南方科技大学
SOUTHERN UNIVERSITY OF SCIENCE AND TECHNOLOGY

计算机科学与工程系
Department of Computer Science and Engineering

计算机科学与工程系
CS321-创新实践 I

基于八叉树的 3D 点云压缩实现

作者: 王宇航 12012208 叶成伟 12010524
指导教师: 于仕琪
时间:2023/1/5

I. 项目简介

当前，3D 数据的应用越来越广泛，三维点云作为 3D 数据的重要组成部分，发展潜力巨大。然而，三维点云所表示的详细位置信息背后，往往伴随着巨大的数据量，这对于点云数据的处理造成了极大的不便。因此，三维点云的压缩有着极强的现实意义。本项目的研究内容为，在 OpenCV5.x 分支中，基于于八叉树实现可选分辨率的点云数据压缩，通过该算法，丰富 OpenCV5.x 对点云数据的处理功能。本项目旨在改善自动驾驶领域的数据传输与存储效率，为社会的实际需求，自动驾驶的落地贡献一份力量。

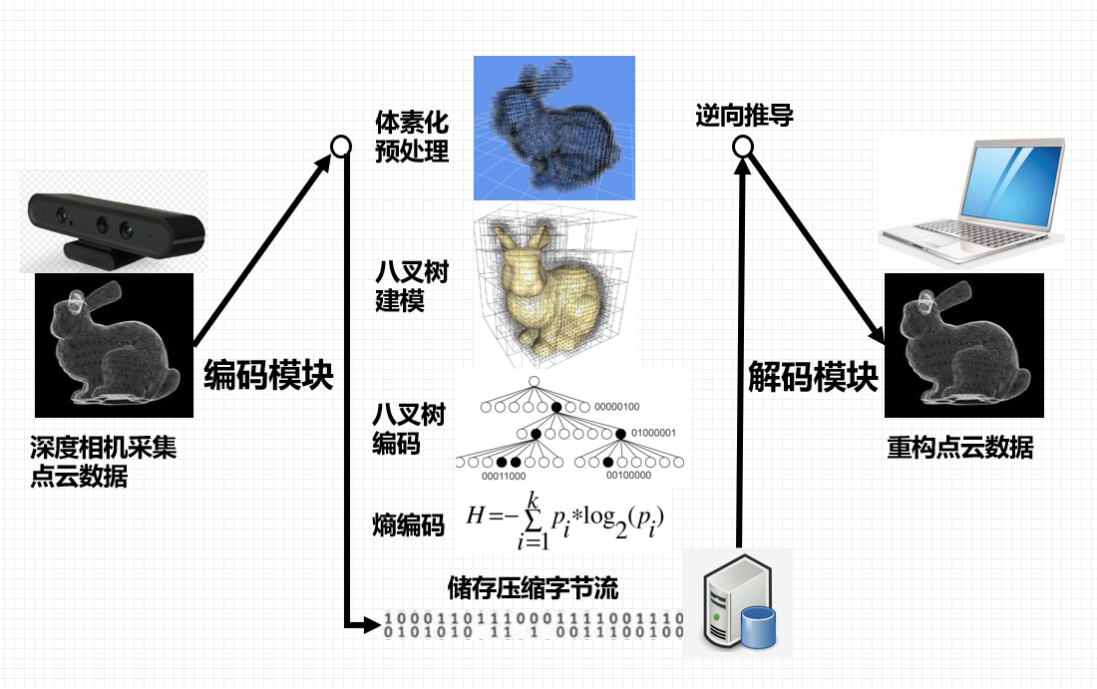


图 1. 项目流程

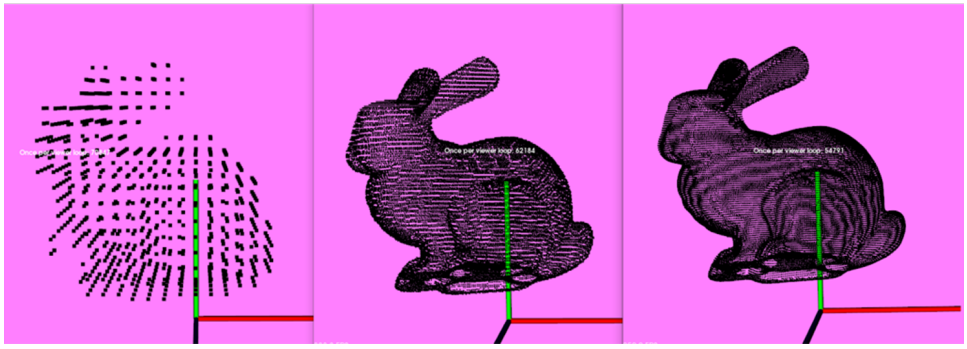


图 2. 实现效果

II. 代码实现

本项目研究历程主要分为三个阶段，分别为准备前置知识阶段，代码实现阶段，测试分析阶段。下面将分别介绍各阶段遇到的问题及解决方法。

1) 准备前置知识: 在此阶段，项目成员主要通过阅读论文来了解目前点云压缩的现状，有哪些主流方法。接着，熟悉 OpenCV 库以及 PCL 库等 3D 视觉处理库，学习八叉树，kDtree 等数据结构。最终确定了通过八叉树该数据结构来进行点云的压缩，并确定了压缩的具体方案以及还原的方案。

2) 代码实现: 在此阶段，任务是针对压缩方案给出具体实现。代码实现主要分为以下几个步骤：1. 点云预处理，2. 根据点云构建八叉树，3. 序列化八叉树，4. 熵编码压缩与解压缩，5. 还原八叉树。下面将分析本项目主要用到的数据结构与算法，其中，n 表示点云文件中点云的个数，d 表示八叉树的深度。

Algorithm 1 findKey

Input: resolution,point,origin

Output: key

$dif \leftarrow point - origin$

$idx \leftarrow dif/resolution$

$key \leftarrow floor(idx)$

$return\ key$

Algorithm1 findkey 为点云预处理的主要算法，先计算出每个点云数据的 key 值，以便插入时采用位运算。key 值有 x,y,z 三个分量的值，分别表示该点云在 x,y,z 三个方向上的体素索引。该算法的时间复杂度为常数时间 O(1)，插入的点云数量为 n 的情况下，点云预处理的总时间复杂度为 O(n)。

Algorithm 2 insertPoint

Input: mask，key, node

Output: none

while mask!=0 **do**

$idx_x, idx_y, idx_z \leftarrow mask\&key$

$idx \leftarrow idx_x + 2 * idx_y + 4 * idx$

$mask \leftarrow mask >> 1$

$insertPoint(mask, key, nodeChild_{idx})$ ▷ $nodeChild_{idx} is\ the\ idx\ child\ of\ node$

end while

Algorithm2 insertPoint 是点云插入构建八叉树的主要伪代码，通过点云预处理所得的 key 值，与八叉树的 mask 值进行位运算，可以快速确定点云在八叉树中的所属节点。其中，mask 的值为 2 的八叉树深度减一次幂。假设八叉树的深度为 d，每次插入点云，该算法会递归调用 d-1 次，故每次点云插入的时间复杂度为 o(d)，总的时间复杂度为 O(nd)。

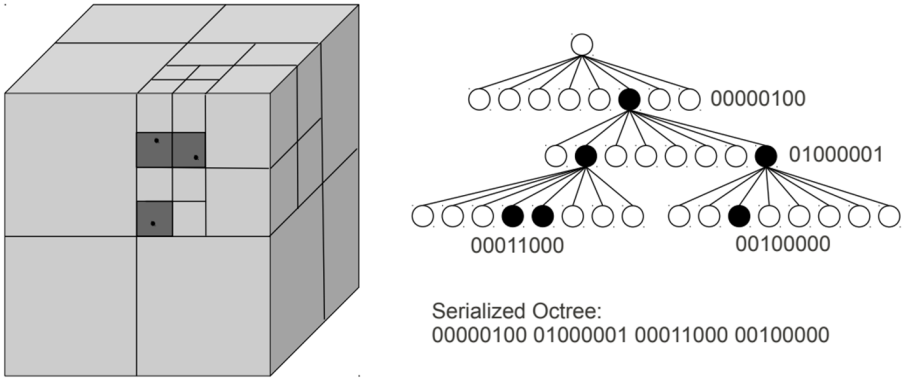


图 3. 序列化八叉树

序列化八叉树即为树的层序遍历，通过队列辅助完成。由于该算法较为常见，此处不再赘述。时间复杂度为 $O(8^d)$。最后得到一个字节数组，字节数组中的每个字节对应着一个八叉树节点的信息。一个字节的八位比特分别表示该节点的八个孩子是否存在。

Algorithm 3 restore

Input: charArray

Output: node

root ← *create a root node*
queue ← *create a queue which element is node and initialize with root*
index ← 0
while *queue is not empty* **do**
    tmp ← *pop queue*
    char ← *charArray*[*index*]
    crate child for every bits of char if it is 1
    insert child into tmp and queue
    index ← *index* + 1
end while
return root

Algorithm3 restore 即为根据字节流还原八叉树的伪代码。输入的字节流的长度即为八叉树的节点个数，在八叉树为完全八叉树时，节点数为 $\sum_{i=1}^{d} 8^{i-1}$,while 函数需执行 $\sum_{i=1}^{d} 8^{i-1}$ 次，每次执行为常数时间，故时间复杂度为 $O(\sum_{i=1}^{d} 8^{i-1})=O(8^d)$。

3) 测试分析: 在实现点云压缩的功能后，需要对其进行可行性，正确性等的测试。测试分析主要分为生成数据压缩测试与现实场景数据压缩测试。生成数据即人为地生成较为刁钻的数据进行点云压缩，而现实场景数据则为 3D 相机拍摄实景所得。针对所得还原结果，验证其正确性后，进行了压缩效果的量化分析，包括压缩率，误差等指标。

III. 成果总览

本项目完成了开题报告中点云压缩的基本任务。能够针对实际中仪器扫描所得的点云数据进行压缩，最终输出二进制 bin 文件作为压缩结果。同时，针对压缩后 bin 文件，提供了对应的解压缩方法，还原出原本的点云数据。与此同时，针对压缩的重要指标，例如压缩率，还原的准确率，各项指标与压缩精度的关系等，进行了科学严谨的评估。

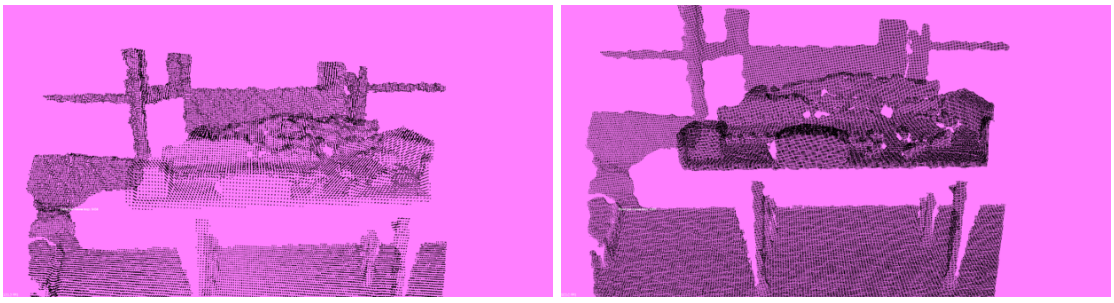
1) 可选精度的压缩： 压缩流程看作八叉树层数由浅至深看作整体不断细分的过程。根节点框定整个点云的空间，初始几层大致确定点云的大致结构，后续层中的大部分节点都在“定位”某个单独的点的坐标。具体表现为，随着点不断被划分到不同子空间，节点的空间中很快只剩下一个点，后续节点的划分实质上是在精确该点在空间的位置范围。

由于本项目采取八叉树作为描述点云位置信息的数据结构，八叉树深度越深，所描述的位置信息越准确。故可以通过改变八叉树的深度来实现可选精度的压缩。本项目采用 3D 相机获取现实生活中的点云数据，进行效果的演示。

(a) 实物场景　　(b) 原始点云 (无颜色)

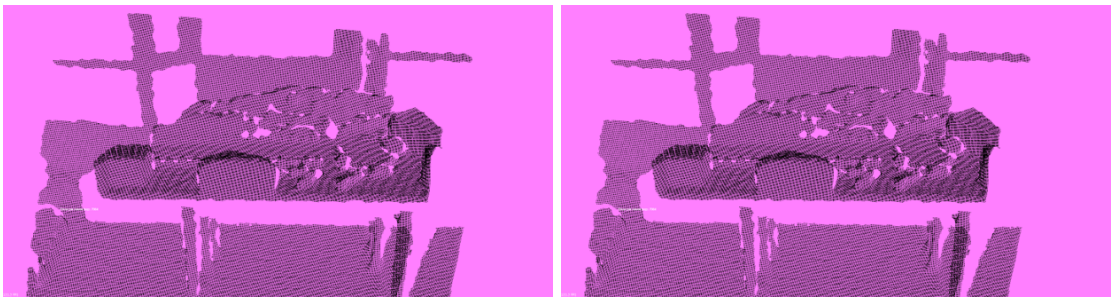
图 4. 压缩所用数据

如上图所示，分别是实物场景与相机所获取的原始点云数据。本项目采取学校的沙盒模型作为原始点云数据，该图像有足够的信息用于压缩。针对该点云数据，分别进行精度为 10，1，0.1，0.01 的压缩。此处的单位为毫米级别。将压缩后的结果进行解压缩，还原结果如下所示。



(a) 精度 10　　(b) 精度 1

图 5. 还原所得数据



(a) 精度 0.1　　(b) 精度 0.01

图 6. 还原所得数据

由此可以初步认为，压缩与解压缩的流程是有正确性保证的，还原后的数据保留了原始点云的主要信息。同时，不同压缩精度下的图像也存在着细微的差别。由于该点云采用实际场景中的点云，故不同精度的压缩展现出来的效果差异不够显著，接下来采用点云数据集“斯坦福的兔子”作为示例，该数据集较好的展示了不同精度压缩的效果差异。

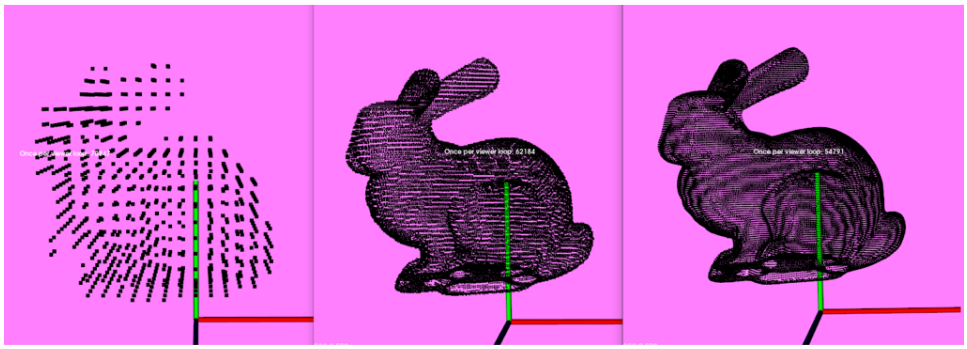
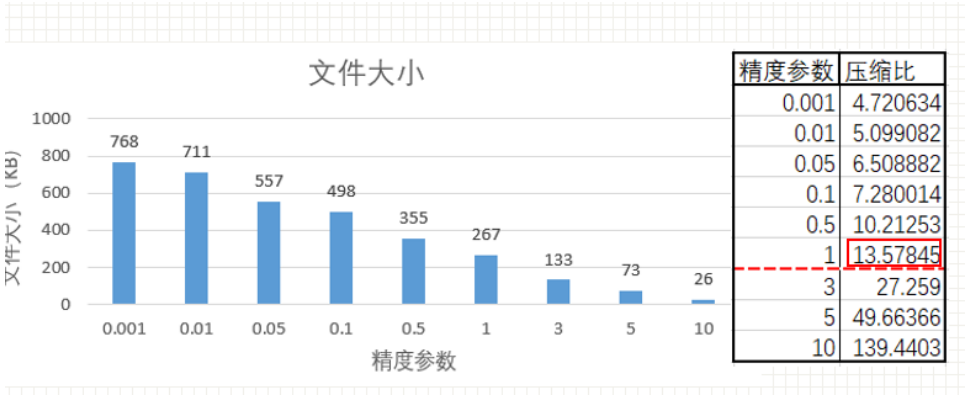


图 7. 不同压缩精度下还原效果

可以发现，不同压缩精度下结果有较大差异。故自定义精度，既可以保留数据有用的信息，忽略无关的细节部分，同时降低了数据的大小。给点云压缩提供了较大的灵活性。下面将对压缩精度与压缩率，误差之间的关系进行分析。

2) 压缩率探究: 压缩率是本项目的重要目标。本项目测试了不同参数设置下的压缩，得到如下结果



精度参数	压缩比
0.001	4.720634
0.01	5.099082
0.05	6.508882
0.1	7.280014
0.5	10.21253
1	13.57845
3	27.259
5	49.66366
10	139.4403

图 8. 精度与压缩比的关系

由于相机精度为 1mm，故 1mm 的精度设置为均衡压缩率和失真度的理想参数，在该精度下，压缩比可以达到 13.578：1。

关于压缩文件的编码效率，本项目采用 Bits per point（bpp）作为压缩率的指标，结果如下。

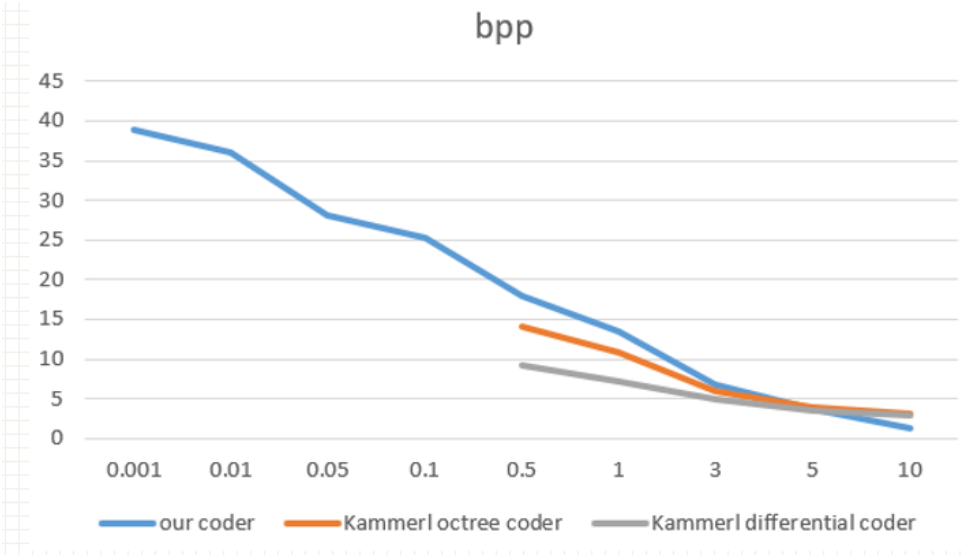


图 9. bpp 横向比较

3) 压缩还原的准确性: 这里将针对实际的点云数据“校园沙盒场景”,，在不同的压缩精度下，进行压缩前与压缩还原后的对比。量化压缩前后所产生的误差。具体做法为：将压缩前点云数据中的每一个点与压缩还原后点云数据中的每一个点一一对应起来。比较两个点之间的三个分量上的差异以及两个点的距离差。最后全部指标求平均值。下表展示了精度为 0.01，0.1，1 情况下误差的具体信息。

可以看到，在不同的精度下，偏移量的绝对值比精度小了一个数量级，说明三个分量上的偏移均在可接受的范围内，而偏移量本身则小了两个数量级，说明点云的重心基本保持不变。最后，点云间的距离偏差也比精度小了一个数量级，说明压缩前与压缩后，点云的距离发生的改变同样在可接受的范围内。下表则更加直观的展示了各个指标偏差量与精度的关系，在不同精度下，各个指标的偏差大小所占精度大小的百分比变化不大，精度越小，此百分比越大。

精度	x 偏移绝对值	y 偏移绝对值	z 偏移绝对值	x 偏移	y 偏移	z 偏移	距离偏差
0.01	0.002715	0.002521	0.005374	-0.000492	-0.000362	0.005374	0.0078
0.1	0.025029	0.025049	0.050049	0.000308	-0.000427	0.050049	0.064114
1	0.25	0.247248	0.5	0.002036	0.001521	0.5	0.639317

表 I
不同压缩精度下的误差信息

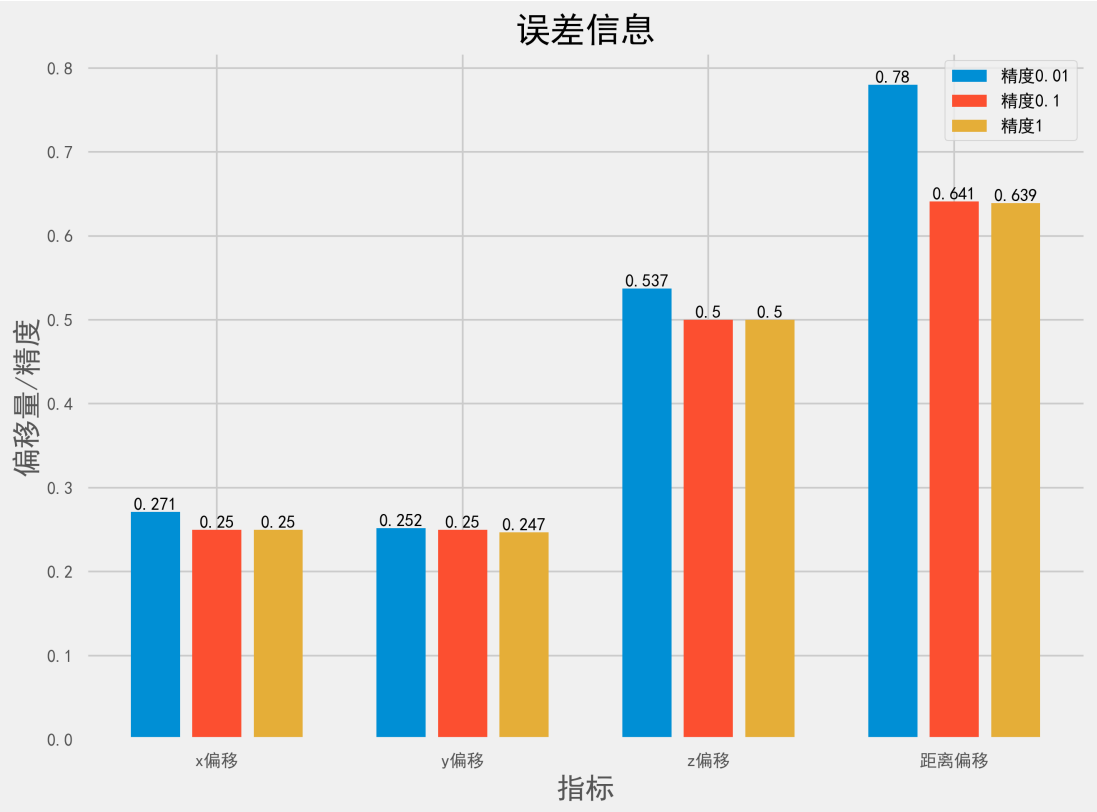


图 10. 误差信息可视化

以上为误差的初步观察，实际上本项目用于衡量失真度的指标为 PSNR，也是大部分相关论文使用的方法。

$$PSNR = 10 * log_{10}\frac{MAX^2}{MSE} \approx 20 * log_{10}\frac{MAX}{E_{dist}}$$

峰值信号大小 MAX 定义为根节点（即包围盒）的对角线长度；噪声定义为重建点云与原点云对应点欧式距离的均值

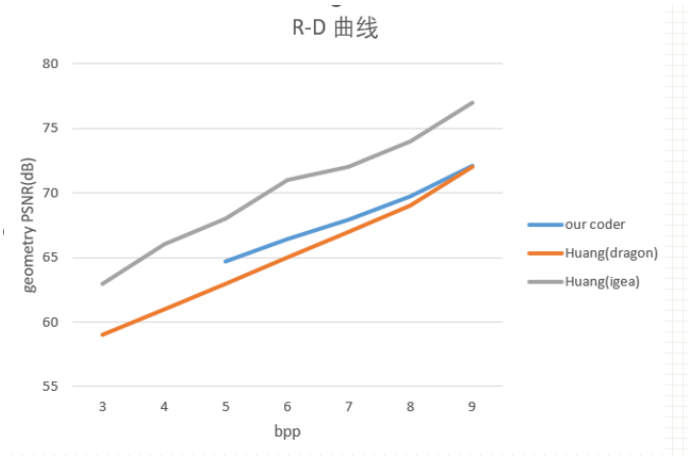


图 11. R-D 曲线与 Huang 论文中的 Octree coder 数据对比

图 12. 不同 PSNR 下失真度的展示

IV. 未来改进方向

1) 关于进一步提升压缩率的探究: 目前本项目已经实现了基于固定深度八叉树进行点云压缩和解压缩。然而，一个已知的问题是，到达一定深度后，许多节点只含有一个点。此时，在到达最深层前每深入一层都会带来 8bit 的代价，然而如果对该点直接编码相对于这个节点的坐标，每层只需要 x，y, z 各 1bit，共 3bit 的代价。八叉树依靠点之间的空间关系，当一个节点内有许多点时占位码长度并不会额外增加，而当面对孤立点时，八叉树的编码方式效率不如直接编码。

当编码过程中判定当前八叉树节点所包含的点为孤立时，将直接编码该点的剩余坐标。孤立的条件采用 6N 法则，即当前节点的父节点相邻六个同等级节点都为空。由于采用层序遍历，可以保证所有父节点的信息都是可获取的。

同时，邻居节点的信息还可用于提升甚至预测当前节点的编码方式，如大面积平面中的节点。

2) 动态点云压缩: 目前项目针对的是单帧点云的压缩。现实生活中的应用，如自动驾驶，实时点云传输等，都需要处理动态点云。而 3D 点云数据必将产生大量时间冗余。动态点云即多个点云帧组成的“视频”，现实是动态点云中大部分静态物体（如墙面）的点集处在固定的面上，然而点云本身只是大量点数据的集合，是无序的，没办法体现相邻帧的关联，也不能保证组成一个面的点集在下一帧同样的面上处于同样坐标。八叉树则能很好的在顶层保留目标的三维结构。

未来项目将尝试实现双缓冲机制，识别关键帧用正常方法压缩，而非关键帧编码前后差值。

3) 变码率传输： 由于压缩流是以层序遍历的顺序输出，而八叉树的每一层在物理意义上可以视作不同清晰率下的 3D 结构。这意味着接收方可以在获取到目标清晰度后随时停止接收，发送方也只需要在本地存放一个高清晰度的八叉树压缩文件，即可满足不同清晰度的发送。这对于减轻网络拥塞具有较大意义。

4) opencv 库兼容： 由于本项目的代码部分在 opencv5.x 下进行开发，需要解决与其他已有的八叉树功能兼容问题。

参考文献

[1] Cao C, Preda M, Zaharia T. (2019). 3D Point Cloud Compression: A Survey. 1-9. 10.1145/3329714.3338130.

[2] Graziosi D, Nakagami O, Kuma S, Zaghetto A, Suzuki T, Tabatabai A. (2020). An overview of ongoing point cloud compression standardization activities: video-based (V-PCC) and geometry-based (G-PCC). APSIPA Transactions on Signal and Information Processing. 9. 10.1017/ATSIP.2020.12.

[3] Inc. 8i Labs. (2017). 8i Voxelized Full Bodies, version 2 –A Voxelized Point Cloud Dataset. ISO/IEC JTC1/SC29/WG11 m40059 ISO/IEC JTC1/SC29/WG1 M74006, Geneva, CH

[4] MPEG 3DG, Call for proposals for point cloud compression v2, ISO/IEC JTC 1/SC 29/WG 11 N16763, 2017.

[5] MPEG 3DG, G-PCC codec description v12, ISO/IEC JTC 1/SC 29/WG 7 N0151, 2021.

[6] Schwarz S, Preda M, Baroncini V, et al. Emerging MPEG Standards for Point Cloud Compression[J]. IEEE Journal on Emerging and Selected Topics in Circuits and Systems, 2018:1-1.