

# 三维地震数据体的切片播放算法

汪在荣<sup>1</sup>, 刘益和<sup>2</sup>

(内江师范学院计算机科学学院, 四川 内江 641110)

**摘要:**目前三维数据可视化技术已经广泛应用于地震解释中,而绝大多数的三维地震解释,是通过数据体的切片实行的。切片显示步骤是:确定切面多边形、对数据体的重采样和图像合成、切片的图像绘制。并且通常需要对切片实现播放、缩放等操作,此时,对于有限的内存采用传统的方法实现庞大的三维地震数据体的切片播放是相当的困难的。文章提出了一种将三维地震数据进行分块存储及显示的技术,即三维数据的砖块组织结构,并建立了快速索引机制,及砖块结构的调度算法。实验证明,该算法对三维地震数据切片的播放具有实时性及高效性。

**关键词:**三维数据可视化;切片播放;砖块结构;三维地震数据切片;调度算法

**中图分类号:**TP393.17

**文献标识码:**A

## 引言

三维地震数据的最主要特点之一就是数据量非常庞大,通常是 1GB 以上。面对如此庞大的三维数据,可以采用纹理贴图<sup>[1]</sup>和光线投射算法<sup>[2]</sup>等实现体绘制。但在绝大多数的三维地震解释中,专业人员希望能够更清晰地了解这些数据真正的构成形态,从而准确地判断油气藏的存储位置,提高生产效率,降低生产费用,这些都是通过对数据体的切片显示来完成。所谓切片就是以平行于三维地震体的三个方向 X、Y、Z 中某一方向的平面去切三维数据体得出的切面,它包括两个垂直剖面纵测线与横测线,和水平切面。三组正交切片显示的基本过程是:确定切面多边形、数据重采样与图像合成、切片的图像绘制<sup>[3]</sup>。在地震解释中用户要求对三维体数据沿 X、Y、Z 方向上进行相应的动态显示操作,即切片的播放。而但传统的三维地震数据可视化软件并没有提供切片的播放功能或是由于存在反复从外存中重采样数据使得切片的显示无法跟上播放速度。因此,本文基于实际项目开发过程中,提出了一种砖块组织结构及其快速的索引调度机制,并以此为基础,将原始三维地震数据转化为砖块结构文件,实现了三维地震数据体三组

正交切片任意速度的播放。

## 1 三维地震数据的砖块结构表示

对于海量的三维数据体在切片播放时要求切片的更新显示与速度同步,但由于播放速度使得在某一位置切片只要能够反映数据的分布情况即可,对其具体的精度要求并不高,但是当播放暂停时,此时显示的切片应该能准确反映在当前位置的数据图形。如果不采取相应的简化策略,直接对切片进行绘制,处理起来极为不便,严重影响运行效率和显示速度<sup>[4]</sup>。故要想实现实时显示与播放,必须进行数据简化(抽稀),简化后的数据能够描绘出原切片数据的分布趋势。但是,当切片播放暂停或停止时,只有重新调用未经采样前的数据进行切片绘制,才能保证图形的精度。因此,不失一般性,本文提出当切片正在播放时,则调用经过抽稀后的数据,这样提高了切片的绘制速度,当播放暂停或停止时,立即调用此切片位置的原始数据,这样则保证了图形的精度。本文在参考文献[5]的砖块结构算法上,进行内存优化和调度算法上的创新,从而更快速的实现三维地震数据体切片的播放。

对于海量的三维地震数据处理来说,显然不能将其

收稿日期:2011-05-27

基金项目:四川省教育厅自然科学重点科研项目(09ZA055)

作者简介:汪在荣(1975-),男,四川简阳人,讲师,硕士,主要从事通信网络及网络安全方面的研究。

全部调入内存中,只能在需要时才将所需的数据读入内存中进行处理。相对于 CPU 的速度来说,磁盘 I/O 是瓶颈之所在<sup>[6]</sup>,在切片播放或静态显示时,处理的数据只是一部分数据。

结合上述需求和三维地震数据的特点,本文在参考文献[5]提出的基础上,并结合 I/O 传输问题,对三维地震数据进行分块,这样既降低了地震数据体的操作难度又简化和提高了绘制效率。

三维地震数据的分块大小通常取 2 的幂次方,本文中经过实践分析,参见表 1 各 2 的幂次分块进行数据传输的耗时对比,从表中可以看出,对于 256 级的灰度图像,以 64 \* 64 \* 64(256K)为最佳,按照这样的大小进行三维数据的分块,每次硬盘传送一块的数据耗时非常短。

表 1 数据分块

分块大小 (KB)	传送数据大小为 4000KB 的 I/O 次数	冗余数据大小 (KB)
16	250	0
32	125	0
64	63	32
128	32	32
256	16	160
512	8	416
1024	4	928

从表 1 中可以看出,就分块而言,并无标准可循,可以按任意规则进行数据分块,但在应用中则必须结合索引、磁盘 I/O、效率等进行考虑。不规则的分块将影响索引构建(索引用于查找定位数据)、索引操作及图形重构的效率。除此之外,分块太大或太小都将影响系统的有效性能,如果分块数据过大,则可能导致读入的冗余数据过多;反之,如果分块数据过小,则导致频繁的磁盘寻址和读写操作,使得所需数据的 I/O 访问时间增加<sup>[8]</sup>。

在介绍砖块组织结构前,先了解一下三维地震数据场。它是由横测线(xLine)、纵测线(inLine)和时间(time)确定的坐标系统。而(xLine, inLine)则构成了地震数据的道。如图 1 所示,显示了三维地震数据场和以块划分后砖块编号顺序。

根据前面的介绍,将原始三维地震数据转化为砖块结构文件来实现切片的播放,转换后的数据分为索引文件、采样文件以及砖块文件三部分。图 2 为砖块文件结构示意图。

索引文件由砖块数据整体描述信息块和道记录描述信息两部分组成。其中砖块数据整体描述信息块的主要作用是对砖块文件进行整体描述,它记录了原始数据中 time、xLine、inLine 三个方向的砖块数,三个方向标

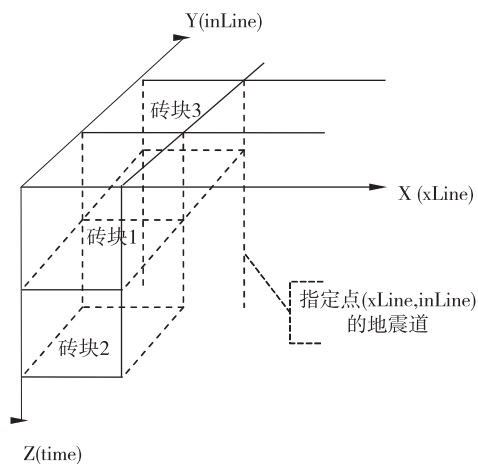


图 1 三维数据场按块划分及砖块编号顺序

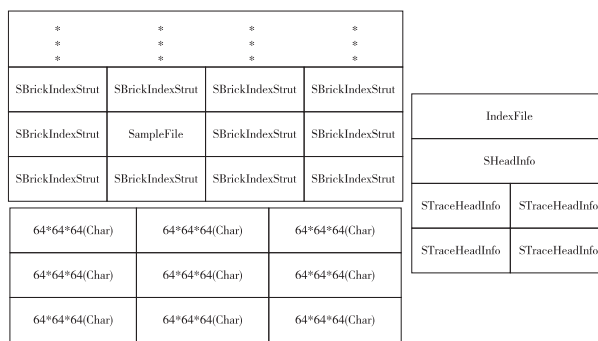


图 2 砖块文件结构示意图

号的最值,速度(振幅)最值,大地坐标最值,本文中使用的结构体进行存储。索引文件的另一部分是原始数据中每一道记录的描述信息,包括每道的 xLine 号、inLine 号及该道对应的大地坐标,它也是由相应的结构体进行存放管理,然后再利用一个三维数组(按照 Time、Inline、xLine 的顺序进行组织)存储所有道的数据结构。索引文件为砖块文件和采样文件数据提取提供参数,其主要作用是描述砖块结构。

砖块文件是砖块结构的主要部分,它存储了完整的绘图数据。砖块文件中存放的是依次排列的大小为 64 \* 64 \* 64 字节的小砖块,这些小砖块按照 time、inLine、xLine 顺序进行组织存放。考虑到不同原始数据的速度(振幅)数据类型可能不相同,占用的存储空间也不同,而且有的数据类型占用的存储空间较大(如 double 类型),因此砖块文件将原始数据值统一转化为 Char 类型的一个颜色值,由于颜色变化是用户关心的焦点,所以进行上述类型的转化不仅大大减小了存储空间,而且还不会影响绘图效果。

采样文件是根据采样间隔从原始数据中提取的数据,是原始数据经过抽稀后的文件,它存储了部分原始

绘图数据,虽没有原始数据详细,但通过它能够对原始数据进行定性和定量分析,能够反映原始数据的整体情况。砖块文件存储的是原始数据的所有数据点,数据量较大,不可能一次性调入内存进行处理,将数据存放在硬盘上,又受硬盘传输速率的限制,不可能实时的调入内存,也就不能进行实时动态播放。采样文件能够反映原始数据的整体情况且数据量很小,可以一次性调入内存,因此能够实时动态播放。采样文件由经过抽稀后的部分原始数据(实际中可利用一个三维数组进行存储)和每一个小砖块文件在整体砖块文件中的偏移量(实际中可利用结构体进行存储)两部分组成。

以上三个文件是在将原始三维地震数据转换为砖块文件时生成。对于 xLine 方向的砖块数为 maxXLine/64, inLine 方向的砖块数为 maxInLine/64, time 方向是由每道的采样点来确定的,故该方向砖块数为 sampleCount/64,因此,总砖块数为:

$$brickCount = \frac{\maxInLine}{64} \times \frac{\maxXLine}{64} \times \frac{\text{sampleCount}}{64} \quad (1)$$

而砖块的编号是按 time、inLine、xLine 方向顺序编号,并以索引文件结构体描述形式将其它数据填入索引文件中,最后生成的索引文件是 .index 格式。下一步是生成砖块文件(brick 格式),砖块文件中存放的是按照砖块编号将原始三维数据的颜色值顺序写入,与此同时,生成采样文件(sample 格式)并写该文件,以纵测线(inLine)数为外重循环,以横测线(xLine)为第二重循环,以 time 方向的每道采样点数(sampleCount)为内层循环,读取每一个原始数据点的颜色值写入 brick 文件中,同时,判断该点是否被采样,若是,则同时将该点写入 sample 文件中,直至所有的数据点全部转换结束。brick 文件和 sample 文件都是以划分后的砖块为单位进行组织的。

## 2 砖块结构的 FIFO 调度

至此,已经阐述了将原始三维地震数据的砖块结构表示。由于 sample 文件和 brick 文件存放的分别是采样后的砖块数据和原始砖块数据,这些数据都存放在磁盘上,在切片播放或停止时,需要确定当前切片数据在砖块文件或采样文件中的具体位置和大小,然后将其调入内存中。因此,砖块结构的调度是实现切片显示或播放的一个重要环节,也是本文研究的另一个重点。

无论切片的播放还是切片的显示都要求能够获取更快的速度,特别是实时显示。而磁盘数据的读取速度较内存数据要慢,为了加快数据的处理速度,减少对磁

盘的 I/O 操作,可以将当前需要的数据存放在事先分配的内存中,每次只处理内存中的数据。

本文根据系统内存的大小,对转换为砖块结构后的数据设置相应的高速缓存,高速缓存也按块分配,即将整个内存块划分为大小相同的单元,每块的大小与 brick 文件或 sample 文件大小相同,当缓冲区数据需要更新时,每次读取分块后的文件中的一块数据。这里 brick 文件是用于切片停止播放时显示准确的信息,而 sample 文件是用于切片播放时简化后的数据,两者用于不同的目的,但两者都是以砖块组织的,所以它们的调度方式是一样的。本文以 brick 文件的调度为例,描述缓冲区的调度方法。为了更好的进行数据调度,采用建立内存块索引的方法进行数据的调度管理。在缓冲数据更新时读取数据采用 FIFO 算法。

文中采用一个辅助数据结构来存放整个三维体数据的所有砖块的状态和信息。结构体表示为:

```
typedef struct _tagBrickInfo
{
    unsigned char status; //0:表示常驻内存,1:硬盘
    unsigned int index; //砖块在内存索引表中的索引位置
} SBrickInfo;
```

该辅助数据结构是用数组来存放,因为数组是一种简单的数据结构,而且其访问速度也最快。砖块的调度步骤如下:

- (1) 建立三维体数据的辅助数据结构;
- (2) 建立一个内存块索引表和开辟砖块数据缓冲区;
- (3) 计算切片所需要的砖块号,并结合 .index 文件取得砖块在 .brick 文件中的偏移位置;
- (4) 当某一砖块或某几个砖块数据需要被调入内存中时,修改辅助数据结构相应砖块号的 status 标志为 0,表示该砖块数据已经被调入内存,同时按 FIFO 调度算法计算出该砖块在内存块索引表中的索引位置(即查找一个内存缓冲区),若该索引位置为空,则将在该索引位置中写入存放的砖块号以及指向该砖块数据的指针,并将该数据读入内存中;若该索引位置已经有砖块数据,则修改该索引位置的砖块号所在的辅助数据结构信息(即将原先存放的砖块号的 SBrickInfo 结构体信息分别置 status 为 1, index 为初始值 MAX),再将内存索引表中该索引位置的 index 设置为需要被替换的新砖块号,同时将砖块数据指针指向的原来砖块数据进行释放,让该指针指向新的砖块数据。

图 3 反映了辅助数据结构与内存索引表的关系。

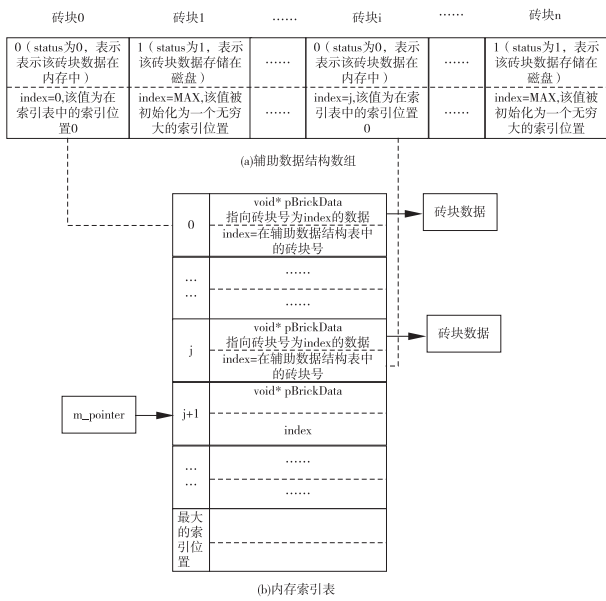
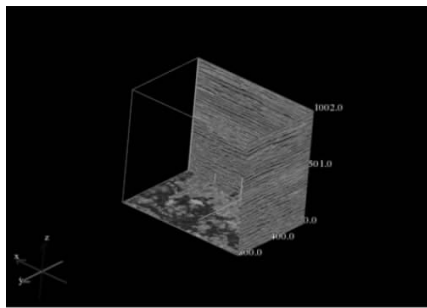


图3 辅助数据结构与内存索引表的关系

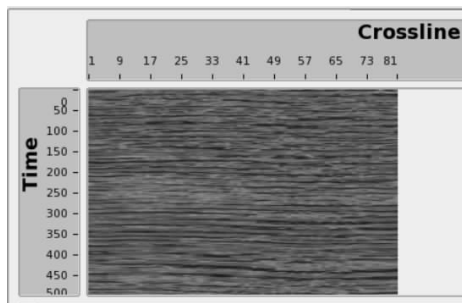
### 3 切片播放算法

利用前面介绍的砖块结构和砖块结构的调度, 本文通过以下3个步骤进行三维地震数据体的切片播放或显示: (1) 利用砖块结构对相应的数据结构进行初始化; (2) 根据当前图形状态获取切片图形数据。如是播放暂停或停止状态, 则计算出当前位置各切片相应的砖块号, 并根据砖块索引判断是否需要将该数据调入内存; 如是播放状态, 则计算出当前播放位置各个切片在采样文件中的位置, 并读取相应的采样文件中的数据。(3) 根据相应的数据运用一定的图形图像原理进行图形绘制。

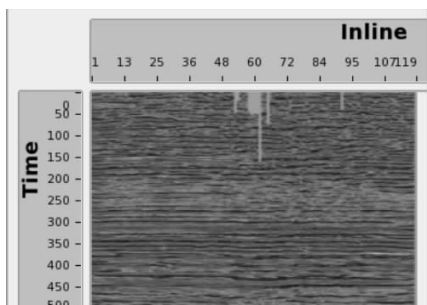
该方法运用于实际的地震数据处理软件开发中, 图4显示了使用该方法进行切片显示与播放的图形。图(a)中显示的是使用砖块结构算法进行切片播放时的三维立体图, 此时三个方向的切片分别是 inline 号为1的切片(图(b)所示), crossline 号为1的切片(图(c)所示), 采样时间 time 为0的切片(图(d)所示)。



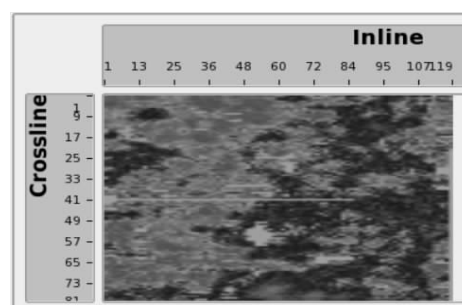
(a) (inline, crossline, time)的三维图



(b) inline=1的切片图



(c) crossline=1的切片图



(d) time=0ms的切片图

图4 切片的显示与三个方向的切片播放

### 4 结束语

采用以上的切片播放和调度算法, 使得在处理庞大的三维地震数据量时不受计算机内存容量的限制, 并且数据量的大小也不影响处理和显示的速度, 大大提高了海量三维地震数据在图形显示和切片播放时的速度, 可实现三维地震数据切片的实时播放。

### 参考文献:

- [1] 夏冰心. 基于纹理映射的三维地震数据可视化研究[D]. 南京理工大学, 2008.
- [2] 张二华, 高林, 马仁安, 等. 三维地震数据可视化原理及方法[J]. CT理论与应用研究, 2007, 16(3): 20-28.
- [3] Xue Daqing, Roger Crawfis. Efficient Splatting Using Modern Graphics Hardware[J]. Proceedings of the Computer Graphics, Imaging and Vision: New Trends, 2005, 8(3): 1-21.
- [4] David Laur, Hanrahan P. Hierarchical Footprint method: A

- Progressive Refinement Algorithm for Volume Rendering [C].New York,NY,USA:ACM SIGGRAPH Computer Graphics,1991:285-288.
- [5] 夏凡.地震数据处理系统中三维可视化的研究与实现[D].成都电子科技大学,2010.
- [6] Peter Lindstrom,Valerio Pascucci.Visualization of Large Terrains Made Easy[C].In:Proceedings of the IEEE Visualization'01,2001,363-371.
- [7] Dachsbacher C, Stamminger M. Rendering Procedural Terrain by Geometry Image Warping[C].In:Proceedings of the Eurographics'04,2004,103-110.
- [8] Hua Yi-Xin.Theory and Technology of Geographics Information System[M].Beijing:PLA Press,2001:1-2.
- [9] 钟晓霞.大范围复杂场景的简化与漫游技术研究[D].南京理工大学,2003.
- [10] 张伟.显微图像拼接系统设计与实现[D].北京邮电大学,2009.

## Play of Slices Algorithm of 3-D Seismic Data Volume

WANG Zai-rong<sup>1</sup>, LIU Yi-he<sup>2</sup>

(School of Computer Science, Neijiang Normal University, Neijiang 641110, China)

**Abstract:** At present, three-dimensional data visualization technology has been widely applied in seismic interpretation, however most of 3D seismic data is interpreted through the slices of the volume. The traditional display steps of slices were as follows: determine the section polygon, re-sample data and compound the image, rendering the slice images. Also, it needs to achieve operations with these slices such as playing, zooming, etc. Simultaneously, it is quite difficult for the limited memory to realize the play of slices of huge 3D seismic data. Therefore, a technology is proposed which stored and display 3D seismic data in blocks. It means the brick structure of 3D seismic data. The mechanism for fast indexing scheduling algorithm of brick structure is established. Experiments show that the algorithm is real-time playback and efficiency for three-dimensional slice of seismic data.

**Key words:** three-dimensional data visualization; play of slices; brick structure; seismic data volume; scheduling algorithm