Img 图像数据格式分析及超大数据量快速读取方法*

朱 政，刘仁义，刘 南
（浙江大学 GIS 省重点实验室，浙江 杭州 310028）

摘 要：Img 格式是一种常用的遥感图像格式，在遥感图像处理软件中应用较多，但对其格式进行深入剖析和读写操作介绍的有关文献较少。在自主开发图像处理软件平台研发工作中，读写 Img 格式是一项基础、重要的工作。对 Img 格式进行了探讨，并给出了读取 Img 格式文件的实例。

关键词：Img 格式；结构列表；节点
中图法分类号：TP391 **文献标识码**：A **文章编号**：1001-3695(2003)08-0060-02

Analysis of Img Format and a Method of Reading for the Large Img Image

ZHU Zheng, LIU Ren-yi, LIU Nan
(*Key Laboratory of GIS, Zhejiang University, Hangzhou Zhejiang* 310028, *China*)

Abstract: Img is a familiar remote sensing image format, used frequently in the remote sensing image disposal software. But the literature about Img format and the operation of reading and writing this format is less. Reading and writing Img format is an important and basic task in developing self-determination software. This paper not only discusses the Img format, but also gives an example of reading Img format

Key words: Img Format; Structure List; Node

1 引言

地理信息系统(GIS)中图像数据处理一直是一个重要的研究课题。图像数据格式的读写是其它工作的基础和前提。目前图像数据的处理和应用格式很多，Img 格式是一种组织结构比较好的数据格式，在各种 GIS 平台的图像处理系统中应用很多。但关于 Img 数据格式的详细分析很少有人介绍，笔者在承担的国家“863”项目中，因研究工作的需要，在 Img 图像格式的读取方面做了一些探索和实践，实现了高质量快速读取超大数据量多波段 Img 图像。

2 Img 文件的基本格式

2.1 Img 格式基本对象

Img 格式非常灵活，由一系列节点构成(图 1)。

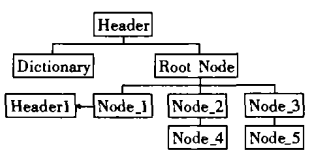

图 1 Img 格式的基本结构图

每一节点存储不同的信息，都有自己的头文件，节点与节点之间的联系通过节点的头文件来实现。每一个 Img 格式由 Ehfa _ HeaderTag，Ehfa _ File，Ehfa _ Entry 三个基本对象组成。由于没有自己的头文件，不能成为 Img 格式的节点。这三个对象的格式如下：

(1)Img 格式总的头文件 Ehfa _ HeaderTag 对象。

```
Char[16] label;    /* Img 文件的标志 EHFA _ HEADER _ TAG */
Long headerPtr;    /* 节点 Ehfa _ File 的位置 */
```

(2)Ehfa _ File 对象的格式。

```
Long version;              /* version number,通常是 1 */
Long freelist;
Long rootEntryPtr;         /* 根节点的位置 */
Short entryHeaderLength;   /* 每一个节点头文件的长度 */
Long dictionaryPtr;        /* 各个节点存储结构列表的位置 */
```

Ehfa _ File 对象里存储着两个非常重要的信息：一个是 rootEntryPtr，一个是 Dictionary。rootEntryPtr 存储根节点的地址，为文件真正信息的开始位置，其它节点读取从 rootEntryPtr 开始；Dictionary 存储该图像文件中所有的节点的存储结构列表的地址。

(3)每一个节点的基本信息以及各个节点的联系是通过 Ehfa _ Entry 对象来实现的。Ehfa _ Entry 是每一个节点的头文件的存储结构。Ehfa _ Entry 对象格式如下：

```
Long next;        /* 下一个节点的位置 */
Long prev;        /* 前一个节点的位置 */
Long parent;      /* 父节点的位置 */
Long child;       /* 第一个子节点的位置 */
Long data;        /* 数据的存放位置 */
Long datasize;    /* 数据大小 */
Char[64] name;    /* 节点的名字 */
Char[32] type;    /* 节点的存储结构 */
TIME modTime;     /* 此节点的修改时间 */
```

前面四个变量是各个节点之间联系的纽带，变量 Name 是一个由程序员定义的变量，一般与节点存储的信

收稿日期：2002-09-07；**修返日期**：2002-09-23
基金项目：国家“863”攻关资助项目(2001AA630301)

息相关。变量 Type 存放所在节点的存储结构的名字,如 Eimg _ Layer。任何一个节点头文件的变量 Type 存储的结构在文件尾的结构列表里都有说明。

2.2 Img 格式的节点存储结构列表

节点的存储结构列表一般放在文件尾,存储地址存放在对象 Ehfa _ File 的 Dictionary 变量里。在 Img 格式的图像文件中,包含的每一个节点的结构在未读取该图像前都是未知的。在从根节点向下读取数据之前,必须先读取这个结构列表。采用字符数组读取的 Img 格式结构列表如下:

```
{1: lversion, 1: lfreeList, 1: lrootEntryPtr, 1: sentryHeaderLength, 1: ldictionaryPtr,}
Ehfa _ File,
{1: lnext, 1: lprev, 1: lparent, 1: lchild, 1: ldata, 1: ldatasize, 64: cname, 32: ctype, 1: tmodeTime,}
Ehfa _ Entry,…
```

上面只列出了两个存储结构:Ehfa _ File 和 Ehfa _ Entry,后面是存储结构的名称,里面是存储结构的具体格式。里面各个变量以逗号隔开,逗号之间为每一个变量的存储类型及存储数量等,例如:

```
1:lversion,/* 数字 1 表示一个变量;字母 l 表示是 Long 型的;Version 表示变量的名称 */
64:cname,/* 数字 64 表示 64 个变量,字母 c 表示 Char 型,Name 表示是变量的名称 */
```

每一个节点的存储结构都按照上述方式组织。读取了这个列表以后,才有可能从根节点开始按照节点的存储结构读取每一个节点的信息。一般来说,用户不必修改通用节点的存储结构。只有当用户需要加一些特殊意义的节点时,才按照自己的方式来定义存储结构。

2.3 Img 格式的一个重要节点及子节点

Eimg _ Layer 是 Img 图像格式中一个最基本的节点结构,它描述了一个图层的基本信息,包括图层的行列数、块的高度和宽度等。Eimg _ Layer 有很多子节点,包括 RasterDMS(数据节点)、Projection(投影信息)、Map _ Info(图像边界信息)、Statistics(统计值信息)等。一个波段对应一个以 Eimg _ Layer 为存储结构的节点。多波段的 Img 格式图像有多个以 Eimg _ Layer 为存储结构的节点,分别以名字来区别,如 band1,band2 等。Eimg _ Layer 节点的存储结构如下:

```
Long width;          /* 图层的宽度 */
Long height;         /* 图层的长度 */
Enum layerType;      /* 图层的类型 */
Enum pixelType;      /* 图层像素的存储类型 */
Long blockWidth;     /* 图层块的宽度 */
Long blockHeight ;   /* 图层块的长度 */
```

变量 Pixeltype 存储文件中一个像素占的字节数。Img 图像格式支持多种存储类型,如 Unsigned 4-bit, Unsigned 8-bit, Signed 8-bit, Unsigned 16-bit, Signed 16-bit, Unsigned 32-bit, Signed 32-bit, Float 32-bit, Float 64-bit 等。Blockwidth, Blockheight 将在下一小节中进行介绍。

3 Img 文件读取方法

3.1 基本信息的读取

根据头文件 Type 中存储的结构名称,从根节点出发,首先读取节点的头文件,到 Dictionary 里面找到存储结构,然后,按照 Data 里面存储的地址按结构读取具体的数据,最后根据头文件里变量 Next 的值进行下一个节点的读取,以此类推,直到最后一个节点。当然也可以先读全部头文件,把头文件中的信息都保存下来,根据头文件的信息,选取一些自己需要的节点进行读取。

3.2 图像数据的读取

Img 图像格式除了可以灵活地存储各种信息外,还有一个重要的特点是图像的分块存储。一幅 Img 图像按照其行列数被分成了 n 块,如 512 × 512(行 × 列)的图像被分成了 64 块(横向为 8 行,纵向为 8 列),每一块的大小是 64 × 64。块的大小根据图像的行、列数的不同而不同,一般情况是 64 × 64 大小。Img 格式中用来存储图像数据的节点是 RasteDMS。

Edms _ State 的存储结构如下:

```
long numvirtualblocks;                  /* 块的个数 */
long numobjectsperblock;                /* 块的大小 */
long nextobjectnum;                     /* 留用字段 */
enum compressiontype;                   /* 是否压缩 */
Edms _ VirtualBlockInfo blockinfo;      /* 块的信息 */
Edms _ FreelDList freelist;             /* 留用字段 */
TIME modTime modTime;                   /* 修改时间 */
```

Edms _ VirtualBlockInfo 结构如下:

```
Short filecode;      /* 一般是 0 */
Long offset;         /* 块数据的存储地址 */
Long size;           /* 块的大小 */
Enum logvalid;       /* 一般是 1 */
Enum compression;    /* 是否压缩 */
```

在读取任意一个 Img 格式图像时,搜索每一个节点。当节点为 RasterDMS 时,用 VC + +6.0 读取方法如下:

```
int fg _ dms = 0;
if(fg _ dms < nBands)                    //nBands 是波段数
{
fread(&blocknum,4,1,fp);                 //块的个数
fread(&blocksize,4,1,fp);                //块的大小
fread(&bb,4,1,fp);                       //furture use
fread(&compressiontype,2,1,fp);          //compressiontype
fread(&virbloinfonum,4,1,fp);            //virtualblockinfo 的重复个数
fread(&pvirbloinfo,4,1,fp);              //virtualblockinfo 的地址
fseek(fp,pvirbloinfo,SEEK _ SET);
for(int zz = 0;zz < virbloinfonum;zz + + )
{
fread(&filecode,2,1,fp);          //filecode
fread(&offset,4,1,fp);            //每一个 Block 的地址 Offset
//把各波段的每一块的地址加到动态数组 Blockaray 里面去
blockarray[fg _ dms].Add(offset);
fread(&bblocksize,4,1,fp);        //块的大小,以字节计
fread(&logvalid,2,1,fp);
fread(&compressiontype,2,1,fp);
}
fg _ dms + + ;
}
```

上面的这一段程序得到了每一块的地址和数据大小,块的位置存储在 Blockarray 数组指针里,数据大小存储在 Bblocksize,根据存储地址和大小就可读取各个波段的各个块的数据。同一波段的数据是连续存储的,可以一次性读入,如:

```
fseek(fp,blockarray[0].GetAt(0),SEEK _ SET); //blockarray[0].
//"GetAt(0)是某一个波段的第一块的存储地址,从前面的程序中得来
fread(pdemint,1,width * height,fp);
//一次性读入此波段的全部块的数据,pdemint 是地址指针
```

由于 Img 格式是按块存储的,这样读出来的数据流还不能显示,必须经过下列的方法重新整理:

```
Int z = o;
for(i = 0;i < Heightunit;i + + ) //Heightunit 是指在列上分成的块数
```

(下转第 87 页)

IP 的。TCP/IP 是 SAN 的构建技术,所得到的 SAN 也是基于 IP 的 SAN,它应用的场合是将分散于 Internet/Intranet 上的存储设备、服务器(这些设备不是基于 FC)等连接起来,以构建一个跨越 MAN/WAN 的无处不在的存储网络。FCIP 中,必须仍由 FC 互连设备来构建基于 FC 的 SAN,而 IP 网络的作用是将分散的 FC-SAN 互连成一个整体,它应用的场合是原来已经存在的基于 FC 的 SAN,然后需要将分散的 SAN 互连起来。iFCP 则是利用 IP 网络来构建一个突破距离限制的基于 FC 的 SAN,它应用在目前大量的 FC 终端设备存在的场合下,将这些 FC 设备连入 IP 网络以便构建一个跨越 MAN/WAN 的 FC-SAN;iFCP 也可以像 FCIP 一样,用在将分散的 FC-SAN 互连起来的场合。不同的应用场合使得它们不能简单地以优劣来区分,在各自的应用场合下都具有其它技术难以比拟的优点,很难相互取代。尤其在目前大量 FC 设备存在的情况下,它们相互补充,平行地向前发展。

另外,FCIP 立足于 FC 仍是构建 SAN 的理想选择,只是利用 IP 网络来扩展 FC-SAN 跨越的距离,因此 FCIP 中,占主导地位的仍是 FC。而 iFCP 则不然,它认为 FC 不是构建 SAN 的最理想的选择,无论是在数据中心还是一些远程应用,必须采用新的构建技术,即 IP 来构建。但它正视目前大量的 FC 终端设备存在的现实,支持 FC 终端设备直接连入 SAN 或互连 FC-SAN,iFCP 中占主导地位的却是 IP。至于 iSCSI,则根本不存在 FC,而是由 TCP/IP 完全取代 FC,它构建的是一个基于 TCP/IP 的遍布 Internet/Intranet 的无处不在的存储网络。考虑到未来网络将统一在 TCP/IP 的基础之上,那么 iSCSI 应该更适合技术的长远发展。

参考文献:

[1] 韩德志.网络存储技术的探讨[J].微电脑应用,2001,(3).

[2] Intel iSCSI Protect[EB/OL].http://sourceforge.net/projects/intel-iscsi,2002.

[3] Julian Satran,et al.iSCSI[EB/OL].http://www.ietf.org/internet-drafts/draft-ietf-ips-iscsi-11.pdf,2002.

[4] M Rajagopal,et al.Fibre Channel Over TCP/IP (FCIP)[EB/OL].http://www.ietf.org/internet-drafts/draft-ietf-ips-fco-ver-tcpip-09.pdf,2002.

[5] Charles Monia,et al.iFCP-A Protocol for Internet Fibre Channel Storage Networking[EB/OL].http://www.ietf.org/internet-drafts/draft-ietf-ips-ifcp-10.pdf,2002.

[6] Fibre Channel Industry Association. Fibre Channel Standard [EB/OL].http://www.fibrechannel.com,2002.

[7] R Weber,et al.FC Frame Encapsulation[EB/OL].http://www.ietf.org/internet-drafts/draft-ietf-ips-fcencapsulation-06.pdf,2002.

[8] Josh Tseng,et al.Internet Storage Name Service(iSNS)[EB/OL].http://www.ietf.org/internet-drafts/draft-ietf-ips-isns-09.txt,2002.

作者简介:

韩德志,博士生,研究方向为基于网络的存储系统、基于 IP 的存储区域网;鄢让,讲师,研究方向为网络存储及多媒体教学;傅湘林,博士生,研究方向为基于网络的存储系统、基于 IP 的存储区域网。

(上接第 61 页)

```
for(j = 0;j < blockheight;j + + ) //Blockheight 是指块的高度
for(int k = 0;k < widthunit;k + + )
//Widthunit 是指在行上分成的块数
for(int p = 0;p < blockwidth;p + + ) //Blockwidth 是指块的宽度
{
pdata[z] = pdemint[i * blockwidth * blockheight * widthunit + j * blockwidth + k * blockwidth * blockheight + p]; //Pdata 是地址指针
z + + ;
}
```

经过上述方法整理过的数据流 Pdata,就可以按照正常的图像格式显示方式显示。

图 2 是一幅北京奥体中心的融合三波段遥感图像。

图 2　北京奥体中心的 Img 图像在 Maxplorer 中显示

当读取超大数据量的遥感图像时,为了提高显示速度,节省内存空间,不采用上面一次性读入的方法,而是采用显示用到哪些块才读哪些块,整理哪些块数据的方法。Img 图像的每一个块在文件中的位置都是已知的,因此定位每一个块能快速实现,这得益于 Img 格式按块存储的特性,这是其它图像格式不能做到的。图像(图 2)格式为 Img,分辨率为 0.61 米;行、列数分别是 1 025 和 768,总共分为 204 个块,块的大小是 64 × 64,用上述方法读取并显示在自主开发的 GIS 专业图像处理平台 Maxplorer 中。

4　结束语

Img 格式图像由于存储格式的特殊性,在读写方面都有其优越性,如读取速度快,色彩质量好,存储的信息量多。本文介绍的对 Img 格式的读取方法,为 Img 格式的普遍应用提供了有效的途径。

参考文献:

[1] David Kruglinski,Scot Wingo,et al.Programming Microsoft Visual C + + [M].北京:北京希望电子出版社,1999.

[2] www.wotsit.org,2002-07[EB/OL].

[3] 章毓晋.图像处理与分析[M].北京:清华大学出版社,1999.

作者简介:

朱政(1979-),女,硕士研究生,主要从事三维虚拟现实和图像处理研究;刘仁义(1960-),男,副教授,博士,主要研究领域为面向对象的空间数据库理论及 GIS 应用系统技术开发;刘南(1944-),男,教授,博士生导师,现任中国地理学会常务理事,浙江省地理学会理事长,浙江省资源与环境信息系统重点实验室主任,主要研究领域为 GIS 理论及应用。