基于压缩域的脑成像大数据体可视化方法

时学凯[1] 王文珂[2] 黄 辉[1] 李思昆[1] 傅艺绮[1]
(国防科学技术大学计算机学院 长沙 410073)[1]
(国防科学技术大学海洋科学与工程研究院 长沙 410073)[2]

摘 要 脑科学是当今国际科技研究的前沿邻域,而对高精度脑成像数据进行可视化是脑神经科学在结构成像方面的基础性需求。针对高精度脑成像数据可视化过程中存在的数据量大以及绘制效率低的问题,提出了基于分类分层矢量量化和完美空间哈希相结合的压缩域可视化方法。首先对体数据进行分块,记录每块的平均值并依据块内体数据的平均梯度值是否为0进行分类;其次运用分层矢量量化对平均梯度值不为0的块进行压缩;然后用分块完美空间哈希技术存储压缩得到两个索引值;最后对上面的压缩体数据进行解码得到恢复体数据,采用分块完美空间哈希对原始体数据与恢复体数据作差得到的残差数据进行压缩。绘制时,只需将压缩得到的数据作为纹理加载到GPU内,即可在GPU内完成实时解压缩绘制。实验结果表明,在保证较好图像重构质量的前提下,该算法减少了数据的存储空间,提高了体可视化的绘制效率,从而可以在单机上处理较大的数据。

关键词 体可视化,分类分层矢量量化,完美空间哈希,神经回路,GPU
中图法分类号 TP391 **文献标识码** A **DOI** 10.11896/j.issn.1002-137X.2017.03.007

Volume Rendering Method of Mass Brain Imaging Data Based on Compression Domain

SHI Xue-kai[1] WANG Wen-ke[2] HUANG Hui[1] LI Si-kun[1] FU Yi-qi[1]
(School of Computer, National University of Defense Technology, Changsha 410073, China)[1]
(Institute of Ocean Science and Engineering, National University of Defense Technology, Changsha 410073, China)[2]

Abstract Nowadays, brain science is the forefront field of international scientific and technological research, while the visualization of the high-precision brain imaging data is the fundamental requirements of the structural imaging of brain neuroscience. Aiming at the problems of great quantity of data and low efficiency during rendering the brain imaging data, a compression domain visualization algorithm based on the combination of flag based classical hierarchical vector quantization and perfect spatial hashing was put forward. Firstly, the volume data is blocked, the average of each block is recorded and then the blocks are classfied according to their average gradient value. Secondly, the hierarchical vector quantization is used to compress the blocks of whose average gradient is not 0. Thirdly, the perfect spatial hashing technology based on blocking is used to store two index values obtained by compressing. Finally, the above compressed data is decompressed to obtain the recovered volume data, and then the perfect spatial hashing based on blocking is applied to compress the differential volume data obtained by making the original volume data minus the recovered volume data. When rendering, the compressed data is reloaded as textures to GPU, then decompression and visualization can be done in real time. The experiment results show that the algorithm reduces the data storage space and improves the compression ratio and can make the single machine handle larger data under the premise of ensuring the better quality of image reconstruction.

Keywords Volume visualization, Flag based classical hierarchical vector quantization, Perfect spatial hashing, Neural circuits, GPU

脑是自然界最复杂的系统之一,支配着人类和动物的一切活动。神经回路是大脑行使一切功能的物质基础,由于脑的高度复杂性,以单神经元分辨率技术采集脑的样本数据,通过对高精度大规模神经回路样本数据的高效处理与可视分析,深入认识、准确掌握脑的神经回路结构和功能,破解大脑之谜,是我国社会和科技发展的重大需求。

到稿日期:2016-02-01 返修日期:2016-05-07 本文受国家重点基础研究计划(973计划)项目:灵长类神经回路精细结构成像的新方法和新工具(2015CB755604)资助。

时学凯(1989—),男,硕士生,主要研究方向为虚拟现实与可视化,E-mail:shi_xuekai@163.com;王文珂(1981—),男,博士,副研究员,主要研究方向为虚拟现实与可视化;黄 辉(1978—),博士生,主要研究方向为虚拟现实与可视化;李思昆(1941—),教授,主要研究方向为虚拟现实与可视化、SOC设计方法学;傅艺绮(1992—),硕士,主要研究方向为高可信软件。

运用目前国内外最先进的 MOST[1] 系列技术的高精度光学脑成像设备对小鼠脑结构采样，即可得到数百 GB 到数 TB 的全脑成像数据。即使采用数十个可视化结点进行并行可视计算，每个结点仍需要处理数 GB 到数十 GB 的数据。由于数据规模巨大，直接利用原始数据进行体可视化效率较低，因此在可视化之前，需要对原始数据进行数据压缩处理，减少数据量，进而提高可视化效率。

压缩域体绘制(Compression Domain Volume Rendering，CDVR)[2] 是一种融合了压缩和体绘制的方法，其通过对压缩后的数据进行实时解压并绘制，能有效地解决当前大规模体数据直接体绘制中存在的体绘制效率低、显存容量小和主存到显存带宽小的问题。矢量量化作为一种非对称的压缩方法，即压缩较复杂但解压非常简单，已成为 CDVR 中体压缩算法的一个不错选择。Ning 等[3-4] 最先运用矢量量化方法压缩体数据并实现了压缩域的体绘制。Kraus 等[5] 最先在 GPU 端用 VQ(Vector Quantization)进行静态体数据的解压缩和绘制。Schneider 等[2] 实现了层次化的矢量量化方法(Hierarchical Vector Quantization，HVQ)，提高了压缩效率和绘制质量。Fout 等[6] 运用矢量量化和变换编码技术，提出了一种非对称的基于 KLT 变换与分区矢量量化编码相结合的体压缩策略(Transform Vector Quantization)。赵利平等[7] 对 HVQ 进行了改进，提出了一种高效的分类分层矢量量化(Flag Based Classical Hierarchical Vector Quantization，FCHVQ)，使得在空体素较多的体数据上可取得更好的压缩比。丁治宇等[8] 提出了一套以层次矢量量化和完美空间哈希为基础的多变量体数据压缩域体绘制方法，缩小了误差，进而提高了绘制质量。

通过对高精度脑成像大数据进行大量统计可知，其中存在较多的空体素，所以拟采用文献[7]提出的分类分层矢量量化(FCHVQ)方法对我们的高精度脑成像大数据进行压缩预处理。当采用 FCHVQ 对高精度脑成像大数据进行压缩时，虽然压缩率较高，但相比丁治宇[8] 的方法，其解压得到的数据与原始数据有较大误差，导致可视化质量较丁治宇的方法不高，而且 FCHVQ 的解压和绘制是在 CPU 上进行的，导致体绘制帧率不高。当采用文献[9]的方法对我们的高精度脑成像大数据进行压缩时，虽然降低了压缩数据与原始数据的误差，但是其压缩比低于 FCHVQ 算法。

基于此，本文提出了基于分类分层矢量量化和完美空间哈希相结合的压缩域体可视化方法。首先对体数据进行分块，计算块的平均值，将其保存到块平均值数组内，并根据块内体数据的平均梯度值是否为 0 对块进行分类；其次使用分层矢量量化对非 0 的数据块进行压缩，用 PCA(主成分分析)分裂法产生初始码表，并运用 LBG 对码表进行优化；然后运用分块完美空间哈希技术存储对非 0 块压缩得到的两个索引值；最后对上面的压缩体数据进行解码得到恢复体数据，将原始体数据和恢复体数据作差得到残差体数据，通过对残差体数据设置阈值来判定体素是否是有效体素，并运用分块完美空间哈希对残差体数据进行压缩。以上步骤都是在 CPU 端实现的。绘制时，将压缩得到的数据构造成纹理查找表加载到 GPU 内，然后在 GPU 内通过纹理查找进行实时解压缩，并运用标准的光线投射流程进行直接体绘制，得到该体数据的压缩域绘制结果。

1 完美空间哈希

1.1 完美空间哈希技术

完美空间哈希是 Lefebvre 等[9] 提出的一种用哈希表来表达稀疏数据的压缩技术。此方法适用于多维空间数据，其实质是充分利用数据的空间连续性，用紧凑的哈希表来表达稀疏的空间数据。

Zhou 等[10] 用径向基函数近似表达原始数据，然后应用完美空间哈希表压缩原始数据与近似数据之间的稀疏差。叶楩等[11] 使用八叉树结构逼近原始不规则体数据，进而用完美空间哈希存储逼近误差。文献[8]使用分层的矢量量化方法逼近原始体数据，解压得到恢复体数据，然后用完美空间哈希压缩原始体数据与恢复体数据之间的稀疏残差体数据。

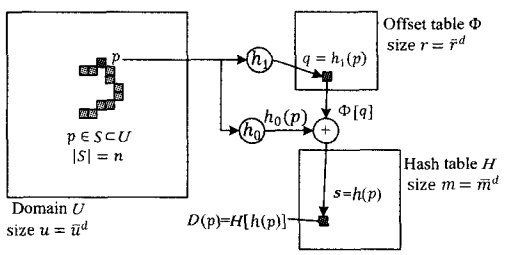


图 1 哈希函数的定义[9]

完美空间哈希构建的过程如图 1 所示。设置一个阈值来判断一个体素是否是有效体素。U 代表整个体素，其中 $u=\bar{u}^d$，S 代表有效体素集合，有效体素的个数 $n=|S|$，p 是其中的一个有效体素，那么数据密度为 $\rho=n/u$。q 表示偏移表的位置，s 表示哈希表的位置，$\bar{r}$ 和 $\bar{m}$ 分别表示偏移表和哈希表的尺寸，$D(p)$ 表示 $p(p\in S)$ 点数据值，Φ 和 H 分别表示偏移表和哈希表的值，h_0 和 h_1 是两个不完美的哈希函数。其中：

$$h_0(p)=p \bmod \bar{r} \tag{1}$$

$$q=h_1(p)=p \bmod \bar{m} \tag{2}$$

完美空间哈希的表达式如下：

$$H(p)=h_0(p)+\Phi(h_1(p)) \bmod \bar{m} \tag{3}$$

随着商业图像硬件的迅猛发展，基于 GPU 加速的直接体可视化已经成为分析体数据的通用方法。然而当前的 GPU 并不支持位(0 或 1)操作，为了在 GPU 端实现完美空间哈希技术，采用文献[9]提出的通过保存有效体素的坐标值来标记其是有效体素：让标记表的尺寸等于哈希表的尺寸，当将有效体素散列到哈希表的 H 处时，同时将该有效体素的坐标值保存到标记表的 H 处。其在具体实现中用 3 个三维纹理来表示 3 个查找表：将哈希表构造为三维的哈希纹理查找表，将偏移表、标记分别构造为三通道的三维偏移纹理查找表和三通道的三维标记纹理查找表。

1.2 分块哈希

由文献[9]可知，当原始体数据尺寸 $\bar{u}$ 较大时，完美空间哈希的构造将会非常耗时，而且当 $\bar{u}>256$ 时，需要用整型数据来保存有效体素的坐标值，这样就会造成压缩率不高。鉴于此，可以对较大尺寸的体数据进行分块哈希，这既有利于运用多线程对每块体数据进行哈希处理，进而缩短哈希时间，也有利于采用无符号的字符型数据保存有效体素的坐标值，进而提高压缩比。本文采用的分块大小为 256×256×256。

分块后，按如下步骤对每块进行标号：

1)沿着 X 轴方向进行标号；

2)递增 Y 轴的值，重复 1)；

3)重复 2)，直到沿 Y 轴标号完成；

4)递增 Z 轴的值，重复 1)—3)，直到沿 Z 轴标号完成。

图 2 为将体数据分为 8 块时，对每块体数据标号的示意图。对体数据空间中的任一点 p，其网格坐标为(x,y,z)，其中 x,y,z 都是整数，那么它所在块的标号为：

$$Tag(p)=(z/256)\times bY\times bX+(y/256)\times bX+(x/256) \tag{4}$$

其中，bY 和 bX 分别表示沿 Y 轴方向的分块数和沿 X 轴方向的分块数，“/”表示整除运算。

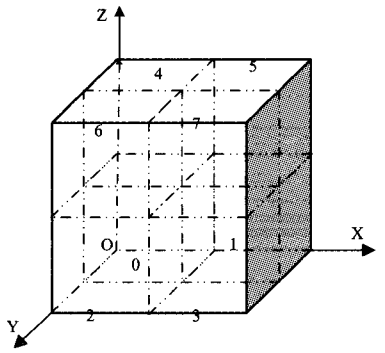


图 2　标号示意图

由于 OpenGL 着色语言支持的最大纹理单元数为 32，当分块较多时，将每块得到的 3 个表(即哈希表、偏移表和标记表)分别作为纹理加载到 GPU 内是不可行的。因此，将每块得到的哈希表、偏移表和标记表按照块标号的递增顺序组成一个更大的一维哈希数组、一维偏移数组和一维标记数组，进而将它们构建为纹理查找表加载到 GPU 内实现解压缩。假设体数据被分为 n 块，分块哈希纹理查找表的构建步骤如下：

1)记录每块哈希表的尺寸$\overline{m_i}$，计算哈希纹理查找表的尺寸$h=\lceil\sqrt[2]{\sum_0^{n-1}\overline{m_i}}\rceil$；

2)构建一维数组，数组的大小为 h^3，把每块哈希表中的哈希值按照块标号的递增顺序存入构建的一维数组中；

3)将一维数组作为三维纹理加载到 GPU 内。

分块偏移纹理查找表和分块标记纹理查找表的构建过程与分块哈希纹理查找表的构建步骤一样，其中标记纹理查找表的尺寸为 h，偏移纹理查找表的尺寸为 $o=\sqrt[2]{\sum_0^{n-1}\overline{r_i}}$，$\overline{r_i}$为第 i 块偏移表的尺寸。

在 GPU 内的解码阶段，针对体数据空间中任意一点 $p(x,y,z)$，通过式(4)得到 p 所在块的标号 $Tag(p)$，让点 p 的坐标对 256 取模得到点 p 在块内的网格坐标(x',y',z')，由块号、块内网格坐标和每块 3 个表的尺寸即可得到点 p 在一维数组中的下标 i。纹理坐标分量的计算公式如下：

$$texZ=float(i/(s*s))/float(s-1) \tag{5}$$

$$texY=float(i\%(s*s))/float(s-1) \tag{6}$$

$$texX=float(i\%s)/float(s-1) \tag{7}$$

其中，$float$ 表示将整数转变为浮点型数据，其为 OpenGL 着色语言的构造函数，s 为查找表的尺寸。

由式(5)—式(7)可得到点 p 的偏移纹理坐标，通过纹理查找可得到点 p 的偏移值；由偏移值加上块内网格坐标，通过式(3)、式(5)—式(7)可得到点 p 的标记纹理坐标，通过纹理查找可得到标记值；将标记值和块内网格坐标进行比较，若相等，则说明点 p 是有效体素，然后由哈希纹理坐标等于标记纹理坐标，查找哈希纹理得到的值即为点 p 的数值；否则，点 p 为无效体素。

2　矢量量化

由文献[7]可知，矢量量化是将 K 维的输入矢量 X 用 M 个码表中的某一个矢量来表示，并用该索引值代替 X，解码则是利用索引值在码表中查找相应的矢量块。当 M 较小时，每个索引只需要很少的位数即可达到解压缩的目的。假设压缩比(Rate)用原始数据的容量除以压缩后数据的容量来表示，则有：

$$Rate=OriginalDataSize/CompressedDataSize \tag{8}$$

对于矢量块内的体素 p，其块内坐标为(x,y,z)，其中 x,y,z 取值为 0,1,2,3，则 p 在 64 维向量和 8 维向量的偏移地址分别为：

$$Offeset_{64}=z\times 16+y\times 4+x \tag{9}$$

$$Offeset_{8}=z/2\times 4+y/2\times 2+x/2 \tag{10}$$

2.1　FCHVQ

提高压缩比、降低复杂度和减少失真一直是矢量量化器追逐的目标[12]。大量实验表明，大部分体数据中超过 60%的体素是空体素[13]，而且体数据之间是有关联性的。基于以上考虑，文献[7]提出了 FCHVQ 算法，然而该算法是在 CPU 内实现的，导致绘制帧率较低，为了提高绘制帧率，本文改进了 FCHVQ 算法，使其能在 GPU 内解压缩绘制。改进后的主要步骤如下。

1)将体数据分成 4×4×4 大小的数据块，计算块内平均值，并保存到块平均值数组内，然后根据块内数据平均的梯度值进行分类：梯度值小于某个阈值的块视为 0 块，其他视为非 0 块；

2)对非 0 块的次高层和最高层采用矢量量化压缩，运用 PCA 分裂法得到初始码表，并用 LBG 进行码表优化；

3)用分块完美空间哈希技术存储对非 0 块压缩得到的两个索引值。

2.2　梯度计算

FCHVQ 考虑体数据相关性的特点，采用类似向前差分(或者采用向后差分)来逼近块梯度。假设 $G(x,y,z)$ 表示中心点 X 的梯度值，$f(x,y,z)$ 表示体素点 $X(x,y,z)$ 的值。则：

$$\begin{aligned}G(x,y,z)=&|f(x+1,y,z)-f(x,y,z)|+|f(x,y+1,\\&z)-f(x,y,z)|+|f(x,y,z+1)-f(x,y,\\&z)|\end{aligned} \tag{11}$$

那么，块平均梯度值：

$$avG=\frac{1}{n*n*n}\sum_{z_i}^{z_i+n}\sum_{y_i}^{y_i+n}\sum_{x_i}^{x_i+n}G(x,y,z) \tag{12}$$

其中，n 表示沿坐标轴的分块数。由式(11)、式(12)可知，当块内的体素值非常接近时，得到的梯度值较小，可认为块内体素值比较接近，可用体素平均值来表示该块内的数据。本文设定一个阈值，如果平均梯度值小于该阈值，则将此数据块归

为 0 块，否则归为非 0 块。对于梯度值较大的块可用文献[2]中的方法，用一个平均值和两个码表索引值表示。

2.3 码表的设计与优化

采用主成分分析分裂法初始码表，步骤如下：

1)将所有块内的数据按照一定规则组织成矢量，并归入到一个类 C 中，计算 C 的平均矢量 Y，将 C 中所有矢量与平均矢量 Y 的欧氏总和记为 $dist$。构建一个双向链表，链表中每一个节点代表一个类 C，节点按照 $dist$ 值从大到小排序。

2)每次从链表中取出头节点并将其从链表中删除，得到一个类 C_max，计算得到它的协方差矩阵 CON。

3)计算协方差矩阵 CON 的最大特征值 λ 及其对应的特征向量 α。

4)对 C_max 中的所有矢量 β，计算 (β,α) 点乘积，若其小于 0，则将 β 归入到一个类 C_1 中，否则将其归入到类 C_r 中。

5)计算 C_1 和 C_r 的平均矢量，及各自类内的矢量到其类的平均矢量的欧氏距离和 $distl$，$distr$，按照 $dist$ 的大小将 C_1和 C_r 插入链表中。

6)重复 2)－5)，直到链表节点数达到码表大小，那么链表节点的类平均矢量即为码元矢量，所有节点的平均矢量即构成了初始码表。

与文献[2]不同，FCHVQ 仅仅对梯度值非零的向量进行码表初始化。

在得到初始码表之后，运用 LBG 对初始码表进行优化，其核心思想为 K 均值聚类算法，步骤如下：

1)由 PCA 算法得到初始码表的码元矢量作为 K-means 每个类的初始中心，码表大小即为 K 的取值，即 K-means 分类的个数。

2)对每个类，计算类内的每个矢量到其所在类的平均矢量的距离之和 $dist\_old$；对所有的矢量，计算其和每一个类的平均矢量的欧氏距离，将其归类到距离最近的平均矢量所在的类中。

3)对所有的类重新计算其平均矢量。

4)计算所有矢量到其所在类的平均矢量的欧氏距离之和 $dist\_new$，与 $dist\_old$ 作差得到 $dist\_change$，当 $dist\_change/dist\_old$ 的绝对值小于一定的阈值时，则退出，否则回到 2)。

由于 PCA 生成的码表已非常准确，LBG 码表精炼仅仅对码表做了细微的调整，因此 LBG 的码表精炼迭代次数较少。

2.4 FCHVQ 解码

由于对梯度值是否为 0 进行了分类，对不同的类采用了不同的压缩方法，因此解码时也采用不同的解码方法，步骤如下：

1)首先确定体素所在数据块在块平均值数组中的位置，得到块的平均值。

2)依据分块完美空间哈希的解码技术判断该数据块是否有索引值，如果有索引值，得到块的两个向量值，根据式(9)、式(10)计算得到体素的两层平均值，三者相加即为体素 p 的解码值，否则平均值即为体素 p 的解码值。

2.5 残差压缩

由于 FCHVQ 属于有损压缩，存在数据损失，为了降低压缩损失、减少失真，使得恢复数据尽量接近原始体数据，因此采用分块完美空间哈希算法压缩由原始体数据与恢复体数据作差得到的残差体数据。

解码时，首先利用 FCHVQ 算法的解码步骤得到体素的恢复值，其次利用分块完美空间哈希的解码步骤得到残差值，最后将恢复值和残差值相加，即可得到体素的解码值。

3 压缩域体绘制

3.1 光线投射

光线投射算法(Ray Casting)[14]是图像空间的体绘制技术，其基本原理为：从视点出发，通过屏幕空间中的每个像素点向对象空间发射光线，光线在穿越数据场的过程中，对体数据进行采样，在每条光线上形成一系列按深度排序的采样点，将同一条光线上的所有采样点数值映射成光学属性(颜色和不透明度)并按深度顺序累加，形成最终的像素颜色[15]。

3.2 基于 GPU 的压缩域绘制

3.2.1 纹理构造

将两个码表构造为二维纹理、块平均值数组；哈希表、偏移表、标记表构造为三维纹理，加载到 GPU 内，以供解压缩使用。

3.2.2 纹理与绘制

CDVR 将 GPU 的解压缩和体绘制耦合在一起，使用光线投射技术作为直接体绘制方法。将构造的纹理加载到 GPU 内，进行实时的解压缩。

解码步骤如下：

1)首先利用分块完美空间哈希技术分块解码的算法判断体素 p 所在块是否有两个索引值。

2)若有，则利用式(9)和式(10)计算体素 p 的两层平均值，通过访问块平均值纹理得到块平均值，再将平均值与两层平均值相加，即得到恢复体素值；否则块平均值即为体素 p 的恢复体素值。

3)利用分块完美空间哈希技术分块解码的算法得到体素 p 的残差值。

4)将恢复体素值加上残差值即为解码得到的体素 p 的解码值。

将上述得到的 p 的解码值作为 p 的体素值，按照标准的光线投射流程进行直接体绘制，得到该体数据的压缩域体绘制结果。

此外，本文所提及的所有纹理的滤波器模式设定为 GL_NEAREST，以免插值变化，寻址纹理的地址取样属性 GL_WRAP 设定为 GL_REPEAT，即相当于将寻址纹理的数据重复平铺于原始体数据空间，以达到节省空间和压缩的效果。

4 实验结果与分析

本文实验在 4 核 Inter i5-3470 3.20GHz CPU，8GB 内存，NVIDIA GeForce GTX 480(4095MB 显存)图形卡的 PC 机上完成。

4.1 实验数据

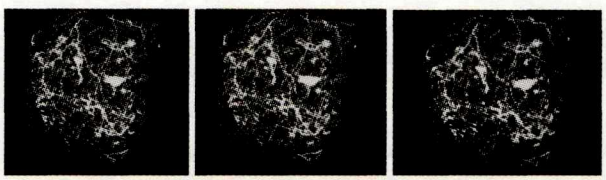
通过对饲养 21 周的 GFP-Mline 小鼠脑细胞进行 $0.5\times0.5\times2um^3$ 采样，得到单张冠状面数据。

选取了小鼠脑细胞的两个不同区域，形成了 1024×1024×1024和 2048×2048×1024 的立方体网格，用 8 位的灰度值表示每个体素，其大小为 1GB 和 4GB。将块内平均梯度值阈值设为 64，残差阈值设为 10。

4.2　实验结果分析

图 3 是 1GB 数据的绘制效果，图 4 是 4GB 数据的绘制效果。图 3 和图 4 中的(a)为本文方法的绘制效果，图 3 和图 4 中的(b)为采用 HVQ 和 PSH 方法的绘制结果，图 3 和图 4 中的(c)为采用 FCHVQ 方法的绘制结果。由图 3 与图 4 的可视化效果比较可知，图 3 和图 4 中的(a)和(b)能够保留更多的神经回路细节，而且绘制效果要比图 3 和图 4 中的(c)看上去更加细腻。

从表 1 中可以看出，当数据较稀疏时，相比文献[9]的 HVQ，PSH 方法，本文方法的压缩比较高；本文方法没有 FCHVQ 的压缩率高，主要是因为其在 CPU 端压缩绘制，用位来标记块是否为非 0 块，然而本文方法的绘制质量要明显高于 FCHVQ。



(a)FCHVQ,PSH　　(b)HVQ,PSH　　(c)FCHVQ

图 3　3 种方法绘制小鼠部分脑区的结果示意图

(a)FCHVQ,PSH　　(b)HVQ,PSH　　(c)FCHVQ

图 4　3 种方法绘制小鼠部分脑区的结果示意图

表 1　不同压缩方法的对比

压缩方法	压缩比/1G	压缩比/4G
HVQ,PSH	19	21
FCHVQ	33	34
FCHVQ,PSH	25	24

结束语　本文针对高精度脑成像数据可视化过程中存在的数据量大以及绘制效率低的问题，提出了基于分类分层矢量量化和完美空间哈希相结合的压缩域可视化算法。首先，将体数据进行分块处理，并将各块内数据的平均值的计算结果保存到块平均值数组中；然后通过计算各块内的平均梯度值来判断该块是否为 0 块，对于 0 块，仅用一个平均值来表示块的数据，对于非 0 块，采用三层数据结构表示块的数据；接着将上述压缩数据进行解压，得到恢复体数据，让原始数据与恢复体数据相减，得到残差体数据，通过设置一定的阈值来判断残差体素是否为有效体素，然后用分块完美空间哈希压缩残差体数据。当进行体绘制时，只需将解码时需要的数据作为纹理加载到 GPU 内，并在 GPU 内进行实时的解压缩就可得到体素值，最后利用标准的光线投射方法进行直接体绘制，得到该体数据的压缩域体绘制结果。

所提算法目前主要考虑静态数据，未来可探索时变体数据的 CDVR 流水线。

参 考 文 献

[1] LI A A, GONG H, ZHANG B, et al. Micro-optical Sectioning tomography to obtain a high-resolution atlas of the mouse brain [J]. Science, 2010, 330(6009): 1404-1408.

[2] SCHNEIDER R J, WESTERMANN R. Compression domain volume rendering[C]// Visualization, 2003. VIS. IEEE, 2003: 293-300.

[3] NING P, HESSLINK L. Vector quantization for volume rendering[C]// Proceedings of the 1992 workshop on Volume visualization. ACM, 1992: 69-74.

[4] NING P, HESSELINK L. Fast volume rendering of compressed data[C]// IEEE Conference on Visualization, 1993. Visualization'93, Proceedings. IEEE, 1993, 11-18.

[5] KRAUS M, ERTL T. Adaptive texture maps[C]// Proceedings of the ACM SIGGRAPH/EURO-GRAPHICS conference on Graphics hardware. Eurographics Association, 2002: 7-15.

[6] FOUT N, AKIBA H, MA K L, et al. High quality rendering of compressed volume data formats[C]// Proceedings of the Seventh Joint Eurographics/IEEE VGTC conference on Visualization. Eurographics Association, 2005: 77-84.

[7] ZHAO L P, XIAO D G, LI K L, et al. An Efficient Algorithm for Large-Scale Volume Data Compression and its Application in Seismic Data Processing[J]. Journal of Computer-Aided Design & Computer Graphics, 2009, 21(11): 1606-1611. (in Chinese)
赵利平，肖德贵，李肯立，等. 一种高效体数据压缩算法及其在地震数据处理中的应用[J]. 计算机辅助设计与图形学学报，2009，21(11)：1606-1611.

[8] DING Z Y. Efficient Visualization of Multivariate Spatial Data [D]. Zhejiang: Zhejiang University, 2014. (in Chinese)
丁治宇. 多变量空间数据场的高效可视化[D]. 杭州：浙江大学，2014.

[9] LEFEBVRE S, HOPPE H. Perfect spatial hashing[J]. ACM Trans. Graph., 2006, 25(3): 579-588.

[10] ZHOU K, ZHONG R, LIN S, et al. Real-time smoke rendering using compensated ray marching [J]. ACM Transactions on Graphics, 2008, 27(3): 1-12.

[11] YE S, LI X, WANG G Z, et al. GPU-Friendly Regularization and Volume Rendering of Tetrahedral Volumetric Datasets [J]. Journal of Computer-Aided Design & Computer Graphics, 2011, 23(6): 933-940. (in Chinese)
叶楔，李昕，王桂珍，等. 适用于 gpu 的四面体体数据规则化与可视化[J]. 计算机辅助设计与图形学学报，2011，23(6)：933-940.

[12] 孙圣和，陆哲明. 矢量量化技术及应用[M]. 北京：科学出版社，2002.

[13] TONG X, TANG Z S. 3D Texture Hardware Assisted Volume Rendering with Space Leaping[J]. Chinese Journal of Computers, 1998, 21(9): 807-812. (in Chinese)
童欣，唐泽圣. 基于空间跳跃的三维纹理硬件体绘制算法[J]. 计算机学报，1998，21(9)：807-812.

[14] LEVOY M. Display of surfaces from volume data[J]. IEEE Computer Graphics and Application, 1988, 8(3): 29-37.

[15] 李思昆，蔡勋，王文珂，等. 大规模流场科学计算可视化[M]. 长沙：国防工业出版社，2013.