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
CS326-创新实践 II

基于八叉树的 3D 点云压缩实现

作者: 叶成伟 12010524 王宇航 12012208 陈张杰 12012524
指导教师: 于仕琪
时间:2023/3/28

I. 项目背景

在人类历史上的长久岁月中，我们一直生活在一个三维世界中。但由于技术的限制，人们只能在二维平面上绘制现实世界的投影，从岩壁、帆布、纸张、到胶片。这种方式只能呈现物体的表面信息，而缺乏对其立体形态和空间结构的精准表达。然而，随着数字化时代的到来和 3D 技术的迅猛发展，人们终于能够在真正的三维语言体系下记录、处理甚至呈现 3D 物体。这一技术突破使得我们能够更加准确地模拟和理解真实世界中的物体，更加便捷地进行工程设计、医学手术规划等活动，也使得机器人和自动驾驶系统更加智能和自适应。

目前,3D 技术已经广泛运用在大量领域, 如工业设计、生医影像、科研分析、仿真等, 并随着 3D 数据探测, 采集技术的进步, 将在未来扩展更多的可能性, 如虚拟现实、全息投影、自主导航、无人驾驶、文化遗产保护等方面。

数字化 3D 数据有多种储存方式, 其中主流的方式是 3D 点云。3D 点云代表三维空间中一系列点的集合, 每个点拥有坐标属性 (x,y,z), 以及颜色, 法向量等属性。点云的集合构成了物体的表面，相比传统的多边形网格，点云更真实、更密集地呈现现实世界中的物体。激光雷达扫描（LiDAR）是目前主流的 3D 点云采集技术，通过密集地精确测距可以获得高质量的 3D 点云数据。

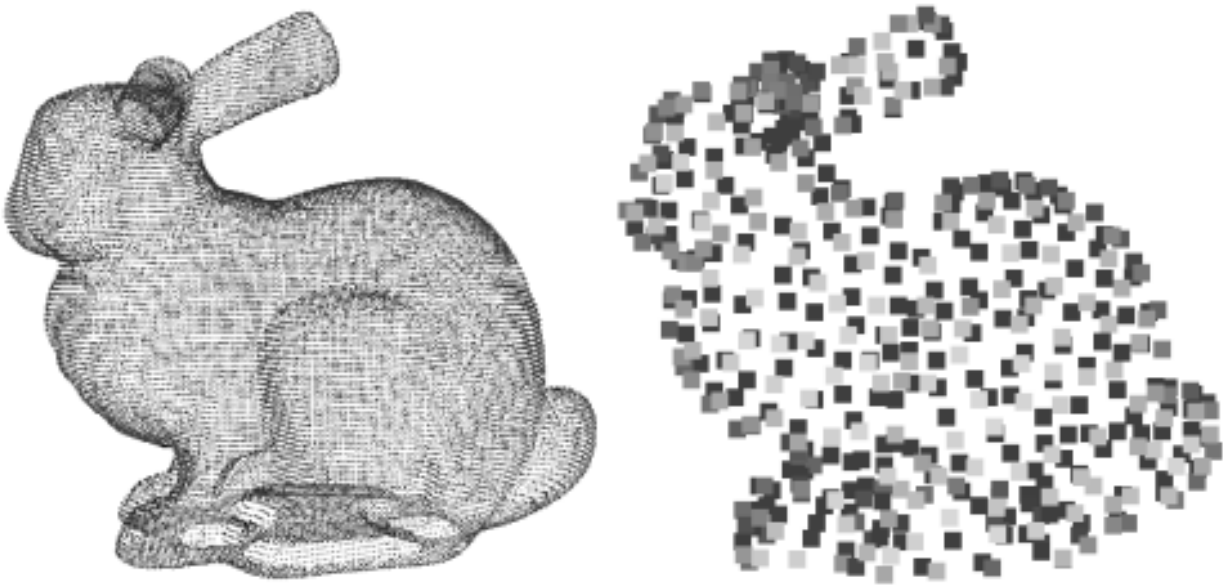


图 1: 斯坦福兔子
From Cao C, Preda M, Zaharia T. (2019). 3D Point Cloud Compression: A Survey. 1-9. 10.1145/3329714.3338130.

然而,3D 数据往往承载着巨量几何信息与特征, 想让一个动态点云达到视觉上平滑流畅的清晰度和帧率, 需要每帧约 100 万个点, 帧率达到 30fps, 占据约 500Mbps 到 1Gbps 带宽 [1] [2]。可以预见, 如果不对 3D 点云进行体积上的压缩, 数据传输将承受巨大的带宽开销。这其中一个典型的例子就是自动驾驶。自动驾驶领域中广泛使用 3D 点云来刻画车辆周围环境，但未经压缩的原始点云数据存储和传输面临着极大的困难。如果不对原始点云数据进行压缩，点云处理相关算法难以真正落地。因此,3D 点云的压缩成为了一个新兴的研究方向。2020 年,Motion Picture Experts Group(MPEG) 推出了两个 3D 点云压缩标准, 其中 Video-based Point Cloud Compression(V-PCC) 主要基于已成熟的图像与视频压缩技术,Geometry based Point Cloud Compression (G-PCC) 则是直接根据其三维几何信息进行压缩。本项目拟采用 G-PCC 标准，进行点云的压缩与解压缩。

	queen	longdress	loot	redandblack	soldier
Average number of points (in 300 frames)	1,005,000	834,000	794,000	727,000	1,076,000
Bitrates for transmitting uncompressed video (Mbytes/s)	514.47	542.22	490.61	448.21	681.96

图 2: 带宽估计

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

II. 研究内容

本项目的研究内容为: 在 OpenCV 5.x 分支中, 基于八叉树实现可选分辨率的点云数据压缩算法。本项目程序接收 3D 点云数据作为输入, 构建八叉树用于承载点云数据。接着，采用多种不同的方式，如直接编码，预测编码，熵编码对八叉树进行编码，最终输出一串比特流作为压缩结果。与此同时，该算法支持点云的颜色压缩。针对上述的压缩方案，提供对应的解压缩方案。

在上一学期，我们已实现基础的八叉树点云压缩算法，本次项目延续创新实践 I 的题目，因此，本项目的重点将放在对八叉树的编码与解码模块的优化。针对此模块，项目团队目前计划实现多种编码方式，以应对不同编码场合，通过多种编码方法的结合，达到较为理想的压缩率。接下来将详细叙述每种编码方法。

A. 熵编码

熵编码模块是本项目的核心之一，它在八叉树点云压缩中扮演了重要角色，可以显著提高压缩比，并为点云数据的传输和存储带来更高的效率。

熵编码是一种无损数据压缩技术，它根据数据（符号）出现的概率来动态地分配可变长度的编码来实现压缩。具体而言，出现频率高的符号将被分配较短的编码，出现频率低的符号将被分配较长的编码，此类编码方法接近于香农极限。熵编码最常用的算法是 Huffman 编码和 Arithmetic 编码，本项目实现了后者作为熵编码模块4。

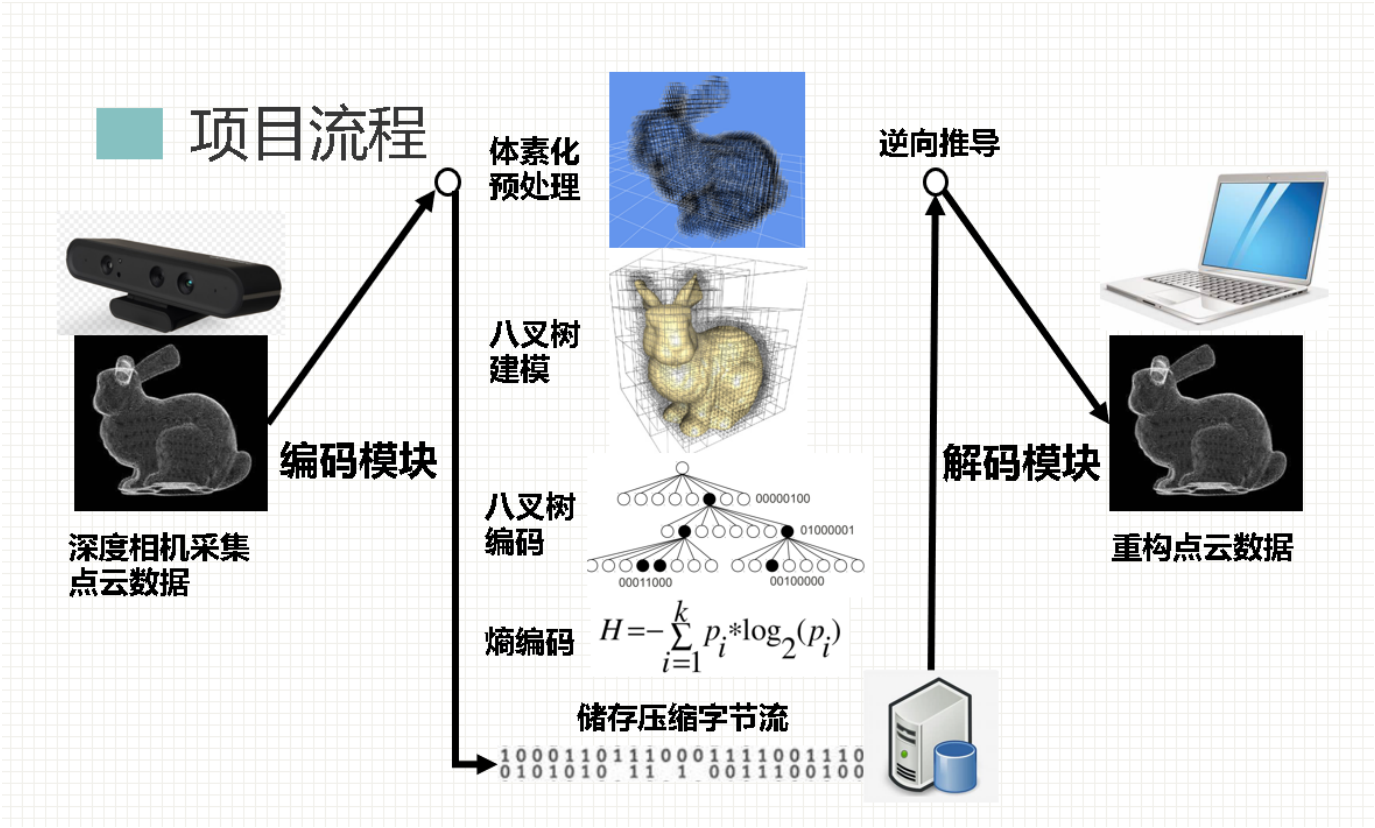


图 3: 已实现的点云压缩算法流程

```
OctreeCompress::encodeCharVectorToStream(const std::vector<unsigned char>& inputCharVector_arg,
                                         std::ostream& outputByteStream_arg) {
OctreeCompress::decodeStreamToCharVector(std::istream& inputByteStream_arg,
                                         std::vector<unsigned char>& outputCharVector_arg)
```

图 4: 熵编码模块编解码接口

B. 编码模块

八叉树5a是一种树形结构，它将 3D 空间递归地划分为 8 个子空间，并将每个子空间分配到 8 个子节点中。

占位码（occupancy code）是最简单的八叉树节点表示方法，一个占位码为一个字节（8bit），每一比特表示相应位置的子节点是否存在。在八叉树点云压缩中，熵编码模块被用于对八叉树所有节点 BFS 序列化后所得的占位码序列进行压缩。

根据统计分析5b我们发现，八叉树节点占位码的分布极不均匀。原因主要为两种：

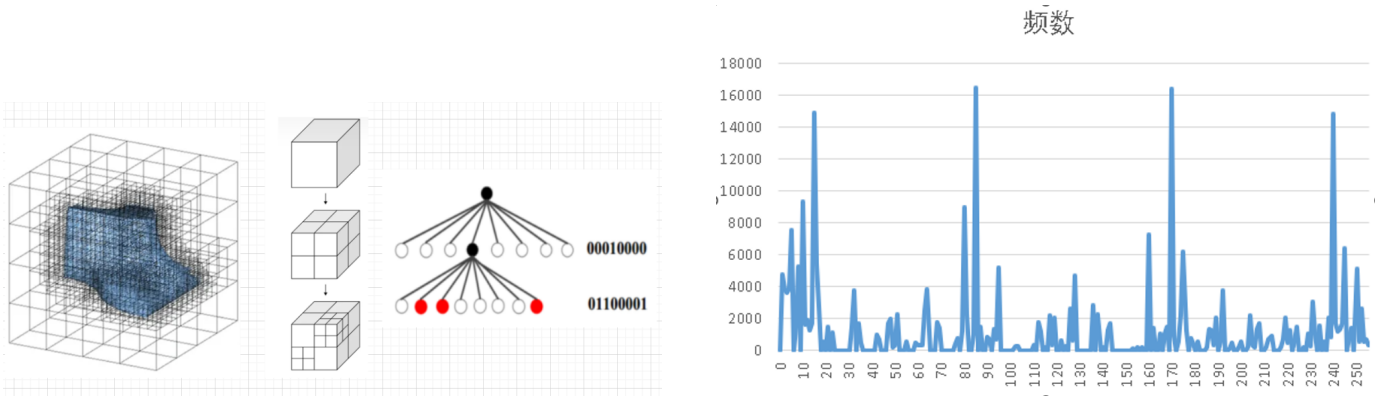
一、点云的分布

我们关注的点云数据来自深度相机或自动驾驶，反映真实世界中 3D 物体的表面。因此，点云数据中的点通常是集中在物体表面上的，相较于弥散在整个空间的随机点云，具有整体密度上的稀疏性和局部的聚类特征。

二、八叉树的深度

随着八叉树深度的增加，每个节点所对应的空间尺度不断缩小，从而对应的点云密度也会变得越来越低，直至尺度小到可以精确划分开点云的每一个点。这意味着深度较深的八叉树节点的分支将逐渐稀疏，直至变为“一叉树”，这意味着八叉树节点的占位码概率分布也会倾向于子节点少的稀疏情况。

因此，八叉树节点占位码的不均匀分布为熵编码提供了较大的压缩空间。基于八叉树的编码方式充分地利用了点云的上述空间特性，可以有效地对点云数据进行压缩。



(a) 八叉树，占位码示意图　　(b) 占位码分布

图 5: 八叉树编码

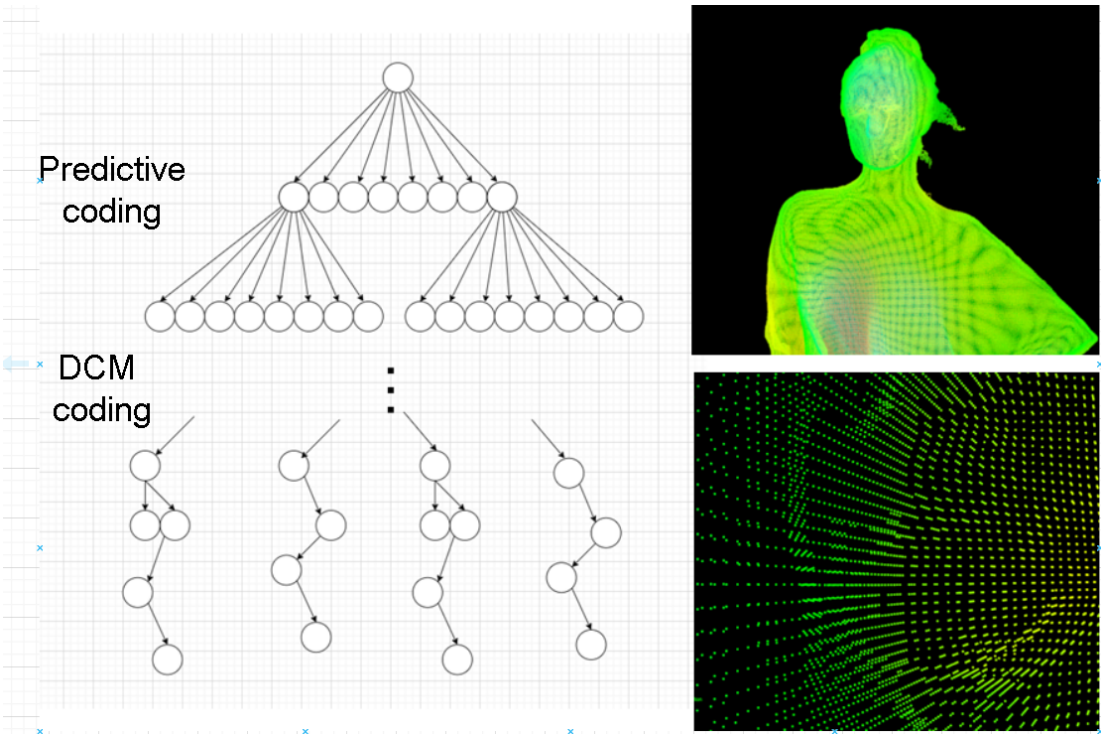


图 6: 对不同情况采用不同的编码模式

C. 直接编码

在 G-PCC 中，直接编码模式（DCM）用于压缩八叉树深度较深时的节点数据6，其中对于一些孤立的点，直接编码可以提高压缩效率。

在八叉树划分层数较深或是点云存在噪点的情况下，某些节点可能只包含一个点，这种节点被称为孤立点。对于孤立点，预测编码模式中的预测算法将无法发挥作用，因为没有与之相关的上下文（即已经编码的邻域）信息。因此，使用预测编码模式可能会浪费比较多的比特数。

在直接编码模式中，每个节点只编码它所包含的点的位置信息。如果节点中只有一个点，它将被视作孤立点，否则将采用预测编码模式。通过采用直接编码模式，对于孤立点的处理可以得到更好的压缩效果，同时还能提早结束八叉树的遍历，减少压缩时间。

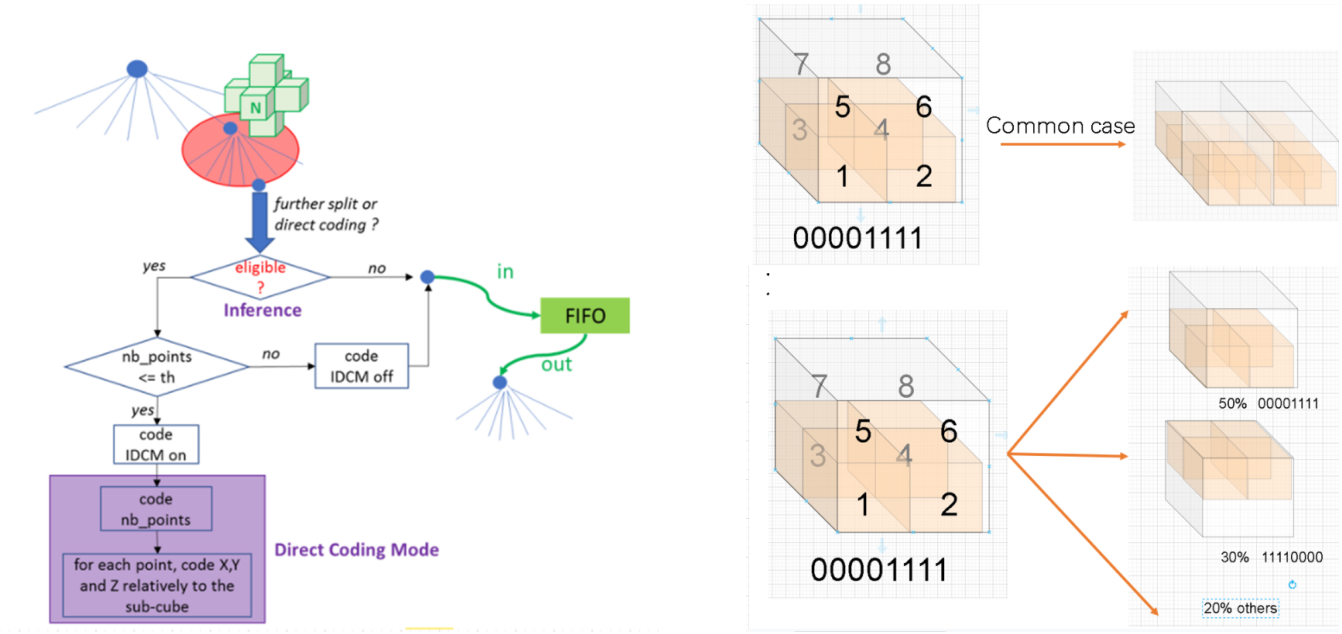
孤立点的判断有两种：父节点判断和 6N 判断。第一种判断方式即父节点仅存在当前这一个子节点。6N 判断则查看当前节点相邻的 6 个节点是否有被占用。后者更符合孤立点的定义，故本项目将实现 6N 判断。

D. 预测编码

预测编码模式是通过利用八叉树节点的空间位置关系来进行预测的。具体而言，对于一个需要编码的节点，预测编码模式会根据其在八叉树中的位置和邻近节点的信息，预测其占位码的值。这种预测编码的方式可以利用空间相邻节点之间的相关性，减小熵编码所需的位数，从而实现更高效的压缩。

在经过统计后，我们也发现：已知特定的邻域节点信息，当前节点的占位码往往是可预测的。该节点的占位码将倾向于与邻域点形成一个平面7b。预测方式可以选择基于统计和基于 MLS 平面拟合两种方式。

为了将预测得到的信息用在压缩上，预测编码模式将编码占位码预测值和真实值的残差，并送入熵编码模块进行压缩。



(a) DCM 模式流程　　(b) 根据父节点占位码预测子节点

图 7: 直接编码与预测编码

E. 颜色编码

在图像压缩领域中，Haar 小波变换是一种常用的数学技术，它通过将图像分解成一系列小波系数来捕捉图像的高频成分，可以有效减小数字图像的大小，同时保留数字图像的基本特征。

RAHT(Region-Adaptive Hierarchical Transform, 区域自适应层次变换) 是一种层次子带变换，也是 Haar 小波变换在 3D 点云上的实现。它的思路是根据八叉树中低层数节点相关的颜色来预测下一层节点的颜色。RAHT 算法对八叉树进行反向构建，每一步都将同一级别的两个体素组成为更大的体素，直到到达根节点，每一次都会产生 RAHT 系数。之后，对 RAHT 变换系数进行量化，然后使用算术编码器 (AC) 对系数进行熵编码达到压缩效果。

以上均为针对单帧点云实现的压缩算法，针对动态点云，项目团队拟采用双缓冲八叉树，充分利用点云帧间的相关性，完成点云的压缩。对于每一帧点云，仅仅需要编码其与上一帧点云的差异，这种差异，利用两帧点云对应的八叉树可以清晰描述出来。值得注意的是，绝大部分的帧间差距对于点云帧本身的大小几乎可以忽略。因此，针对动态点云，选取某一确定的时间间隔，将处于每次间隔节点的单帧点云作为此次间隔的初始帧，在此间隔内，往后的每一帧均编码与前一帧的差值即可。通过此方法，可以极大地提升动态点云压缩的压缩率。

F. 帧间编码

基于八叉树的动态点云压缩的帧间编码，考虑到点云数据在时间上的连续性，利用了时间冗余性，即压缩相邻帧的点云数据时，当前帧的点云数据可以借鉴前一帧的点云数据进行压缩。

在基于八叉树的动态点云压缩中，帧间编码过程可以分为以下几个步骤：

1. 先对前一帧的点云数据进行八叉树编码，并记录下每个节点的占位码。
2. 对当前帧的点云数据进行八叉树编码，并记录下每个节点的占位码。
3. 利用八叉树的结构比较前一帧和当前帧的八叉树节点，找出发生变化的节点。
4. 对发生变化的节点进行编码，包括新节点的属性、删除节点的占位码以及属性发生变化的节点的属性差值。
5. 将编码结果与前一帧未发生变化的节点的占位码拼接起来，形成当前帧的压缩结果。

在这个过程中，由于利用了时间冗余性，对于前一帧和当前帧相似的部分，可以直接拷贝前一帧的占位码，从而节省了编码所需的空间。

III. 研究意义

本项目拟实现的点云压缩具有以下特点：由知名视觉开源库 OpenCV 集成，支持可选精度的压缩，提供多种压缩的方案，支持颜色压缩，动态点云压缩。其意义具体体现在：

- 由于本算法为 OpenCV 所集成，故丰富了 OpenCV5.x 对于点云数据的处理功能，拓宽了其引用面。同时，由于 OpenCV 本身庞大的用户基数，此算法能够得到快速的，全面的反馈，反过来推动此算法的进步与完善。
- 由于支持可选精度的压缩，用户可以针对不同的应用场景选定合适的精度，忽略点云数据中所不关心的细节部分，而保留所关注的信息，使点云压缩算法具有更好的灵活性与实用性
- 计算机视觉的最终体现是三维视觉，而三维视觉的表达方式则是点云。考虑到存储空间与传输带宽的限制，本项目期望通过减轻点云数据在存储与传输过程中所面临的压力，为点云相关的算法在现实场景中的使用提供先决条件，推动相关算法如点云语义分割，目标检测等算法的落地。

IV. 项目安排

本项目安排分为三个时间段，每个时间段存在相应的交付成果。

- 第 10 周，完成直接编码模块与预测编码模块。
- 第 12 周，中期答辩，完成颜色编码模块，具有可以展示三种编码方式的 demo。
- 第 16 周，项目结题，实现双缓冲八叉树的结构，对程序进行测试与分析，并得出最终成果。

V. 预期成果

- 实现基于八叉树可选分辨率的点云数据压缩算法。针对八叉树的编码，可提供多种方式如直接编码，预测编码，熵编码。与此同时，针对颜色点云，支持点云的颜色压缩，针对动态点云，能够采用双缓冲八叉树描述。针对上述的压缩方案，提供对应的解压缩方案。

- 本项目能够遵循 OpenCV 的代码规范，提供详细的文档描述，最终被集成。
- 针对点云压缩的压缩率，速度，与重建质量三个指标进行了算法的评价与改进。采用 Bits per point (bpp) 作为压缩率的评估准则，Peak Signal-to-Noise Ratio（PSNR）作为还原重建质量的评估准则。
- 分别在不同的数据集上完成了算法的测试工作。数据集：http://graphics.stanford.edu/data/3Dscanrep/。

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.