 DWORD biClrUsed; //实际使用的色彩表中的颜色索引数
 DWORD biClrImportant; //重要颜色数
}BITMAPINFOHEADER;
```

位图信息头结构的大小是 40(字节),这正是 0x28000000 换算的十进制数。接着,我们看到了图像的宽度和高度 本例是 80×48)。现在让我们仔细看一下该 AVI 文件的图像位深度:由于位图信息头结构的大小、图像宽度、图像高度占 12 个字节,加上目标设备位面数占两个字节(0x0100),因此位深度是 0x1800(占两个字节), 稍加换算即可知道为 24。这表明图像是 24 位的,是典型的 RGB 彩色模型位图。

虽然位图信息头结构的大小是 40 字节, 但是整个"strf"块的大小是 0x2C040000,即 1 068 个字节,这中间大部分是颜色表。

颜色表后面是"JUNK"四字符码,如图 3 所示,

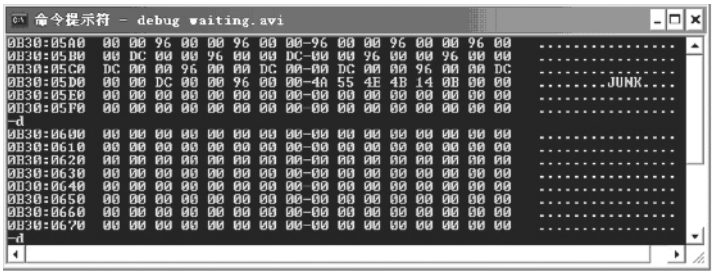

图 3 JUNK列表

这是 JUNK 列表。

JUNK 列表在一般的 AVI 格式分析中都没有介绍[4],但是现在我们遇到了。这说明 AVI 文件的格式很灵活。实际上 JUNK 列表中的数据均为零,这从图 3 看得很清楚。JUNK 列表起占位作用,用户可以在其中写入各种信息(所以并非垃圾块)。从图 3 还可以看出,JUNK 列表的大小是 0x140B0000,即 3 904 个字节。

越过 JUNK 列表, 就是最重要的图像数据本身了,如图 4 所示。图像数据是一个列表,所以用"LIST"开始, 后面是这个 LIST 的大小, 为 0x54C20100,即 115 284 字节,可见这是整个 AVI 文件的主体。LIST 大小后面是"movi"四字符码,表示视频块的开始。

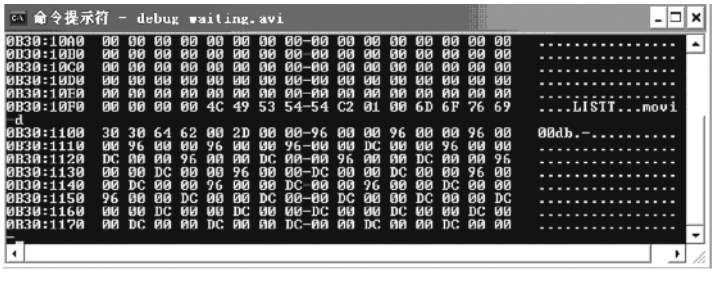

图 4 图像数据列表

视频图像序列到底如何排列呢? 原来,图像数据列表中包含着以帧为单位的视频序列图像块。每一块(也就是一帧) 以"00db"开始,接着是块大小,这里是 0x002D0000,即 11 520 字节,这正是 80×48×3 的计算结果。所以,后面的每一帧都是以"00db"开始的块。了解这一点很重要,程序在读取一帧图像

数据时必须越过这 8 个字节。

至此，我们通过一个特例，对 AVI 文件的结构按照存储顺序作了说明。有了这样一个直观认识，再来看 AVI 规范的描述就比较容易理解了。

2 AVI 文件的一般结构

图5 是 AVI 文件格式的一般描述。结合前面的示例，可以看出：整个 AVI 文件在逻辑上由文件头和数据（data）构成。而数据是指文件包含的列表，如 header list、info list、movi list、indx list 等。列表又可以包含子列表和块，如 stream header list 列表、AVI header 块等。这里，已经不难理解什么是列表、什么是块了。显然，列表是含有子列表和块的数据类型，而块则是基本的数据类型。总之，一个 AVI 文件就是通过列表和块来组织数据的。

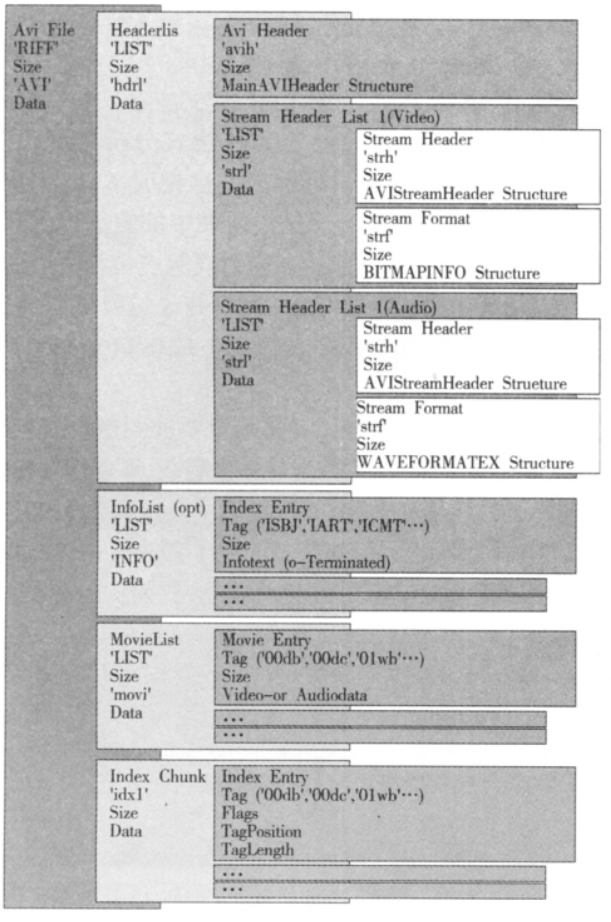

图 5 AVI 文件的一般结构

本文虽然以一个仅包含视频数据的 AVI 文件来说明文件结构，但是已经不难理解含有音频序列的 AVI 文件了。图 5 中，stream header list 1 后面可以再跟一个 stream header list 2，这个子列表就是用来描述音频流的，与视频流的描述类似，音频流列表中含有波形声音数据结构 WAVEFORMATEX structure，它描述了波形音频的特征。如果一个 AVI 文件同时含有音频和视频序列，那么在后面的 Movie list 中，就会交错地摆放音频帧与视频帧。而一个音频帧就是以‘00wb’四字符码开始的。

由于 AVI 文件格式具有很大的灵活性，因此有时候一个具体的 AVI 文件还可能包含信息列表、索引列表，甚至 JUNK 列表。

3 结 语

AVI 是目前最为复杂的 RIFF 文件，理解它的前提条件还包括：熟练掌握 DIB 图像的相关数据结构和图像的实际存储结构（包括压缩和非压缩两种情形）、波形音频的相关数据结构和存储结构。掌握了 AVI 文件结构，再辅以相关知识，就能顺利操作视频和音频数据了[5-6]。

参考文献：

[1] 黄晓鹰.Visual C++6.0 显示 AVI 文件的研究[J].光电工程，2003,(3):26.
[2] 谢维，李华.视频文件格式及转换[J].电子与电脑，2002,(5):12-16.
[3] 冯传岗.当代视频文件格式之纵观[J].现代电视技术，2005,(2):21-25.
[4] 吕四化，史萍.视频文件格式研究[J].北京广播学院学报（自然科学版），2004,(4):11-17.
[5] 周俊.谈谈 AVI 文件的修复[J].电脑知识与技术（经验技巧），2003,(1):8-11.
[6] 许先斌，朱平，安晖.扩展格式 AVI 文件播放的编程方法[J].计算机工程与设计，2003,(3):65-68.