基于改进八叉树的三维点云压缩算法

黄源[1,2]，达飞鹏[1,2]，唐林[1]

[1]东南大学自动化学院，江苏 南京 210096；

[2]复杂工程系统测量与控制教育部重点实验室，江苏 南京 210096

摘要 针对大数据环境下，三维模型的传输和存储需求，提出了一种基于八叉树的三维点云有损压缩算法。该算法改进了八叉树分割的停止条件，可以在适当的深度停止分割并确保体素大小合适。同时在分割的基础上通过建立 K 邻域，利用简单有效的统计方法去除原始点云的离群点。在数据结构上，对每个节点分配位掩码，通过操纵位掩码，在遍历时对数据查询和操作，并优化随后的点位置编码。该算法可以有效地移除离群点和表面杂点，并在区间编码上提高了点云压缩效率。实验结果表明，该算法较完整地保留了三维点云数据的关键信息，取得了良好的压缩率并缩短了压缩时间。

关键词 图像处理；点云压缩；八叉树；K 邻域；区间编码

中图分类号 TN958.98 **文献标识码** A

doi：10.3788/AOS201737.1210003

Three-Dimensional Point Cloud Compression Algorithm Based on Improved Octree

Huang Yuan[1,2], Da Feipeng[1,2], Tang Lin[1]

[1] *School of Automation, Southeast University, Nanjing, Jiangsu* 210096, *China*;

[2] *Key Laboratory of Measurement and Control of Complex Systems of Engineering, Ministry of Education, Nanjing, Jiangsu* 210096, *China*

Abstract Aiming at the transmission and storage requirements of three-dimensional model in the large data environment, a three-dimensional point cloud lossy compression algorithm based on the octree is presented. The stop condition of the octree segmentation is improved, so the segmentation can be stopped at an appropriate depth, and the proper size of voxel is ensured. At the same time, the K neighborhood is established based on the segmentation and the outliers of original point cloud are removed by simple and effective statistical method. In the data structure, each node is assigned to a bit mask. The data query and manipulation are traversed by manipulating the bit mask. Then the subsequent point position coding are optimized. The proposed algorithm effectively removes the outliers and miscellaneous points on the surface, and improves the efficiency of point cloud compression in range encoding. The experimental results show that this algorithm can preserve the key information of three-dimensional point cloud data more completely, obtain a good compression rate and shorten compression time.

Key words image processing; point cloud compression; octree; K neighborhood; range encoding

OCIS codes 100.6890; 100.4999; 100.5010

1 引 言

通过三维测量技术可以获得复杂物体的点云数据，其中常见的方法有激光三维测量法[1-2]和光栅投影测量法[3-4]。然而随着技术的发展，测量设备的精度也随之提高，获取物体的三维数据数量级也越来越大，这给后续的传输和储存都带来了挑战。为了节省存储空间和提高传输效率，众多学者开始对数据压缩领域展开

收稿日期：2017-07-03；**收到修改稿日期**：2017-08-17

基金项目：国家自然科学基金(61405034、51475092、61462072)

作者简介：黄源(1988—)，男，博士研究生，主要从事三维点云及图像处理等方面的研究。E-mail：whhbb@163.com

导师简介：达飞鹏(1968—)，男，教授，博士生导师，主要从事三维信息获取与处理、三维生物特征识别、智能控制等方面的研究。E-mail：dafp@seu.edu.cn

研究。相比较目前已经较为成熟的多边形网格压缩领域，三维点云压缩是一个比较新的概念，考虑到在实际应用中，用多边形网格来表示高精度模型需要数量级较大的网格，维护和存储都将消耗较多的内存和时间。同时，当投影转换到平面的多边形数目比平面像素还多时，多边形网格在平面上的投影将会比一个平面像素更小，此时利用点来替代网格作为模型的基本单元将更有优势。因此基于点的图形学成为了近些年的研究热点[5-7]。而根据编译解码后数据精度或点云个数是否损失，可以将压缩算法分为有损点云压缩和无损点云压缩。在工程上，点云模型的位置信息和属性信息都使用浮点数进行表示，而浮点数的精度较大，实际工程中并不需要这么高的精度，因此目前提出的多数算法都属于有损范围。

Kalaiah 等[8]利用 k-means 聚类的方法对点云空间进行层次划分，然后采用主成分分析（PCA）将点云的几何信息及属性信息划分为一系列的高斯概率函数，通过对这些概率分布函数进行随机采样实现重构和绘制，构建并实现了一个可以在各种平台上运行的点云压缩传输和绘制框架；Botsch 等[9]提出了一种针对点采样曲面区域进行编码绘制的算法，存储和绘制点云数据。算法将点坐标 p 编码为递归子分八叉树中最小包围单元的中心，每次细分的字节编码提供每个节点的树平衡信息；Peng 等[10]将类似的八叉树编码方法运用到网格模型压缩中，将掩码压缩分为两步并引入预测技术，在第二步预测时运用了模型中的拓扑关系；Schnabel 等[11]运用了与文献[10]相似的方法，与之不同的是，该算法在第一步也引入了预测技术，针对密集采样的点云数据算法取得了较为理想的像素深度（bpp）和峰值信噪比（PSNR）值；Huang 等[12]也提出了基于八叉树结构的压缩算法，该算法利用重新排序的掩码来预测八叉树中非空子节点的概率，从而将掩码中的 1 推到一端，使得利用算术编码时掩码的熵值减小，从而提高压缩率；根据点集的几何结构，Bordignon 等[13]提出了一种自适应的二叉空间分割（BSP）压缩算法。为了充分利用几何信息，在空间剖分过程中，确保 BSP 中每个单元的切面与单元中点的主方向垂直，然后采用渐进编码的方法将几何编码的开销分布在每个点上。

与此同时，在利用三维系统获取点云数据的过程中，由于采集设备精度、操作者经验、采集环境因素以及被测物体表面变化和数据拼接配准等操作的影响，采集得到的点云数据往往存在一些离主体点云即离被测物体点云较远的离散点，即离群点[14-15]。不同的设备获取的离群点结构也有不同。原始点云中离群点的干扰，会使得点云的处理变得困难。目前，离群点移除也受到了国内外很多研究人员的重视，提出了基于局部点云特征移除方案，对采样点处曲率变化率或法向量进行运算，但是该类计算量较大且复杂，很有可能得到错误的数值，最终可能引发点云后期处理的失败。

在上述研究的基础上，针对目前所存在的问题，本文利用改进的八叉树算法对数据进行分割，从实际应用角度出发采用统计的方法去除离群点，并且在后续操作中优化了数据点的位置编码。通过压缩与解压缩后的效果以及与其他现有压缩算法的对比实验可以看出所提算法在保证了原始数据精度的前提下，有效地提高了压缩率，降低了压缩时间。

2　算法简介

基于无序的点云模型，所提出的压缩算法主要步骤如下：

1）八叉树分割：根据读入点云数据坐标的 X、Y、Z 的最大最小值构造包围盒，根据八叉树划分条件划分包围盒。

2）离群点移除：在分割的基础上，建立 K 邻域，并在八叉树最深处叶子节点分配位掩码，根据点的邻域的统计分析和位操作移除离群点。

3）点位置细节编码和区间编码：为了提高解压精度，对每个体素所占有的点做处理。在执行序列化的同时，利用广度优先遍历查询和编码点的局部细节，对含有超过一个点的体素，计算点和体素中心的拓扑关系，产生与各自体素中心相关的位置细节参数流。最后利用区间编码输入至集合输入流，输出的点云压缩数据被写至一个文件中。

算法流程如图 1 所示。

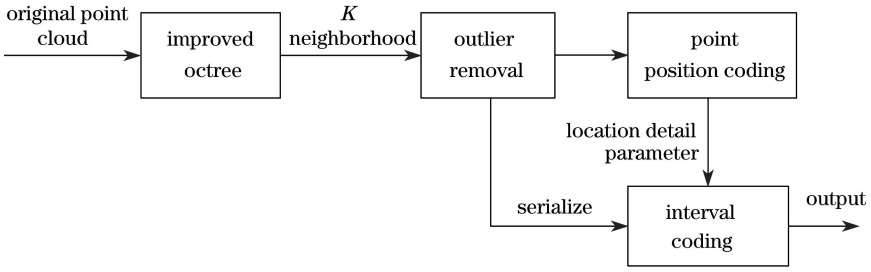

图 1　本文算法流程图

Fig. 1　Flowchart of the proposed algorithm

3　基于改进八叉树的点云压缩算法

3.1　八叉树分割

3.1.1　简单八叉树结构

八叉树结构在空间划分上有很强的优势。先在空间形成包围盒，八叉树将包围盒分成八个相同时间空间复杂度的子区间。八叉树的存储方式沿用了二维四叉树的有关方法。存储方法充分利用了八叉树结构在空间的相关性。因此，所占存储空间比三维体素阵列要少。八叉树的主要优点在于可以方便地对编码之后的点进行集合计算（例如求交、并、异或），这大大提高了计算效率，这也是其他方法难以与之匹敌的地方。不仅如此，由于通过八叉树划分之后的空间具有良好的有序性和分层性，也提高了点云的计算精度和速度。

八叉树结构通过对三维空间的几何实体进行空间划分，通过循环递归的方法对大小为 $2^n\times2^n\times2^n$ 的八叉树空间进行 8 等分划分，经过划分所得的空间具有相同的时间和空间复杂度，如果划分所得的最小包围盒具有相同的属性，则包围盒构成八叉树的一个节点；否则将继续对空间进行划分，对于 $2^n\times2^n\times2^n$ 大小的空间对象，最多剖分 n 次。图 2 为在某一分割停止条件下八叉树分割的点云模型的包围盒示意图及其局部放大细节。

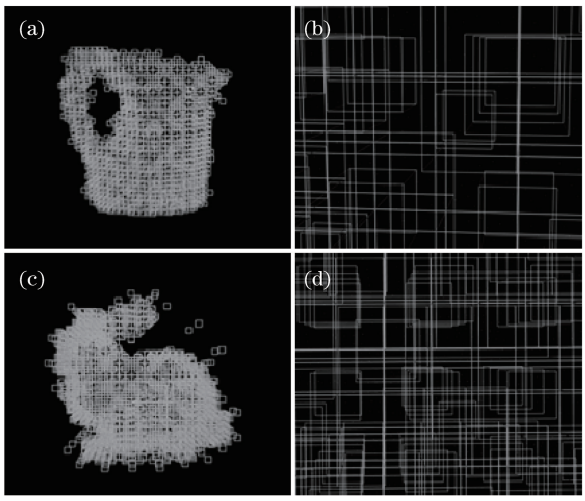

图 2　包围盒示意图及局部放大细节。(a)水壶模型包围盒示意图；(b)水壶模型的局部放大细节；(c)兔子模型包围盒示意图；(d)兔子模型的局部放大细节

Fig. 2　Bounding box schematic and local enlargement details. (a) Bounding box schematic of kettle model; (b) local enlargement details of kettle model; (c) bounding box schematic of bunny model; (d) local enlargement details of bunny model

在完成逐层划分之后，对数据进行编码，编码方式为：假设点云坐标 $P(x,y,z)$ 和树中节点 (a,b,c) 相对应且树中任一个节点都与一个最小包围盒一一对应；节点的编码为 $M=m_{n-1}\cdots m_2m_1m_0$，$m_{n-1}$ 为节点在 n 层的节点序号。

首先利用空间坐标计算出节点索引值：

$$\begin{cases} a = \mathrm{int}\left(\dfrac{x - x_{\min}}{\lambda}\right) \\ b = \mathrm{int}\left(\dfrac{y - y_{\min}}{\lambda}\right) \\ c = \mathrm{int}\left(\dfrac{z - z_{\min}}{\lambda}\right) \end{cases}, \tag{1}$$

式中 x、y、z 为点在空间的坐标，$x_{\min}$、$y_{\min}$、$z_{\min}$ 为点云在空间中的最小坐标值，λ 为八叉树分辨率，int 表示计算结果取整。

其编码可用二进制表示为

$$\begin{cases} a = a_{n-1}2^{n-1} + a_{n-2}2^{n-2} + a_{n-3}2^{n-3} + \cdots + a_0 2^0 \\ b = b_{n-1}2^{n-1} + b_{n-2}2^{n-2} + b_{n-3}2^{n-3} + \cdots + b_0 2^0 \\ c = c_{n-1}2^{n-1} + c_{n-2}2^{n-2} + c_{n-3}2^{n-3} + \cdots + c_0 2^0 \end{cases}。 \tag{2}$$

节点序号 m_i 和 a_i，b_i，c_i 之间的关系可表示为

$$m_i = a_i + b_i \cdot 2 + c_i \cdot 4。 \tag{3}$$

若知道八叉树某一节点序号 m_i，利用 m_i 和 a_i，b_i，c_i 之间的关系得出：

$$a_i = (m_i \bmod 2), b_i = [(m_i/2) \bmod 2], c_i = [(m_i/4) \bmod 2], \tag{4}$$

式中 mod 为取模运算符，利用上述公式可以由点 A 包围盒邻接的立方体求出节点在树中的编码，即可从根节点沿路径检测出节点，按此规则，在点 A 搜索出距离最近的 K 个点，并得到点 A 的 K 邻域 $N|A|$。

通过八叉树结构对每个点搜索其邻域。查询非空叶子节点所在立方体的数据点和周围 26 个叶子节点立方体中的数据点，找到最近的 K 个点作为点云的 K 邻域，如果邻域内点的个数不足 K 个，则扩大搜索范围至 124 个立方体，最终完成 K 邻域的建立。

3.1.2　分割停止条件

在实际操作中，对于分割停止条件，并无确定的标准。目前一般使用八叉树分辨率进行限制。但当分辨率过小时，过多的分割会造成内存浪费；当分辨率过大时，体素就越大，由于利用体素形心近似表示原始点，因此造成的误差越大。本文提出了一种新的分割停止条件。以构造的包围盒为划分对象，根据分割停止标准划分最小包围盒，其中分割停止标准的建立基于叶子节点/点云比率、八叉树分辨率 λ、层数之间的关系，计算点云总数和非空节点之间的差值与点云总数的比值，即在划分过程中所产生的一定精度损失率，表示为

$$\begin{cases} \lambda^3 > \dfrac{V_{\mathrm{cube}}}{8^n} \\ \sqrt[3]{2V_{\mathrm{cube}}} \geqslant \lambda \geqslant \lambda_1 \\ t = \dfrac{M - P}{M} \end{cases}, \tag{5}$$

式中 P 表示空间中最小包围盒的数量，M 为当前层点云中点的数量，λ 为八叉树分辨率，λ_1 为扫描仪精度，t 为精度损失率，n 为层数，V_{cube} 为包围盒的体积。调节分辨率 λ，确保 t 在[0，0.1]之间，图 3 为调整 λ 时，不同层数 n 所对应的分割示例，P_{voxel} 为当前层所对应的最小包围盒。

3.2　离群点移除

对于完成分割后的数据，利用熊邦书等[15]的方法建立 K 邻域，然后对每个点的邻域进行统计分析。通过计算点云集中点到邻域的距离来处理离群点。针对空间的每一个点，得到其所在邻域内点的平均距离和标准差。假设所得数值呈高斯形态分布，设定离群点的阈值，平均距离在阈值之外的点，将其定义为离群点并移除点云空间。

算法首先通过广度遍历对八叉树分层遍历，对最深层的叶子节点添加掩码。如图 4 所示，假设八叉树只分为 2 层，最深层点的掩码可表示为 1000010001001000。

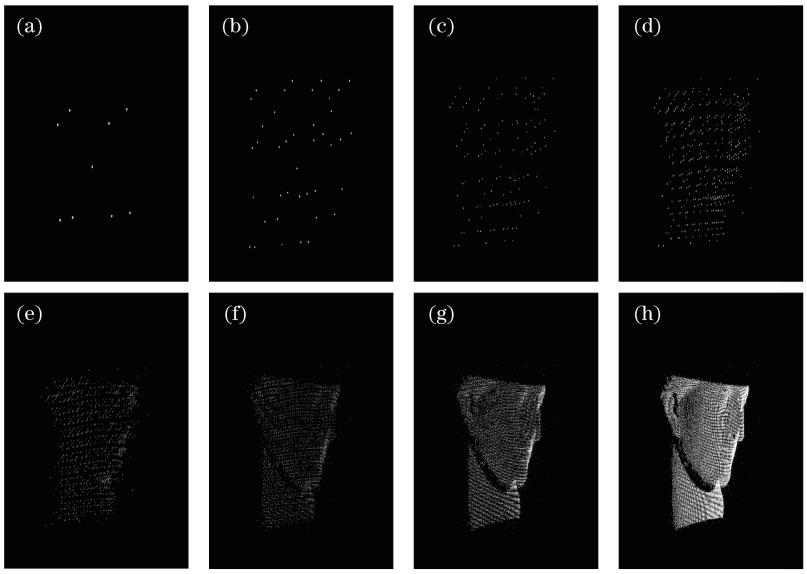

图 3　八叉树分割，每个体素至少含一个点。(a) $n=1$，$P_{\text{voxel}}=9$；(b) $n=2$，$P_{\text{voxel}}=40$；(c) $n=3$，$P_{\text{voxel}}=137$；(d) $n=4$，$P_{\text{voxel}}=435$；(e)$n=5$，$P_{\text{voxel}}=1426$；(f) $n=6$，$P_{\text{voxel}}=4815$；(g) $n=7$，$P_{\text{voxel}}=15845$；(h) $n=8$，$P_{\text{voxel}}=45859$

Fig. 3　Octree segmentation, each voxel containing at least one point. (a) $n=1$, $P_{\text{voxel}}=9$; (b) $n=2$, $P_{\text{voxel}}=40$; (c) $n=3$, $P_{\text{voxel}}=137$; (d) $n=4$, $P_{\text{voxel}}=435$; (e)$n=5$, $P_{\text{voxel}}=1426$; (f) $n=6$, $P_{\text{voxel}}=4815$; (g) $n=7$, $P_{\text{voxel}}=15845$; (h) $n=8$, $P_{\text{voxel}}=45859$

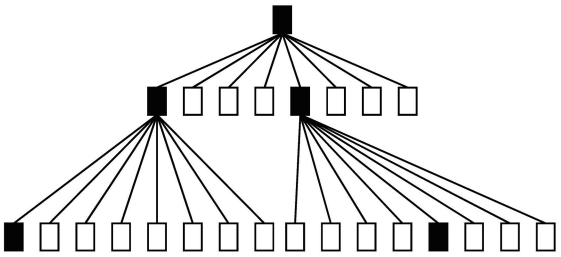

图 4　两层八叉树掩码结构

Fig. 4　Mask architecture of two layer octree

节点 N 的位掩码表示为 S^N，遍历查询的离群点 K 的位掩码表示为

$$S^N=\{S_i^N\}=\{S_1^N S_2^N\cdots S_7^N S_8^N\},\tag{6}$$

式中 S^N 的长度为 8。如果给定节点 $S^N=1$ 表示节点包围盒至少有一个点。$S^N=0$ 表示包围盒为空。

离群点去除的方法如下：设 N_n 为新八叉树 S_n 的根部，当判断八叉树 S_n 没有遍历至底端时，N_n 赋值为有遍历顺序 S_n 的非空节点，如果 S_n 的节点 N_n 满足 N_K 一样的离群点条件则 $S^K=\text{bitewiseAND}(S^K,0)$，否则继续遍历并将 N_n 赋值为 S_n 的新的非空节点，直到遍历完八叉树为止。其中 bitewiseAND 为按位与操作。

3.3　点位置细节编码

根据八叉树特性，所有被分配到一个特定体素的点只会被体素中心表示。在实际处理中，并不能保证 100% 的分割(即每个体素只含一个点)，因此采用额外的点位置编码来提高精度。

对于八叉树分割之后含有超过一个点的体素，采用点位置编码，并计算每个点坐标 X、Y、Z 与体素中心 A 之间的距离，把 A 点与每个点之间的矢量关系编码存储起来，因此解码时可以准确地还原位置。如图 5 所示，基于扫描目标的获取精度，离散化的局部点坐标用正整数表示，整数值受体素分辨率和特定精度的限制。在随后的区间编码时，点位置参数能够有效地被较小的位表示。该技术可有效减少计算复杂度，尤其适用于点密集并且曲率面较大的点云。

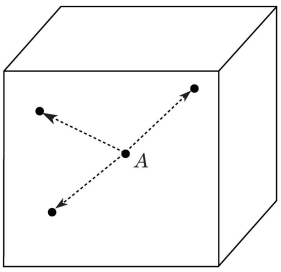

图 5　点位置细节编码

Fig. 5　Detail encoding of point position

3.4　区间编码

最后对输入的数据流进行熵编码，常用的方法有哈夫曼编码、算术编码和区间编码。综合考虑结合效率和压缩率，选择了区间编码。

区间编码是在整数区间上进行整数运算，区间长度为 s 或表示为 b^d，其中 b 为基数。设有一整数区间 $i\in[L,H)$，L 为区间下沿，H 为区间上沿。区间范围 $R=H-L+1$。T 为符号的总计频率。f_S 为符号 S 的频率。令 F_S 为符号 S 的累计频率。累积频度是符号小于 S 的其他符号的频率总和：

$$F_S=\sum_{x<S}f_x。\tag{7}$$

假设 A_i 是信息中的第 i 个想要编码的字母，$1\leqslant i\leqslant k$。根据频率表选择 s 大小的长度编码 A_1，留下 R_1 的区间长度编码 A_2，留下 R_2 的区间长度编码 A_3，以此类推，区间长度的计算公式为

$$R(0)=s,R_i=\left[\frac{R(i-1)}{T}\cdot(F_S+f_S)\right]-\left[\frac{R(i-1)}{T}\cdot F_S\right],\tag{8}$$

如果 $B_j=\sum_{i=1}^{j}\left[\frac{R(i-1)}{T}\cdot F_S\right]$ 在长度 s 上的完整范围是 $[Bk,Bk+Rk)$，当 Rk 小于一定阈值时，从 $[B_k,B_k+R_k)$ 中提取一个数值来表示整个压缩数据。

4　实　　验

本文算法实验仿真编译结果在 Microsoft Visual Studio 2012 32 位 win7 系统上进行，并对八叉树分割、离群点移除以及最后的压缩结果进行了分析。结合实验室在点云处理中常用的点云模型，验证了本文算法的实用性和有效性。点云模型如图 6 所示，图 6(a)为 Stanford 大学所提供的兔子模型，图 6(b)为三维重建实验中常用的 3D 模型，图 6(c)为水壶模型经过离群点移除后的模型，图 6(d)、(e)分别为人脸模型和塑料模型利用实验室光栅投影三维测量得到的点云模型。

表 1 为不同模型的八叉树分割，选择合适的 λ 值，使得在八叉树最深层满足 $t\in[0,0.1]$，意味着基本实现了每个最小体素包含一个点，t 值越小，在近似和后期操作时造成的误差越小。λ 值需要通过(5)式不断逼近。对实验室的点云设置 $\lambda=0.0018$ m。

表 1　八叉树分割

Table 1　Octree segmentation

Model	λ	Number of plies	t
Kettle	0.00180	7	0.0042
Plastic	0.00183	8	0.0050

完成八叉树分割之后，为了保证表面平滑和不受噪声点干扰，在所建立的 K 邻域内进行离群点移除。移除原则为：确保所滤除点中离群点占绝大多数，一般离群点分布在边缘处以及在光栅投影黑影处。对 K 值和 a 值的选取视所拍摄的物体表面复杂度和大小而定，对于同一物体，一般选择相同的 K 值和 a 值，首先确定 K 值大小，a 值受 K 值限制。目前并没有针对 K 值的一个自动预测器，因此 K 值一般选择经验值。以塑料模型为例，选择 $K=50$，$a=3.0$，滤除点数为 274，离群点移除结果如图 7 所示。

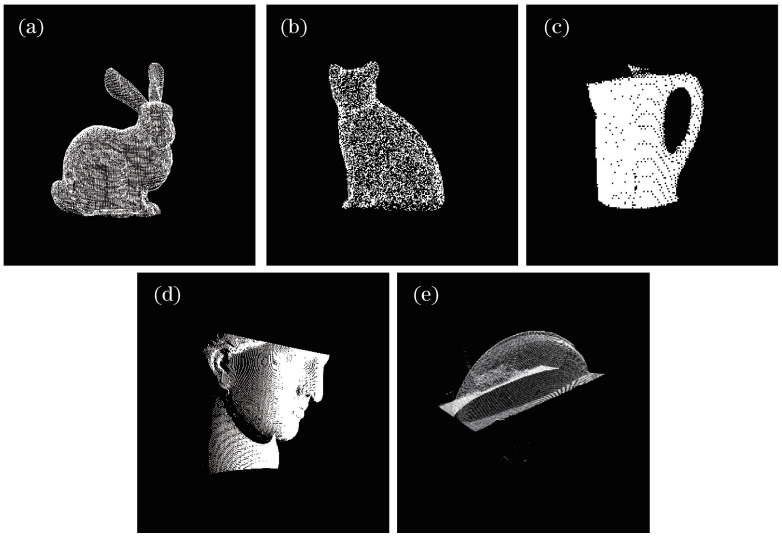

图 6　实验所用点云模型。(a)兔子模型,点数为 31607;(b)猫模型,点数为 10000;(c)水壶模型,点数为 19062;(d)人脸模型,点数为 46111;(e)塑料模型,点数为 24673

Fig. 6　Point cloud models used in experiment. (a) Bunny model, point is 31607; (b) cat model, point is 10000; (c) kettle model, point is 19062; (d) face model, point is 46111; (e) plastic model, point is 24673

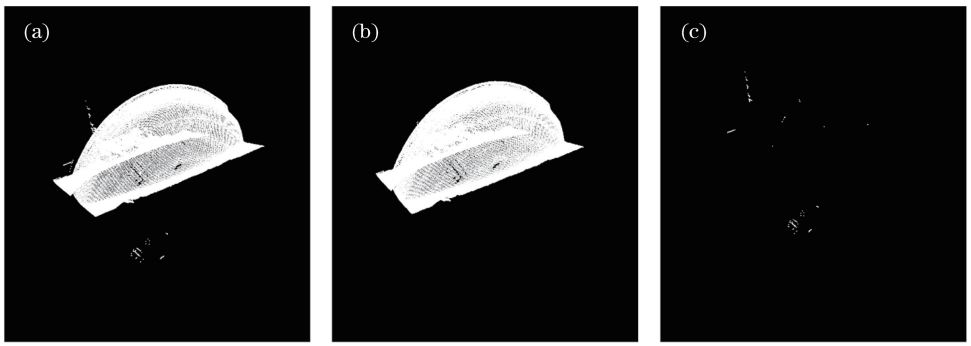

图 7　离群点移除结果。(a)原始数据;(b)移除离群点后的点云数据;(c)移除的离群点

Fig. 7　Results of removing outliers. (a) Initial data; (b) point cloud data after removing outliers; (c) removed outliers

从图 7 可以看出,离群点主要分布在模型的下侧和表面附近,为了更好地观察移除效果,对点的大小放大一倍。在确保点的低损失率的前提下,利用本文算法可有效滤除离群点。因为造成了点的损失,本文算法对于实验室点云属于有损压缩。不过对于点云模型兔子、猫和水壶,在进行解压缩之后,点的个数没有损失,且损失的精度也在可控范围内,可以与其他无损压缩技术相比较。

图 8 为兔子模型的解压缩示意图,根据(5)式设置不同的 λ 得到的 t 值不同。图 8(a)～(e)为解压缩前后的兔子模型,图 8(b)～(f)为模型背部的细节放大。解压缩前后的点云在整体上基本没有变化,但由于解压缩计算过程中,在最小包围盒内部使用中心点近似取代原始点,因此不可避免地造成了点位置精度的损失。通过图 8(d)可以观察得到,当 $t\notin[0,0.1]$ 时,点的偏移会比较大,较大的偏移破坏了区域平滑性和曲率一致性。红色标记区域点的位置具有较为明显的偏移。可以通过(5)式调节 λ 值使得 t 满足 $t\in[0,0.1]$来降低偏移量,提高解压缩精度,图 8(f)中 $t\in[0,0.1]$,可以看出点的偏移明显减小,点云区域的特性得以较好的保留。

另外还将本文算法与文献[16]、[17]算法进行了性能对比。文献[16]算法利用生成树划分空间,并对浮点数做处理;在压缩时间和压缩率上都逊于文献[17]算法,其中压缩率采用字节每点(Bpp)为单位。文献[17]中采用八叉树对空间划分,主要应用于动态点云压缩,不过利用该算法对静态点云也取得了较好的压缩率。

由图 9 可知,文献[17]和本文算法对比于文献[16]算法无论是在压缩率还是压缩时间上都取得了极高的提升。主要是因为文献[16]算法构造最小生成树并分块对浮点数进行处理,算法的关键放在了浮点数的

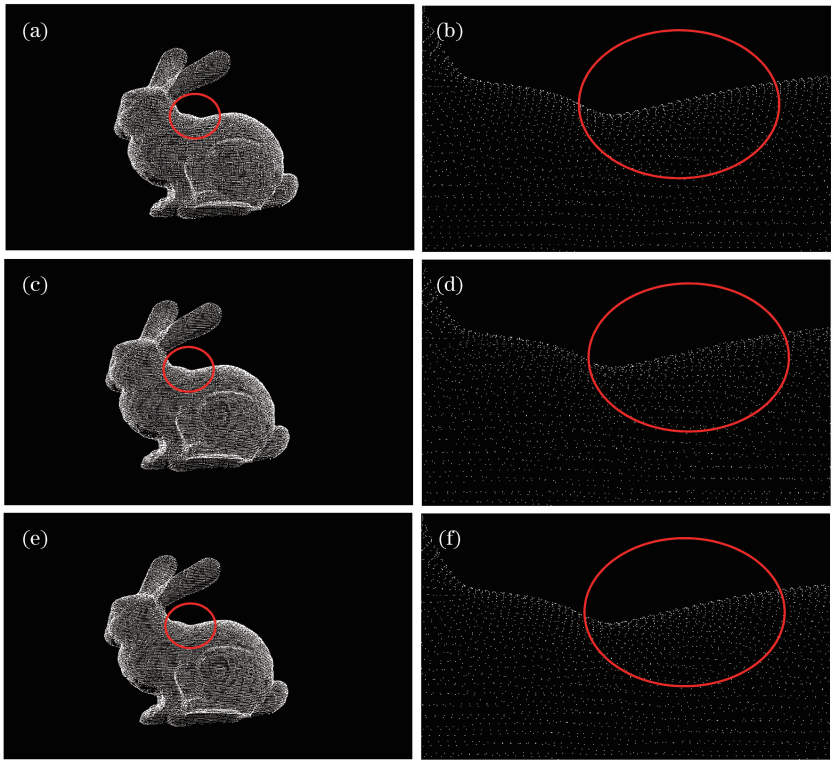

图 8　兔子模型解压缩示意图。(a)解压缩前的兔子模型;(b)图(a)局部放大细节;(c) $t=0.25$ 时解压缩结果;(d)图(c)局部放大细节;(e) $t=0.005$ 时解压缩结果;(f)图(e)局部放大细节

Fig. 8　Decompression schematic of bunny model. (a) Bunny model before decompression; (b) local amplification details of Fig. (a); (c) decompression result when $t=0.25$; (d) local amplification details of Fig. (c); (e) decompression result when $t=0.005$; (f) local amplification details of Fig. (e)

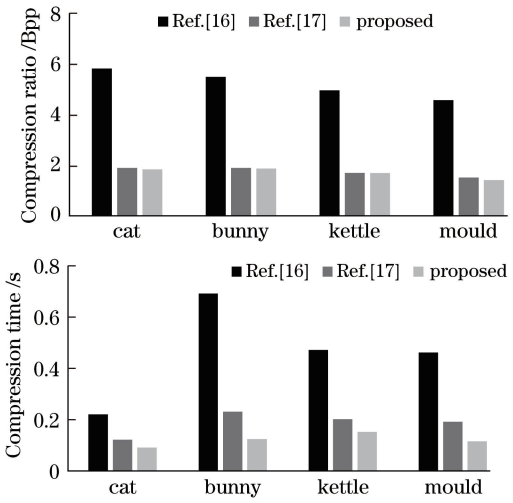

图 9　压缩率及压缩时间对比

Fig. 9　Comparison of compression ratio and compression time

分析,因而对空间拓扑分析不足,导致空间冗余处理的效果不理想。而本文算法和文献[17]算法利用八叉树进行空间划分并建立了清晰的拓扑关系,当确定点坐标精度(稍小于光栅采集精度)时,与文献[17]算法相比,针对静态点云压缩,本文算法并不考虑时间冗余,并且在划分时引入了掩码编码,方便在查询和遍历八叉树时访问和删除点。在相同点坐标精度的情况下,本文算法取得了和文献[17]算法同样优秀的压缩效果,并减少了压缩时间。而且在点的细节方面多做了处理,增加了空间复杂度,因此在解码时,能根据点坐标精度更加准确地还原点的位置和个数。

5 结 论

现代化点云采集设备所得到的点云数据量越来越庞大，如何在实际应用中高效地对数据进行传输和存储是目前的热点研究问题。针对静态点云，实现了一种基于改进八叉树的点云有损压缩算法，该算法能够有效地快速实现点云模型的压缩。该算法改进了八叉树的分割条件，优化了后续的区间编码。由实验结果可知，该算法具有良好的压缩率和较理想的压缩时间，后续将采用车辆、船只以及现代城市的部分等三维数据对算法进行进一步的测试及改进，以验证其能否真正满足实际应用中对三维点云数据的压缩处理需求。后续研究将会增加在区间编码中频率模型的复杂性，如果再结合有序点云，有望实现更高的压缩效率。

参 考 文 献

[1] Deng Wenjun, Ye Jingyang, Zhang Tie. Acquisition and denoising algorithm of laser point cloud oriented to robot polishing[J]. Acta Optica Sinica, 2016, 36(8): 0814002.
邓文君, 叶景杨, 张铁. 面向机器人磨抛的激光点云获取及去噪算法[J]. 光学学报, 2016, 36(8): 0814002.

[2] Stone E E, Skubic M. Fall detection in homes of older adults using the Microsoft Kinect[J]. IEEE Journal of Biomedical & Health Informatics, 2015, 19(1): 290-301.

[3] An Dong, Gai Shaoyan, Da Feipeng. A new model of three-dimensional shape measurement system based on fringe projection[J]. Acta Optica Sinica, 2014, 34(5): 0512004.
安冬, 盖绍彦, 达飞鹏. 一种新的基于条纹投影的三维轮廓测量系统模型[J]. 光学学报, 2014, 34(5): 0512004.

[4] Shi L, Yang X, Pan H L. 3D face visualization using grid light[J]. Computing in Science and Engineering, 2008, 10(2): 48-54.

[5] Kobbelt L, Botsch M. A survey of point-based techniques in computer graphics[J]. Computer & Graphics, 2004, 28(6): 801-814.

[6] Gross M, Pfister H. Point based graphics[M]. San Fransisco: Morgan Kaufmann Publishers, 2007.

[7] Lü Shuai, Da Feipeng, Huang Yuan. A fast and lossy compression algorithm for point-cloud models based on data type conversion[J]. Journal of Graphics, 2016, 37(2): 199-205.
律帅, 达飞鹏, 黄源. 基于数据类型转换的点云快速有损压缩算法[J]. 图学学报, 2016, 37(2): 199-205.

[8] Kalaiah A, Varshney A. Statistical geometry representation for efficient transmission and rendering[J]. ACM Transactions on Graphics, 2005, 24(2): 348-373.

[9] Botsch M, Wiratanaya A, Kobbelt L. Efficient high quality rendering of point sampled geometry[C]. Proceedings of the 13th Eurographics Workshop on Rendering, 2002: 53-64.

[10] Peng J L, Kuo C C J. Progressive geometry encoder using octree-based space partitioning[C]. IEEE International Conference on Multimedia and Expo, 2004, 1: 1-4.

[11] Schnabel R, Klein R. Octree-based point-cloud compression[C]. Proceedings of the 3rd Eurographics/IEEE VGTC Conference on Point-Based Graphics, 2006: 111-120.

[12] Huang Y, Peng J L, Kuo C C J, *et al*. A generic scheme for progressive point cloud coding[J]. IEEE Transactions on Visualization and Computer Graphics, 2008, 14(2): 440-453.

[13] Bordignon A, Lewiner T, Lopes H, *et al*. Point set compression through BSP quantization[C]. Proceedings of Brazilian Symposium on Computer Graphics and Image Processing, 2006: 229-236.

[14] Nie Jianhui, Hu Ying, Ma Zi. Outlier detection of scattered point cloud by classification[J]. Journal of Computer-Aided Design & Computer Graphics, 2011, 23(9): 1526-1532.
聂建辉, 胡英, 马孜. 散乱点云离群点的分类识别算法[J]. 计算机辅助设计与图形学学报, 2011, 23(9): 1526-1532.

[15] Xiong Bangshu, He Mingyi, Yu Huajing. Algorithm for finding k-nearest neighbors of scattered points in three dimension[J]. Journal of Computer-Aided Design & Computer Graphics, 2004, 16(7): 909-912.
熊邦书, 何明一, 俞华璟. 三维散乱数据的 k 个最近邻域快速搜索算法[J]. 计算机辅助设计与图形学报, 2004, 16(7): 909-912.

[16] Wang Pengjie, Pan Zhigeng, Xu Mingliang, *et al*. A fast and lossless compression algorithm for point-based models based on local minimal spanning tree[J]. Journal of computer Research and Development, 2011, 48(7): 1263-1268.
王鹏杰, 潘志庚, 徐明亮, 等. 基于局部最小生成树的点模型快速无损压缩算法[J]. 计算机研究与发展, 2011, 48(7): 1263-1268.

[17] Kammerl J, Blodow N, Rusu R B, *et al*. Real-time compression of point cloud streams[C]. IEEE International Conference on Robotics and Automation, 2012: 778-785.