

(12) 按照专利合作条约所公布的国际申请

(19) 世界知识产权组织
国际局

(43) 国际公布日
2025 年 1 月 9 日 (09.01.2025)



(10) 国际公布号
WO 2025/007355 A1

(51) 国际专利分类号:

H04N 19/597 (2014.01)

(21) 国际申请号:

PCT/CN2023/106178

(22) 国际申请日:

2023 年 7 月 6 日 (06.07.2023)

(25) 申请语言:

中文

(26) 公布语言:

中文

(71) 申请人: **OPPO 广东移动通信有限公司 (GUANGDONG OPPO MOBILE TELECOMMUNICATIONS CORP., LTD.)** [CN/CN]; 中国广东省东莞市长安镇乌沙海滨路 18 号, Guangdong 523860 (CN)。

(72) 发明人: 孙泽星(SUN, Zexing); 中国广东省东莞市长安镇乌沙海滨路 18 号, Guangdong 523860 (CN)。

(74) 代理人: 北京派特恩知识产权代理有限公司(**CHINA PAT INTELLECTUAL PROPERTY OFFICE**); 中国北京市海淀区苏州街 3 号大恒科技大厦南座五层 503, Beijing 100080 (CN)。

(81) 指定国(除另有指明, 要求每一种可提供的国家保护): AE, AG, AL, AM, AO, AT, AU, AZ, BA, BB, BG, BH, BN, BR, BW, BY, BZ, CA, CH, CL, CN, CO, CR, CU,

CV, CZ, DE, DJ, DK, DM, DO, DZ, EC, EE, EG, ES, FI, GB, GD, GE, GH, GM, GT, HN, HR, HU, ID, IL, IN, IQ, IR, IS, IT, JM, JO, JP, KE, KG, KH, KN, KP, KR, KW, KZ, LA, LC, LK, LR, LS, LU, LY, MA, MD, MG, MK, MN, MU, MW, MX, MY, MZ, NA, NG, NI, NO, NZ, OM, PA, PE, PG, PH, PL, PT, QA, RO, RS, RU, RW, SA, SC, SD, SE, SG, SK, SL, ST, SV, SY, TH, TJ, TM, TN, TR, TT, TZ, UA, UG, US, UZ, VC, VN, WS, ZA, ZM, ZW。

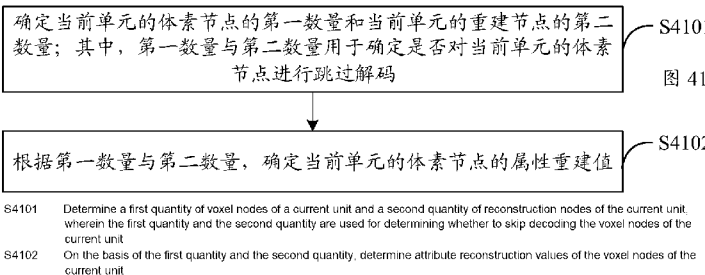
(84) 指定国(除另有指明, 要求每一种可提供的地区保护): ARIPO (BW, CV, GH, GM, KE, LR, LS, MW, MZ, NA, RW, SC, SD, SL, ST, SZ, TZ, UG, ZM, ZW), 欧亚 (AM, AZ, BY, KG, KZ, RU, TJ, TM), 欧洲 (AL, AT, BE, BG, CH, CY, CZ, DE, DK, EE, ES, FI, FR, GB, GR, HR, HU, IE, IS, IT, LT, LU, LV, MC, ME, MK, MT, NL, NO, PL, PT, RO, RS, SE, SI, SK, SM, TR), OAPI (BF, BJ, CF, CG, CI, CM, GA, GN, GQ, GW, KM, ML, MR, NE, SN, TD, TG)。

本国际公布:

— 包括国际检索报告(条约第 21 条(3))。

(54) **Title:** ENCODING METHOD, DECODING METHOD, CODE STREAM, ENCODER, DECODER, AND STORAGE MEDIUM

(54) 发明名称: 编解码方法、码流、编码器、解码器以及存储介质



(57) **Abstract:** Disclosed in embodiments of the present application are an encoding method, a decoding method, a code stream, an encoder, a decoder, and a storage medium. The method comprises: determining a first quantity of voxel nodes of a current unit and a second quantity of reconstruction nodes of the current unit, wherein the first quantity and the second quantity are used for determining whether to skip decoding the voxel nodes of the current unit; and on the basis of the first quantity and the second quantity, determining attribute reconstruction values of the voxel nodes of the current unit. In this way, skip-decoding of the voxel nodes of the current unit can reduce the time complexity of attribute encoding and decoding and improve the attribute encoding and decoding efficiency of a point cloud, thereby improving the encoding and decoding performance of the point cloud.

(57) 摘要: 本申请实施例公开了一种编解码方法、码流、编码器、解码器以及存储介质, 该方法包括: 确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量; 其中, 第一数量与第二数量用于确定是否对当前单元的体素节点进行跳过解码; 根据第一数量与第二数量, 确定当前单元的体素节点的属性重建值。这样, 在当前单元的体素节点跳过解码时, 可以降低属性编解码的时间复杂度, 提高点云的属性编解码效率, 进而提升点云的编解码性能。

WO 2025/007355 A1

编解码方法、码流、编码器、解码器以及存储介质

技术领域

本申请实施例涉及点云编解码技术领域，尤其涉及一种编解码方法、码流、编码器、解码器以及存储介质。

5 背景技术

在基于几何的点云压缩（Geometry-based Point Cloud Compression, G-PCC）编解码框架中，点云的几何信息和属性信息是分开进行编码的。其中，G-PCC 的属性编码可以包括：预测变换（Predicting Transform, PT）、提升变换（Lifting Transform, LT）以及区域自适应分层变换（Region Adaptive Hierarchical Transform, RAHT）。

10 在进行 RAHT 变换中，需要对每一层的节点依次进行变换、预测以及编解码等过程，这样会导致 RAHT 属性变换编解码的复杂度增大，进而无法有效地去除属性的冗余性，导致属性编码效率较低。

发明内容

本申请实施例提供一种编解码方法、码流、编码器、解码器以及存储介质，能够降低属性编解码的时间复杂度，提高点云的属性编解码效率，进而提升点云的编解码性能。

15 本申请实施例的技术方案可以如下实现：

第一方面，本申请实施例提供了一种解码方法，应用于解码器，该方法包括：

确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量；其中，第一数量与第二数量用于确定是否对当前单元的体素节点进行跳过解码；

根据第一数量与第二数量，确定当前单元的体素节点的属性重建值。

20 第二方面，本申请实施例提供了一种编码方法，应用于编码器，该方法包括：

确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量；

根据第一数量与第二数量，确定当前单元的体素节点的属性信息是否跳过编码。

第三方面，本申请实施例提供了一种码流，该码流是根据待编码信息进行比特编码生成的；其中，待编码信息包括下述至少一项：

25 预测模式标识信息、当前单元的重复节点的属性信息和当前层的节点对应的量化后的系数残差；其中，所述预测模式标识信息用于指示所述当前单元是否启动跳过编码模式。

第四方面，本申请实施例提供了一种编码器，该编码器包括第一确定单元和编码单元，其中：

第一确定单元，配置为确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量；

编码单元，配置为根据第一数量与第二数量，确定当前单元的体素节点的属性信息是否跳过编码。

30 第五方面，本申请实施例提供了一种编码器，该编码器包括第一存储器和第一处理器，其中：

第一存储器，用于存储能够在第一处理器上运行的计算机程序；

第一处理器，用于在运行计算机程序时，执行如第二方面所述的方法。

第六方面，本申请实施例提供了一种解码器，该解码器包括第二确定单元和第二重建单元，其中：

第二确定单元，配置为确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量；其

35 中，第一数量与第二数量用于确定是否对当前单元的体素节点进行跳过解码；

第二重建单元，配置为根据第一数量与第二数量，确定当前单元的体素节点的属性重建值。

第七方面，本申请实施例提供了一种解码器，该解码器包括第二存储器和第二处理器，其中：

第二存储器，用于存储能够在第二处理器上运行的计算机程序；

第二处理器，用于在运行计算机程序时，执行如第一方面所述的方法。

40 第八方面，本申请实施例提供了一种计算机可读存储介质，该计算机可读存储介质存储有计算机程序，所述计算机程序被执行时实现如第一方面所述的方法、或者实现如第二方面所述的方法。

本申请实施例提供了一种编解码方法、码流、编码器、解码器以及存储介质，无论是编码端还是解码端，首先确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量，然后根据第一数量与第二数量，确定当前单元的体素节点的属性重建值。其中，第一数量与第二数量用于确定当前单元的

体素节点是否跳过编解码。也就是说,在对每个体素节点进行属性重建时,优化了每个体素节点是否进行属性编解码的判断条件,具体为如果当前单元不存在重复节点,即第一数量与第二数量相同,此时不需要编解码当前单元的体素节点,从而在保证点云属性的编解码效率基础上,可以降低点云属性编解码的时间复杂度,并且还可以节省码率,进而提升了点云的编解码性能。

5 附图说明

- 图 1A 为一种三维点云图像示意图;
图 1B 为一种三维点云图像的局部放大图;
图 2A 为一种点云图像的六个观看角度示意图;
图 2B 为一种点云图像对应的数据存储格式示意图;
10 图 3 为一种点云编解码的网络架构示意图;
图 4A 为一种 G-PCC 编码器的组成框架示意图;
图 4B 为一种 G-PCC 解码器的组成框架示意图;
图 5A 为一种 Z 轴方向的低平面位置示意图;
图 5B 为一种 Z 轴方向的高平面位置示意图;
15 图 6 为一种节点编码顺序示意图;
图 7A 为一种平面标识信息示意图;
图 7B 为另一种平面标识信息示意图;
图 8 为一种当前节点的兄弟姐妹节点示意图;
图 9 为一种激光雷达与节点的相交示意图;
20 图 10 为一种处于相同划分深度以及相同坐标的邻域节点示意图;
图 11 为一种当前节点位于父节点的低平面位置示意图;
图 12 为一种当前节点位于父节点的高平面位置示意图;
图 13 为一种激光雷达点云平面位置信息的预测编码示意图;
图 14 为一种 IDCM 编码示意图;
25 图 15 为一种旋转激光雷达获取点云的坐标转换示意图;
图 16 为一种 X 轴或 Y 轴方向的预测编码示意图;
图 17A 为一种通过水平方位角来进行预测 Y 平面的角度示意图;
图 17B 为一种通过水平方位角来进行预测 X 平面的角度示意图;
图 18 为另一种 X 轴或 Y 轴方向的预测编码示意图;
30 图 19A 为一种子块包括的三个交点示意图;
图 19B 为一种利用三个交点拟合的三角面片集示意图;
图 19C 为一种三角面片集的上采样示意图;
图 20 为一种基于距离的 LOD 构造过程的示意图;
图 21 为一种 LOD 生成过程的可视化结果示意图;
35 图 22 为一种属性预测的编码流程示意图;
图 23 为一种金字塔结构的组成示意图;
图 24 为另一种金字塔结构的组成示意图;
图 25 为一种层间最近邻查找的 LOD 结构示意图;
图 26 为一种基于空间关系进行最近邻查找结构示意图;
40 图 27A 为一种共面的空间关系示意图;
图 27B 为一种共面和共线的空间关系示意图;
图 27C 为一种共面、共线和共点的空间关系示意图;
图 28 为一种基于快速查找的层间预测示意图;
图 29 为一种属性层内最近邻查找的 LOD 结构示意图;
45 图 30 为一种基于快速查找的层内预测示意图;
图 31 为一种基于块进行邻域查找结构示意图;
图 32 为一种提升变换的编码流程示意图;
图 33 为一种 RAHT 变换结构示意图;
图 34 为一种 RAHT 沿 x、y、z 三方向的变换过程示意图;
50 图 35A 为一种 RAHT 正变换的过程示意图;

图 35B 为一种 RAHT 逆变换的过程示意图；
图 36 为一种属性编码块的结构示意图；
图 37 为一种 RAHT 属性预测变换编码的整体流程示意图；
图 38 为一种当前块的邻域预测关系示意图；
5 图 39 为一种属性变换系数的计算过程示意图；
图 40 为一种 RAHT 属性帧间预测编码的结构示意图；
图 41 为本申请实施例提供的一种解码方法的流程示意图一；
图 42 为本申请实施例提供的一种 RAHT 属性编码层的结构示意图；
图 43 为本申请实施例提供的一种解码方法的流程示意图二；
10 图 44 为本申请实施例提供的一种编码方法的流程示意图一；
图 45 为本申请实施例提供的一种编码方法的流程示意图二；
图 46 为本申请实施例提供的一种编码方法的流程示意图三；
图 47 为本申请实施例提供的一种编码器的组成结构示意图；
图 48 为本申请实施例提供的一种编码器的具体硬件结构示意图；
15 图 49 为本申请实施例提供的一种解码器的组成结构示意图；
图 50 为本申请实施例提供的一种解码器的具体硬件结构示意图；
图 51 为本申请实施例提供的一种编解码系统的组成结构示意图。

具体实施方式

为了能够更加详尽地了解本申请实施例的特点与技术内容,下面结合附图对本申请实施例的实现进行详细阐述,所附附图仅供参考说明之用,并非用来限定本申请实施例。

除非另有定义,本文所使用的所有的技术和科学术语与属于本申请的技术领域的技术人员通常理解的含义相同。本文中所使用的术语只是为了描述本申请实施例的目的,不是旨在限制本申请。

在以下的描述中,涉及到“一些实施例”,其描述了所有可能实施例的子集,但是可以理解,“一些实施例”可以是所有可能实施例的相同子集或不同子集,并且可以在不冲突的情况下相互结合。

25 还需要指出,本申请实施例所涉及的术语“第一\第二\第三”仅是用于区别类似的对象,不代表针对对象的特定排序,可以理解地,“第一\第二\第三”在允许的情况下可以互换特定的顺序或先后次序,以使这里描述的本申请实施例能够以除了在这里图示或描述的以外的顺序实施。

点云(Point Cloud)是物体表面的三维表现形式,通过光电雷达、激光雷达、激光扫描仪、多视角相机等采集设备,可以采集得到物体表面的点云(数据)。

30 点云是空间中一组无规则分布的、表达三维物体或场景的空间结构及表面属性的离散点集,图 1A 展示了三维点云图像和图 1B 展示了三维点云图像的局部放大图,可以看到点云表面是由分布稠密的点所组成的。

2 维图像在每一个像素点均有信息表达,分布规则,因此不需要额外记录其位置信息;然而点云中的点在三维空间中的分布具有随机性和不规则性,因此需要记录每一个点在空间中的位置,才能完整地表达一幅点云。与二维图像类似,采集过程中每一个位置均有对应的属性信息,通常为 RGB 颜色值,颜色值反映物体的色彩;对于点云来说,每一个点所对应的属性信息除了颜色信息以外,还有比较常见的是反射率(reflectance)值,反射率值反映物体的表面材质。因此,点云数据通常包括点的位置信息和点的属性信息。其中,点的位置信息也可称为点的几何信息。例如,点的几何信息可以是点的三维坐标信息(x, y, z)。点的属性信息可以包括颜色信息和/或反射率等等。例如,反射率可以是一维反射率信息(r);颜色信息可以是任意一种色彩空间上的信息,或者颜色信息也可以是三维颜色信息,如 RGB 信息。在这里,R 表示红色(Red, R),G 表示绿色(Green, G),B 表示蓝色(Blue, B)。再如,颜色信息可以是亮度色度(YCbCr, YUV)信息。其中,Y 表示明亮度(Luma),Cb(U)表示蓝色色差,Cr(V)表示红色色差。

45 根据激光测量原理得到的点云,点云中的点可以包括点的三维坐标信息和点的反射率值。再如,根据摄影测量原理得到的点云,点云中的点可以包括点的三维坐标信息和点的三维颜色信息。再如,结合激光测量和摄影测量原理得到点云,点云中的点可以包括点的三维坐标信息、点的反射率值和点的三维颜色信息。

如图 2A 和图 2B 所示为一幅点云图像及其对应的数据存储格式。其中,图 2A 提供了点云图像的六个观看角度,图 2B 由文件头信息部分和数据部分组成,头信息包含了数据格式、数据表示类型、点云总点数、以及点云所表示的内容。例如,点云为“.ply”格式,由 ASCII 码表示,总点数为 207242,每

个点具有三维坐标信息 (x, y, z) 和三维颜色信息 (r, g, b)。

点云可以按获取的途径分为：

静态点云：即物体是静止的，获取点云的设备也是静止的；

动态点云：物体是运动的，但获取点云的设备是静止的；

5 动态获取点云：获取点云的设备是运动的。

例如，按点云的用途分为两大类：

类别一：机器感知点云，其可以用于自主导航系统、实时巡检系统、地理信息系统、视觉分拣机器人、抢险救灾机器人等场景；

10 类别二：人眼感知点云，其可以用于数字文化遗产、自由视点广播、三维沉浸通信、三维沉浸交互等点云应用场景。

点云可以灵活方便地表达三维物体或场景的空间结构及表面属性，并且由于点云通过直接对真实物体采样获得，在保证精度的前提下能提供极强的真实感，因而应用广泛，其范围包括虚拟现实游戏、计算机辅助设计、地理信息系统、自动导航系统、数字文化遗产、自由视点广播、三维沉浸远程呈现、生物组织器官三维重建等。

15 点云的采集主要有以下途径：计算机生成、3D 激光扫描、3D 摄影测量等。计算机可以生成虚拟三维物体及场景的点云；3D 激光扫描可以获得静态现实世界三维物体或场景的点云，每秒可以获得百万级点云；3D 摄影测量可以获得动态现实世界三维物体或场景的点云，每秒可以获得千万级点云。这些技术降低了点云数据获取成本和时间周期，提高了数据的精度。点云数据获取方式的变革，使大量点云数据的获取成为可能，伴随着应用需求的增长，海量 3D 点云数据的处理遭遇存储空间和传输带宽限制的瓶颈。

20 示例性地，以帧率为 30 帧每秒 (fps) 的点云视频为例，每帧点云的点数为 70 万，每个点具有坐标信息 xyz (float) 和颜色信息 RGB (uchar)，则 10s 点云视频的数据量大约为 $0.7\text{million} \times (4\text{Byte} \times 3 + 1\text{Byte} \times 3) \times 30\text{fps} \times 10\text{s} = 3.15\text{GB}$ ，其中，1Byte 为 10bit；而 YUV 采样格式为 4:2:0，帧率为 24fps 的 1280×720 二维视频，其 10s 的数据量约为 $1280 \times 720 \times 12\text{bit} \times 24\text{fps} \times 10\text{s} \approx 0.33\text{GB}$ ，10s 的两视角三维视频的数据量约为 $0.33 \times 2 = 0.66\text{GB}$ 。由此可见，点云视频的数据量远超过相同时长的二维视频和三维视频的数据量。因此，为更好地实现数据管理，节省服务器存储空间，降低服务器与客户端之间的传输流量及传输时间，点云压缩成为促进点云产业发展的关键问题。

也就是说，由于点云是海量点的集合，存储点云不仅会消耗大量的内存，而且不利于传输，也没有这么大的带宽可以支持将点云不经过压缩直接在网络层进行传输，因此，需要对点云进行压缩。

30 目前，可对点云进行压缩的点云编码框架可以是运动图像专家组 (Moving Picture Experts Group, MPEG) 提供的基于几何的点云压缩 (Geometry-based Point Cloud Compression, G-PCC) 编解码框架或基于视频的点云压缩 (Video-based Point Cloud Compression, V-PCC) 编解码框架，也可以是 AVS 提供的 AVS-PCC 编解码框架。G-PCC 编解码框架可用于针对第一类静态点云和第三类动态获取点云进行压缩，其可以是基于点云压缩测试平台 (Test Model Compression 13, TMC13)，V-PCC 编解码框架可用于针对第二类动态点云进行压缩，其可以是基于点云压缩测试平台 (Test Model Compression 2, TMC2)。故 G-PCC 编解码框架也称为点云编解码器 TMC13，V-PCC 编解码框架也称为点云编解码器 TMC2。

35 本申请实施例提供了一种包含解码方法和编码方法的点云编解码系统的网络架构，图 3 为本申请实施例提供的一种点云编解码的网络架构示意图。如图 3 所示，该网络架构包括一个或多个电子设备 13 至 1N 和通信网络 01，其中，电子设备 13 至 1N 可以通过通信网络 01 进行视频交互。电子设备在实施的过程中可以为各种类型的具有点云编解码功能的设备，例如，所述电子设备可以包括手机、平板电脑、个人计算机、个人数字助理、导航仪、数字电话、视频电话、电视机、传感设备、服务器等，本申请实施例不作限制。其中，本申请实施例中的解码器或编码器就可以为上述电子设备。

其中，本申请实施例中的电子设备具有点云编解码功能，一般包括点云编码器 (即编码器) 和点云解码器 (即解码器)。

45 下面以 G-PCC 编解码框架为例进行相关技术的说明。

可以理解，在点云 G-PCC 编解码框架中，针对待编码的点云数据，首先通过片 (slice) 划分，将点云数据划分为多个 slice。在每一个 slice 中，点云的几何信息和每个点所对应的属性信息是分开进行编码的。

50 图 4A 示出了一种 G-PCC 编码器的组成框架示意图。如图 4A 所示，在几何编码过程中，对几何信息进行坐标转换，使点云全都包含在一个包围盒 (Bounding Box) 中，然后再进行量化，这一步量化主要起到缩放的作用，由于量化取整，使得一部分点云的几何信息相同，于是再基于参数来决定是否移除重复点，量化和移除重复点这一过程又被称为体素化过程。接着对包围盒进行八叉树划分或者预测树构

建。在该过程中,针对划分的叶子结点中的点进行算术编码,生成二进制的几何比特流;或者,针对划分产生的交点(Vertex)进行算术编码(基于交点进行表面拟合),生成二进制的几何比特流。在属性编码过程中,几何编码完成,对几何信息进行重建后,需要先进行颜色转换,将颜色信息(即属性信息)从RGB颜色空间转换到YUV颜色空间。然后,利用重建的几何信息对点云重新着色,使得未编码的属性信息与重建的几何信息对应起来。属性编码主要针对颜色信息进行,在颜色信息编码过程中,主要有两种变换方法,一是依赖于细节层次(Level of Detail, LOD)划分的基于距离的提升变换,二是直接进行区域自适应分层变换(Region Adaptive Hierarchical Transform, RAHT),这两种方法都会将颜色信息从空间域转换到频域,通过变换得到高频系数和低频系数,最后对系数进行量化,再对量化系数进行算术编码,可以生成二进制的属性比特流。

图4B示出了一种G-PCC解码器的组成框架示意图。如图4B所示,针对所获取的二进制比特流,首先对二进制比特流中的几何比特流和属性比特流分别进行独立解码。在对几何比特流的解码时,通过算术解码-重构八叉树/重构预测树-重建几何-坐标逆转换,得到点云的几何信息;在对属性比特流的解码时,通过算术解码-反量化-LOD划分/RAHT-颜色逆转换,得到点云的属性信息,基于几何信息和属性信息还原待编码的点云数据(即输出点云)。

需要说明的是,在如图4A或图4B所示,目前G-PCC的几何编解码可以分为基于八叉树的几何编解码(用虚线框标识)和基于预测树的几何编解码(用点划线框标识)。

对于基于八叉树的几何编码(Octree geometry encoding, OctGeomEnc)而言,基于八叉树的几何编码包括:首先对几何信息进行坐标转换,使点云全都包含在一个包围盒中。然后再进行量化,这一步量化主要起到缩放的作用,由于量化取整,使得一部分点的几何信息相同,根据参数来决定是否移除重复点,量化和移除重复点这一过程又被称为体素化过程。接下来,按照广度优先遍历的顺序不断对包围盒进行树划分(例如八叉树、四叉树、二叉树等),对每个节点的占位码进行编码。在相关技术中,某公司提出了一种隐式几何的划分方式,首先计算点云的包围盒(2^{d_x} , 2^{d_y} , 2^{d_z}),假设 $d_x > d_y > d_z$,该包围盒对应为一个长方体。在几何划分时,首先会基于x轴一直进行二叉树划分,得到两个子节点;直到满足 $d_x = d_y > d_z$ 条件时,才会基于x和y轴一直进行四叉树划分,得到四个子节点;当最终满足 $d_x = d_y = d_z$ 条件时,会一直进行八叉树划分,直到划分得到的叶子结点为 $1 \times 1 \times 1$ 的单位立方体时停止划分,对叶子结点中的点进行编码,生成二进制码流。在基于二叉树/四叉树/八叉树划分的过程中,引入两个参数:K、M。参数K指示在进行八叉树划分之前二叉树/四叉树划分的最大次数;参数M用来指示在进行二叉树/四叉树划分时对应的最小块边长为 2^M 。同时K和M必须满足条件:假设 $d_{\max} = \max(d_x, d_y, d_z)$, $d_{\min} = \min(d_x, d_y, d_z)$,参数K满足: $K \geq d_{\max} - d_{\min}$;参数M满足: $M \geq d_{\min}$ 。参数K与M之所以满足上述的条件,是因为目前G-PCC在几何隐式划分的过程中,划分方式的优先级为二叉树、四叉树和八叉树,当节点块大小不满足二叉树/四叉树的条件时,才会对节点一直进行八叉树的划分,直到划分到叶子节点最小单位 $1 \times 1 \times 1$ 。基于八叉树的几何信息编码模式可以通过利用空间中相邻点之间的相关性来对点云的几何信息进行有效的编码,但是对于一些较为平坦的节点或者具有平面特性的节点,通过利用平面编码可以进一步提升点云几何信息的编码效率。

示例性地,图5A和图5B提供了一种平面位置示意图。其中,图5A示出了一种Z轴方向的低平面位置示意图,图5B示出了一种Z轴方向的高平面位置示意图。如图5A所示,这里的(a)、(a0)、(a1)、(a2)、(a3)均属于Z轴方向的低平面位置,以(a)为例,可以看到当前节点中被占据的四个子节点都位于当前节点在Z轴方向的低平面位置,那么可以认为当前节点属于一个Z平面并且在Z轴方向是一个低平面。同理,如图5B所示,这里的(b)、(b0)、(b1)、(b2)、(b3)均属于Z轴方向的高平面位置,以(b)为例,可以看到当前节点中被占据的四个子节点位于当前节点在Z轴方向的高平面位置,那么可以认为当前节点属于一个Z平面并且在Z轴方向是一个高平面。

进一步地,以图5A中的(a)为例,对八叉树编码和平面编码效率进行比较,图6提供了一种节点编码顺序示意图,即按照图6所示的0、1、2、3、4、5、6、7的顺序进行节点编码。在这里,如果对图5A中的(a)采用八叉树编码方式,那么当前节点的占位信息表示为:11001100。但是如果采用平面编码方式,首先需要编码一个标识符表示当前节点在Z轴方向是一个平面,其次如果当前节点在Z轴方向是一个平面,还需要对当前节点的平面位置进行表示;其次仅仅需要对Z轴方向的低平面节点的占位信息进行编码(即0、2、4、6四个子节点的占位信息),因此基于平面编码方式对当前节点进行编码,仅仅需要编码6个比特(bit),相比相关技术的八叉树编码可以减少2个bit的表示。基于此分析,平面编码相比八叉树编码具有较为明显的编码效率。因此,对于一个被占据的节点,如果在某一个维度上采用平面编码方式进行编码,首先需要当前节点在该维度上的平面标识(planarMode)和平面位置(PlanePos)信息进行表示,其次基于当前节点的平面信息来对当前节点的占位信息进行编码。示例性地,图7A示出了一种平面标识信息示意图。如图7A所示,这里在Z轴方向为一个低平面;对应

地, 平面标识信息的取值为真 (true) 或者 1, 即 $\text{planarMode}_z=\text{true}$; 平面位置信息为低平面 (low), 即 $\text{PlanePosition}_z=\text{low}$ 。图 7B 示出了另一种平面标识信息示意图。如图 7B 所示, 这里在 Z 轴方向不

为一个平面; 对应地, 平面标识信息的取值为假 (false) 或者 0, 即 $\text{planarMode}_z=\text{false}$ 。
需要注意的是, 对于 PlaneMode_i : 0 代表当前节点在 i 轴方向不是一个平面, 1 代表当前节点在 i

轴方向是一个平面。若当前节点在 i 轴方向是一个平面, 则对于 PlanePosition_i : 0 代表当前节点在 i 轴

方向是一个平面, 并且平面位置为低平面, 1 表示当前节点在 i 轴方向上是一个高平面。其中, i 表示

坐标维度, 可以为 X 轴方向、Y 轴方向或者 Z 轴方向, 故 $i=0, 1, 2$ 。

在 G-PCC 标准中, 判断一个节点是否满足平面编码的条件以及在该节点满足平面编码条件时, 需

要对该节点的平面标识和平面位置信息的预测编码。

在本申请实施例中, 当前 G-PCC 标准中存在三种判断节点是否满足平面编码的判断条件, 下面对

其逐一进行详细说明。

一、根据节点在每个维度上的平面概率进行判断。

(1) 确定当前节点的局部区域密度 ($\text{local_node_density}$);

(2) 确定当前节点在每个维度上的概率 $\text{Prob}(i)$ 。

在节点的局部区域密度小于阈值 Th (例如 $\text{Th}=3$) 时, 利用当前节点在三个坐标维度上的平面概率

$\text{Prob}(i)$ 和阈值 Th_0 、 Th_1 和 Th_2 进行比较, 其中 $\text{Th}_0<\text{Th}_1<\text{Th}_2$ (例如, $\text{Th}_0=0.6$, $\text{Th}_1=0.77$, $\text{Th}_2=0.88$),

这里可以利用 Eligible_i ($i=0, 1, 2$) 表示每个维度上是否启动平面编码: $\text{Eligible}_i=\text{Prob}(i)\geq\text{threshold}$ 。

需要注意的是, threshold 是进行自适应变化的, 例如, 当 $\text{Prob}(0)>\text{Prob}(1)>\text{Prob}(2)$ 时, 则 Eligible_i

的设置如下:

$$\text{Eligible}_0=\text{Prob}(0)\geq\text{Th}_0;$$

$$\text{Eligible}_1=\text{Prob}(1)\geq\text{Th}_1;$$

$$\text{Eligible}_2=\text{Prob}(2)\geq\text{Th}_2.$$

当 $\text{Prob}(1)>\text{Prob}(0)>\text{Prob}(2)$ 时, 则 Eligible_i 的设置如下:

$$\text{Eligible}_0=\text{Prob}(0)\geq\text{Th}_1;$$

$$\text{Eligible}_1=\text{Prob}(1)\geq\text{Th}_0;$$

$$\text{Eligible}_2=\text{Prob}(2)\geq\text{Th}_2.$$

在这里, $\text{Prob}(i)$ 的更新具体如下:

$$\text{Prob}(i)_{\text{new}}=(L\times\text{Prob}(i)+\delta(\text{coded node}))/L+1 \quad (1)$$

其中, $L=255$; 另外, 若 coded node 节点是一个平面, 则 $\delta(\text{coded node})$ 为 1; 否则 $\delta(\text{coded node})$ 为

0。

在这里, $\text{local_node_density}$ 的更新具体如下:

$$\text{local_node_density}_{\text{new}}=\text{local_node_density}+4*\text{numSiblings} \quad (2)$$

其中, $\text{local_node_density}$ 初始化为 4, numSiblings 为该节点的兄弟姐妹节点数目。示例性地, 图 8

示出了一种当前节点的兄弟姐妹节点示意图。如图 8 所示, 当前节点为用斜线填充的节点, 用网格填充

的节点为兄弟姐妹节点, 那么当前节点的兄弟姐妹节点数目为 5 (包括当前节点自身)。

二、根据当前层的点云密度来判断当前层节点是否满足平面编码。

利用当前层点的密度来判断是否对当前层的节点进行平面编码。假设当前待编码点云的点数为

pointCount , 经过推断直接编码模式 (Infer Direct Coding Model, IDCM) 编码已经重建出的点数为

$\text{numPointCountRecon}$, 又因为八叉树是基于广度优先遍历的顺序进行编码, 因此可以得到当前层待编码

的节点数目假设为 nodeCount , 那么判断当前层是否启动平面编码假设为 $\text{planarEligibleKOctreeDepth}$,

具体为: $\text{planarEligibleKOctreeDepth}=(\text{pointCount}-\text{numPointCountRecon})<\text{nodeCount}\times 1.3$ 。

其中, 若 $(\text{pointCount}-\text{numPointCountRecon})$ 小于 $\text{nodeCount}\times 1.3$, 则 $\text{planarEligibleKOctreeDepth}$

为 true; 若 $(\text{pointCount}-\text{numPointCountRecon})$ 不小于 $\text{nodeCount}\times 1.3$, 则 $\text{planarEligibleKOctreeDepth}$

为 false。这样, 当 $\text{planarEligibleKOctreeDepth}$ 为 true 时, 则在当前层所有节点都进行平面编码; 否则

在当前层所有节点都不进行平面编码, 仅仅采用八叉树编码。

三、根据激光雷达点云的采集参数来判断当前节点是否满足平面编码。

图 9 示出了一种激光雷达与节点的相交示意图。如图 9 所示, 用网格填充的节点同时被两个激光射

线 (Laser) 穿过, 因此当前节点在 Z 轴垂直方向上不是一个平面; 用斜线填充的节点足够小到不能同

时被两个 Laser 同时穿过, 因此斜线填充的节点在 Z 轴垂直方向上有可能是一个平面。

进一步地, 针对满足平面编码条件的节点, 可以对平面标识信息和平面位置信息进行预测编码。

首先, 平面标识信息的预测编码。

在这里, 仅仅采用三个上下文信息进行编码, 即各个坐标维度上的平面标识分开进行上下文设计。

其次，平面位置信息的预测编码。

应理解，针对非激光雷达点云平面位置信息的编码而言，平面位置信息的预测编码可以包括：

(a) 利用邻域节点的占位信息进行预测得到当前节点的平面位置信息为三元素：预测为低平面、预测为高平面和无法预测；

5 (b) 与当前节点在相同划分深度以及相同坐标下的节点与当前节点之间的空间距离：“近”和“远”；

(c) 与当前节点在相同划分深度以及相同坐标下的节点如果是一个平面，则确定该节点的平面位置；

(d) 坐标维度 ($i=0, 1, 2$)。

需要说明的是，在本申请实施例，确定出与当前节点在相同划分深度以及相同坐标下的节点和当前节点之间的空间距离之后，如果该空间距离小于预设距离阈值，那么可以确定该空间距离为“近”；或者，如果该空间距离大于预设距离阈值，那么可以确定该空间距离为“远”。

示例性地，图 10 示出了一种处于相同划分深度以及相同坐标的邻域节点示意图。如图 10 所示，加粗的大立方体表示父节点 (Parent node)，其内部网格填充的小立方体表示当前节点 (Current node)，并且示出了当前节点的交点位置 (Vertex position)；白色填充的小立方体表示处于相同划分深度以及相同坐标的邻域节点，当前节点与邻域节点之间的距离为空间距离，可以判断为“近”或“远”；另外，如果该邻域节点为一个平面，那么还需要该邻域节点的平面位置 (Planar position)。

这样，如图 10 所示，当前节点为网格填充的小立方体，则在相同的八叉树划分深度等级下，以及相同的垂直坐标下查找邻域节点为白色填充的小立方体，判断两个节点之间的距离为“近”和“远”，并且参考节点的平面位置。

20 进一步地，在本申请实施例，图 11 示出了一种当前节点位于父节点的低平面位置示意图。如图 11 所示，(a)、(b)、(c) 示出了三种当前节点位于父节点的低平面位置的示例。具体说明如下：

①如果点填充节点的子节点 4 到 7 中有任何一个被占用，而所有网格填充节点都未被占用，则极有可能在当前节点 (用斜线填充) 中存在一个平面，且该平面位置较低。

25 ②如果点填充节点的子节点 4 到 7 都未被占用，而任何网格填充节点被占用，则极有可能在当前节点 (用斜线填充) 中存在一个平面，且该平面位置较高。

③如果点填充节点的子节点 4 到 7 均为空节点，网格填充节点均为空节点，则无法推断平面位置，故标记为未知。

④如果点填充节点的子节点 4 到 7 中有任何一个被占用，而网格填充节点中有任何一个被占用，此时也无法推断出平面位置，因此将其标记为未知。

30 在本申请实施例，图 12 示出了一种当前节点位于父节点的高平面位置示意图。如图 12 所示，(a)、(b)、(c) 示出了三种当前节点位于父节点的高平面位置的示例。具体说明如下：

①如果网格填充节点的子节点 4 到 7 中有任何一个节点被占用，而点填充节点未被占用，则极有可能在当前节点 (用斜线填充) 中存在一个平面，且平面位置较低。

35 ②如果网格填充节点的子节点 4 到 7 均未被占用，而点填充节点被占用，则极有可能在当前节点 (用斜线填充) 中存在平面，且平面位置较高。

③如果网格填充节点的子节点 4 到 7 都是未被占用的，而点填充节点是未被占用的，此时无法推断平面位置，因此标记为未知。

④如果网格填充节点的子节点 4 到 7 中有一个被占用，而点填充节点被占用，此时无法推断平面位置，因此标记为未知。

40 还应理解，针对激光雷达点云平面位置信息的编码而言，图 13 示出了一种激光雷达点云平面位置信息的预测编码示意图。如图 13 所示，在激光雷达的发射角度为 θ_{bottom} 时，这时候可以映射为低平面 (Bottom virtual plane)；在激光雷达的发射角度为 θ_{top} 时，这时候可以映射为高平面 (Top virtual plane)。

也就是说，通过利用激光雷达采集参数来预测当前节点的平面位置，通过利用当前节点与激光射线相交的位置来将位置量化为多个区间，最终作为当前节点平面位置的上下文信息。具体计算过程如下：假设激光雷达的坐标为 $(x_{\text{Lidar}}, y_{\text{Lidar}}, z_{\text{Lidar}})$ ，当前节点的几何坐标为 (x, y, z) ，那么首先计算当前节点相对于激光雷达的垂直正切值 $\tan \theta$ ，计算公式如下：

$$\tan \theta = \frac{z - z_{\text{Lidar}}}{\sqrt{(x - x_{\text{Lidar}})^2 + (y - y_{\text{Lidar}})^2}} \quad (3)$$

进一步地，又因为每个 Laser 会相对于激光雷达有一定偏移角度，因此还需要计算当前节点相对于 Laser 的相对正切值 $\tan \theta_{\text{corr}, L}$ ，具体计算如下：

$$50 \quad \tan \theta_{\text{corr}, L} = \frac{z - z_{\text{Lidar}} - z_L}{\sqrt{(x - x_{\text{Lidar}})^2 + (y - y_{\text{Lidar}})^2}} = \tan \theta - \frac{z_L}{r} \quad (4)$$

最终会利用当前节点的相对正切值 $\tan \theta_{\text{corr}, L}$ 来对当前节点的平面位置进行预测，具体如下，假设当

前节点下边界的正切值为 $\tan(\theta_{\text{bottom}})$ ，上边界的正切值为 $\tan(\theta_{\text{top}})$ ，根据 $\tan \theta_{\text{corr},L}$ 将平面位置量化为 4 个量化区间，即确定平面位置的上下文信息。

但是基于八叉树的几何信息编码模式仅对空间中具有相关性的点有高效的压缩速率，而对于在几何空间中处于孤立位置的点来说，使用直接编码模式 (Direct Coding Model, DCM) 可以大大降低复杂度。

5 对于八叉树中的所有节点，DCM 的使用不是通过标志位信息来表示的，而是通过当前节点父节点和邻居信息来进行推断得到。判断当前节点是否具有 DCM 编码资格的方式有三种，具体如下：

(1) 当前节点没有兄弟姐妹子节点，即当前节点的父节点只有一个孩子节点，同时当前节点父节点的父节点仅有两个被占据子节点，即当前节点最多只有一个邻居节点。

10 (2) 当前节点的父节点仅有当前节点一个占据子节点，同时与当前节点共用一个面的六个邻居节点也都属于空节点。

(3) 当前节点的兄弟姐妹节点数目大于 1。

15 示例性地，图 14 提供了一种 IDCM 编码示意图。如果当前节点不具有 DCM 编码资格将对其进行八叉树划分，若具有 DCM 编码资格将进一步判断该节点中包含的点数，当点数小于阈值 (例如 2) 时，则对该节点进行 DCM 编码，否则将继续进行八叉树划分。当应用 DCM 编码模式时，首先需要编码当前节点是否是一个真正的孤立点，即 IDCM_flag，当 IDCM_flag 为 true 时，则当前节点采用 DCM 编码，否则仍然采用八叉树编码。在当前节点满足 DCM 编码时，需要编码当前节点的 DCM 编码模式，目前存在两种 DCM 模式，分别是：(a) 仅仅只有一个点存在 (或者是多个点，但是属于重复点)；(b) 含有两个点。最后需要编码每个点的几何信息，假设节点的边长为 2^d 时，对该节点几何坐标的每一个分量进行编码时需要 d 比特，该比特信息直接被编进码流中。这里需要注意的是，在对激光雷达点云进行编码时，通过利用激光雷达采集参数来对三个维度的坐标信息进行预测编码，从而可以进一步提升几何信息的编码效率。

进一步地，下面针对 IDCM 编码的过程进行详细介绍。

当前节点满足 DCM 编码模式时，首先编码当前节点的点数目 numPoints；根据不同的 DirectMode 来对当前节点的点数目进行编码：

25 (1) 如果当前节点不满足 DCM 节点的要求，则直接退出 (即点数大于 2 个点，并且不是重复点)。

(2) 当前节点含有的点数 numPoints 小于或等于 2，则编码过程如下：

i) 首先编码当前节点的 numPoints 是否大于 1；

30 ii) 如果当前节点只有一个点并且几何编码环境为几何无损编码，则需要编码当前节点的第二个点不是重复点。

(3) 当前节点含有的点数 numPoints 大于 2，则编码过程如下：

i) 首先编码当前节点的 numPoints 小于或等于 1；

ii) 其次编码当前节点的第二个点是一个重复点，其次编码当前节点的重复点数目是否大于 1，当重复点数目大于 1 时，需要对剩余的重复点数目进行指数哥伦布解码。

35 在编码完成当前节点的点数目之后，对当前节点中包含点的坐标信息进行编码。下面将分别对激光雷达点云和面向人眼点云进行详细介绍。

(一) 面向人眼点云。

(1) 如果当前节点中仅仅只含有一个点，则会对点的三个维度方向的几何信息进行直接编码 (Bypass coding)；

40 (2) 如果当前节点中含有两个点，则会首先通过利用点的几何坐标得到优先编码的坐标轴 directAxis。这里需要注意的是，目前比较的坐标轴只包含 x 轴和 y 轴，不包含 z 轴。假设当前节点的几何坐标为 nodePos，则判断的方式如下：

directAxis = ! (nodePos[0] < nodePos[1]) (5)

也就是会将节点坐标几何位置小的轴作为优先编码的坐标轴 directAxis，其次按照如下方式首先对优先编码的坐标轴 directAxis 几何信息进行编码。假设优先编码的轴对应的代编码几何 bit 深度为 nodeSizeLog2，并假设两个点的坐标分别为 pointPos[0] 和 pointPos[1]。具体编码过程如下：

45 Bool sameBit=true;
while(nodeSizeLog2 && sameBit){
int mask=1<< nodeSizeLog2;
50 --nodeSizeLog2;
bool bit0=!!(pointPos[0]& mask)
bool bit1=!!(pointPos[1]& mask)
sameBits=bit0==bit1;

```

        entropyCodeSameBit(sameBits); ///<entropy coding
        if(sameBits)
            encodePosBit(bit0); ///<Bypass coding
    }

```

5 在编码完成优先编码的坐标轴 `dirextAxis` 之后，再继续对当前节点的几何坐标进行直接编码。假设每个点的剩余编码 bit 深度为 `nodeSizeLog2`，则具体编码过程如下：

```

    for(int axisIdx=0;axisIdx<3;++axisIdx)
    for(int mask=(1<< nodeSizeLog2[axisIdx])>>1;mask;mask>>1)
        encodePosBit(!(pointPos[axisIdx]&mask));

```

10 (二) 面向激光雷达点云。

如果当前节点中含有两个点，则会首先通过利用点的几何坐标得到优先编码的坐标轴 `dirextAxis`，假设当前节点的几何坐标为 `nodePos`，则判断的方式如下：

```

    dirextAxis = ! (nodePos[0] < nodePos[1])

```

也就是会将节点坐标几何位置小的轴作为优先编码的坐标轴 `dirextAxis`，这里需要注意的是，目前比较的坐标轴只包含 x 轴和 y 轴，不包含 z 轴。其次按照如下方式首先对优先编码的坐标轴 `dirextAxis` 几何信息进行编码，假设优先编码的轴对应的代编码 bit 深度为 `nodeSizeLog2`，并假设两个点的坐标分别为 `pointPos[0]` 和 `pointPos[1]`。具体编码过程如下：

```

    Bool sameBit=true;
    while(nodeSizeLog2&& sameBit){
    20         int mask=1<< nodeSizeLog2;
            --nodeSizeLog2;
            bool bit0=!!( pointPos[0]& mask)
            bool bit1=!!( pointPos[1]& mask)
            sameBits=bit0==bit1;
    25         entropyCodeSameBit(sameBits);
            if(sameBits)
                encodePosBit(bit0);
    }

```

在编码完成优先编码的坐标轴 `dirextAxis` 之后，再对当前节点的几何坐标进行编码。

30 由于激光雷达点云可以得到激光雷达点云的采集参数，通过利用可以预测当前节点的几何坐标信息，从而可以进一步提升点云的几何信息编码效率。同样的首先利用当前节点的几何信息 `nodePos` 得到一个直接编码的主轴方向，其次利用已经完成编码的方向的几何信息来对另外一个维度的几何信息进行预测编码。同样假设直接编码的轴方向是 `directAxis`，并且假设直接编码中的代编码 bit 深度为 `nodeSizeLog2`，则编码方式如下：

```

    for(int mask=(1<< nodeSizeLog2)>>1;mask;mask>>1);
        encodePosBit(!(pointPos[directAxis]&mask));

```

这里需要注意的是，在这里会将 `directAxis` 方向的几何精度信息全部编码。

示性地，图 15 提供了一种旋转激光雷达获取点云的坐标转换示意图。其中，在笛卡尔坐标系下，对于每一个节点的 (x,y,z) 坐标，均可以转换为用 (R, φ, i) 表示。另外，激光扫描器 (Laser Scanner) 40 可以按照预设角度进行激光扫描，在 i 的不同取值下，可以得到不同的 θ(i)。例如，在 i 等于 1 时，这时候可以得到 θ(1)，对应的扫描角度为 -15°；在 i 等于 2 时，这时候可以得到 θ(2)，对应的扫描角度为 -13°；在 i 等于 10 时，这时候可以得到 θ(10)，对应的扫描角度为 +13°；在 i 等于 9 时，这时候可以得到 θ(19)，对应的扫描角度为 +15°。

这样，在编码完成 `directAxis` 坐标方向的所有精度之后，会首先计算当前点所对应的 `LaserIdx`，即图 15 中的 `pointLaserIdx` 号，并且计算当前节点的 `LaserIdx`，即 `nodeLaserIdx`；其次会利用节点的 `LaserIdx` 即 `nodeLaserIdx` 来对点的 `LaserIdx` 即 `pointLaserIdx` 进行预测编码，其中节点或者点的 `LaserIdx` 的计算方式如下。假设点的几何坐标为 `pointPos`，激光射线的起始坐标为 `LidarOrigin`，并且假设 Laser 的数目为 `LaserNum`，每个 Laser 的正切值为 $\tan\theta_i$ ，每个 Laser 在垂直方向上的偏移位置为 Z_i ，则：

```

    Int bestLaserIdx=0;
    Int Distortion=INT_MAX;
    For(int LaserIdx=0; LaserIdx<numLaser;++ LaserIdx){
    50         int radius =  $\sqrt{(\text{pointPos}[0] - \text{LidarOrigin}[0])^2 + (\text{pointPos}[1] - \text{LidarOrigin}[1])^2}$ 
            int invRadius=1/ radius
            int Z=pointPos[2]+  $Z_i$ 
            int tanTheta=  $Z \times \text{invRadius}$ 

```

```

        if(std::abs(tanTheta-tanθi)<Distortion){
            Distortion= std::abs(tanTheta-tanθi);
            bestLaserIdx= LaserIdx;
        }
    }

```

5

在计算得到当前点的 LaserIdx 之后，首先会利用当前节点的 LaserIdx 对点的 pointLaserIdx 进行预测编码。在编码完成当前点的 LaserIdx 之后，对当前点三个维度的几何信息利用激光雷达的采集参数进行预测编码。

10

示例性地，图 16 示出了一种 X 轴或 Y 轴方向的预测编码示意图。如图 16 所示，用网格填充的方框表示当前点（Current node），用斜线填充的方框表示已编码点（Already coded node）。在这里，首先利用当前点对应的 LaserIdx 得到对应的水平方位角的预测值，即 φ_{pred} ；其次利用当前点对应的节点几何信息得到节点对应的水平方位角 φ_{node} 。其中，假设节点的几何坐标为 nodePos，则水平方位角 φ 与节点几何信息之间的计算方式如下：

$$\varphi = \arctan(\text{nodePos}[1]/\text{nodePos}[0]) \quad (6)$$

15

通过利用激光雷达的采集参数，可以得到每个 Laser 的旋转点数 numPoints，即代表每个激光射线旋转一圈得到的点数，则可以利用每个 Laser 的旋转点数计算得到每个 Laser 的旋转角速度 deltaPhi，计算方式如下：

$$\text{deltaPhi} = \frac{2\pi}{\text{numPoints}} \quad (7)$$

20

进一步地，利用节点的水平方位角 φ_{node} 以及当前点对应的 Laser 前一个编码点的水平方位角 φ_{pred} 计算得到当前点对应的水平方位角预测值 $\varphi_{predPoint}$ ，即如图 17A 和图 17B 所示的水平方位角的预测值。其中，图 17A 示出了一种通过水平方位角来进行预测 Y 平面的角度示意图，图 17B 示出了一种通过水平方位角来进行预测 X 平面的角度示意图。在这里，对于当前点对应的水平方位角预测值 $\varphi_{predPoint}$ ，计算方式如下：

$$\varphi_{predPoint} = \frac{\varphi_{pred} - \varphi_{node}}{\text{deltaPhi}} \times \text{deltaPhi} + \varphi_{pred} \quad (8)$$

25

示例性地，图 18 示出了另一种 X 轴或 Y 轴方向的预测编码示意图。如图 18 所示，用网格填充的部分（左侧）表示低平面，用点填充的部分（右侧）表示高平面， φ_{left} 表示当前节点的低平面水平方位角， φ_{right} 表示当前节点的高平面水平方位角， φ_{pred} 表示当前节点对应的水平方位角预测值。

这样，通过利用水平方位角的预测值 $\varphi_{predPoint}$ 以及当前节点的低平面水平方位角 φ_{left} 和高平面水平方位角 φ_{right} 来对当前节点的几何信息进行预测编码。具体如下所示：

30

```

int angLeL = φleft - φpred;
int angLeR = φright - φpred;
int context = (angLeL ≥ 0 && angLeR ≥ 0) || (angLeL < 0 && angLeR < 0)? 0: 2;
int minAngle = std::min(abs(angLeL), abs(angLeR));
int maxAngle = std::max(abs(angLeL), abs(angLeR));
context += maxAngle > minAngle? 0: 1;
context += maxAngle > minAngle? 0: 4.

```

35

在编码完成点的 LaserIdx 之后，会利用当前点所对应的 LaserIdx 对当前点的 Z 轴方向进行预测编码，即当前通过利用当前点的 x 和 y 信息计算得到雷达坐标系的深度信息 radius，其次利用当前点的激光 LaserIdx 得到当前点的正切值以及垂直方向的偏移量，则可以得到当前点的 Z 轴方向的预测值即 Z_{pred}。具体如下所示：

40

```

int radius = √((pointPos[0] - LidarOrigin[0])2 + (pointPos[1] - LidarOrigin[1])2);
int tanTheta= tanθlaserIdx;
int zOffset= ZlaserIdx;
Zpred= radius × tanTheta- zOffset.

```

45

进一步地，利用 Z_{pred} 对当前点的 Z 轴方向的几何信息进行预测编码得到预测残差 Z_{res}，最终对 Z_{res} 进行编码。

50

需要注意的是，在节点划分到叶子节点时，在几何无损编码的情况下，需要对叶子节点中的重复点数目进行编码。最终对所有节点的占位信息进行编码，生成二进制码流。另外 G-PCC 目前引入了一种平面编码模式，在对几何进行划分的过程中，会判断当前节点的子节点是否处于同一平面，如果当前节点的子节点满足同一平面的条件，会用该平面对当前节点的子节点进行表示。

对于基于八叉树的几何解码而言，解码端按照广度优先遍历的顺序，在对每个节点的占位信息解码之前，首先会利用已经重建得到的几何信息来判断当前节点是否进行平面解码或者 IDCM 解码，如果

当前节点满足平面解码的条件,则会首先对当前节点的平面标识和平面位置信息进行解码,其次基于平面信息来对当前节点的占位信息进行解码;如果当前节点满足 IDCM 解码的条件,则会首先解码当前节点是否是一个真正的 IDCM 节点,如果是一个真正的 IDCM 解码,则会继续解析当前节点的 DCM 解码模式,其次可以得到当前 DCM 节点中的点数目,最后对每个点的几何信息进行解码。对于既不满足平面解码也不满足 DCM 解码的节点,会对当前节点的占位信息进行解码。通过按照这样的方式不断解析得到每个节点的占位码,并且依次不断划分节点,直至划分得到 $1 \times 1 \times 1$ 的单位立方体时停止划分,解析得到每个叶子节点中包含的点数目,最终恢复得到几何重构点云信息。

下面对 IDCM 解码的过程进行详细介绍。

与编码端的处理过程类似,首先利用先验信息来决定节点是否启动 IDCM,即 IDCM 的启动条件如下:

(1) 当前节点没有兄弟姐妹子节点,即当前节点的父节点只有一个孩子节点,同时当前节点父节点的父节点仅有两个被占据子节点,即当前节点最多只有一个邻居节点。

(2) 当前节点的父节点仅有当前节点一个占据子节点,同时与当前节点共用一个面的六个邻居节点也都属于空节点。

(3) 当前节点的兄弟姐妹节点数目大于 1。

进一步地,当节点满足 DCM 编码的条件时,首先解码当前节点是否是一个真正的 DCM 节点,即 IDCM_flag;当 IDCM_flag 为 true 时,则当前节点采用 DCM 编码,否则仍然采用八叉树编码。

其次解码当前节点的点数目 numPoints,具体的解码方式如下所示:

i) 首先解码当前节点的 numPoints 是否大于 1;

ii) 如果解码得到当前节点的 numPoints 大于 1,则继续解码第二个点是否是一个重复点;如果第二个点不是重复点,则这里可以隐性推断出满足 DCM 模式的第二种,只含有两个点;

iii) 如果解码得到当前节点的 numPoints 小于等于 1,则继续解码第二个点是否是一个重复点;如果第二个点不是重复点,则这里可以隐性推断出满足 DCM 模式的第二种,只含有一个点;如果解码得到第二个点是一个重复点,则可以推断出满足 DCM 模式的第三种,含有多个点,但是都是重复点,则继续解码重复点的数目是否大于 1 (熵解码),如果大于 1,则继续解码剩余重复点的数目 (利用指数哥伦布进行解码)。

如果当前节点不满足 DCM 节点的要求,则直接退出 (即点数大于 2 个点,并且不是重复点)。

在解码完成当前节点的点数目之后,对当前节点中包含点的坐标信息进行解码。下面将分别对激光雷达点云和面向人眼点云进行详细介绍。

(一) 面向人眼点云。

(1) 如果当前节点中仅仅只含有一个点,则会对点的三个维度方向的几何信息进行直接解码 (Bypass coding);

(2) 如果当前节点中含有两个点,则会首先通过利用点的几何坐标得到优先解码的坐标轴 dirextAxis,这里需要注意的是,目前比较的坐标轴只包含 x 和 y 轴,不包含 z 轴。假设当前节点的几何坐标为 nodePos,则判断的方式如下:

$$\text{dirextAxis} = !(\text{nodePos}[0] < \text{nodePos}[1]) \quad (9)$$

也就是会将节点坐标几何位置小的轴作为优先解码的坐标轴 dirextAxis,其次按照如下方式首先对优先解码的坐标轴 dirextAxis 几何信息进行解码。假设优先解码的轴对应的待解码几何 bit 深度为 nodeSizeLog2,并假设两个点的坐标分别为 pointPos[0]和 pointPos[1]。具体编码过程如下:

Bool sameBit=true;

while(nodeSizeLog2 && sameBit){

pointPos[0][dirextAxis]<<1;

pointPos[1][dirextAxis]<<1;

--nodeSizeLog2;

int bit=0;

deEntropyCodeSameBit(sameBits); ///<entropy coding

if(sameBits){

bit=decodePosBit(); ///<Bypass coding

pointPos[0][dirextAxis]= bit

pointPos[1][dirextAxis]= bit

}else

pointPos[1][dirextAxis]= 1 ///<这里的原因是因为在编码的时候,会对两个点

在优先编码轴的方向上进行排序，所以可以保证 $\text{pointPos}[0][\text{dirextAxis}] < \text{pointPos}[1][\text{dirextAxis}]$ ，因此在解码的时候，如果两个点的 bit 位信息不相同，则可以推断出第一个点的 bit 位为 0，第二个点的 bit 位为 1。

}

在解码完成优先解码的坐标轴 dirextAxis 之后，再继续对当前点的几何坐标进行直接解码。假设每个点的剩余编码 bit 深度为 nodeSizeLog2 ，并假设点的坐标信息为 pointPos ，则具体解码过程如下：

```
for(int axisIdx=0;axisIdx<3;++axisIdx)
  for(int idx= nodeSizeLog2[axisIdx]; idx; idx--){
    pointPos[axisIdx]<<1;
    pointPos[axisIdx]=decodePosBit();
  }
```

(二) 面向激光雷达点云。

如果当前节点中含有两个点，则会首先通过利用点的几何坐标得到优先解码的坐标轴 dirextAxis ，假设当前节点的几何坐标为 nodePos ，则判断的方式如下：

$\text{dirextAxis} = !(\text{nodePos}[0] < \text{nodePos}[1])$ (10)

也就是会将节点坐标几何位置小的轴作为优先解码的坐标轴 dirextAxis ，这里需要注意的是，目前比较的坐标轴只包含 x 轴和 y 轴，不包含 z 轴。其次按照如下方式首先对优先编码的坐标轴 dirextAxis 几何信息进行解码，假设优先解码的轴对应的代编码几何 bit 深度为 nodeSizeLog2 ，并假设两个点的坐标分别为 $\text{pointPos}[0]$ 和 $\text{pointPos}[1]$ 。具体编码过程如下：

```
Bool sameBit=true;
while(nodeSizeLog2&& sameBit){
  pointPos[0][ dirextAxis]<<1;
  pointPos[1][ dirextAxis]<<1;
  --nodeSizeLog2;
  int bit=0;
  deEntropyCodeSameBit(sameBits); ///<entropy coding
  if(sameBits){
    bit=decodePosBit();///<Bypass coding
    pointPos[0][ dirextAxis]= bit
    pointPos[1][ dirextAxis]= bit
  }else
```

$\text{pointPos}[1][\text{dirextAxis}] = 1$ ///<这里的原因是因为在编码的时候，会对两个点在优先编码轴的方向上进行排序，所以可以保证 $\text{pointPos}[0][\text{dirextAxis}] < \text{pointPos}[1][\text{dirextAxis}]$ ，因此在解码的时候，如果两个点的 bit 位信息不相同，则可以推断出第一个点的 bit 位为 0，第二个点的 bit 位为 1。

}

在解码完优先解码的坐标轴 dirextAxis 之后，再对当前点的几何坐标进行解码。

同样的首先利用当前节点的几何信息 nodePos 得到一个直接解码的主轴方向，其次利用已经完成解码的方向的几何信息来对另外一个维度的几何信息进行解码。同样假设直接解码的轴方向是 directAxis ，并且假设直接解码中的待解码 bit 深度为 nodeSizeLog2 ，则解码方式如下：

```
for(int idx= nodeSizeLog2[directAxis]; idx; idx--){
  pointPos[directAxis]<<1;
  pointPos[directAxis]=decodePosBit();
}
```

这里需要注意的是，在这里会将 directAxis 方向的几何精度信息全部解码。

在解码完成 directAxis 坐标方向的所有精度之后，会首先计算当前节点的 LaserIdx ，即 nodeLaserIdx ；其次会利用节点的 LaserIdx 即 nodeLaserIdx 来对点的 LaserIdx 即 pointLaserIdx 进行预测解码，其中节点或者点的 LaserIdx 的计算方式跟编码端相同。最终对当前点的 LaserIdx 与节点的 LaserIdx 预测残差信息进行解码得到 ResLaserIdx ，则解码方式如下：

$\text{PointLaserIdx} = \text{nodeLaserIdx} + \text{ResLaserIdx}$ (11)

在解码完成当前点的 LaserIdx 之后，对当前点三个维度的几何信息利用激光雷达的采集参数进行预测解码。具体算法如下：

如图 11 所示，首先利用当前点对应的 LaserIdx 得到对应的水平方位角的预测值，即 φ_{pred} ；其次利用当前点对应的节点几何信息得到节点对应的水平方位角度 φ_{node} 。其中，假设节点的几何坐标为 nodePos ，则水平方位角 φ 与节点几何信息之间的计算方式如下：

$$\varphi = \arctan(\text{nodePos}[1]/\text{nodePos}[0]) \quad (12)$$

通过利用激光雷达的采集参数,可以得到每个 Laser 的旋转点数 numPoints,即代表每个激光射线旋转一圈得到的点数,则可以利用每个 Laser 的旋转点数计算得到每个 Laser 的旋转角速度 deltaPhi,计算方式如下:

$$\text{deltaPhi} = \frac{2\pi}{\text{numPoints}} \quad (13)$$

进一步地,利用节点的水平方位角 φ_{node} 以及当前点对应的 Laser 前一个编码点的水平方位角 φ_{pred} 计算得到当前点对应的水平方位角预测值 $\varphi_{\text{predPoint}}$,即如图 17A 和图 17B 所示的水平方位角的预测值。计算方式如下:

$$\varphi_{\text{predPoint}} = \frac{\varphi_{\text{pred}} - \varphi_{\text{node}}}{\text{deltaPhi}} \times \text{deltaPhi} + \varphi_{\text{pred}} \quad (14)$$

这样,通过利用水平方位角的预测值 $\varphi_{\text{predPoint}}$ 以及当前节点的低平面水平方位角 φ_{left} 和高平面的水平方位角 φ_{right} 来对当前节点的几何信息进行预测解码。具体如下所示:

```

int angLeL =  $\varphi_{\text{left}} - \varphi_{\text{pred}}$ ;
int angLeR =  $\varphi_{\text{right}} - \varphi_{\text{pred}}$ ;
int context = ( $\text{angLeL} \geq 0 \&\& \text{angLeR} \geq 0$ ) || ( $\text{angLeL} < 0 \&\& \text{angLeR} < 0$ ) ? 0 : 2;
int absAngleL = abs(angLeL);
int absAngleR = abs(angLeR);
context += absAngleL > absAngleR ? 0 : 1;
context += maxAngle > minAngle << 1 ? 4 : 0;

```

在解码完成点的 LaserIdx 之后,会利用当前点所对应的 LaserIdx 对当前点的 Z 轴方向进行预测解码,即当前通过利用当前点的 x 和 y 信息计算得到雷达坐标系的深度信息 radius,其次利用当前点的激光 LaserIdx 得到当前点的正切值以及垂直方向的偏移量,则可以得到当前点的 Z 轴方向的预测值即 Z_{pred} 。具体如下所示:

```

int radius =  $\sqrt{(\text{pointPos}[0] - \text{LidarOrigin}[0])^2 + (\text{pointPos}[1] - \text{LidarOrigin}[1])^2}$ ;
int tanTheta =  $\tan\theta_{\text{laserIdx}}$ ;
int zOffset =  $Z_{\text{laserIdx}}$ ;
 $Z_{\text{pred}} = \text{radius} \times \tan\Theta - \text{zOffset}$ ;

```

进一步地,利用解码得到的 Z_{res} 和 Z_{pred} 来重建恢复得到当前点 Z 轴方向的几何信息。

对于基于三角面片集 (triangle soup, trisoup) 的几何信息编码而言,在基于 trisoup 的几何信息编码框架中,同样也要先进行几何划分,但区别于基于二叉树/四叉树/八叉树的几何信息编码,该方法不需要将点云逐级划分到边长为 $1 \times 1 \times 1$ 的单位立方体,而是划分到子块 (block) 边长为 W 时停止划分,基于每个 block 中点云的分布所形成的表面,得到该表面与 block 的十二条边所产生的至多十二个交点 (vertex)。依次编码每个 block 的 vertex 坐标,生成二进制码流。

对于基于 trisoup 的点云几何信息重建而言,在解码端进行点云几何信息重建时,首先解码 vertex 坐标用于完成三角面片重建,该过程如图 19A、图 19B 和图 19C 所示。其中,图 19A 所示的 block 中存在 3 个交点 (v_1, v_2, v_3),利用这 3 个交点按照一定顺序所构成的三角面片集被称为 triangle soup,即 trisoup,如图 19B 所示。之后,在该三角面片集上进行采样,将得到的采样点作为该 block 内的重建点云,如图 19C 所示。

对于基于预测树的几何编码 (Predictive geometry coding, PredGeomTree) 而言,基于预测树的几何编码包括:首先对输入点云进行排序,目前采用的排序方法包括无序、莫顿序、方位角序和径向距离序。在编码端通过利用两种不同的方式建立预测树结构,其中包括:KD-Tree (高时延慢速模式) 和低时延快速模式 (利用激光雷达标定信息)。在利用激光雷达标定信息时,将每个点划分到不同的 Laser 上,按照不同的 Laser 建立预测树结构。接下来基于预测树的结构,遍历预测树中的每个节点,通过选取不同的预测模式对节点的几何位置信息进行预测得到预测残差,并且利用量化参数对几何预测残差进行量化。最终通过不断迭代,对预测树节点位置信息的预测残差、预测树结构以及量化参数等进行编码,生成二进制码流。

对于基于预测树的几何解码而言,解码端通过不断解析码流,重构预测树结构,其次通过解析得到每个预测节点的几何位置预测残差信息以及量化参数,并且对预测残差进行反量化,恢复得到每个节点的重构几何位置信息,最终完成解码端的几何重构。

在几何编码完成后,需要对几何信息进行重建。目前,属性编码主要针对颜色信息进行。首先,将颜色信息从 RGB 颜色空间转换到 YUV 颜色空间。然后,利用重建的几何信息对点云重新着色,使得未编码的属性信息与重建的几何信息对应起来。在颜色信息编码中,主要有两种变换方法,一是依赖于

LOD 划分的基于距离的提升变换, 二是直接进行 RAHT 变换, 这两种方法都会将颜色信息从空间域转换到频域, 通过变换得到高频系数和低频系数, 最后对系数进行量化并编码, 生成二进制码流, 具体参见图 4A 和图 4B 所示。

进一步地, 在利用几何信息来对属性信息进行预测时, 可以利用莫顿码进行最近邻居搜索, 点云中每点对应的莫顿码可以由该点的几何坐标得到。计算莫顿码的具体方法描述如下所示, 对于每一个分量用 d 比特二进制数表示的三维坐标, 其三个分量可以表示为:

$$x = \sum_{\ell=1}^d 2^{d-\ell} x_{\ell}, y = \sum_{\ell=1}^d 2^{d-\ell} y_{\ell}, z = \sum_{\ell=1}^d 2^{d-\ell} z_{\ell} \quad (15)$$

其中, $x_{\ell}, y_{\ell}, z_{\ell} \in \{0,1\}$ 分别是 x, y, z 的最高位 ($\ell=1$) 到最低位 ($\ell=d$) 对应的二进制数值。莫顿码 M 是对 x, y, z 从最高位开始, 依次交叉排列 $x_{\ell}, y_{\ell}, z_{\ell}$ 到最低位, M 的计算公式如下所示:

$$M = \sum_{\ell=1}^d 2^{3(d-\ell)} (4x_{\ell} + 2y_{\ell} + z_{\ell}) = \sum_{\ell'=1}^{3d} 2^{3d-\ell'} m_{\ell'} \quad (16)$$

其中, $m_{\ell'} \in \{0,1\}$ 分别是 M 的最高位 ($\ell'=1$) 到最低位 ($\ell'=3d$) 的值。在得到点云中每个点的莫顿码 M 后, 将点云中的点按莫顿码由小到大的顺序进行排列, 并将每个点的权重值 w 设为 1。

还可以理解, 对于 G-PCC 编解码框架而言, 通用测试条件如下:

(1) 测试条件共 4 种:

- 条件 1: 几何位置有限度有损、属性有损;
- 条件 2: 几何位置无损、属性有损;
- 条件 3: 几何位置无损、属性有限度有损;
- 条件 4: 几何位置无损、属性无损。

(2) 通用测试序列包括 Cat1A, Cat1B, Cat3-fused, Cat3-frame 共四类, 其中 Cat2-frame 点云只包含反射率属性信息, Cat1A、Cat1B 点云只包含颜色属性信息, Cat3-fused 点云同时包含颜色和反射率属性信息。

(3) 技术路线: 共 2 种, 以几何压缩所采用的算法进行区分。

技术路线 1: 八叉树编码分支。

在编码端, 将包围盒依次划分得到子立方体, 对非空的 (包含点云中的点) 的子立方体继续进行划分, 直到划分得到的叶子结点为 $1 \times 1 \times 1$ 的单位立方体时停止划分, 在几何无损编码情况下, 需要对叶子节点中所包含的点数进行编码, 最终完成几何八叉树的编码, 生成二进制码流。

在解码端, 解码端按照广度优先遍历的顺序, 通过不断解析得到每个节点的占位码, 并且依次不断划分节点, 直至划分得到 $1 \times 1 \times 1$ 的单位立方体时停止划分, 在几何无损解码的情况下, 需要解析得到每个叶子节点中包含的点数, 最终恢复得到几何重构点云信息。

技术路线 2: 预测树编码分支。

在编码端, 通过利用两种不同的方式建立预测树结构, 其中包括: 基于 KD-Tree (高时延慢速模式) 和利用激光雷达标定信息 (低时延快速模式), 利用激光雷达标定信息, 可以将每个点划分到不同的 Laser 上, 按照不同的 Laser 建立预测树结构。接下来基于预测树的结构, 遍历预测树中的每个节点, 通过选取不同的预测模式对节点的几何位置信息进行预测得到预测残差, 并且利用量化参数对几何预测残差进行量化。最终通过不断迭代, 对预测树节点位置信息的预测残差、预测树结构以及量化参数等进行编码, 生成二进制码流。

在解码端, 解码端通过不断解析码流, 重构预测树结构, 其次通过解析得到每个预测节点的几何位置预测残差信息以及量化参数, 并且对预测残差进行反量化, 恢复得到每个节点的重构几何位置信息, 最终完成解码端的几何重构。

还需要说明的是, 在如图 4A 或图 4B 所示, 目前 G-PCC 编码框架包含三种属性编码方法: 预测变换 (Predicting Transform, PT)、提升变换 (Lifting Transform, LT) 以及区域自适应分层变换 (Region Adaptive Hierarchical Transform, RAHT)。其中, 前两者是以 LOD 的生成顺序为依据对点云预测编码, RAHT 则是依据八叉树的构建层级自下而上对属性信息进行自适应变换。下面将分别对这三种点云属性编码方法进行具体介绍。

(a) 点云属性信息的预测编码。

目前 G-PCC 的属性预测模块采用一种基于分层 (Level-of-details, LoDs) 结构的最近邻属性预测编码方案, LOD 的构造方法包括基于距离的 LOD 构造方案、基于固定采样率的 LOD 构造方案以及基于八叉树的 LOD 构造方案等。在基于距离阈值的 LOD 构造方案中, 构造 LOD 之前首先对点云进行 Morton 排序, 来保证相邻点之间具有较强的属性相关性。图 20 为一种基于距离的 LOD 构造过程的示意图。如图 20 所示, 根据用户提前预设的 L 个曼哈顿 (Manhattan) 距离 (d/l), $l=0,1,\dots,L-1$; 将点云划分成 L 个不同的点云细节层 (R/l), $l=0,1,\dots,L-1$, 其中 (d/l) $l=0,1,\dots,L-1$ 满足 $d/l < d/l-1$ 。LOD 的构造过程如下所述:

(1) 首先将点云中所有点都标记为未访问过, 建立一个集合 V 用来存储已经访问过的点集; (2) 对于每一次迭代 l , 通过对点云中的点进行遍历, 如果当前点已经被访问过, 则忽略该点, 否则计算当前点到点集 V 的最小距离 D , 如果 $D < dl$, 则忽略该点; 否则将当前点标记为已访问并将当前点加入细化层 R_l 和点集 V ; (3) 细节层次 LOD_l 中的点由细化层 $R_0, R_1, R_2 \dots R_l$ 中的点构成; (4) 不断重复上述步骤, 直至所有的点都被标记为已访问。

在 LOD 的结构基础上, 每个点的属性值通过利用同一层或更高一层 LOD 中点的属性重建值进行线性加权预测, 其中参考预测邻居的最大数目由编码器高层语法元素决定。对于每个点的属性, 在编码端利用率失真优化算法选取通过利用搜索到的 N 个最近邻点的属性进行加权预测或者选择单个最近邻点的属性进行预测, 最后对选取的预测模式以及预测残差进行编码。

$$Attr_i' = Round\left(\frac{1}{N} \sum_{m \in p_i} \frac{\frac{1}{D_m^2}}{\sum_{m \in p_i} \frac{1}{D_m^2}} Attr_m\right) \quad (17)$$

其中, N 代表点 i 最近邻点集中预测点的数目, P_i 代表点 i 的 N 个最近邻点的集合, D_m 代表了最近邻点 m 到当前点 i 的空间几何距离, $Attr_m$ 代表了最近邻点 m 重建之后的属性值, $Attr_i'$ 代表了对当前点 i 的属性预测值, 点数 N 为提前预设的数值。

为了权衡属性编码效率和不同 LOD 层之间的并行处理, 在编码器高层语法元素引入了一个开关可以控制是否引入 LOD 层内预测。如果开启则启动 LOD 层内预测, 可以利用同一 LOD 层内的点进行预测。需要注意的是, 当 LOD 层的数目为 1 时, 总是使用 LOD 层内预测。

图 21 为一种 LOD 生成过程的可视化结果示意图。如图 21 所示, 这里提供了一种基于距离的 LOD 生成过程的主观示例。具体是 (从左向右): 第一层中的点是代表点云的外轮廓; 随着细节层的增加, 点云细节描述逐渐清晰。

图 22 为一种属性预测的编码流程示意图。如图 22 所示, 针对 $G-PCC$ 属性预测的具体流程, 对于原始点云, 首先搜索第 K 个点的三个近邻点, 然后进行属性预测; 根据第 K 个点的属性预测值与第 K 个点的属性原始值进行差值计算, 可以得到第 K 个点的预测残差; 然后进行量化与算术编码, 最终生成属性码率。

(i) 最优预测值选取:

LOD 构建完成以后, 根据 LOD 的生成顺序, 首先从已编码的数据点中找到当前待编码点的三个最近邻点。将这 3 个最近邻点的属性重建值, 作为当前待编码点的候选预测值; 然后, 根据率失真优化 (Rate-Distortion Optimal, RDO) 从中选择最优的预测值。例如, 当编码图 20 中点 P_2 的属性值时, 将最近邻居点 P_4 属性值的预测变量索引设为 1; 将次近邻点 P_5 和三近邻点 P_0 的属性预测变量索引分别设为 2 和 3; 将点 P_0 、 P_5 和 P_4 的加权平均值的预测变量索引设为 0, 如表 1 所示; 最后, 利用 RDO 选择最佳预测变量。其中加权平均的公式如下所示:

$$\hat{a}_i = Round\left(\sum_{j=0}^2 \frac{\tilde{w}_{ij}}{\sum_{j=0}^2 \tilde{w}_{ij}} \tilde{a}_j\right) \quad (18)$$

其中, $\tilde{w}_{ij}$ 表示近邻点 j 到当前点 i 的空间几何权重:

$$\tilde{w}_{ij} = \frac{1}{(x_i - x_{ij})^2 + (y_i - y_{ij})^2 + (z_i - z_{ij})^2} \quad (19)$$

$\hat{a}_i$ 表示对当前点 i 的属性预测值, j 表示 3 个近邻点的索引, $\tilde{a}_j$ 代表了近邻点重建之后的属性值, x_i, y_i, z_i 是当前点 i 的几何位置坐标, x_{ij}, y_{ij}, z_{ij} 为近邻点 j 的几何坐标。

示例性地, 表 1 提供了一种属性编码的候选预测项样本示例。

表 1

预测模式	预测值
0	三近邻的属性加权平均
1	P_4 (一近邻的属性值)
2	P_5 (二近邻的属性值)
3	P_0 (三近邻的属性值)

(ii) 属性预测残差及量化:

通过上述预测得到当前点 i 的属性预测值 $(\hat{a}_i)_{i \in 0 \dots k-1}$ (k 为点云的总点数)。令 $(a_i)_{i \in 0 \dots k-1}$ 为当前点的属性原始值, 则属性残差 $(r_i)_{i \in 0 \dots k-1}$ 记为:

$$r_i = a_i - \hat{a}_i \quad (20)$$

进一步对预测残差进行量化:

$$Q_i = \frac{r_i}{Q_s} \quad (21)$$

其中, Q_i 表示当前点 i 的量化后的属性残差, Q_s 为量化步长 (Quantization step, Q_s), 可以由 CTC 规定的量化参数 QP (Quantization Parameter, QP) 计算得出。

(iii) 编码端重建属性值:

5 编码端重建的目的是为了后续点的预测。在重建属性值之前要对残差进行反量化, 记 $\hat{r}_i$ 为反量化后的残差:

$$\hat{r}_i = Q_i \times Q_s \quad (22)$$

$\hat{r}_i$ 与预测值 $\hat{a}_i$ 相加得到点 i 的重建值 $\tilde{a}_i$:

$$\tilde{a}_i = \hat{r}_i + \hat{a}_i \quad (23)$$

10 在基于 LOD 划分的基础上进行属性最近邻查找时, 目前存在两大类算法: 帧内最近邻查找和帧间最近邻查找。其中, 帧间的最近邻查找算法具体如下, 帧内的最近邻查找可以分为层间最近邻查找和层内最近邻查找两种算法。

(i) 帧内最近邻查找:

15 帧内最近邻查找分为层间最近邻查找和层内最近邻查找两种算法。LOD 划分之后, 类似一个金字塔结构, 如图 23 所示。

在一种具体的实现方式中, 对于层间最近邻查找, 金字塔结构如图 24 所示。图 25 为一种层间最近邻查找的 LOD 构造过程示意图。如图 25 所示, 基于几何信息划分得到不同的 LOD 层, 得到 LOD0、LOD1 和 LOD2, 利用 LOD0 中的点去预测下一层 LOD 中点的属性在层间最近邻查找的过程中。

20 下面将对帧内最近邻查找的整个过程进行详细地介绍。

在整个 LOD 的划分过程中, 存在三个集合 $O(k)$ 、 $L(k)$ 以及 $I(k)$ 。其中, k 为 LOD 划分时 LOD 层的索引, $I(k)$ 为当前 LOD 层划分时的输入点集, 经过 LOD 划分, 得到 $O(k)$ 集合以及 $L(k)$ 集合, $O(k)$ 集合存储的是采样点集, $L(k)$ 为当前 LOD 层中的点集。即整个 LOD 划分的过程如下:

(1) 初始化。

25 if $k=0$, $L(k) \leftarrow \{\}$; 否则, $L(k) \leftarrow L(k-1)$;

$O(k) \leftarrow \{\}$;

(2) 利用 LOD 划分算法, 将采样点存入 $O(k)$, 其余的点划分到 $L(k)$;

(3) 进行下一次迭代时 $I \leftarrow O(k)$ 。

30 这里需要注意的是, 由于整个 LOD 划分的过程是基于莫顿码进行划分的, 因此 $O(k)$ 、 $L(k)$ 以及 $I(k)$ 存储的是点对应的莫顿码索引。

在进行层间最近邻查找时, 即 $L(k)$ 集合中的点在 $O(k)$ 集合中进行最近邻查找, 具体的查找算法如下:

以基于空间关系进行最近邻查找为例, 在对当前点 P 进行预测时, 通过利用点 P 对应的父块 (Block B) 进行邻居搜索, 如图 26 所示, 搜索与当前父块共面、共线邻居块内的点来进行属性预测。

35 其中, 图 27A 示出了一种共面的空间关系示意图, 这里共有 6 个与当前父块具有关系的空间块。图 27B 示出了一种共面和共线的空间关系示意图, 这里共有 18 个与当前父块具有关系的空间块。图 27C 示出了一种共面、共线和共点的空间关系示意图, 这里共有 26 个与当前父块具有关系的空间块。

首先, 利用当前点的坐标得到对应的空间块, 其次在之前已编码的 LOD 层中进行最近邻查找, 查找与当前块共面、共线和共点的空间块来得到当前点的 N 近邻。

40 当进行共面、共线和共点最近邻查找之后, 仍然没有得到当前点的 N 近邻, 则会基于快速查找算法来得到当前点的 N 近邻, 具体算法如下:

如图 28 所示, 当进行属性层间预测时, 首先利用当前待编码点的几何坐标得到当前点所对应的莫顿码, 其次基于当前点的莫顿码在参考帧中查找到第一个大于当前点莫顿码的参考点(j), 其次在 $[j-\text{searchRange}, j+\text{searchRange}]$ 范围内进行最近邻查找。

45 其余具体的更新最近邻的算法和帧间最近邻查找算法一致, 在这里不在详述, 具体的算法会在帧间最近邻查找算法中提到。

可以理解的是, 在本申请实施例中, 一个视频帧可以理解为一幅图像。示例性地, 当前帧可以理解为当前图像, 参考帧可以理解为参考图像。

50 在另一种具体的实现方式中, 对于层内最近邻查找, 图 29 示出了一种属性层内最近邻查找的 LOD 结构示意图。如图 29 所示, 如果层内预测算法开启, 即语法元素 `EnableRefferingSameLoD=1`, 那么可以允许在层内最近邻查找, 如对于 LOD1 层, 当前点 P_6 的最近邻点可以为 P_1 , 其他层不允许; 如果语法元素 `EnableRefferingSameLoD=0`, 那么允许在其他层进行层间查找, 如对于 LOD1 层, 当前点 P_6 的

最近邻点可以为 P4。也就是说，当层内预测算法开启时，会在同一层 LOD 内，在同层已编码的点集中进行最近邻查找，得到当前点的 N 近邻（同样进行层间最近邻查找）。

在进行属性层内预测时，会基于快速查找算法进行最近邻查找，具体的算法如图 30 所示。其中，当前点用网格表示，假设当前点的莫顿码索引为 i，则会在 $[i+1, i+searchRange]$ 进行最近邻查找。具体的最近邻查找算法与帧间基于块的快速查找算法一致，在这里不再详述。

(ii) 帧间最近邻查找：

图 28 为一种属性帧间预测示意图。如图 28 所示，当进行属性帧间预测时，首先利用当前待编码点的几何坐标得到当前点所对应的莫顿码，其次基于当前点的莫顿码在参考帧中查找到第一个大于当前点莫顿码的参考点(j)，其次在 $[j-searchRange, j+searchRange]$ 范围内进行最近邻查找。

目前的帧内和帧间进行最近邻查找时，是基于块进行邻域查找的，具体的参见图 31。如图 31 所示，在对当前点（莫顿码索引为 i）进行邻域查找时，首先将参考帧中的点按照莫顿码划分成 N（N=3）个层，具体的划分算法如下：

第一层：将假设参考帧的点为 numPoints，首先将参考帧中的点每 $M(M=2^5=32)$ 个点划分到一个块中；

第二层：在第一层的基础上，同样按照莫顿码的顺序对第一层的块每 $M(M=2^5=32)$ 个块划分到一个块中；

第三层：在第二层的基础上，同样按照莫顿码的顺序对第二层的块每 $M(M=2^5=32)$ 个块划分到一个块中；

最终得到如图 31 所示的预测结构。

在基于如图 31 所示的预测结构来进行属性预测，假设当前待编码点的莫顿码索引为 i，首先在参考帧中得到第一个大于等于当前点莫顿码的点，索引为 j。其次基于 j 计算得到参考点的块索引，具体计算方式如下：

第一层： $BucketSize_0=2^5=32$;

第二层： $BucketSize_1=2^5=32 \times BucketSize_0=1024$;

第三层： $BucketSize_2=2^5=32 \times BucketSize_1=32768$ 。

假设当前点的预测帧中的参考范围为 $[j-searchRange, j+searchRange]$ ，利用 j-searchRange 计算得到第三层的起始索引，j+searchRange 计算得到第三层的终止索引；其次，首先在第三层的块中判断第二层的一些块是否需要最近邻查找，其次到第二层，对于第一层中的每个块判断是否需要最近邻查找，如果第一层的某些块需要进行最近邻查找，则会对第一层中的一些块中点进行逐点判断来更新最近邻。

下面介绍一下，基于索引计算块的算法，假设当前点对应的莫顿码索引为 index，那么对应的第三层块的索引为：

$$idx_2 = index / BucketSize_2 \quad (24)$$

在得到第三层的块索引 idx_2 之后，可以利用 idx_2 得到当前块在第二层对应的块的起始索引和终止索引：

$$startIdx1 = idx_2 \times BucketSize_1 \quad (25)$$

$$endIdx = idx_2 \times BucketSize_1 + BucketSize_1 - 1 \quad (26)$$

同样，基于同样的算法基于第二层块的索引得到第一层块的索引。

在基于块进行最近邻查找时，会首先判断当前块是否需要最近邻查找，也就是筛选块的最近邻查找。每个空间块可以通过两个变量进行得到 minPos 和 maxPos，minPos 表示的是块的最小值，maxPos 表示的是块的最大值。

假设当前点查找的 N 近邻中最远点的距离为 Dist，待编码点的坐标为 (x,y,z)，当前块表示为 (minPos,maxPos)，其中 minPos 为包围盒三个维度上的最小值，maxPos 为包围盒三个维度上的最大值，则当前点与包围盒之间的距离 D 计算如下：

$int\ dx = int(std::max(std::max(minPos[0] - point[0], 0), point[0] - maxPos[0]));$

$int\ dy = int(std::max(std::max(minPos[1] - point[1], 0), point[1] - maxPos[1]));$

$int\ dz = int(std::max(std::max(minPos[2] - point[2], 0), point[2] - maxPos[2]));$

$D = dx + dy + dz;$

当 D 小于等于 Dist，才会去遍历当前块中的点。

(b) 点云属性信息的提升变换编码。

图 32 为一种提升变换的编码流程示意图。提升变换同样是基于 LOD 对点云属性进行预测编码。与预测变换的不同之处在于，提升变换首先会对 LOD 进行高低层的划分，按照 LOD 生成层的逆序进行预测，并且在预测的过程中引入了更新算子来对低层 LOD 中点的量化权重进行更新，以提高预测的

准确性。这是由于低层 LOD 中点的属性值会频繁的用于高层 LOD 中点的属性值预测，低层 LOD 中的点应具有更大的影响力。

步骤 1: 分割过程。

分割过程是将完整的 LOD 层分为低 LOD 层 $L(N)$ 和高 LOD 层 $H(N)$ 。如果某点云有三层 LOD，即 $(LOD_l)_{l=0,1,2}$ ，经过分割后， LOD_2 为高 LOD 层，记为 $H(N)$ ， $(LOD_l)_{l=0,1}$ 为低 LOD 层，记为 $L(N)$ 。

步骤 2: 预测过程。

高层 LOD 中的点从低层中选取最近邻点的属性信息作为当前待编码点的属性预测值 $P(N)$ ，预测残差 $D(N)$ 记为：

$$D(N) = H(N) - P(N) \quad (27)$$

步骤 3: 更新过程。

对高层 LOD 中的属性预测残差 $D(N)$ 进行更新，得到 $U(N)$ ，并利用 $U(N)$ 对低层 LOD 中点的属性值进行提升，如下式所示：

$$L'(N) = L(N) + U(N) \quad (28)$$

上述过程将依据 LOD 从高到低的顺序，不断迭代直至最低层 LOD。

由于基于 LOD 的预测方案使得 LOD 低层中的点具有更大的影响力，基于提升小波变换的变换方案通过引入量化权重，并且根据预测残差 $D(N)$ 以及预测点和相邻点之间的距离来更新预测残差，最后利用变换过程中的量化权重来对预测残差进行自适应量化。这里需要注意的是，在解码端可以通过几何重构来确定每个点的量化权重值，因此不要对量化权重进行编码。

(c) 区域自适应分层变换。

区域自适应分层变换 (RAHT) 是一种哈尔小波变换，它可以将点云属性信息从空域变换到频域，进一步减少点云属性之间的相关性。其主要思想是按照八叉树结构，采用自底向上的方式对每一层中的节点分别从 X、Y、Z 三个维度进行变换 (如图 34)，并迭代直至八叉树的根节点。如图 33 所示，其基本思想是基于八叉树的层级结构进行小波变换，将属性信息与八叉树节点相关联，对于同一父节点中被占据节点的属性沿着自底向上的方式进行递归变换，对于每一层中的节点分别从 X、Y、Z 三个维度进行变换，直至变换至八叉树的根节点。在分层变换的过程中，将同层节点变换之后得到的低通/低频 (DC) 系数传递到下一层的节点继续进行变换，而所有的高通/高频 (AC) 系数可以通过算术编码器进行编码。

在变换过程中，同一层节点变换之后的 DC 系数 (直流分量) 将传递到上一层继续变换，而每一层变换后的 AC 系数 (交流分量) 将进行量化编码。下面将介绍主要的变换过程。

图 35A 为一种 RAHT 正变换的过程示意图，图 35B 为一种 RAHT 逆变换的过程示意图。针对 RAHT 对应的变换与逆变换过程，假设 $g'_{L,2x,y,z}$ 和 $g'_{L,2x+1,y,z}$ 为 L 层中互为近邻点的两个属性 DC 系数。经过线性变换后，L-1 层的信息为 AC 系数 $f'_{L-1,x,y,z}$ 和 DC 系数 $g'_{L-1,x,y,z}$ ；然后， $f'_{L-1,x,y,z}$ 将不再进行变换，直接进行量化编码， $g'_{L-1,x,y,z}$ 将继续寻找近邻点进行变换，如果寻找不到，则将其直接传递至 L-2 层，即 RAHT 变换仅对存在邻居点的节点有效，没有邻居点的节点将直接传递至上一层。在上述变换过程中， $g'_{L,2x,y,z}$ 和 $g'_{L,2x+2,y,z}$ 对应的权重 (该节点内非空子节点的个数) 分别为 $w'_{L,2x,y,z}$ 和 $w'_{L,2x+1,y,z}$ (简称为 w'_0 和 w'_1)， $g'_{L-1,x,y,z}$ 的权重为 $w'_{L-1,x,y,z}$ ，则通用变换公式为：

$$\begin{bmatrix} g'_{L-1,x,y,z} \\ f'_{L-1,x,y,z} \end{bmatrix} = T_{w_0,w_1} \begin{bmatrix} g'_{L,2x,y,z} \\ g'_{L,2x+1,y,z} \end{bmatrix} \quad (29)$$

其中， T_{w_0,w_1} 为变换矩阵：

$$T_{w_0,w_1} = \frac{1}{\sqrt{w'_0 + w'_1}} \begin{bmatrix} \sqrt{w'_0} & \sqrt{w'_1} \\ -\sqrt{w'_1} & \sqrt{w'_0} \end{bmatrix} \quad (30)$$

变换矩阵会随着各点对应的权重自适应变化更新。上述过程会依据八叉树的划分结构不断迭代更新，直至八叉树的根节点。

在一种具体的实现方式中，针对区域自适应分层帧内预测变换编码，可以基于 RAHT 变换编码的基础上进行预测。如图 33 所示，RAHT 属性变换基于八叉树层级的顺序，由体素级别不断进行变换直至得到根节点，从而完成整个属性的分层变换编码。在预测变换编码中，同样基于八叉树的层级顺序进行属性预测变换编码，但是是从根节点不断进行变换直至到体素级别。在每一次 RAHT 属性变换的过程中，是基于 $2 \times 2 \times 2$ 的块进行属性预测变换编码。具体的如图 36 所示。如图 36 所示，可以看到网格填充块为当前待编码块，斜线填充块为与当前待编码块共面和共线的一些邻域块。其中，当前块的属性通过如下方式进行归一化处理：

$$A_{node} = \sum_{p \in node} \text{attribute}(p);$$

$$w_{node} = \sum_{p \in node} 1 = \{p \in node\};$$

$$a_{node} = A_{node} / w_{node}.$$

首先, 可以通过当前块中包含点的属性得到当前块的属性, 即: A_{node} 。通过对当前块中包含点属性进行简单的相加, 其次利用当前块的属性与当前块中点的个数进行归一化处理得到当前块属性的均值 a_{node} 。利用当前块属性的均值进行属性变换编码。具体编码过程参见图 37。

5 如图 37 所示, 这里示出了 RAHT 属性预测变换编码的整体流程。其中, (a) 为当前块以及共面和共线的一些邻域块, (b) 为经过归一化处理后的块, (c) 为经过上采样后的块, (d) 为当前块的属性, (e) 为通过利用当前块的邻域属性进行线性加权拟合得到预测块的属性, 最终将对两者分别进行属性变换, 得到 DC 和 AC 系数, 对 AC 系数进行预测编码。

10 其中, 当前块的预测属性可以通过利用如图 38 所示进行线性拟合得到。如图 38 所示, 首先得到当前块的 19 个邻域块, 其次利用邻域块与当前块的每个子块之间的空间几何距离对每个子块的属性进行线性加权预测, 最终利用线性加权得到的预测块属性进行变换。具体的属性变换如图 39 所示。

在图 39 中, (d) 表示属性原始值, 对应的属性变换系数如下:

$$\begin{bmatrix} AC_{1,orig}^* \\ \vdots \\ AC_{k-1,orig} \end{bmatrix} = T_{node} \begin{bmatrix} A_{1,orig}/\sqrt{w_1} \\ \vdots \\ A_{k,orig}/\sqrt{w_k} \end{bmatrix} \quad (31)$$

(e) 表示属性预测值, 对应的属性变换系数如下:

$$15 \quad \begin{bmatrix} AC_{1,up}^* \\ \vdots \\ AC_{k-1,up} \end{bmatrix} = T_{node} \begin{bmatrix} A_{1,up}/\sqrt{w_1} \\ \vdots \\ A_{k,up}/\sqrt{w_k} \end{bmatrix} \quad (32)$$

根据属性原始值与属性预测值进行减法运算, 可以得到预测残差如下:

$$\begin{bmatrix} DC_{depth\ d-1} \\ AC_{1,res} \\ \vdots \\ AC_{k-1,res} \end{bmatrix} = \begin{bmatrix} DC_{depth\ d-1} \\ AC_{1,orig} \\ \vdots \\ AC_{k-1,orig} \end{bmatrix} - \begin{bmatrix} 0 \\ AC_{1,up} \\ \vdots \\ AC_{k-1,up} \end{bmatrix} \quad (33)$$

20 在另一种具体的实现方式中, 针对区域自适应分层帧间预测变换编码, 在 G-PCC 属性帧间预测编码方案一中, 类似帧内预测编码的过程。首先, 基于几何信息构建 RAHT 属性变换编码结构, 即: 由体素级别不断进行变换直至得到根节点, 从而完成整个属性的分层变换编码。按照这样的方式, 构建得到帧内编码结构和帧间编码结构。其中, RAHT 属性的帧间编码结构可以参见图 40。

如图 40 所示, 首先利用当前待编码节点的几何信息在参考帧中得到待编码节点的同位预测节点, 其次利用参考节点的几何信息和属性信息得到当前待编码节点的预测属性。

其中, 根据如下两种不同的方式得到当前待编码节点的属性预测值:

25 ①当前节点的帧间预测节点有效: 即同位节点存在, 则将预测节点的属性直接作为当前待编码节点的属性预测值;

②当前节点的帧间预测节点无效: 即同位节点不存在, 则利用帧内相邻节点的属性预测值作为待编码节点的属性预测值。

最终, 利用得到的属性预测值来对当前待编码节点的属性进行预测。从而完成整个属性的预测编码。

30 在又一种具体的实现方式中, 针对区域自适应分层帧间预测变换编码, 在 G-PCC 属性帧间预测编码方案二中, 与帧内预测编码以及帧间预测编码方案一不同的是, 如果启动帧间预测编码方案二, 那么首先会基于当前待编码节点的几何信息构建 RAHT 属性变换编码结构, 即由体素级别不断进行节点合并, 直至得到整个 RAHT 变换树的根节点, 从而完成整个属性的变换编码分层结构。其次, 在根据 RAHT 变换结构, 由根节点进行划分, 得到每个节点的 N 个子节点 (N 小于等于 8), 在帧间预测编码方案二中, 会首先利用 RAHT 变换对 N 个子节点的属性进行独立正交变换, 得到 DC 系数和 AC 系数, 其次按照以下方式对 N 个子节点的 AC 系数进行属性帧间预测:

①当前节点的帧间预测节点有效: 即同位节点存在, 则将预测节点的属性直接作为当前待编码节点的属性预测值

40 ②当前节点可以在参考帧的缓存中查找到与当前节点位置完全相同的节点: 即同位节点存在, 则将同位节点中包含的 M 个子节点的 AC 系数直接作为当前节点 N 个子节点的 AC 系数属性预测值。

如果预测节点的 AC 系数不为零: 则将预测节点的 AC 系数直接作为预测值;

如果预测节点的 AC 系数为零, 则会将帧内预测对应子节点的 AC 系数作为预测值。

③当前节点的帧间预测节点无效: 即同位节点不存在, 则利用帧内相邻节点的属性预测值作为待编码节点的属性预测值。

简单来讲,在 G-PCC 的 RAHT 属性变换编码时,可以按照由根节点到子节点顺序进行编解码。首先,利用当前层节点的几何信息,依次按照 Z、Y 和 X 的顺序进行恢复得到当前层的子节点,其次,利用父节点层已经重建的属性来对当前层节点的属性进行预测解码,从而恢复得到当前层节点的属性,直至变换到子节点,即体素级别。但是在 G-PCC 中,如果当前层节点的数目与当前层节点子节点的数目完全一致时,此时表明当前层中的每个节点仅仅只有一个子节点,即当前层不会产生 AC 系数,但是在已有的编码方案中,仍然需要对当前层的节点依次进行变换、预测等过程,这样会导致 RAHT 属性变换编解码的复杂度增大,引入了冗余操作。同样的,在已有的 RAHT 编码方案中,编解码端首先完成非节点层属性信息的编码以及解码(即节点的大小大于等于 $1 \times 1 \times 1$),最终在对体素级别节点的属性信息进行编码和解码,原因是由于点云中会存在一种情况,即点云中存在重复点,因此需要先完成非体素级别点的属性信息编码和解码,其次在完成体素级别点属性信息的编码和解码。由于在当前编码单元中不存在重复点时,仍然需要对体素级别点属性信息进行编码及解码过程,进一步提升了属性变换编解码/解码的时间复杂度。

基于此,本申请实施例提供了一种编码方法,首先确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量;然后根据第一数量与第二数量,确定当前单元的体素节点的属性信息是否跳过编码。本申请实施例提供了还一种解码方法,首先确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量;其中,第一数量与第二数量用于确定是否对当前单元的体素节点进行跳过解码;然后根据第一数量与第二数量,确定当前单元的体素节点的属性重建值。

如此,在对每个体素节点进行属性重建时,优化了每个体素节点是否进行属性编解码的判断条件,具体为如果当前单元不存在重复节点,即第一数量与第二数量相同,此时不需要编解码当前单元的体素节点,从而在保证点云属性的编解码效率基础上,可以降低点云属性编解码的时间复杂度,并且还可以节省码率,进而提升了点云的编解码性能。

下面将结合附图对本申请各实施例进行详细说明。

在本申请的一实施例中,参见图 41,其示出了本申请实施例提供的一种解码方法的流程示意图。

如图 41 所示,该方法可以包括:

S4101: 确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量;其中,第一数量与第二数量用于确定是否对当前单元的体素节点进行跳过解码。

需要说明的是,本申请实施例的解码方法应用于点云解码器(可简称为“解码器”)。其中,该方法可以是指点云解码方法,具体是一种点云属性解码方法,更具体地,可以是一种点云属性 RAHT 变换预测的跳过解码方法。在这里,主要是针对当前层的节点是否跳过解码以及在划分到体素级别时的体素节点是否跳过解码进行优化,从而可以降低属性变换编解码的时间复杂度,并且对属性变换的编解码效率不会产生任何影响。

还需要说明的是,在本申请实施例中,当前单元可以是当前待解码的当前解码单元,这里可以是一个片(slice)。在一些实施例中,该方法还可以包括:对当前单元中的节点进行划分,确定至少一层;在最后一层的节点被划分到体素级别时,确定当前单元的体素节点以及体素节点的第一数量。

示例性地,在 RAHT 属性变换中,RAHT 属性变换的顺序是从根节点依次进行划分,直至划分到体素级别,具体为划分到 $1 \times 1 \times 1$ 大小的单位立方体时停止划分,从而完成整个点云属性的编码和重建。在这里,如图 42 所示,每次沿着 Z 方向、Y 方向和 X 方向做一次下采样所得到的层即为一个 RAHT 变换层,即 layer。然后直至划分到 $1 \times 1 \times 1$ 大小的单位立方体时表示已经划分到体素级别,这时候可以确定体素节点以及体素节点的第一数量。

还需要说明的是,在本申请实施例中,体素节点表示由根节点划分到体素级别时对应的节点,体素节点的大小为 $1 \times 1 \times 1$;重建节点表示当前单元中进行属性重建的节点。其中,在当前单元中的节点进行属性解码之前,当前单元中的节点的几何信息已经全部解码完成,这时候可以确定当前单元的重建节点的第二数量,从而能够判断第一数量与第二数量是否一致。也就是说,在本申请实施例中,在对体素节点进行属性解码之前,首先需要判断第一数量与第二数量是否相同,进而来确定当前单元的体素节点是否进行跳过解码。

在一些实施例中,若第一数量与第二数量相同,则跳过解码当前单元的体素节点。

在一些实施例中,若第一数量与第二数量不同,则对体素节点中的重复节点进行属性解码,以及跳过解码体素节点中除重复节点之外的剩余体素节点。

在本申请实施例中,当前单元划分的体素节点中可能会存在重复节点,因此第一数量与第二数量有可能不一致。其中,如果当前单元划分的体素节点中不存在重复节点,那么第一数量与第二数量一致;反之,如果当前单元划分的体素节点中存在重复节点,那么第一数量与第二数量不一致,此时需要针对

重复节点进行属性解码。在这里，重复节点（也可简称为“重复点”）是指几何信息一样、但属性信息不同的多个节点。

S4102: 根据第一数量与第二数量，确定当前单元的体素节点的属性重建值。

需要说明的是，在本申请实施例中，对于当前单元的体素节点的属性重建值，可以是直接复制当前单元的重建节点的属性重建值，但是其中的重复节点的属性重建值则无法复制得到，此时需要通过解码码流来确定。也就是说，这里可以是由第一数量与第二数量之间的大小来决定如何确定当前单元的体素节点的属性重建值。

在一些实施例中，在第一数量与第二数量相同时，根据第一数量与第二数量，确定当前单元的体素节点的属性重建值，可以包括：将当前单元的体素节点的属性重建值设置为当前单元的重建节点的属性重建值。

示例性地，如果第一数量与第二数量相同，那么可以跳过解码当前单元的体素节点的属性信息，这时候可以将体素节点的属性重建值直接复制为当前单元的重建节点的属性重建值。

在一些实施例中，在第一数量与第二数量不同时，根据第一数量与第二数量，确定当前单元的体素节点的属性重建值，可以包括：解码码流，确定重复节点的属性重建值；将重复节点的属性重建值作为当前单元中第一重建节点的属性重建值，以及将体素节点中除重复节点之外的剩余体素节点的属性重建值设置为当前单元中除第一重建节点之外的剩余重建节点的属性重建值。

示例性地，如果第一数量与第二数量不同，表明当前单元划分的体素节点中存在有重复节点，这时候可以解码确定重复节点的属性重建值，然后将重复节点的属性重建值作为当前单元中第一重建节点的属性重建值，以及将重复节点之外的剩余体素节点的属性重建值复制为当前单元中除第一重建节点之外的剩余重建节点的属性重建值。

示例性地，对于几何信息相同、属性信息不同的多个节点（即重复节点），编码端可以采用计时器来实现把这多个节点的属性信息进行相加，将相加值确定为该几何信息对应节点的属性信息；后续在解码端，同样可以采用计时器来实现从这个属性信息中将这多个节点各自的属性信息分别剥离出来，以得到重复节点的属性重建值。

在一些实施例中，解码码流，确定重复节点的属性重建值，可以包括：设置计时器；在重复节点的属性重建值开始解码时，启动计时器，并在计时器的计时达到预设数值时，确定重复节点的属性重建值全部解码完成。

需要说明的是，在第一数量与第二数量不同时，还可以根据第一数量与第二数量之间的差值，确定体素节点中的重复节点的第三数量；其中，计时器的设置与第三数量有关联关系。

示例性地，可以采用一个计时器来确定重复节点的属性重建值是否全部解码完成。如果计时器处于正计数模式，那么在重复节点的属性重建值开始解码时，计时器的初始值为0，在计时器计数到第三数量时，此时重复节点的属性重建值全部解码完成；如果计时器处于倒计数模式，那么在重复节点的属性重建值开始解码时，计时器的初始值为第三数量，在计时器计数到0时，此时重复节点的属性重建值全部解码完成。在重复节点的属性重建值全部解码完成之后，意味着后续的体素节点中不存在重复节点，可以跳过这些节点的属性解码，并且将后续剩余体素节点的属性重建值直接复制为最终剩余重建节点的属性重建值。

可以理解地，在本申请实施例中，至少一层包括当前层，其中，当前层可以为当前待解码的 RAHT 变换层，或称为“RAHT 属性解码层”。图 43 为本申请实施例提供的另一种解码方法的流程示意图。如图 43 所示，该方法可以包括：

S4301: 确定当前层的节点的第四数量和当前层的节点对应的子节点的第五数量；其中，第四数量和第五数量用于确定是否对当前层进行跳过解码。

需要说明的是，在本申请实施例中，可以先确定当前层的节点的第四数量，同时可以确定当前层的节点对应的子节点的第五数量。其中，在当前层的节点进行属性解码之前，由于当前层的节点的几何信息已经解码完成，根据当前层的节点的几何信息可以确定出当前层的节点数目（即“第四数量”）和当前层的节点的子节点数目（即“第五数量”）。

进一步地，在本申请实施例中，需要先基于点云中的点的几何信息来构建 RAHT 属性变换结构，可以按照由根节点到子节点的顺序进行解码。利用当前层节点的几何信息，依次按照 Z、Y 和 X 的顺序进行恢复得到当前层的子节点，其次利用上一层的节点已经重建的属性来对当前层节点的属性进行预测解码，从而恢复得到当前层节点的属性重建值，直至变换到体素级别，进而可以得到包括有至少一个 RAHT 变换层的 RAHT 属性变换结构。

需要说明的是，在本申请实施例中，RAHT 属性变换可以是基于八叉树层级的顺序执行的。其中，基于八叉树的层级顺序，可以由体素级别不断进行变换直至到根节点，从而构建出八叉树。然后在预测

变换过程中,同样基于八叉树的层级顺序进行属性预测变换编码,但是是由根节点不断进行变换直至到体素级别。

可以理解的是,在本申请实施例中,可以定义每次沿着预设方向,如Z方向、Y方向和X方向依次做一次下采样得到的层即为一个RAHT变换层,如当前层(layer)。

5 还需要说明的是,在本申请实施例中,对于当前层来说,当前层中可以包括至少一个点。其中,对于当前层中的至少一个点,在对当前层进行解码时,其可以作为当前层中的待解码节点。

进一步地,在本申请实施例中,对于当前层中的每一个点,其对应一个几何信息和一个属性信息;其中,几何信息表征该点的空间关系,属性信息表征该点的属性的相关信息。

10 在这里,属性信息可以为颜色信息,也可以是反射率或者其它属性,本申请实施例不作具体限定。其中,当属性信息为颜色信息时,具体可以为任意颜色空间的属性信息。示例性地,属性信息可以为RGB空间的属性信息,也可以为YUV空间的属性信息,还可以为YCbCr空间的属性信息等等,本申请实施例也不作具体限定。

15 还需要说明的是,在本申请实施例中,在进行RAHT变换编码时,首先是完成非体素级别的节点属性变换以及逆变换,其次完成体素级别的节点变换,因为点云中会存在一种情况,即点云中存在重复节点的现象。

相应地,在本申请实施例中,如果当前层的节点为非体素级别时,那么第四数量可以表征当前层的被占据节点的数量;第五数量则可以表征当前层的节点中被占据的子节点的数量。

相应地,在本申请实施例中,如果当前层的节点为体素级别时,那么第四数量可以表征当前层的被占据节点的数量;第五数量则可以表征待解码的节点的数量。

20 也就是说,在本申请实施例中,第四数量即为当前层的有效节点(即被占据的节点)的数量,而对于非体素级别的节点,第五数量为当前层的节点的有效子节点(即被占据的子节点)的数量,对于体素级别的节点,第五数量为待解码的节点的数量。

进一步地,在本申请实施例中,可以先确定当前层的节点的几何信息;然后再根据几何信息确定当前层的节点对应的子节点,以及第五数量。

25 需要说明的是,在本申请实施例中,对于当前层的当前节点来说,在利用当前节点的几何信息进行对应的子节点的确定时,可以选择利用当前节点的几何信息进行上采样,得到当前节点被占据的子节点(子节点个数为N,其中N的取值最大为8)。

30 示例性地,在一些实施例中,在对当前层的节点的属性信息进行编解码时,可以先得到当前层的节点的数量,即第四数量;同时,在利用当前层的节点的几何信息进行恢复得到当前层的节点的子节点之后,可以得到当前层的节点的子节点的数量,即第五数量。

S4302: 根据第四数量和第五数量,确定当前层的节点对应的子节点的属性重建值。

需要说明的是,在本申请实施例中,在确定当前层的节点对应的第四数量和第五数量之后,可以进一步根据第四数量和第五数量,确定当前层的节点对应的子节点的属性重建值。

35 进一步地,在本申请实施例中,在确定当前层的节点对应的第四数量和第五数量之后,可以利用第四数量和第五数量进行是否跳过编码层(当前层)的属性信息的编解码。

可以理解的是,在本申请实施例中,由于RAHT变换仅对存在邻居点的节点有效,如果当前层的节点的数目与当前层的节点的子节点的数目完全一致时,则可以表明当前层中的每个节点仅仅只有一个子节点;在这种情况下,当前层不会产生AC系数(高频系数),因此可以选择跳过对当前层的节点所依次进行的变换、预测等过程。

40 也就是说,在本申请实施例中,通过利用当前层的节点数目与子节点数目,可以自适应地决定当前层是否可以跳过编解码。其中,判定是否跳过对当前层的编解码处理的关键,在于当前层的节点数目与子节点数目是否相同,即第四数量和第五数量是否相同。

45 在一些实施例中,在根据第四数量和第五数量,确定当前层的节点对应的子节点的属性重建值时,该方法还可以包括:若第四数量和第五数量相同,则将当前层的节点的属性重建值确定为当前层的节点对应的子节点的属性重建值。

可以理解的是,在本申请实施例中,如果当前层的节点对应的第四数量和第五数量相同,即可以确定当前层的节点的数量与当前层的节点所对应的子节点的数量是相同的,那么,便可以认为对于当前层的每一个节点来说,均只对应有一个子节点。

50 相应地,在本申请实施例中,由于RAHT变换仅对存在邻居点的节点有效,在RAHT属性变换的过程中,如果当前层的每一个节点均仅对应有一个子节点,那么可以认为当前层不会产生AC系数,因此可以选择不对当前层的节点依次进行变换、预测等过程,即跳过对当前层的处理,此时,可以称为“跳过解码层”。

相应地,在本申请实施例中,如果确定当前层的节点对应的第四数量和第五数量相同,即确定当前层为跳过解码层,那么可以选择跳过对当前层的节点依次进行变换、预测等过程,而是可以直接将当前层的节点的属性重建值确定为当前层的节点对应的子节点的属性重建值。

也就是说,在本申请实施例中,在 RAHT 属性变换的过程中,如果当前层的每一个节点均仅对应有一个子节点,那么可以选择不再对当前层的节点依次进行变换、预测等过程,从而可以降低 RAHT 属性变换编解码的复杂度。

进一步地,在本申请实施例中,在根据第四数量和第五数量,确定当前层的节点对应的子节点的属性重建值时,若第四数量和第五数量相同,确定当前层为跳过解码层,那么可以选择跳过当前层至下一层,然后将下一层作为当前层,继续判断是否对下一层的节点进行跳过解码。

相应地,在本申请实施例中,对于当前层的节点对应的子节点来说,可以先确定子节点对应的下一层子节点的第六数量;然后根据第五数量和第六数量,确定子节点对应的下一层子节点的属性重建值。

可以理解的是,在本申请实施例中,如果当前层的节点对应的子节点为非体素级别的节点,那么第六数量可以为当前层的节点对应的子节点的下一层有效子节点(即下一层被占据的子节点)的数量,如果当前层的节点对应的子节点为体素级别的节点,那么第六数量可以为待解码的节点的数量。

需要说明的是,在本申请实施例中,在确定第五数量和第六数量之后,可以利用第五数量和第六数量进行是否跳过编码层(当前层的下一层)的属性信息的编解码。

示例性地,在一些实施例中,如果第五数量和第六数量相同,那么可以选择不对当前层的节点的子节点依次进行变换、预测等过程,即跳过对当前层的子节点的处理,而是可以直接将当前层的节点对应的子节点的属性重建值确定为当前层的节点对应的子节点的下一层子节点的属性重建值。

在一些实施例中,在根据第四数量和第五数量,确定当前层的节点对应的子节点的属性重建值时,该方法还可以包括:若当前层的节点对应的第四数量和第五数量不同,则根据当前层的节点确定当前层的节点对应的子节点的属性预测值;基于子节点的属性预测值进行 RAHT 变换,确定当前层的节点对应的高频系数的重建值和低频系数;基于高频系数的重建值和低频系数进行 RAHT 逆变换,确定子节点的属性重建值。

需要说明的是,在本申请实施例中,如果第四数量和第五数量不相同,即可以确定当前层的节点的数量与当前层的节点所对应的子节点的数量是不同的,此时,当前层依然会产生 AC 系数,因此可以继续对当前层的节点依次进行变换、预测等过程,而不会跳过对当前层的处理,此时,可以将当前层作为非跳过解码层。

进一步地,在一些实施例中,在根据当前层的节点确定当前层的节点对应的子节点的属性预测值时,可以包括:确定当前层的节点对应的相邻节点;根据相邻节点对应的属性重建值和相对距离参数,确定当前层的节点对应的子节点的属性预测值。

还需要说明的是,在本申请实施例中,相邻节点可以是指当前节点的邻域节点。其中,相邻节点对应的相对距离参数可以表征当前层的节点对应的子节点与对应的相邻节点的之间的空间几何距离。

示例性地,在本申请实施例中,对于当前层的当前节点,该当前节点包括子节点 1 和子节点 2 这两个子节点,当前节点与相邻节点之间的相对距离参数可以包括子节点 1 与相邻节点的之间的空间几何距离,还可以包括子节点 2 与相邻节点的之间的空间几何距离。

示例性地,在本申请实施例中,在根据当前层的节点确定当前层的节点对应的子节点的属性预测值时,对于当前层的当前节点,可以利用当前节点的邻域节点的重建属性(属性重建值)以及每个邻域节点距离当前节点的子节点的空间几何距离进行线性拟合,最终得到当前节点的每个子节点的属性预测值。

示例性地,在本申请实施例中,对于当前层的当前节点来说,可以先确定当前节点的 19 个相邻节点,其次利用相邻节点与当前节点的每个子节点之间的空间几何距离对每个子节点的属性进行线性加权预测,最终得到每个子节点的属性预测值。

进一步地,在一些实施例中,在基于子节点的属性预测值进行 RAHT 变换,确定当前层的节点对应的高频系数的重建值和低频系数时,可以包括:基于子节点的属性预测值进行 RAHT 变换,确定当前层的节点对应的高频系数的预测值和低频系数;根据高频系数的预测值确定当前层的节点对应的高频系数的重建值。

可以理解的是,在本申请实施例中,对于当前层的当前节点来说,在确定出当前节点的子节点对应的属性预测值之后,可以利用对应的子节点的属性预测值进行 RAHT 属性变换,从而可以得到相应的 DC 系数和 AC 系数,即得到当前节点相应的 DC 系数和 AC 系数。其中,DC 系数即为低频系数,AC 系数即为高频系数。

需要说明的是,在本申请实施例中,对于当前层的当前节点来说,利用子节点对应的属性预测值进行 RAHT 属性变换所获得的 AC 系数可以理解当前节点对应的高频系数的预测值。

进一步地, 在一些实施例, 在根据高频系数的预测值确定当前层的节点对应的高频系数的重建值时, 可以包括: 解码码流, 确定当前层的节点对应的量化后的系数残差; 根据高频系数的预测值和量化后的系数残差, 确定当前层的节点对应的高频系数的重建值。

进一步地, 在一些实施例, 在根据高频系数的预测值和量化后的系数残差, 确定当前层的节点对应的高频系数的重建值时, 可以包括: 对量化后的系数残差进行反量化, 确定当前层的节点对应的反量化残差值; 根据当前层的节点对应的反量化残差值和当前层的节点对应的高频系数的预测值, 确定当前层的节点对应的高频系数的重建值。

示例性的, 在本申请实施例中, 可以对当前层的节点对应的反量化残差值和当前层的节点对应的高频系数的预测值进行求和计算, 进而可以获得当前层的节点对应的高频系数的重建值。

还需要说明的是, 在本申请实施例中, 在确定当前层的节点的低频系数和高频系数的重建值之后, 便可以基于高频系数的重建值和低频系数进行 RAHT 逆变换, 进而可以确定子节点的属性重建值。

示例性地, 在本申请实施例中, 假设 $g'_{L,2x,y,z}$ 和 $g'_{L,2x+1,y,z}$ 为 L 层中互为近邻点的两个属性 DC 系数。经过线性变换后, L-1 层的信息为 AC 系数 $f'_{L-1,x,y,z}$ 和 DC 系数 $g'_{L-1,x,y,z}$; 然后, $f'_{L-1,x,y,z}$ 将不再进行变换, 直接进行量化编码, $g'_{L-1,x,y,z}$ 将继续寻找近邻点进行变换, 如果寻找不到, 则将其直接传递至 L-2 层, 即 RAHT 变换仅对存在邻居点的节点有效, 没有邻居点的节点将直接传递至上一层。在该变换过程中, $g'_{L,2x,y,z}$ 和 $g'_{L,2x+2,y,z}$ 对应的权重(该节点内非空子节点的个数)分别为 $w'_{L,2x,y,z}$ 和 $w'_{L,2x+1,y,z}$ (简写为 w'_0 和 w'_1), $g'_{L-1,x,y,z}$ 的权重为 $w'_{L-1,x,y,z}$, 则通用变换公式为:

$$\begin{bmatrix} g'_{L-1,x,y,z} \\ f'_{L-1,x,y,z} \end{bmatrix} = T_{w'_0,w'_1} \begin{bmatrix} g'_{L,2x,y,z} \\ g'_{L,2x+1,y,z} \end{bmatrix} \quad (34)$$

其中, $T_{w'_0,w'_1}$ 为变换矩阵, 变换矩阵会随着各点对应的权重自适应变化更新。RAHT 的正向变换(也可称为“RAHT 正变换”)如前述的图 35A 所示。

示例性地, 在本申请实施例中, 根据所得到的当前节点的子节点的 DC 系数和 AC 系数进行 RAHT 的逆向变换, 可以恢复得到当前节点的子节点的属性重建值。其中, RAHT 的逆向变换(也可称为“RAHT 反变换”、“RAHT 逆变换”)如前述的图 35B 所示。

也就是说, 在本申请实施例中, 如果当前层的节点对应的第四数量和第五数量不相同, 那么可以继续对当前层的节点依次进行变换、预测等过程。具体地, 对于当前层的当前节点来说, 可以利用当前节点的相邻节点的重建属性以及每个相邻节点距离当前节点的每个子节点的空间几何距离进行线性拟合, 得到当前节点的每个子节点的预测属性; 接着, 利用每个子节点的预测属性进行 RAHT 属性变换得到相应的 DC 和 AC 系数, 然后利用预测节点的 AC 系数(高频系数的预测值)和从码流中解析得到的 AC 系数(系数差值), 来恢复得到当前待解码节点(当前节点)的 AC 系数(高频系数的重建值), 最后便可以利用当前节点的 AC 系数(高频系数的重建值)和 DC 系数进行 RAHT 反变换, 从而恢复得到当前节点的每个子节点的属性重建值。

进一步地, 在本申请实施例中, 如果当前层为非跳过编码层, 那么可以对当前层的节点依次进行变换、预测等过程, 确定出当前层的节点对应的子节点的属性重建值; 然后针对当前层节点的子节点来说, 可以先确定子节点对应的下一层子节点的第六数量; 然后根据第五数量和第六数量, 确定子节点对应的下一层子节点的属性重建值。

也就是说, 在本申请实施例中, 无论当前层是否为跳过编码层, 即无论是否对当前层的节点依次进行变换、预测等过程; 仍然需要不断重复上述步骤, 依然确定其它层的节点数量, 以及对应的子节点数量, 然后再根据节点数量和对应的子节点数量来判断是否执行跳过解码的处理。

相应地, 在本申请实施例中, 不断重复步骤 S4301 至步骤 S4302 的方法, 依次从 RAHT 变换的根节点不断重复直至到 RAHT 的叶子结点层的最后一个节点, 从而完成整个 RAHT 变换的属性解码。

进一步地, 在一些实施例, 该方法还可以包括: 解码码流, 确定预测模式标识信息; 在预测模式标识信息指示当前单元启动跳过解码模式时, 执行第一数量和第二数量的确定步骤, 和/或, 执行第四数量和第五数量的确定步骤。

在本申请实施例中, 预测模式标识信息至少为以下高层语法元素中的其中之一: 属性参数集(Attribute Parameter Set, APS)对应的语法元素和属性块头信息(Attribute Block Head, ABH)对应的语法元素。

在本申请实施例中, 若预测模式标识信息的取值为第一值, 则确定当前单元启动跳过解码模式; 若预测模式标识信息的取值为第二值, 则确定当前单元不启动跳过解码模式。下面针对当前单元的体素节点是否启动跳过解码模式和当前层的节点是否启动跳过解码模式分别进行说明。

在一种具体的实施例中, 该方法还可以包括: 解码码流, 确定第一预测模式标识信息; 在第一预测模式标识信息指示当前单元的体素节点启动跳过解码模式时, 执行第一数量和第二数量的确定步骤。

需要说明的是,在本申请实施例中,若第一预测模式标识信息的取值为第一值,则确定当前单元的体素节点启动跳过解码模式;若第一预测模式标识信息的取值为第二值,则确定当前单元的体素节点不启动跳过解码模式。

还需要说明的是,在本申请实施例中,只有在当前单元的体素节点启动跳过解码模式时,这时候可以进一步确定第一数量和第二数量,然后根据第一数量和第二数量的大小来决定当前单元的体素节点是否跳过解码。具体地,如果两者一致,则跳过解码体素级别的节点属性,否则,会采用一个计时器,当该计时器的剩余重复节点数目为零时,则表示后续中的点不存在重复节点,同样可以跳过处理后续点的属性解码,并且将后续体素节点的属性重建值直接复制为最终剩余重建点的属性重建值。

在另一种具体的实施例中,该方法还可以包括:解码码流,确定第二预测模式标识信息;在第二预测模式标识信息指示当前层的节点启动跳过解码模式时,执行第四数量和第五数量的确定步骤。

需要说明的是,在本申请实施例中,若第二预测模式标识信息的取值为第一值,则确定当前层的节点启动跳过解码模式;若第二预测模式标识信息的取值为第二值,则确定当前层的节点不启动跳过解码模式。

还需要说明的是,在本申请实施例中,只有在当前层的节点启动跳过解码模式时,这时候可以进一步确定第四数量和第五数量,然后根据第四数量和第五数量的大小来决定当前层的节点是否跳过解码,即当前层是否为跳过解码层。具体地,如果两者一致,则确定当前层为跳过解码层,这时候不再需要解码当前层的节点对应的子节点的属性重建值;如果两者不一致,则确定当前层不属于跳过解码层,这时候可以按照相关技术的解码方式进行 RAHT 预测以及解码。

还需要说明的是,在本申请实施例中,第一值与第二值不同,而且第一值和第二值可以是参数形式,也可以是数字形式。具体地,第一预测模式标识信息和第二预测模式标识信息可以是写入在概述(profile)中的参数,也可以是一个标志(flag)的取值,这里不作具体限定。另外,对于第一值和第二值而言,第一值可以设置为1,第二值可以设置为0;或者,第一值可以设置为0,第二值可以设置为1;或者,第一值可以设置为true,第二值可以设置为false;或者,第一值可以设置为false,第二值可以设置为true。其中,在本申请实施例中,第一值设置为1,第二值设置为0,但是不作具体限定。

示例性地,在一些实施例中,以第一值设置为1,第二值设置为0为例,解码码流,确定第二预测模式标识信息的取值;如果第二预测模式标识信息的取值为1,那么即可确定当前层的节点启动跳过解码模式,进而可以根据上述方法进一步确定当前层的节点对应的第四数量和第五数量;如果第二预测模式标识信息的取值为0,那么即可确定当前层的节点不启动跳过解码模式,则可以按照常见的帧内预测的方式或帧间预测的方式对当前层的节点进行属性解码处理。

也就是说,在本申请实施例中,如果解码码流确定的第一预测模式标识信息的取值为第一值,即确定当前单元的体素节点启动跳过解码模式,那么可以执行第一数量和第二数量的确定步骤,即执行图41所示的解码流程;如果解码码流确定的第二预测模式标识信息的取值为第一值,即确定当前层的节点启动跳过解码模式,那么可以执行第四数量和第五数量的确定流程,即执行图43所示的解码流程。

综上所述,在本申请实施例中,在对属性信息进行编解码时,如果当前层的节点数目与当前层的子节点数目一致,则认为当前层属于跳过编码层,因此不需要对当前层进行变换、预测、编码以及解码等过程,从而可以降低属性变换编解码的时间复杂度,并且对属性的编码效率不会产生任何影响。

示例性的,在标准文本的实现中,一种具体的 SPEC 修改如下所示:

10.5.2.2 State variables

The RAHT tree is specified in terms of the following state variables:

The sparse array *RahtCoeff* of transform block coefficients; *RahtCoeff*[*lvl*][*bs*][*bt*][*bv*][*i*] is the *i*-th coefficient for the block located at (*bs*, *bt*, *bv*) in transform level *lvl*. Unset elements shall be inferred to be 0.

The array *RahtBlkLoc* of transform block locations; *RahtBlkLoc*[*lvl*][*nIdx*][*k*] is the location of the *nIdx*-th coded block in transform level *lvl*.

The array *RahtBlkCnt* of node counts per tree level; *RahtBlkCnt*[*lvl*] is the number of blocks in transform level *lvl*.

The variable *RahtLvlCnt*, the number of transform levels.

10.5.2.3 Transform block weight

The weight of the DC transform coefficient for a block located at (*bs*, *bt*, *bv*) in transform level *lvl* is specified by the expression *RahtBlkWeight*[*lvl*][*bs*][*bt*][*bv*]. It is equal to the number of points that the coefficient applies to.

The sum of all block weights in any transform level is equal to the number of coded points (*PointCnt*).

A block's weight is equal to the sum of its child block weights.

RahtBlkWeight[*lvl*][*bs*][*bt*][*bv*] :=

RahtBlkWeight = 0

```

for (ptIdx = 0; ptIdx < PointCnt; ptIdx++)
    RahtBlkWeight += isPointInSubtree[ptIdx]
Where
    isPointInSubtree[ptIdx] :=
5      bs == AttrPos[ptIdx][0] >> lvl
      && bt == AttrPos[ptIdx][1] >> lvl
      && bv == AttrPos[ptIdx][2] >> lvl
10.5.2.4 Number of transform levels and per-level block order
    The root node of the transform tree is the lowest block in the tree with a DC coefficient that spans the
10  entire geometry; i.e. it has a weight equal to the number of coded points, PointCnt.
    The tree level containing the root node is RahtRootLvl:
    RahtRootLvl := RahtLvlCnt - 1
    Within a transform level, blocks are ordered for coefficient coding by ascending Morton-coded block
    location, as specified by the derivation of RahtBlkLoc. Empty blocks are ignored.
15  for (RahtLvlCnt = 0; !done; RahtLvlCnt++)
    for (mIdx = 0, nIdx = 0, wSum = 0; wSum < PointCnt; mIdx++) {
        (bs, bt, bv) = FromMorton(mIdx)

        wSum += RahtBlkWeight[RahtLvlCnt][bs][bt][bv]
20      if (RahtBlkWeight[RahtLvlCnt][bs][bt][bv] == 0)
          continue

        RahtBlkCnt[RahtLvlCnt]++
        RahtBlkLoc[RahtLvlCnt][nIdx][0] = bs
        RahtBlkLoc[RahtLvlCnt][nIdx][1] = bt
        RahtBlkLoc[RahtLvlCnt][nIdx][2] = bv
        nIdx++

        done = RahtBlkWeight[RahtLvlCnt][bs][bt][bv] == PointCnt
30    }

10.5.2.5 2×2×2 transform block coefficient weights
    Transform coefficient weights are specified for each directional stage of the two-point transform for
    2×2×2 transform blocks by the expression RahtCoeffWeightM[lvl][stage][bs][bt][bv][m]; the parameter(s):
    bs, bt and bv specify a transform block location in tree level lvl, lvl > 0;
35  m specifies the transform coefficient index in forward transform stage stage.
    RahtCoeffWeightM[lvl][stage][bs][bt][bv][m] := RahtCoeffWeight[lvl][stage][s][t][v]
    where
        s := 2 × bs + FromMorton[m][0]
        t := 2 × bt + FromMorton[m][1]
40      v := 2 × bv + FromMorton[m][2]
    Within a block, coefficient weights are determined iteratively starting from its child block weights (stage
    0) to the transform block coefficient weights of stage 3. At each transform stage and for each pair of
    inverse-transformed values a and b, the weight for the DC (wL) and AC (wH) coefficient is the sum of the
    weights for a and b. If the weight for either a or b is 0, the AC coefficient weight is 0.
45  The expression RahtCoeffWeight[lvl][stage][s][t][v] specifies the derivation of a weight in transform
    stage stage for the coefficient corresponding to the block located at (s, t, v) in tree level lvl - 1.
    RahtCoeffWeight[lvl][stage][s][t][v] :=
        stage == 0 ? RahtBlkWeight[lvl - 1][s][t][v] :
        stage == 1 ? v % 2 == 0 ? wL[0][0][1] : wH[0][0][-1] :
50      stage == 2 ? t % 2 == 0 ? wL[0][1][0] : wH[0][-1][0] :
        stage == 3 ? s % 2 == 0 ? wL[1][0][0] : wH[-1][0][0] : na
    where
        wL[ds][dt][dv] := wSum[ds][dt][dv]
        wH[ds][dt][dv] := wSum[ds][dt][dv] × wHnz[ds][dt][dv]
55      wSum[ds][dt][dv] := wP[s][t][v] + wP[s + ds][t + dt][v + dv]
        wHnz[ds][dt][dv] := wP[s][t][v] × wP[s + ds][t + dt][v + dv] > 0
        wP[s][t][v] := RahtCoeffWeight[lvl][stage - 1][s][t][v]
        RahtBlkWeight[lvl][s][t][v] = RahtCoeffWeight[lvl][3][2 × s][2 × t][2 × v]; lvl > 0.

```

In the example of Figure 18, block B has stage 0 coefficient weights of 1, 3 and 1; stage 1 and 2 weights of 1, 4 and 1; and stage 3 weights of 5, 4 and 5. *RahtCoeffWeight*[1][1][3][0][2] would be 4.

10.5.3 Coefficient order

10.5.3.1 General

Subclause 10.5.3 specifies the correspondence between coded transform coefficients and the transform tree.

Starting from the root of the transform tree and proceeding in breadth-first order, coefficients are coded for each transform block; all transform blocks within one tree level are coded before those of the next level. Within a tree level, blocks shall be traversed in ascending Morton order of block location.

The order of coefficients within a transform block is specified by 10.5.3.2 for $2 \times 2 \times 2$ blocks (tree levels greater than 0) and 10.5.3.3 for blocks of co-located points (tree level 0).

The mapping from the coded order to the transform tree is specified in terms of the following variables:

Lvl, the index of the mapped transform level.

CoeffIdx, the index into the decoded coefficient array *AttrCoeff* for the next mapped coefficient.

CoeffIdx = 0

for (*Lvl* = *RahtLvlCnt*; *Lvl* ≥ 0; *Lvl*--) {

if (*Lvl* > 0) {

... /* See 10.5.3.2 */

if (*RahtBlkCnt*[*Lvl*] == *RahtBlkCnt*[*Lvl*])

continue

} else if (*RahtBlkCnt*[*Lvl*] != *PointCnt*)

{

... /* See 10.5.3.3 */

}

}

也就是说，本申请实施例提出的解码方法，在对属性进行 RAHT 解码时，在每一个 RAHT 变换层通过判断当前层的节点数目与当前层的子节点数目是否一致，来决定当前 RAHT 变换层是否属于跳过编码层，即确定是否跳过对当前层的变换、预测等过程。如果当前层的节点数目与当前层的子节点数目一致，则认为当前层属于跳过编码层，不需要进行变换、预测、编码以及解码等过程，从而可以降低属性变换编码解码的时间复杂度，并且对属性的编码效率不会有任何影响。

本实施例提供了一种解码方法，这里提出了一种跳过当前层的解码方式，通过利用当前层的节点数目与子节点数目来自适应地决定当前层是否可以跳过解码，这样可以在保证编解码效率不变的前提下，降低编解码的时间复杂度；另外，这里还提出了一种跳过体素级别节点的解码方式，首先可以得到体素节点数目和重建节点数目，当重建节点数目与体素节点数目一致时，则认为当前单元中不存在重复节点，同样可以跳过解码体素级别的节点属性；如此，在保证点云属性的编解码效率基础上，可以降低点云属性编解码的时间复杂度，并且还可以节省码率，进而提升了点云的编解码性能。

在本申请的一实施例中，参见图 44，其示出了本申请实施例提供的一种编码方法的流程示意图。如图 44 所示，该方法可以包括：

S4401：确定当前单元的体素节点的第一数量和所述当前单元的重建节点的第二数量。

需要说明的是，本申请实施例的编码方法应用于点云编码器（可简称为“编码器”）。其中，该方法可以是指点云编码方法，具体是一种点云属性编码方法，更具体地，可以是一种点云属性 RAHT 变换预测的跳过编码方法。在这里，主要是针对当前层的节点是否跳过编码以及在划分到体素级别时的体素节点是否跳过编码进行优化，从而可以降低属性变换编解码的时间复杂度，并且对属性变换的编解码效率不会有任何影响。

还需要说明的是，在本申请实施例中，当前单元可以是当前待编码的当前编码单元，这里可以是一个片（slice）。在一些实施例中，该方法还可以包括：对当前单元中的节点进行划分，确定至少一层；在最后一层的节点被划分到体素级别时，确定当前单元的体素节点以及体素节点的第一数量。

示例性地，在 RAHT 属性变换中，RAHT 属性变换的顺序是从根节点依次进行划分，直至划分到体素级别，具体为划分到 $1 \times 1 \times 1$ 大小的单位立方体时停止划分，从而完成整个点云属性的编码和重建。在这里，如图 42 所示，每次沿着 Z 方向、Y 方向和 X 方向做一次下采样所得到的层即为一个 RAHT 变换层，即 layer。然后直至划分到 $1 \times 1 \times 1$ 大小的单位立方体时表示已经划分到体素级别，这时候可以确定体素节点以及体素节点的第一数量。

还需要说明的是，在本申请实施，例中，体素节点表示由根节点划分到体素级别时对应的节点，体素节点的大小为 $1 \times 1 \times 1$ ；重建节点表示当前单元中进行属性重建的节点。其中，在当前单元中的节点进

行属性编码之前,当前单元中的节点的几何信息已经全部编码完成,这时候可以确定当前单元的重建节点的第二数量,从而能够判断第一数量与第二数量是否一致。

S4402: 根据第一数量与第二数量,确定当前单元的体素节点的属性信息是否跳过编码。

需要说明的是,在本申请实施例中,在对体素节点进行属性编码之前,首先需要判断第一数量与第二数量是否相同,进而来确定当前单元的体素节点是否进行跳过编码。

在一些实施例中,若第一数量与第二数量相同,则跳过编码当前单元的体素节点的属性信息。

在一些实施例中,若第一数量与第二数量不同,则对体素节点中的重复节点进行属性编码,以及跳过编码体素节点中除重复节点之外剩余体素节点的属性信息。

在本申请实施例中,当前单元划分的体素节点中可能会存在重复节点,因此第一数量与第二数量有可能不一致。其中,如果当前单元划分的体素节点中不存在重复节点,那么第一数量与第二数量一致;反之,如果当前单元划分的体素节点中存在重复节点,那么第一数量与第二数量不一致,此时需要针对重复节点进行属性编码。在这里,重复节点(也可简称为“重复点”)是指几何信息一样、但属性信息不同的多个节点。

在一些实施例中,对体素节点中的重复节点进行属性编码,可以包括:对重复节点的属性信息进行编码处理,将所得到的编码比特写入码流。

需要说明的是,在本申请实施例中,在对重复节点的属性信息进行编码处理时,可以包括:设置计时器;在重复节点的属性信息开始编码时,启动计时器,并在计时器的计时达到预设数值时,确定重复节点的属性信息全部编码完成。

还需要说明的是,在本申请实施例中,在第一数量与第二数量不同时,还可以根据第一数量与第二数量之间的差值,确定体素节点中的重复节点的第三数量;其中,计时器的设置与第三数量有关联关系。

示例性地,可以采用一个计时器来确定重复节点的属性重建值是否全部编码完成。如果计时器处于正计数模式,那么在重复节点的属性重建值开始编码时,计时器的初始值为0,在计时器计数到第三数量时,此时重复节点的属性重建值全部编码完成;如果计时器处于倒计时模式,那么在重复节点的属性重建值开始编码时,计时器的初始值为第三数量,在计时器计数到0时,此时重复节点的属性重建值全部编码完成。在重复节点的属性重建值全部编码完成之后,意味着后续的体素节点中不存在重复节点,可以跳过这些节点的属性编码。

进一步地,在一些实施例中,参见图45,在步骤S4401之后,该方法还可以包括:

S4501: 根据第一数量与第二数量,确定当前单元的体素节点的属性重建值。

需要说明的是,在本申请实施例中,对于当前单元的体素节点的属性重建值,可以是直接复制当前单元的重建节点的属性重建值,但是其中的重复节点的属性重建值则无法复制得到。也就是说,这里可以由第一数量与第二数量之间的大小来决定如何确定当前单元的体素节点的属性重建值。

在一些实施例中,在第一数量与第二数量相同时,根据第一数量与第二数量,确定当前单元的体素节点的属性重建值,可以包括:将当前单元的体素节点的属性重建值设置为当前单元的重建节点的属性重建值。示例性地,如果第一数量与第二数量相同,那么可以跳过编码当前单元的体素节点的属性信息,这时候可以将体素节点的属性重建值直接复制为当前单元的重建节点的属性重建值。

在一些实施例中,在第一数量与第二数量不同时,根据第一数量与第二数量,确定当前单元的体素节点的属性重建值,可以包括:将重复节点的属性重建值作为当前单元中第一重建节点的属性重建值,以及将体素节点中除重复节点之外的剩余体素节点的属性重建值设置为当前单元中除第一重建节点之外的剩余重建节点的属性重建值。

示例性地,如果第一数量与第二数量不同,表明当前单元划分的体素节点中存在有重复节点,这时候可以确定重复节点的属性重建值,然后将重复节点的属性重建值作为当前单元中第一重建节点的属性重建值,以及将重复节点之外的剩余体素节点的属性重建值复制为当前单元中除第一重建节点之外的剩余重建节点的属性重建值。另外,在第一数量与第二数量不同时,需要编码重复节点的属性信息,以便解码端可以解码确定这些重复节点的属性重建值。

示例性地,对于几何信息相同、属性信息不同的多个节点(即重复节点),编码端可以采用计时器来实现把这多个节点的属性信息进行相加,将相加值确定为该几何信息对应节点的属性信息;后续在解码端进行属性解码时,采用计时器来实现从这个属性信息中将这多个节点各自的属性信息分别剥离出来,以得到重复节点的属性重建值。

还可以理解地,在本申请实施例中,至少一层包括当前层,其中,当前层可以为当前待解码的 RAHT 变换层,或称为“RAHT 属性编码层”。图46为本申请实施例提供的又一种编码方法的流程示意图。如图46所示,该方法可以包括:

S4601: 确定当前层的节点的第四数量和当前层的节点对应的子节点的第五数量;其中,第四数量

和第五数量用于确定是否对当前层进行跳过编码。

需要说明的是，在本申请实施例中，可以先确定当前层的节点的第四数量，同时可以确定当前层的节点对应的子节点的第五数量。其中，在当前层的节点进行属性编码之前，由于当前层的节点的几何信息已经编码完成，根据当前层的节点的几何信息可以确定出当前层的节点数目（即“第四数量”）和当前层的节点的子节点数目（即“第五数量”）。

进一步地，在本申请实施例中，需要先基于点云中的点的几何信息来构建 RAHT 属性变换结构，可以按照由根节点到子节点的顺序进行编码。利用当前层节点的几何信息，依次按照 Z、Y 和 X 的顺序进行恢复得到当前层的子节点，其次利用上一层的节点已经重建的属性来对当前层节点的属性进行预测编码，从而恢复得到当前层节点的属性重建值，直至变换到体素级别，进而可以得到包括有至少一个 RAHT 变换层的 RAHT 属性变换结构。

需要说明的是，在本申请实施例中，RAHT 属性变换可以是基于八叉树层级的顺序执行的。其中，基于八叉树的层级顺序，可以由体素级别不断进行变换直至到根节点，从而构建出八叉树。然后在预测变换过程中，同样基于八叉树的层级顺序进行属性预测变换编码，但是是由根节点不断进行变换直至到体素级别。

可以理解的是，在本申请实施例中，可以定义每次沿着预设方向，如 Z 方向、Y 方向和 X 方向依次做一次下采样得到的层即为一个 RAHT 变换层，如当前层（layer）。

还需要说明的是，在本申请实施例中，对于当前层来说，当前层中可以包括至少一个点。其中，对于当前层中的至少一个点，在对当前层进行编码时，其可以作为当前层中的待编码节点。

进一步地，在本申请实施例中，对于当前层中的每一个点，其对应一个几何信息和一个属性信息；其中，几何信息表征该点的空间关系，属性信息表征该点的属性的相关信息。

在这里，属性信息可以为颜色信息，也可以是反射率或者其它属性，本申请实施例不作具体限定。其中，当属性信息为颜色信息时，具体可以为任意颜色空间的属性信息。示例性地，属性信息可以为 RGB 空间的属性信息，也可以为 YUV 空间的属性信息，还可以为 YCbCr 空间的属性信息等等，本申请实施例也不作具体限定。

还需要说明的是，在本申请实施例中，在进行 RAHT 变换编码时，首先是完成非体素级别的节点属性变换以及逆变换，其次完成体素级别的节点变换，因为点云中会存在一种情况，即点云中存在重复节点的现象。

相应地，在本申请实施例中，如果当前层的节点为非体素级别时，那么第四数量可以表征当前层的被占据节点的数量；第五数量则可以表征当前层的节点中被占据的子节点的数量。

相应地，在本申请实施例中，如果当前层的节点为体素级别时，那么第四数量可以表征当前层的被占据节点的数量；第五数量则可以表征待编码的节点的数量。

也就是说，在本申请实施例中，第四数量即为当前层的有效节点（即被占据的节点）的数量，而对于非体素级别的节点，第五数量为当前层的节点的有效子节点（即被占据的子节点）的数量，对于体素级别的节点，第五数量为待编码的节点的数量。

进一步地，在本申请实施例中，可以先确定当前层的节点的几何信息；然后再根据几何信息确定当前层的节点对应的子节点，以及第五数量。

需要说明的是，在本申请实施例中，对于当前层的当前节点来说，在利用当前节点的几何信息进行对应的子节点的确定时，可以选择利用当前节点的几何信息进行上采样，得到当前节点被占据的子节点（子节点个数为 N，其中 N 的取值最大为 8）。

示例性地，在一些实施例中，在对当前层的节点的属性信息进行编解码时，可以先得到当前层的节点的数量，即第四数量；同时，在利用当前层的节点的几何信息进行恢复得到当前层的节点的子节点之后，可以得到当前层的节点的子节点的数量，即第五数量。

S4602：根据第四数量和第五数量，确定当前层的节点对应的子节点的属性重建值。

需要说明的是，在本申请实施例中，在确定当前层的节点对应的第四数量和第五数量之后，可以进一步根据第四数量和第五数量，确定当前层的节点对应的子节点的属性重建值。

进一步地，在本申请实施例中，在确定当前层的节点对应的第四数量和第五数量之后，可以利用第四数量和第五数量进行是否跳过编码层（当前层）的属性信息的编解码。

可以理解的是，在本申请实施例中，由于 RAHT 变换仅对存在邻居点的节点有效，如果当前层的节点的数目与当前层的节点的子节点的数目完全一致时，则可以表明当前层中的每个节点仅仅只有一个子节点；在这种情况下，当前层不会产生 AC 系数（高频系数），因此可以选择跳过对当前层的节点所依次进行的变换、预测等过程。

也就是说，在本申请实施例中，通过利用当前层的节点数目与子节点数目，可以自适应地决定当前

层是否可以跳过编解码。其中,判定是否跳过对当前层的编解码处理的关键,在于当前层的节点数目与子节点数目是否相同,即第四数量和第五数量是否相同。

在一些实施例中,在根据第四数量和第五数量,确定当前层的节点对应的子节点的属性重建值时,该方法还可以包括:若第四数量和第五数量相同,则将当前层的节点的属性重建值确定为当前层的节点对应的子节点的属性重建值。

可以理解的是,在本申请实施例中,如果当前层的节点对应的第四数量和第五数量相同,即可以确定当前层的节点的数量与当前层的节点所对应的子节点的数量是相同的,那么,便可以认为对于当前层的每一个节点来说,均只对应有一个子节点。

相应地,在本申请实施例中,由于 RAHT 变换仅对存在邻居点的节点有效,在 RAHT 属性变换的过程中,如果当前层的每一个节点均只对应有一个子节点,那么可以认为当前层不会产生 AC 系数,因此可以选择不对当前层的节点依次进行变换、预测等过程,即跳过对当前层的处理,此时,可以称为“跳过编码层”。

相应地,在本申请实施例中,如果确定当前层的节点对应的第四数量和第五数量相同,即确定当前层为跳过编码层,那么可以选择跳过对当前层的节点依次进行变换、预测等过程,而是可以直接将当前层的节点的属性重建值确定为当前层的节点对应的子节点的属性重建值。

也就是说,在本申请实施例中,在 RAHT 属性变换的过程中,如果当前层的每一个节点均只对应有一个子节点,那么可以选择不再对当前层的节点依次进行变换、预测等过程,从而可以降低 RAHT 属性变换编解码的复杂度。

进一步地,在本申请实施例中,在根据第四数量和第五数量,确定当前层的节点对应的子节点的属性重建值时,若第四数量和第五数量相同,确定当前层为跳过编码层,那么可以选择跳过当前层至下一层,然后将下一层作为当前层,继续判断是否对下一层的节点进行跳过编码。

相应地,在本申请实施例中,对于当前层的节点对应的子节点来说,可以先确定子节点对应的下一层子节点的第六数量;然后根据第五数量和第六数量,确定子节点对应的下一层子节点的属性重建值。

可以理解的是,在本申请实施例中,如果当前层的节点对应的子节点为非体素级别的节点,那么第六数量可以为当前层的节点对应的子节点的下一层有效子节点(即下一层被占据的子节点)的数量,如果当前层的节点对应的子节点为体素级别的节点,那么第六数量可以为待编码的节点的数量。

需要说明的是,在本申请实施例中,在确定第五数量和第六数量之后,可以利用第五数量和第六数量进行是否跳过编码层(当前层的下一层)的属性信息的编解码。

示例性地,在一些实施例中,如果第五数量和第六数量相同,那么可以选择不对当前层的节点的子节点依次进行变换、预测等过程,即跳过对当前层的子节点的处理,而是可以直接将当前层的节点对应的子节点的属性重建值确定为当前层的节点对应的子节点的下一层子节点的属性重建值。

在一些实施例中,在根据第四数量和第五数量,确定当前层的节点对应的子节点的属性重建值时,该方法还可以包括:若当前层的节点对应的第四数量和第五数量不同,则根据当前层的节点确定当前层的节点对应的子节点的属性预测值;基于子节点的属性预测值进行 RAHT 变换,确定当前层的节点对应的高频系数的重建值和低频系数;基于高频系数的重建值和低频系数进行 RAHT 逆变换,确定子节点的属性重建值。

需要说明的是,在本申请实施例中,如果第四数量和第五数量不相同,即可以确定当前层的节点的数量与当前层的节点所对应的子节点的数量是不同的,此时,当前层依然会产生 AC 系数,因此可以继续对当前层的节点依次进行变换、预测等过程,而不会跳过对当前层的处理,此时,可以将当前层作为非跳过编码层。

进一步地,在一些实施例中,在根据当前层的节点确定当前层的节点对应的子节点的属性预测值时,可以包括:确定当前层的节点对应的相邻节点;根据相邻节点对应的属性重建值和相对距离参数,确定当前层的节点对应的子节点的属性预测值。

还需要说明的是,在本申请实施例中,相邻节点可以是指当前节点的邻域节点。其中,相邻节点对应的相对距离参数可以表征当前层的节点对应的子节点与对应的相邻节点的之间的空间几何距离。

示例性地,在本申请实施例中,对于当前层的当前节点,该当前节点包括子节点 1 和子节点 2 这两个子节点,当前节点与相邻节点之间的相对距离参数可以包括子节点 1 与相邻节点的之间的空间几何距离,还可以包括子节点 2 与相邻节点的之间的空间几何距离。

示例性地,在本申请实施例中,在根据当前层的节点确定当前层的节点对应的子节点的属性预测值时,对于当前层的当前节点,可以利用当前节点的邻域节点的重建属性(属性重建值)以及每个邻域节点距离当前节点的子节点的空间几何距离进行线性拟合,最终得到当前节点的每个子节点的属性预测值。

示例性地,在本申请实施例中,对于当前层的当前节点来说,可以先确定当前节点的 19 个相邻节

点,其次利用相邻节点与当前节点的每个子节点之间的空间几何距离对每个子节点的属性进行线性加权预测,最终得到每个子节点的属性预测值。

进一步地,在一些实施例中,在基于子节点的属性预测值进行 RAHT 变换,确定当前层的节点对应的高频系数的重建值和低频系数时,可以包括:基于子节点的属性预测值进行 RAHT 变换,确定当前层的节点对应的高频系数的预测值和低频系数;基于子节点的属性值进行 RAHT 变换,确定当前层的节点对应的高频系数和低频系数;根据当前层的节点对应的高频系数的预测值和当前层的节点对应的高频系数,确定当前层的节点对应的高频系数的重建值。

进一步地,在一些实施例中,在根据当前层的节点对应的高频系数的预测值和当前层的节点对应的高频系数,确定当前层的节点对应的高频系数的重建值时,可以包括:根据当前层的节点对应的高频系数的预测值和当前层的节点对应的高频系数,确定当前层的节点对应的系数残差;根据当前层的节点对应的系数残差,确定当前层的节点对应的高频系数的重建值。

进一步地,在一些实施例中,在根据当前层的节点对应的系数残差,确定当前层的节点对应的高频系数的重建值时,可以包括:对量化后的系数残差进行反量化,确定当前层的节点对应的反量化残差值;根据当前层的节点对应的反量化残差值和当前层的节点对应的高频系数的预测值,确定当前层的节点对应的高频系数的重建值。

需要说明的是,在本申请实施例中,对于当前层的当前节点来说,在确定出当前节点的子节点对应的属性预测值之后,可以利用对应的子节点的属性预测值进行 RAHT 属性变换,从而可以得到相应的 DC 系数和 AC 系数。其中,DC 系数即为低频系数,AC 系数即为高频系数。其中,在本申请实施例中,对于当前层的当前节点来说,利用子节点对应的属性预测值进行 RAHT 属性变换所获得的 AC 系数可以理解当前节点对应的 AC 系数的预测值。

还需要说明的是,在本申请实施例中,对于当前层的当前节点来说,还可以利用子节点的属性值进行 RAHT 变换,确定当前层的节点对应的 AC 系数和 DC 系数。其中,在本申请实施例中,对于当前层的当前节点来说,利用子节点对应的属性值进行 RAHT 属性变换所获得的 AC 系数可以理解当前节点对应的 AC 系数的原始值。

这样,根据当前层的节点对应的 AC 系数的预测值和 AC 系数的原始值,可以确定当前层的节点对应的系数残差;然后对量化后的系数残差进行反量化,确定当前层的节点对应的反量化残差值;再根据当前层的节点对应的反量化残差值和当前层的节点对应的 AC 系数的预测值,可以确定当前层的节点对应的 AC 系数的重建值。

示例性的,在本申请实施例中,可以对当前层的节点对应的反量化残差值和当前层的节点对应的 AC 系数的预测值进行求和计算,进而可以获得当前层的节点对应的 AC 系数的重建值。

进一步地,在一些实施例中,该方法还可以包括:对系数残差进行量化,确定当前层的节点对应的量化后的系数残差;对量化后的系数残差进行编码处理,将所得到的编码比特写入码流。

需要说明的是,在本申请实施例中,将量化后的系数残差写入码流,后续在解码端通过解码码流即可获得量化后的系数残差,然后根据高频系数的预测值和量化后的系数残差,就可以确定当前层的节点对应的高频系数的重建值。

还需要说明的是,在本申请实施例中,在确定当前层的节点的低频系数和高频系数的重建值之后,便可以基于高频系数的重建值和低频系数进行 RAHT 逆变换,进而可以确定子节点的属性重建值。

示例性地,在本申请实施例中,假设 $g'_{L,2x,y,z}$ 和 $g'_{L,2x+1,y,z}$ 为 L 层中互为近邻点的两个属性 DC 系数。经过线性变换后, L-1 层的信息为 AC 系数 $f'_{L-1,x,y,z}$ 和 DC 系数 $g'_{L-1,x,y,z}$; 然后, $f'_{L-1,x,y,z}$ 将不再进行变换,直接进行量化编码, $g'_{L-1,x,y,z}$ 将继续寻找近邻点进行变换,如果寻找不到,则将其直接传递至 L-2 层,即 RAHT 变换仅对存在邻居点的节点有效,没有邻居点的节点将直接传递至上一层。在该变换过程中, $g'_{L,2x,y,z}$ 和 $g'_{L,2x+2,y,z}$ 对应的权重(该节点内非空子节点的个数)分别为 $w'_{L,2x,y,z}$ 和 $w'_{L,2x+1,y,z}$ (简称为 w'_0 和 w'_1), $g'_{L-1,x,y,z}$ 的权重为 $w'_{L-1,x,y,z}$, 则通用变换公式为:

$$\begin{bmatrix} g'_{L-1,x,y,z} \\ f'_{L-1,x,y,z} \end{bmatrix} = T_{w'_0,w'_1} \begin{bmatrix} g'_{L,2x,y,z} \\ g'_{L,2x+1,y,z} \end{bmatrix} \quad (35)$$

其中, $T_{w'_0,w'_1}$ 为变换矩阵,变换矩阵会随着各点对应的权重自适应变化更新。RAHT 的正向变换(也可称为“RAHT 正变换”)如前述的图 35A 所示。

示例性地,在本申请实施例中,根据所得到的当前节点的子节点的 DC 系数和 AC 系数进行 RAHT 的逆向变换,可以恢复得到当前节点的子节点的属性重建值。其中,RAHT 的逆向变换(也可称为“RAHT 反变换”、“RAHT 逆变换”)如前述的图 35B 所示。

也就是说,在本申请实施例中,如果当前层的节点对应的第四数量和第五数量不相同,那么可以继续对当前层的节点依次进行变换、预测等过程。具体地,对于当前层的当前节点来说,可以利用当前节

点的相邻节点的重建属性以及每个相邻节点距离当前节点的每个子节点的空间几何距离进行线性拟合,得到当前节点的每个子节点的预测属性;接着,利用每个子节点的预测属性进行 RAHT 属性变换得到相应的 DC 和 AC 系数,同时,可以通过 RAHT 属性变换来对当前节点的每个子节点的属性进行变换得到 DC 和 AC 系数;接着,可以利用预测节点得到的 AC 系数的预测值来对当前节点的 AC 系数进行预测,进而获得每个子节点的 AC 预测残差系数(系数残差),然后可以对系数残差进行量化编码。另一方面,还可以利用 AC 预测残差系数的反量化残差值以及 AC 系数的预测值,恢复得到当前节点的 AC 重建系数(高频系数的重建值),最终利用当前节点的 AC 系数和 DC 系数进行 RAHT 反变换,从而恢复得到当前节点每个子节点的属性重建值。

进一步地,在本申请实施例中,如果当前层为非跳过编码层,那么可以对当前层的节点依次进行变换、预测等过程,确定出当前层的节点对应的子节点的属性重建值;然后针对当前层节点的子节点来说,可以先确定子节点对应的下一层子节点的第六数量;然后根据第五数量和第六数量,确定子节点对应的下一层子节点的属性重建值。

也就是说,在本申请实施例中,无论当前层是否为跳过编码层,即无论是否对当前层的节点依次进行变换、预测等过程;仍然需要不断重复上述步骤,依然确定其它层的节点数量,以及对应的子节点数量,然后再根据节点数量和对应的子节点数量来判断是否执行跳过编码的处理。

相应地,在本申请实施例中,不断重复步骤 S4601 至步骤 S4602 的方法,依次从 RAHT 变换的根节点不断重复直至到 RAHT 的叶子结点层的最后一个节点,从而完成整个 RAHT 变换的属性编码。

进一步地,在一些实施例中,该方法还可以包括:确定预测模式标识信息;在预测模式标识信息指示当前单元启动跳过编码模式时,执行第一数量和第二数量的确定步骤,和/或,执行第四数量和第五数量的确定步骤。

在本申请实施例中,预测模式标识信息至少为以下高层语法元素中的其中之一:属性参数集(Attribute Parameter Set, APS)对应的语法元素和属性块头信息(Attribute Block Head, ABH)对应的语法元素。

在本申请实施例中,还可以对预测模式标识信息进行编码处理,将所得到的编码比特写入码流。其中,若当前单元启动跳过编码模式,则确定预测模式标识信息的取值为第一值;若当前单元不启动跳过编码模式,则确定预测模式标识信息的取值为第二值。下面针对当前单元的体素节点是否启动跳过编码模式和当前层的节点是否启动跳过编码模式分别进行说明。

在一种具体的实施例中,该方法还可以包括:确定第一预测模式标识信息;在第一预测模式标识信息指示当前单元的体素节点启动跳过编码模式时,执行第一数量和第二数量的确定步骤。

在本申请实施例中,还可以对第一预测模式标识信息进行编码处理,将所得到的编码比特写入码流。其中,若当前单元的体素节点启动跳过编码模式,则确定第一预测模式标识信息的取值为第一值;若当前单元的体素节点不启动跳过编码模式,则确定第一预测模式标识信息的取值为第二值。

还需要说明的是,在本申请实施例中,只有在当前单元的体素节点启动跳过编码模式时,这时候可以进一步确定第一数量和第二数量,然后根据第一数量和第二数量的大小来决定当前单元的体素节点是否跳过编码。具体地,如果两者一致,则跳过编码体素级别的节点属性,否则,会采用一个计时器,当该计时器的剩余重复节点数目为零时,则表示后续中的点不存在重复节点,同样可以跳过处理后续点的属性编码,并且将后续体素节点的属性重建值直接复制为最终剩余重建点的属性重建值。

在另一种具体的实施例中,该方法还可以包括:确定第二预测模式标识信息;在第二预测模式标识信息指示当前层的节点启动跳过编码模式时,执行第四数量和第五数量的确定步骤。

在本申请实施例中,还可以对第二预测模式标识信息进行编码处理,将所得到的编码比特写入码流。其中,若当前层的节点启动跳过编码模式,则确定第二预测模式标识信息的取值为第一值;若当前层的节点不启动跳过编码模式,则确定第二预测模式标识信息的取值为第二值。

还需要说明的是,在本申请实施例中,只有在当前层的节点启动跳过编码模式时,这时候可以进一步确定第四数量和第五数量,然后根据第四数量和第五数量的大小来决定当前层的节点是否跳过编码,即当前层是否为跳过编码层。具体地,如果两者一致,则确定当前层为跳过编码层,这时候不再需要编码当前层的节点对应的子节点的属性重建值;如果两者不一致,则确定当前层不属于跳过编码层,这时候可以按照相关技术的编码方式进行 RAHT 预测以及编码。

还需要说明的是,在本申请实施例中,第一值与第二值不同,而且第一值和第二值可以是参数形式,也可以是数字形式。具体地,第一预测模式标识信息和第二预测模式标识信息可以是写入在概述(profile)中的参数,也可以是一个标志(flag)的取值,这里不作具体限定。另外,对于第一值和第二值而言,第一值可以设置为 1,第二值可以设置为 0;或者,第一值可以设置为 0,第二值可以设置为 1;或者,第一值可以设置为 true,第二值可以设置为 false;或者,第一值可以设置为 false,第二值可以设置为

true。其中,在本申请实施例中,第一值设置为1,第二值设置为0,但是不作具体限定。

示例性地,在一些实施例中,以第一值设置为1,第二值设置为0为例,如果当前层的节点启动跳过编码模式,那么即可确定第二预测模式标识信息的取值为1,进而可以根据上述方法进一步确定当前层的节点对应的第四数量和第五数量;如果当前层的节点不启动跳过编码模式,那么可以确定第二预测模式标识信息的取值为0,则可以按照常见的帧内预测的方式或帧间预测的方式对当前层的节点进行属性编码处理。

也就是说,在本申请实施例中,如果码流中写入的第一预测模式标识信息的取值为第一值,即确定当前单元的体素节点启动跳过编码模式,那么可以执行第一数量和第二数量的确定步骤,即执行图44所示的编码流程;如果码流中写入的第二预测模式标识信息的取值为第一值,即确定当前层的节点启动跳过编码模式,那么可以执行第四数量和第五数量的确定流程,即执行图46所示的编码流程。

进一步地,本申请实施例还提供了一种码流,码流是根据待编码信息进行比特编码生成的;其中,待编码信息可以包括下述至少一项:预测模式标识信息、当前单元的重复节点的属性信息和当前层的节点对应的量化后的系数残差;其中,预测模式标识信息用于指示当前单元是否启动跳过编码模式。

需要说明的是,在本申请实施例中,编码端在确定出待编码信息之后,可以将这些待编码信息写入码流;然后由编码端传输到解码端,后续在解码端,通过解码码流就可得到这些信息,例如预测模式标识信息,进而可以确定当前单元是否启动跳过编码模式。

综上所述,在本申请实施例中,在对属性信息进行编码时,如果当前层的节点数目与当前层的子节点数目一致,则认为当前层属于跳过编码层,因此不需要对当前层进行变换、预测、编码以及解码等过程,从而可以降低属性变换编解码的时间复杂度,并且对属性的编码效率不会产生任何影响。

本实施例提供了一种编码方法,这里提出了一种跳过当前层的编码方式,通过利用当前层的节点数目与子节点数目来自适应地决定当前层是否可以跳过编码,这样可以在保证编解码效率不变的前提下,降低编解码的时间复杂度;另外,这里还提出了一种跳过体素级别节点的编码方式,首先可以得到体素节点数目和重建节点数目,当重建节点数目与体素节点数目一致时,则认为当前单元中不存在重复节点,同样可以跳过编码体素级别的节点属性;如此,在保证点云属性的编解码效率基础上,可以降低点云属性编解码的时间复杂度,并且还可以节省码率,进而提升了点云的编解码性能。

在本申请的又一实施例中,基于前述实施例的编解码方法,本申请实施例首先定义 RAHT 属性编码层,目前的属性 RAHT 变换编码顺序是从根节点依次进行划分直至划分到体素级别($1 \times 1 \times 1$),从而完成整个点云的属性编码和属性重建。在这里,可以定义每次沿着 Z 方向、Y 方向和 X 方向做一下采样得到的层即为一个 RAHT 变换层,即 layer。具体如图42所示。

其次,基于 RAHT 属性编码层,引入跳过编码层的算法。首先,在对当前层节点的属性进行编码/解码时,可以得到当前层的节点数目,以及利用当前层节点的几何信息进行恢复得到当前层节点的子节点之后,可以得到当前层节点的子节点数目,基于两者的大小来决定当前层是否属于跳过编码层:

如果当前层节点的数据与当前层节点子节点数目一致,则当前层属于跳过编码层;

否则,当前层为非跳过编码层。

这样,在得到跳过编码层的启动条件之后,编码端的一种具体算法如下:

步骤1:判断当前层的节点数目与当前层的子节点数目是否一致,如果一致,则当前层属于跳过编码层;否则,为非跳过编码层;

步骤2:如果当前层不属于跳过编码层,则按照相关技术的编码方式进行变换、预测以及编码。

步骤3:如果当前层属于跳过编码层,则直接跳过当前层至下一层。

步骤4:不断重复上述步骤,直至编码到体素级别。

步骤5:对于体素级别的节点,在对体素级别的节点进行属性编码之前,首先判断当前编码单元中体素级别的节点数目是否与重建节点的数目一致,如果一致,则跳过编码体素级别点的属性;否则,会采用一个计时器,当该计时器的剩余重复点数目为零时,则表示后续中的节点不存在重复点,同样可以跳过处理后续节点的属性编码。

在解码端,一种具体算法如下:

步骤1:判断当前层的节点数目与当前层的子节点数目是否一致,如果一致,则当前层属于跳过解码层;否则,为非跳过解码层;

步骤2:如果当前层不属于跳过解码层,则按照相关技术的解码方式进行预测以及解码。

步骤3:如果当前层属于跳过解码层,则直接跳过当前层至下一层。

步骤4:不断重复上述步骤,直至解码到体素级别。

步骤 5: 对于体素级别的节点, 在对体素级别的节点进行属性解码之前, 首先判断当前解码单元中体素级别的节点数目是否与重建节点的数目一致, 如果一致, 则跳过解码体素级别点的属性; 否则, 会采用一个计时器, 当该计时器的剩余重复点数目为零时, 则表示后续中的节点不存在重复点, 同样可以跳过处理后续节点的属性解码, 并且将后续剩余体素节点重建的属性重建值直接复制为最终剩余重建节点的属性重建值。

也就是说, 在本申请实施例中, 这里提出了一种跳过编码层的编码方式, 通过利用当前层的节点数目与子节点数目来自适应地决定当前层是否可以跳过编码, 这样可以在保证编解码效率不变的前提下, 降低编解码的时间复杂度。同样的, 在 RAHT 编码方案中, 编解码端首先完成非节点层属性信息的编码以及解码(即节点的大小大于或等于 $1 \times 1 \times 1$), 最终在对体素级别节点的属性信息进行编码和解码, 原因是由于点云中会存在重复点, 因此需要先完成非体素级别点的属性信息编码和解码, 其次在完成体素级别点属性信息的编码和解码。如此, 在编解码端, 首先可以得到体素级别重建点的数目, 当重建点的数目与所需要重建的节点数目一致时, 则认为当前编码单元(例如 slice)中不存在重复点, 同样可以跳过编码/解码体素级别的点属性信息。

通过上述实施例, 对前述实施例的具体实现进行详细阐述, 从中可以看出, 根据前述实施例的技术方案, 本申请实施例在对属性信息进行 RAHT 编码时, 在每一个 RAHT 属性编码层通过判断当前层的节点数目与当前层节点的子节点数目是否一致, 来决定当前 RAHT 属性编码层是否属于跳过编码层。如果当前层的节点数目与当前层节点的子节点数目一致, 则认为当前层属于跳过编码层, 即不需要进行变换、预测、编码以及解码等过程, 从而可以降低属性变换编码/解码的时间复杂度, 并且对属性的编码效率不会产生任何影响。

在本申请的再一实施例中, 基于前述实施例相同的发明构思, 参见图 47, 其示出了本申请实施例提供的一种编码器的组成结构示意图。如图 47 所示, 该编码器 470 可以包括: 第一确定单元 4701 和编码单元 4702, 其中:

第一确定单元 4701, 配置为确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量;

编码单元 4702, 配置为根据第一数量与第二数量, 确定当前单元的体素节点的属性信息是否跳过编码。

在一些实施例中, 编码单元 4702, 还配置为若第一数量与第二数量相同, 则跳过编码体素节点的属性信息; 若第一数量与第二数量不同, 则对体素节点中的重复节点进行属性编码, 以及跳过编码体素节点中除重复节点之外剩余体素节点的属性信息。

在一些实施例中, 编码单元 4702, 还配置为对重复节点的属性信息进行编码处理, 将所得到的编码比特写入码流。

在一些实施例中, 编码单元 4702, 还配置为设置计时器; 在重复节点的属性信息开始编码时, 启动计时器, 并在计时器的计时达到预设数值时, 确定重复节点的属性信息全部编码完成。

在一些实施例中, 第一确定单元 4701, 还配置为根据第一数量与第二数量之间的差值, 确定体素节点中的重复节点的第三数量; 其中, 计时器的设置与第三数量有关联关系。

在一些实施例中, 参见图 47, 编码器 470 还可以包括第一重建单元 4703, 配置为根据第一数量与第二数量, 确定当前单元的体素节点的属性重建值。

在一些实施例中, 第一重建单元 4703, 还配置为在第一数量与第二数量相同时, 将当前单元的体素节点的属性重建值设置为当前单元的重建节点的属性重建值。

在一些实施例中, 第一重建单元 4703, 还配置为在第一数量与第二数量不同时, 将重复节点的属性重建值作为当前单元中第一重建节点的属性重建值, 以及将体素节点中除重复节点之外的剩余体素节点的属性重建值设置为当前单元中除第一重建节点之外的剩余重建节点的属性重建值。

在一些实施例中, 第一确定单元 4701, 还配置为对当前单元中的节点进行划分, 确定至少一层;

以及在最后一层的节点被划分到体素级别时, 确定当前单元的体素节点以及体素节点的第一数量。

在一些实施例中, 至少一层包括当前层, 第一确定单元 4701, 还配置为确定当前层的节点的第四数量和当前层的节点对应的子节点的第五数量; 其中, 第四数量和第五数量用于确定是否对当前层进行跳过编码;

第一重建单元 4703, 还配置为根据第四数量和第五数量, 确定当前层的节点对应的子节点的属性重建值。

在一些实施例中, 第四数量表征当前层的被占据节点的数量; 第五数量表征当前层的节点中被占据的子节点的数量或待编码的节点的数量。

在一些实施例中,第一确定单元 4701,还配置为若第四数量和第五数量相同,则将当前层的节点的属性重建值确定为当前层的节点对应的子节点的属性重建值。

在一些实施例中,第一确定单元 4701,还配置为若第四数量和第五数量相同,则确定子节点对应的下一层子节点的第六数量;

5 第一重建单元 4703,还配置为根据第五数量和第六数量,确定子节点对应的下一层子节点的属性重建值。

在一些实施例中,第一重建单元 4703,还配置为若第四数量和第五数量不同,则根据当前层的节点确定当前层的节点对应的子节点的属性预测值;基于子节点的属性预测值和子节点的属性值分别进行 RAHT 变换,确定当前层的节点对应的高频系数的重建值和低频系数;以及基于高频系数的重建值和低频系数进行 RAHT 逆变换,确定子节点的属性重建值。

在一些实施例中,第一确定单元 4701,还配置为确定当前层的节点对应的相邻节点;以及根据相邻节点对应的属性重建值和相对距离参数,确定当前层的节点对应的子节点的属性预测值。

在一些实施例中,第一重建单元 4703,还配置为基于子节点的属性预测值进行 RAHT 变换,确定当前层的节点对应的高频系数的预测值和低频系数;基于子节点的属性值进行 RAHT 变换,确定当前层的节点对应的高频系数和低频系数;以及根据当前层的节点对应的高频系数的预测值和当前层的节点对应的高频系数,确定当前层的节点对应的高频系数的重建值。

在一些实施例中,第一重建单元 4703,还配置为根据当前层的节点对应的高频系数的预测值和当前层的节点对应的高频系数,确定当前层的节点对应的系数残差;以及根据当前层的节点对应的系数残差,确定当前层的节点对应的高频系数的重建值。

20 在一些实施例中,第一确定单元 4701,还配置为对系数残差进行量化,确定当前层的节点对应的量化后的系数残差;

编码单元 4702,还配置为对量化后的系数残差进行编码处理,将所得到的编码比特写入码流。

25 在一些实施例中,第一重建单元 4703,还配置为对量化后的系数残差进行反量化,确定当前层的节点对应的反量化残差值;以及根据当前层的节点对应的反量化残差值和当前层的节点对应的高频系数的预测值,确定当前层的节点对应的高频系数的重建值。

在一些实施例中,第一确定单元 4701,还配置为确定当前层的节点的几何信息;根据几何信息确定当前层的节点对应的子节点,以及第五数量。

在一些实施例中,第一确定单元 4701,还配置为确定预测模式标识信息;

30 编码单元 4702,还配置为在预测模式标识信息指示当前单元启动跳过编码模式时,执行第一数量和第二数量的确定步骤,和/或,执行第四数量和第五数量的确定步骤。

在一些实施例中,预测模式标识信息至少为以下高层语法元素中的其中之一:属性参数集对应的语法元素和属性块头信息对应的语法元素。

在一些实施例中,编码单元 4702,还配置为对预测模式标识信息进行编码处理,将所得到的编码比特写入码流。

35 可以理解地,在本申请实施例中,“单元”可以是部分电路、部分处理器、部分程序或软件等等,当然也可以是模块,还可以是非模块化的。而且在本实施例中的各组成部分可以集成在一个处理单元中,也可以是各个单元单独物理存在,也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现,也可以采用软件功能模块的形式实现。

40 所述集成的单元如果以软件功能模块的形式实现并非作为独立的产品进行销售或使用时,可以存储在一个计算机可读存储介质中,基于这样的理解,本实施例的技术方案本质上或者说对现有技术做出贡献的部分或者该技术方案的全部或部分可以以软件产品的形式体现出来,该计算机软件产品存储在一个存储介质中,包括若干指令用以使得一台计算机设备(可以是个人计算机,服务器,或者网络设备等)或 processor(处理器)执行本实施例所述方法的全部或部分步骤。而前述的存储介质包括:U 盘、移动硬盘、只读存储器(Read Only Memory, ROM)、随机存取存储器(Random Access Memory, RAM)、磁碟或者光盘等各种可以存储程序代码的介质。

因此,本申请实施例提供了一种计算机可读存储介质,应用于编码器 470,该计算机可读存储介质存储有计算机程序,所述计算机程序被第一处理器执行时实现前述实施例中任一项所述的方法。

50 基于编码器 470 的组成以及计算机可读存储介质,参见图 48,其示出了本申请实施例提供的编码器 470 的具体硬件结构示意图。如图 48 所示,编码器 470 可以包括:第一通信接口 4801、第一存储器 4802 和第一处理器 4803;各个组件通过第一总线系统 4804 耦合在一起。可理解,第一总线系统 4804 用于实现这些组件之间的连接通信。第一总线系统 4804 除包括数据总线之外,还包括电源总线、控制总线和状态信号总线。但是为了清楚说明起见,在图 48 中将各种总线都标为第一总线系统 4804。其中:

第一通信接口 4801, 用于在与其他外部网元之间进行收发信息过程中, 信号的接收和发送;

第一存储器 4802, 用于存储能够在第一处理器 4803 上运行的计算机程序;

第一处理器 4803, 用于在运行所述计算机程序时, 执行:

确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量;

5 根据第一数量与第二数量, 确定当前单元的体素节点的属性信息是否跳过编码。

可以理解, 本申请实施例中的第一存储器 4802 可以是易失性存储器或非易失性存储器, 或可包括易失性和非易失性存储器两者。其中, 非易失性存储器可以是只读存储器 (Read-Only Memory, ROM)、可编程只读存储器 (Programmable ROM, PROM)、可擦除可编程只读存储器 (Erasable PROM, EPROM)、电可擦除可编程只读存储器 (Electrically EPROM, EEPROM) 或闪存。易失性存储器可以是随机存取存储器 (Random Access Memory, RAM), 其用作外部高速缓存。通过示例性但不是限制性说明, 许多形式的 RAM 可用, 例如静态随机存取存储器 (Static RAM, SRAM)、动态随机存取存储器 (Dynamic RAM, DRAM)、同步动态随机存取存储器 (Synchronous DRAM, SDRAM)、双倍数据速率同步动态随机存取存储器 (Double Data Rate SDRAM, DDRSDRAM)、增强型同步动态随机存取存储器 (Enhanced SDRAM, ESDRAM)、同步连接动态随机存取存储器 (Synchlink DRAM, SLDRAM) 和直接内存总线随机存取存储器 (Direct Rambus RAM, DRRAM)。本申请描述的系统和方法的第一存储器 4802 旨在包括但不限于这些和任意其它适合类型的存储器。

而第一处理器 4803 可能是一种集成电路芯片, 具有信号的处理能力。在实现过程中, 上述方法的各步骤可以通过第一处理器 4803 中的硬件的集成逻辑电路或者软件形式的指令完成。上述的第一处理器 4803 可以是通用处理器、数字信号处理器 (Digital Signal Processor, DSP)、专用集成电路 (Application Specific Integrated Circuit, ASIC)、现成可编程门阵列 (Field Programmable Gate Array, FPGA) 或者其他可编程逻辑器件、分立门或者晶体管逻辑器件、分立硬件组件。可以实现或者执行本申请实施例中的公开的各方法、步骤及逻辑框图。通用处理器可以是微处理器或者该处理器也可以是任何常规的处理器等。结合本申请实施例所公开的方法的步骤可以直接体现为硬件译码处理器执行完成, 或者用译码处理器中的硬件及软件模块组合执行完成。软件模块可以位于随机存储器, 闪存、只读存储器, 可编程只读存储器或者电可擦写可编程存储器、寄存器等本领域成熟的存储介质中。该存储介质位于第一存储器 4802, 第一处理器 4803 读取第一存储器 4802 中的信息, 结合其硬件完成上述方法的步骤。

可以理解的是, 本申请描述的这些实施例可以用硬件、软件、固件、中间件、微码或其组合来实现。对于硬件实现, 处理单元可以实现在一个或多个专用集成电路 (Application Specific Integrated Circuits, ASIC)、数字信号处理器 (Digital Signal Processing, DSP)、数字信号处理设备 (DSP Device, DSPD)、可编程逻辑设备 (Programmable Logic Device, PLD)、现场可编程门阵列 (Field-Programmable Gate Array, FPGA)、通用处理器、控制器、微控制器、微处理器、用于执行本申请所述功能的其它电子单元或其组合中。对于软件实现, 可通过执行本申请所述功能的模块 (例如过程、函数等) 来实现本申请所述的技术。软件代码可存储在存储器中并通过处理器执行。存储器可以在处理器中或在处理器外部实现。

可选地, 作为另一个实施例, 第一处理器 4803 还配置为在运行所述计算机程序时, 执行前述实施例中任一项所述的方法。

本实施例提供了一种编码器, 在该编码器中, 在对每个体素节点进行属性重建时, 优化了每个体素节点是否进行属性编解码的判断条件, 具体为如果当前单元不存在重复节点, 即第一数量与第二数量相同, 此时不需要编解码当前单元的体素节点, 从而在保证点云属性的编解码效率基础上, 可以降低点云属性编解码的时间复杂度, 并且还可以节省码率, 进而提升了点云的编解码性能。

40 在本申请的再一实施例中, 基于前述实施例相同的发明构思, 参见图 49, 其示出了本申请实施例提供的一种解码器的组成结构示意图。如图 49 所示, 该解码器 490 可以包括第二确定单元 4901 和第二重建单元 4902, 其中:

第二确定单元 4901, 配置为确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量; 其中, 第一数量与第二数量用于确定是否对当前单元的体素节点进行跳过解码;

第二重建单元 4902, 配置为根据第一数量与第二数量, 确定当前单元的体素节点的属性重建值。

在一些实施例中, 参见图 49, 解码器 490 还可以包括解码单元 4903, 配置为若第一数量与第二数量相同, 则跳过解码当前单元的体素节点。

50 在一些实施例中, 第二重建单元 4902, 还配置为在第一数量与第二数量相同时, 将当前单元的体素节点的属性重建值设置为当前单元的重建节点的属性重建值。

在一些实施例中, 解码单元 4903, 还配置为若第一数量与第二数量不同, 则对体素节点中的重复节点进行属性解码, 以及跳过解码体素节点中除重复节点之外的剩余体素节点。

在一些实施例中,第二重建单元 4902,还配置为在第一数量与第二数量不同时,解码码流,确定重复节点的属性重建值;将重复节点的属性重建值作为当前单元中第一重建节点的属性重建值,以及将体素节点中除重复节点之外的剩余体素节点的属性重建值设置为当前单元中除第一重建节点之外的剩余重建节点的属性重建值。

5 在一些实施例中,解码单元 4903,还配置为设置计时器;在重复节点的属性重建值开始解码时,启动计时器,并在计时器的计时达到预设数值时,确定重复节点的属性重建值全部解码完成。

在一些实施例中,在第一数量与第二数量不同时,第二确定单元 4901,还配置为根据第一数量与第二数量之间的差值,确定体素节点中的重复节点的第三数量;其中,计时器的设置与第三数量有关联关系。

10 在一些实施例中,第二确定单元 4901,还配置为对当前单元中的节点进行划分,确定至少一层;以及在最后一层的节点被划分到体素级别时,确定当前单元的体素节点以及体素节点的第一数量。

在一些实施例中,至少一层包括当前层,第二确定单元 4901,还配置为确定当前层的节点的第四数量和当前层的节点对应的子节点的第五数量;其中,第四数量和第五数量用于确定是否对当前层进行跳过解码;

15 第二重建单元 4902,还配置为根据第四数量和第五数量,确定当前层的节点对应的子节点的属性重建值。

在一些实施例中,第四数量表征当前层的被占据节点的数量;第五数量表征当前层的节点中被占据的子节点的数量或待解码的节点的数量。

20 在一些实施例中,第二重建单元 4902,还配置为若第四数量和第五数量相同,则将当前层的节点的属性重建值确定为当前层的节点对应的子节点的属性重建值。

在一些实施例中,第二重建单元 4902,还配置为若第四数量和第五数量相同,则确定子节点对应的下一层子节点的第六数量;根据第五数量和第六数量,确定子节点对应的下一层子节点的属性重建值。

25 在一些实施例中,第二重建单元 4902,还配置为若第四数量和第五数量不同,则根据当前层的节点确定当前层的节点对应的子节点的属性预测值;基于子节点的属性预测值进行 RAHT 变换,确定当前层的节点对应的高频系数的重建值和低频系数;以及基于高频系数的重建值和低频系数进行 RAHT 逆变换,确定子节点的属性重建值。

在一些实施例中,第二确定单元 4901,还配置为确定当前层的节点对应的相邻节点;以及根据相邻节点对应的属性重建值和相对距离参数,确定当前层的节点对应的子节点的属性预测值。

30 在一些实施例中,第二重建单元 4902,还配置为基于子节点的属性预测值进行 RAHT 变换,确定当前层的节点对应的高频系数的预测值和低频系数;以及根据高频系数的预测值确定当前层的节点对应的高频系数的重建值。

在一些实施例中,解码单元 4903,还配置为解码码流,确定当前层的节点对应的量化后的系数残差;

35 第二确定单元 4901,还配置为根据高频系数的预测值和量化后的系数残差,确定当前层的节点对应的高频系数的重建值。

在一些实施例中,第二重建单元 4902,还配置为对量化后的系数残差进行反量化,确定当前层的节点对应的反量化残差值;以及根据当前层的节点对应的反量化残差值和当前层的节点对应的高频系数的预测值,确定当前层的节点对应的高频系数的重建值。

40 在一些实施例中,第二确定单元 4901,还配置为确定当前层的节点的几何信息;以及根据几何信息确定当前层的节点对应的子节点,以及第五数量。

在一些实施例中,解码单元 4903,还配置为解码码流,确定预测模式标识信息;以及在预测模式标识信息指示当前单元启动跳过解码模式时,执行第一数量和第二数量的确定步骤,和/或,执行第四数量和第五数量的确定步骤。

45 在一些实施例中,预测模式标识信息至少为以下高层语法元素中的其中之一:属性参数集对应的语法元素和属性块头信息对应的语法元素。

可以理解地,在本实施例中,“单元”可以是部分电路、部分处理器、部分程序或软件等等,当然也可以是模块,还可以是非模块化的。而且在本实施例中的各组成部分可以集成在一个处理单元中,也可以是各个单元单独物理存在,也可以两个或两个以上单元集成在一个单元中。上述集成的单元既可以采用硬件的形式实现,也可以采用软件功能模块的形式实现。

50 所述集成的单元如果以软件功能模块的形式实现并非作为独立的产品进行销售或使用,可以存储在一个计算机可读取存储介质中。基于这样的理解,本实施例提供了一种计算机可读存储介质,应用于解码器 490,该计算机可读存储介质存储有计算机程序,所述计算机程序被第二处理器执行时实现前述

实施例中任一项所述的方法。

基于解码器 490 的组成以及计算机可读存储介质, 参见图 50, 其示出了本申请实施例提供的解码器 490 的具体硬件结构示意图。如图 50 所示, 解码器 490 可以包括: 第二通信接口 5001、第二存储器 5002 和第二处理器 5003; 各个组件通过第二总线系统 5004 耦合在一起。可理解, 第二总线系统 5004 用于实现这些组件之间的连接通信。第二总线系统 5004 除包括数据总线之外, 还包括电源总线、控制总线

和状态信号总线。但是为了清楚说明起见, 在图 50 中将各种总线都标为第二总线系统 5004。其中:

第二通信接口 5001, 用于在与其他外部网元之间进行收发信息过程中, 信号的接收和发送;

第二存储器 5002, 用于存储能够在第二处理器 5003 上运行的计算机程序;

第二处理器 5003, 用于在运行所述计算机程序时, 执行:

确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量; 其中, 第一数量与第二数量

用于确定是否对当前单元的体素节点进行跳过解码;

根据第一数量与第二数量, 确定当前单元的体素节点的属性重建值。

可选地, 作为另一个实施例, 第二处理器 5003 还配置为在运行所述计算机程序时, 执行前述实施例中任一项所述的方法。

可以理解, 第二存储器 5002 与第一存储器 4802 的硬件功能类似, 第二处理器 5003 与第一处理器 4803 的硬件功能类似; 这里不再详述。

本实施例提供了一种解码器, 在该解码器中, 在对每个体素节点进行属性重建时, 优化了每个体素节点是否进行属性编解码的判断条件, 具体为如果当前单元不存在重复节点, 即第一数量与第二数量相同, 此时不需要编解码当前单元的体素节点, 从而在保证点云属性的编解码效率基础上, 可以降低点云属性编解码的时间复杂度, 并且还可以节省码率, 进而提升了点云的编解码性能。

在本申请的再一实施例中, 参见图 51, 其示出了本申请实施例提供的一种编解码系统的组成结构示意图。如图 51 所示, 编解码系统 510 可以包括编码器 5101 和解码器 5102。

在本申请实施例中, 编码器 5101 可以是前述实施例中任一项所述的编码器, 解码器 5102 可以是前述实施例中任一项所述的解码器。

需要说明的是, 在本申请中, 术语“包括”、“包含”或者其任何其他变体意在涵盖非排他性的包含, 从而使得包括一系列要素的过程、方法、物品或者装置不仅包括那些要素, 而且还包括没有明确列出的其他要素, 或者是还包括为这种过程、方法、物品或者装置所固有的要素。在没有更多限制的情况下, 由语句“包括一个……”限定的要素, 并不排除在包括该要素的过程、方法、物品或者装置中还存在另外的相同要素。

上述本申请实施例序号仅仅为了描述, 不代表实施例的优劣。

本申请所提供的几个方法实施例中揭露的方法, 在不冲突的情况下可以任意组合, 得到新的方法实施例。

本申请所提供的几个产品实施例中揭露的特征, 在不冲突的情况下可以任意组合, 得到新的产品实施例。

本申请所提供的几个方法或设备实施例中揭露的特征, 在不冲突的情况下可以任意组合, 得到新的方法实施例或设备实施例。

以上所述, 仅为本申请的具体实施方式, 但本申请的保护范围并不局限于此, 任何熟悉本技术领域的技术人员在本申请揭露的技术范围内, 可轻易想到变化或替换, 都应涵盖在本申请的保护范围之内。因此, 本申请的保护范围应以所述权利要求的保护范围为准。

工业实用性

本申请实施例中, 无论是编码端还是解码端, 确定当前单元的体素节点的第一数量和当前单元的重建节点的第二数量, 然后根据第一数量与第二数量, 确定当前单元的体素节点的属性重建值; 其中, 第一数量与第二数量用于确定当前单元的体素节点是否跳过编解码。以及确定当前层的节点的第四数量和当前层的节点对应的子节点的第五数量; 根据第四数量和第五数量, 确定当前层的节点对应的子节点的属性重建值; 其中, 第四数量和第五数量用于确定当前层是否跳过编解码。这样, 可以利用当前层的节点数目与子节点数目来自适应地决定当前层是否可以跳过解码, 还可以在重建节点数目与体素节点数目一致时, 认为当前单元中不存在重复节点, 同样可以跳过解码体素级别的节点属性; 如此, 在保证点云属性的编解码效率基础上, 可以降低点云属性编解码的时间复杂度, 并且还可以节省码率, 进而提升了点云的编解码性能。

权利要求书

1、一种解码方法，应用于解码器，所述方法包括：

确定当前单元的体素节点的第一数量和所述当前单元的重建节点的第二数量；其中，所述第一数量与所述第二数量用于确定是否对所述当前单元的体素节点进行跳过解码；

5 根据所述第一数量与所述第二数量，确定所述当前单元的体素节点的属性重建值。

2、根据权利要求1所述的方法，其中，所述方法还包括：

若所述第一数量与所述第二数量相同，则跳过解码所述当前单元的体素节点。

3、根据权利要求2所述的方法，其中，所述根据所述第一数量与所述第二数量，确定所述当前单元的体素节点的属性重建值，包括：

10 在所述第一数量与所述第二数量相同时，将所述当前单元的体素节点的属性重建值设置为所述当前单元的重建节点的属性重建值。

4、根据权利要求1所述的方法，其中，所述方法还包括：

若所述第一数量与所述第二数量不同，则对所述体素节点中的重复节点进行属性解码，以及跳过解码所述体素节点中除所述重复节点之外的剩余体素节点。

15 5、根据权利要求4所述的方法，其中，所述根据所述第一数量与所述第二数量，确定所述当前单元的体素节点的属性重建值，包括：

在所述第一数量与所述第二数量不同时，解码码流，确定所述重复节点的属性重建值；

将所述重复节点的属性重建值作为所述当前单元中第一重建节点的属性重建值，以及将所述体素节点中除所述重复节点之外的剩余体素节点的属性重建值设置为所述当前单元中除所述第一重建节点之外的剩余重建节点的属性重建值。

20 6、根据权利要求5所述的方法，其中，所述解码码流，确定所述重复节点的属性重建值，包括：设置计时器；

在所述重复节点的属性重建值开始解码时，启动所述计时器，并在所述计时器的计时达到预设数值时，确定所述重复节点的属性重建值全部解码完成。

25 7、根据权利要求6所述的方法，其中，在所述第一数量与所述第二数量不同时，所述方法还包括：根据所述第一数量与所述第二数量之间的差值，确定所述体素节点中的重复节点的第三数量；其中，所述计时器的设置与所述第三数量有关联关系。

8、根据权利要求1至7中任一项所述的方法，其中，所述确定当前单元的体素节点的第一数量，包括：

30 对所述当前单元中的节点进行划分，确定至少一层；

在最后一层的节点被划分到体素级别时，确定所述当前单元的体素节点以及所述体素节点的第一数量。

9、根据权利要求8所述的方法，其中，所述至少一层包括当前层，所述方法还包括：

35 确定所述当前层的节点的第四数量和所述当前层的节点对应的子节点的第五数量；其中，所述第四数量和所述第五数量用于确定是否对所述当前层进行跳过解码；

根据所述第四数量和所述第五数量，确定所述当前层的节点对应的子节点的属性重建值。

10、根据权利要求9所述的方法，其中，

所述第四数量表征所述当前层的被占据节点的数量；

所述第五数量表征所述当前层的节点中被占据的子节点的数量或待解码的节点的数量。

40 11、根据权利要求9所述的方法，其中，所述根据所述第四数量和所述第五数量，确定所述当前层的节点对应的子节点的属性重建值，包括：

若所述第四数量和所述第五数量相同，则将所述当前层的节点的属性重建值确定为所述当前层的节点对应的子节点的属性重建值。

45 12、根据权利要求9所述的方法，其中，所述根据所述第四数量和所述第五数量，确定所述当前层的节点对应的子节点的属性重建值，包括：

若所述第四数量和所述第五数量相同，则确定所述子节点对应的下一层子节点的第六数量；

根据所述第五数量和所述第六数量，确定所述子节点对应的下一层子节点的属性重建值。

13、根据权利要求9所述的方法，其中，所述根据所述第四数量和所述第五数量，确定所述当前层的节点对应的子节点的属性重建值，包括：

50 若所述第四数量和所述第五数量不同，则根据所述当前层的节点确定所述当前层的节点对应的子节

点的属性预测值;

基于所述子节点的属性预测值进行 RAHT 变换, 确定所述当前层的节点对应的高频系数的重建值和低频系数;

基于所述高频系数的重建值和所述低频系数进行 RAHT 逆变换, 确定所述子节点的属性重建值。

5 14、根据权利要求 13 所述的方法, 其中, 所述根据所述当前层的节点确定所述当前层的节点对应的子节点的属性预测值, 包括:

确定所述当前层的节点对应的相邻节点;

根据所述相邻节点对应的属性重建值和相对距离参数, 确定所述当前层的节点对应的子节点的属性预测值。

10 15、根据权利要求 13 所述的方法, 其中, 所述基于所述子节点的属性预测值进行 RAHT 变换, 确定所述当前层的节点对应的高频系数的重建值和低频系数, 包括:

基于所述子节点的属性预测值进行 RAHT 变换, 确定所述当前层的节点对应的高频系数的预测值和所述低频系数;

根据所述高频系数的预测值确定所述当前层的节点对应的所述高频系数的重建值。

15 16、根据权利要求 15 所述的方法, 其中, 所述根据所述高频系数的预测值确定所述当前层的节点对应的所述高频系数的重建值, 包括:

解码码流, 确定所述当前层的节点对应的量化后的系数残差;

根据所述高频系数的预测值和所述量化后的系数残差, 确定所述当前层的节点对应的所述高频系数的重建值。

20 17、根据权利要求 16 所述的方法, 其中, 所述根据所述高频系数的预测值和所述量化后系数残差, 确定所述当前层的节点对应的所述高频系数的重建值, 包括:

对所述量化后的系数残差进行反量化, 确定所述当前层的节点对应的反量化残差值;

根据所述当前层的节点对应的反量化残差值和所述当前层的节点对应的高频系数的预测值, 确定所述当前层的节点对应的高频系数的重建值。

25 18、根据权利要求 9 所述的方法, 其中, 所述方法还包括:

确定所述当前层的节点的几何信息;

根据所述几何信息确定所述当前层的节点对应的子节点, 以及所述第五数量。

19、根据权利要求 9 所述的方法, 其中, 所述方法还包括:

解码码流, 确定预测模式标识信息;

30 在所述预测模式标识信息指示所述当前单元启动跳过解码模式时, 执行所述第一数量和所述第二数量的确定步骤, 和/或, 执行所述第四数量和所述第五数量的确定步骤。

20、根据权利要求 19 所述的方法, 其中, 所述预测模式标识信息至少为以下高层语法元素中的其中之一: 属性参数集对应的语法元素和属性块头信息对应的语法元素。

21、一种编码方法, 应用于编码器, 所述方法包括:

35 确定当前单元的体素节点的第一数量和所述当前单元的重建节点的第二数量;

根据所述第一数量与所述第二数量, 确定所述当前单元的体素节点的属性信息是否跳过编码。

22、根据权利要求 21 所述的方法, 其中, 所述根据所述第一数量与所述第二数量, 确定所述当前单元的体素节点的属性信息是否跳过编码, 包括:

若所述第一数量与所述第二数量相同, 则跳过编码所述体素节点的属性信息;

40 若所述第一数量与所述第二数量不同, 则对所述体素节点中的重复节点进行属性编码, 以及跳过编码所述体素节点中除所述重复节点之外剩余体素节点的属性信息。

23、根据权利要求 22 所述的方法, 其中, 所述对所述体素节点中的重复节点进行属性编码, 包括:

对所述重复节点的属性信息进行编码处理, 将所得到的编码比特写入码流。

24、根据权利要求 23 所述的方法, 其中, 所述对所述重复节点的属性信息进行编码处理, 包括:

45 设置计时器;

在所述重复节点的属性信息开始编码时, 启动所述计时器, 并在所述计时器的计时达到预设数值时, 确定所述重复节点的属性信息全部编码完成。

25、根据权利要求 24 所述的方法, 其中, 所述方法还包括:

50 根据所述第一数量与所述第二数量之间的差值, 确定所述体素节点中的重复节点的第三数量; 其中, 所述计时器的设置与所述第三数量有关联关系。

26、根据权利要求 21 所述的方法, 其中, 所述方法还包括:

根据所述第一数量与所述第二数量, 确定所述当前单元的体素节点的属性重建值。

27、根据权利要求 26 所述的方法，其中，所述根据所述第一数量与所述第二数量，确定所述当前单元的体素节点的属性重建值，包括：

在所述第一数量与所述第二数量相同时，将所述当前单元的体素节点的属性重建值设置为所述当前单元的重建节点的属性重建值。

5 28、根据权利要求 26 所述的方法，其中，所述根据所述第一数量与所述第二数量，确定所述当前单元的体素节点的属性重建值，包括：

在所述第一数量与所述第二数量不同时，将所述重复节点的属性重建值作为所述当前单元中第一重建节点的属性重建值，以及将所述体素节点中除所述重复节点之外的剩余体素节点的属性重建值设置为所述当前单元中除所述第一重建节点之外的剩余重建节点的属性重建值。

10 29、根据权利要求 21 至 28 中任一项所述的方法，其中，所述确定当前单元的体素节点的第一数量，包括：

对所述当前单元中的节点进行划分，确定至少一层；

在最后一层的节点被划分到体素级别时，确定所述当前单元的体素节点以及所述体素节点的第一数量。

15 30、根据权利要求 29 所述的方法，其中，所述至少一层包括当前层，所述方法还包括：

确定所述当前层的节点的第四数量和所述当前层的节点对应的子节点的第五数量；其中，所述第四数量和所述第五数量用于确定是否对所述当前层进行跳过编码；

根据所述第四数量和所述第五数量，确定所述当前层的节点对应的子节点的属性重建值。

20 31、根据权利要求 30 所述的方法，其中，

所述第四数量表征所述当前层的被占据节点的数量；

所述第五数量表征所述当前层的节点中被占据的子节点的数量或待编码的节点的数量。

32、根据权利要求 30 所述的方法，其中，所述根据所述第四数量和所述第五数量，确定所述当前层的节点对应的子节点的属性重建值，包括：

25 若所述第四数量和所述第五数量相同，则将所述当前层的节点的属性重建值确定为所述当前层的节点对应的子节点的属性重建值。

33、根据权利要求 30 所述的方法，其中，所述根据所述第四数量和所述第五数量，确定所述当前层的节点对应的子节点的属性重建值，包括：

若所述第四数量和所述第五数量相同，则确定所述子节点对应的下一层子节点的第六数量；

根据所述第五数量和所述第六数量，确定所述子节点对应的下一层子节点的属性重建值。

30 34、根据权利要求 30 所述的方法，其中，所述根据所述第四数量和所述第五数量，确定所述当前层的节点对应的子节点的属性重建值，包括：

若所述第四数量和所述第五数量不同，则根据所述当前层的节点确定所述当前层的节点对应的子节点的属性预测值；

35 基于所述子节点的属性预测值和所述子节点的属性值分别进行 RAHT 变换，确定所述当前层的节点对应的高频系数的重建值和低频系数；

基于所述高频系数的重建值和所述低频系数进行 RAHT 逆变换，确定所述子节点的属性重建值。

35、根据权利要求 34 所述的方法，其中，所述根据所述当前层的节点确定所述当前层的节点对应的子节点的属性预测值，包括：

确定所述当前层的节点对应的相邻节点；

40 根据所述相邻节点对应的属性重建值和相对距离参数，确定所述当前层的节点对应的子节点的属性预测值。

36、根据权利要求 34 所述的方法，其中，所述基于所述子节点的属性预测值和所述子节点的属性值分别进行 RAHT 变换，确定所述当前层的节点对应的高频系数的重建值和低频系数，包括：

45 基于所述子节点的属性预测值进行 RAHT 变换，确定所述当前层的节点对应的高频系数的预测值和所述低频系数；

基于所述子节点的属性值进行 RAHT 变换，确定所述当前层的节点对应的高频系数和所述低频系数；

根据所述当前层的节点对应的高频系数的预测值和所述当前层的节点对应的高频系数，确定所述当前层的节点对应的高频系数的重建值。

50 37、根据权利要求 36 所述的方法，其中，所述根据所述当前层的节点对应的高频系数的预测值和所述当前层的节点对应的高频系数，确定所述当前层的节点对应的高频系数的重建值，包括：

根据所述当前层的节点对应的高频系数的预测值和所述当前层的节点对应的高频系数，确定所述当

前层的节点对应的系数残差;

根据所述当前层的节点对应的系数残差,确定所述当前层的节点对应的高频系数的重建值。

38、根据权利要求 37 所述的方法,其中,所述方法还包括:

对所述系数残差进行量化,确定所述当前层的节点对应的量化后的系数残差;

5 对所述量化后的系数残差进行编码处理,将所得到的编码比特写入码流。

39、根据权利要求 38 所述的方法,其中,所述根据所述当前层的节点对应的系数残差,确定所述当前层的节点对应的高频系数的重建值,包括:

对所述量化后的系数残差进行反量化,确定所述当前层的节点对应的反量化残差值;

10 根据所述当前层的节点对应的反量化残差值和所述当前层的节点对应的高频系数的预测值,确定所述当前层的节点对应的高频系数的重建值。

40、根据权利要求 30 所述的方法,其中,所述方法还包括:

确定所述当前层的节点的几何信息;

根据所述几何信息确定所述当前层的节点对应的子节点,以及所述第五数量。

41、根据权利要求 30 所述的方法,其中,所述方法还包括:

15 确定预测模式标识信息;

在所述预测模式标识信息指示所述当前单元启动跳过编码模式时,执行所述第一数量和所述第二数量的确定步骤,和/或,执行所述第四数量和所述第五数量的确定步骤。

42、根据权利要求 41 所述的方法,其中,所述预测模式标识信息至少为以下高层语法元素中的其中之一:属性参数集对应的语法元素和属性块头信息对应的语法元素。

20 43、根据权利要求 41 所述的方法,其中,所述方法还包括:

对所述预测模式标识信息进行编码处理,将所得到的编码比特写入码流。

44、一种码流,其中,所述码流是根据待编码信息进行比特编码生成的;其中,待编码信息包括下述至少一项:

25 预测模式标识信息、当前单元的重复节点的属性信息和当前层的节点对应的量化后的系数残差;其中,所述预测模式标识信息用于指示所述当前单元是否启动跳过编码模式。

45、一种编码器,所述编码器包括第一确定单元和编码单元,其中:

所述第一确定单元,配置为确定当前单元的体素节点的第一数量和所述当前单元的重建节点的第二数量;

30 所述编码单元,配置为根据所述第一数量与所述第二数量,确定所述当前单元的体素节点的属性信息是否跳过编码。

46、一种编码器,所述编码器包括第一存储器和第一处理器,其中:

所述第一存储器,用于存储能够在所述第一处理器上运行的计算机程序;

所述第一处理器,用于在运行所述计算机程序时,执行如权利要求 21 至 43 中任一项所述的方法。

47、一种解码器,所述解码器包括第二确定单元和第二重建单元,其中:

35 所述第二确定单元,配置为确定当前单元的体素节点的第一数量和所述当前单元的重建节点的第二数量;其中,所述第一数量与所述第二数量用于确定是否对所述当前单元的体素节点进行跳过解码;

所述第二重建单元,配置为根据所述第一数量与所述第二数量,确定所述当前单元的体素节点的属性重建值。

48、一种解码器,所述解码器包括第二存储器和第二处理器,其中:

40 所述第二存储器,用于存储能够在所述第二处理器上运行的计算机程序;

所述第二处理器,用于在运行所述计算机程序时,执行如权利要求 1 至 20 中任一项所述的方法。

49、一种计算机可读存储介质,其中,所述计算机可读存储介质存储有计算机程序,所述计算机程序被执行时实现如权利要求 1 至 20 中任一项所述的方法、或者实现如权利要求 21 至 43 中任一项所述的方法。

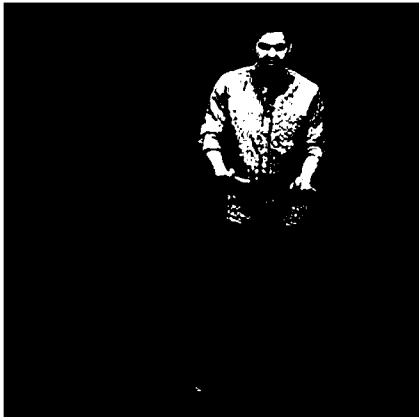


图 1A

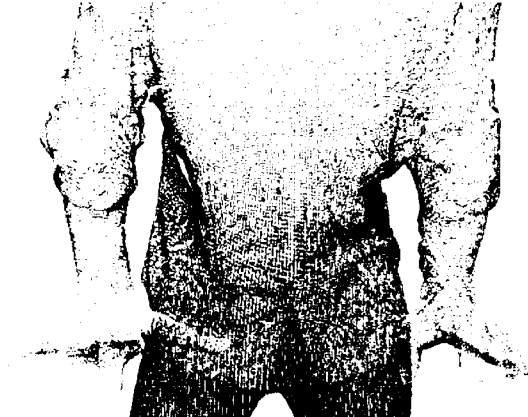


图 1B



图 2A

1	ply
2	format ascii 1.0
3	element vertex 207242
4	property float x
5	property float y
6	property float z
7	property uchar red
8	property uchar green
9	property uchar blue
10	end_header
11	75 318 0 0 142 0
12	75 319 0 0 143 0
13	75 319 1 1 9 9
14	77 315 0 1 9 9

图 2B

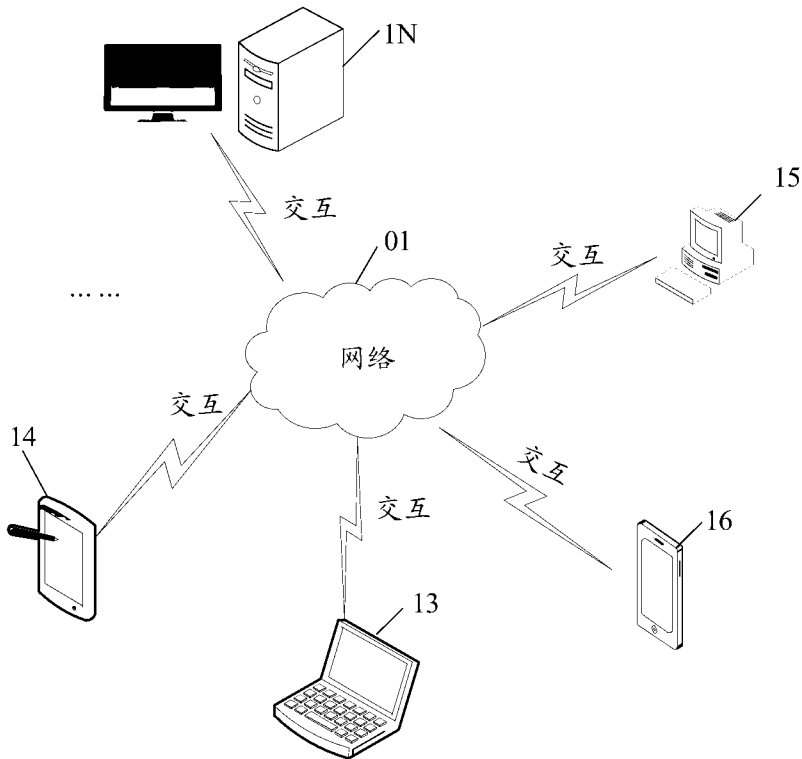


图 3

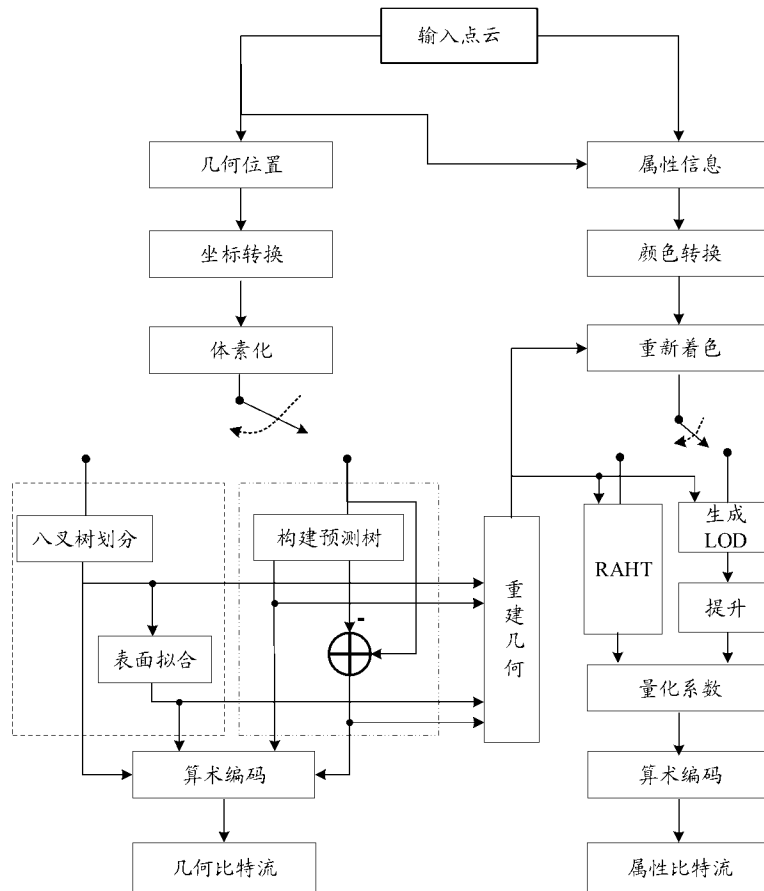


图 4A

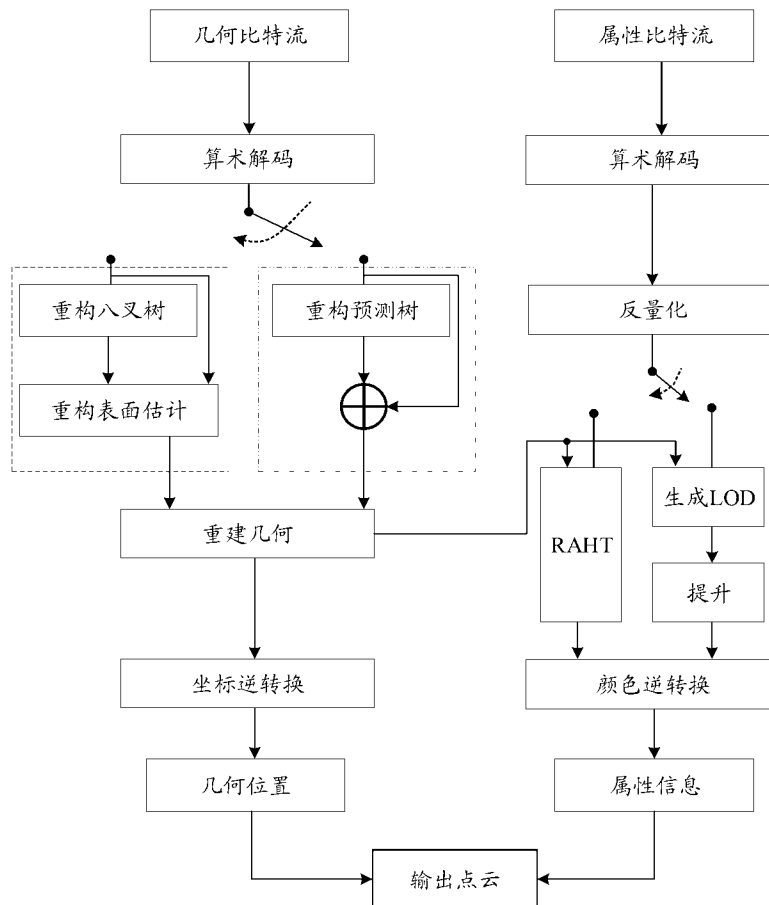
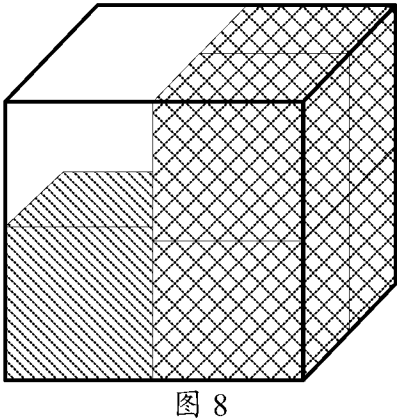
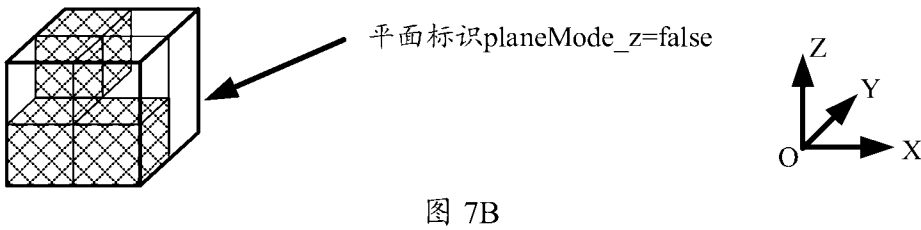
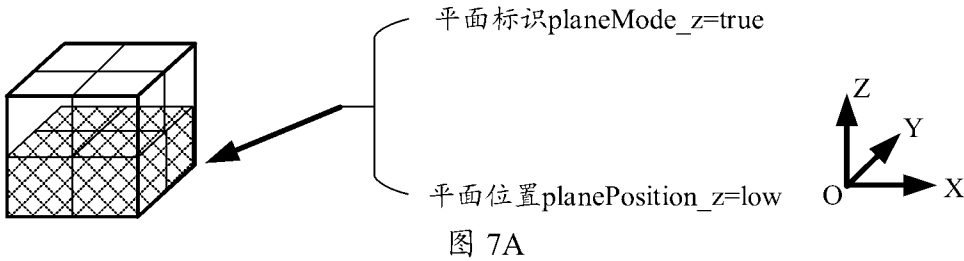
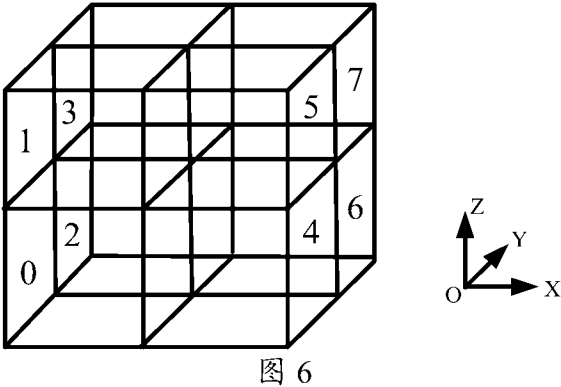
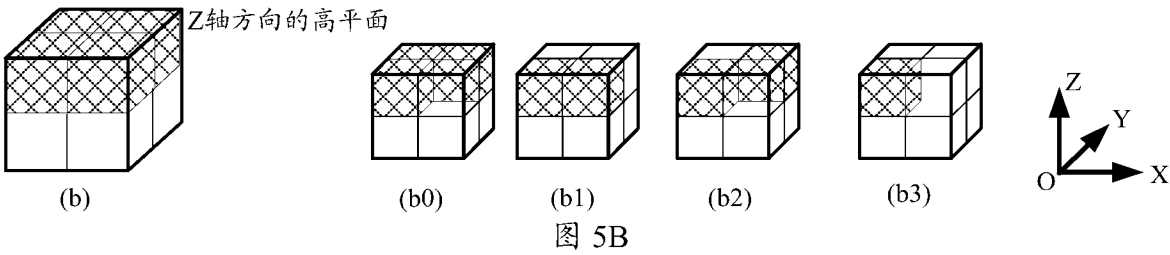
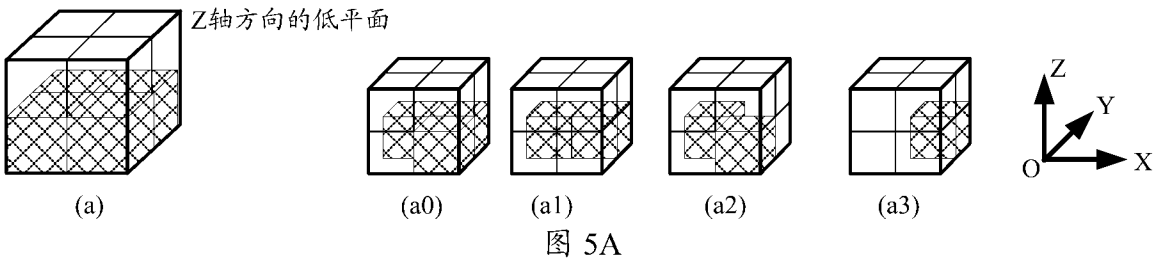


图 4B



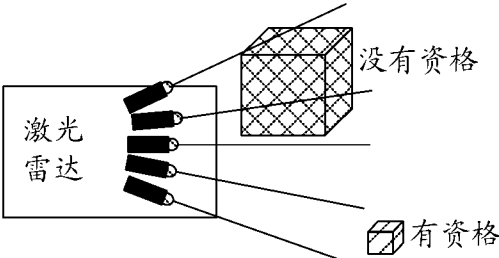


图 9

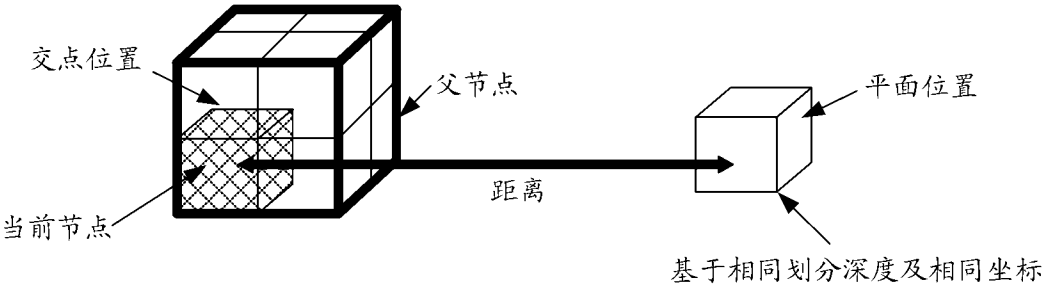


图 10

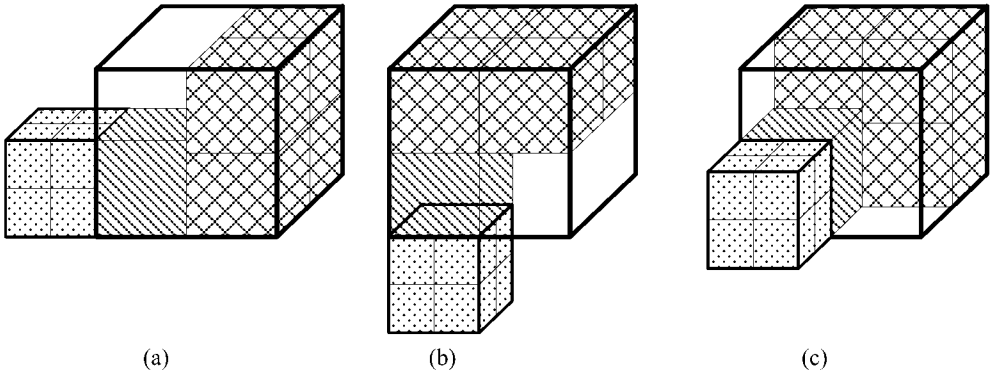


图 11

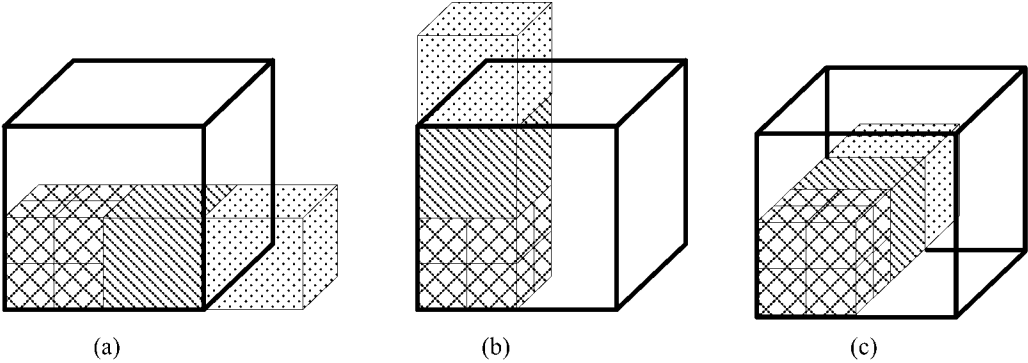


图 12

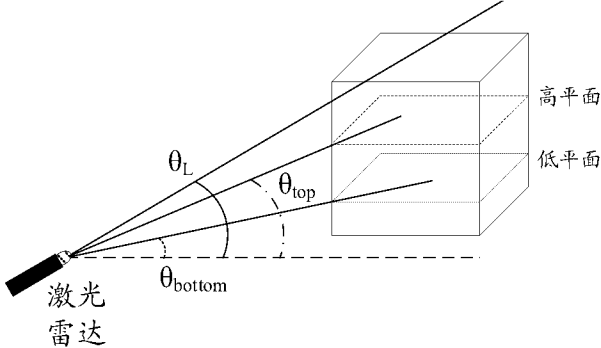


图 13

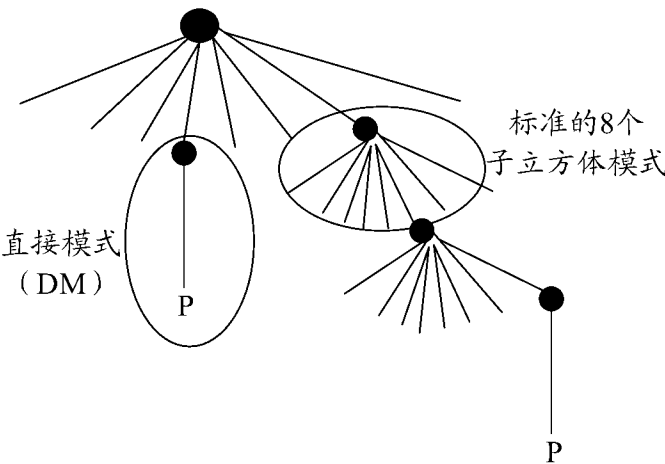


图 14

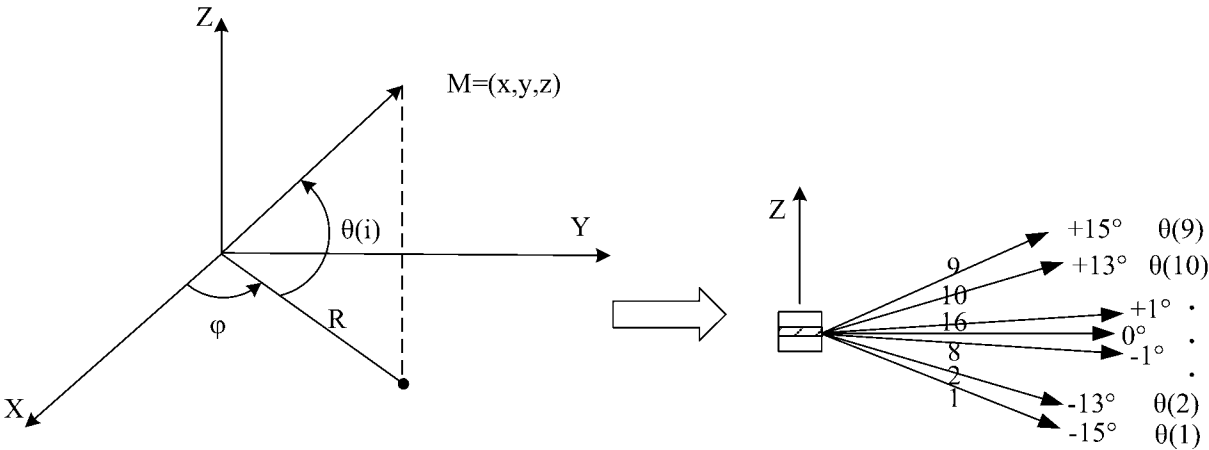


图 15

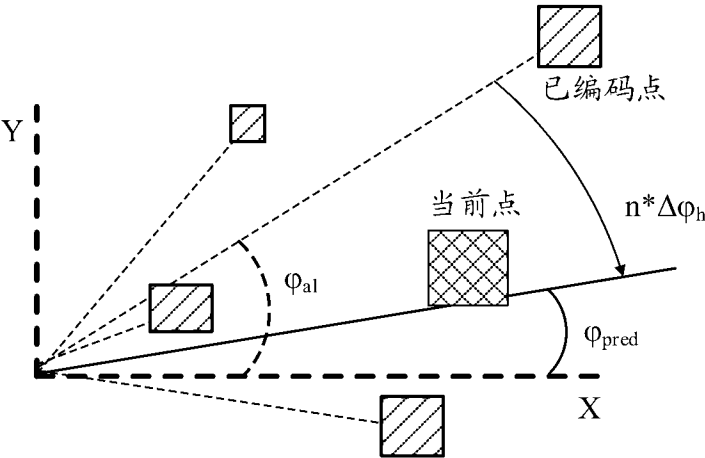


图 16

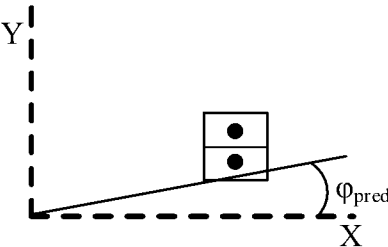


图 17A

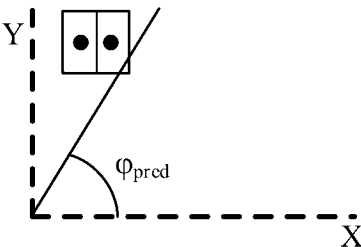


图 17B

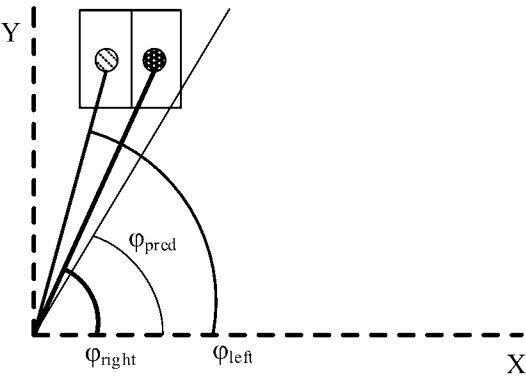


图 18

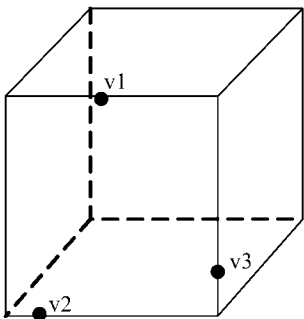


图 19A

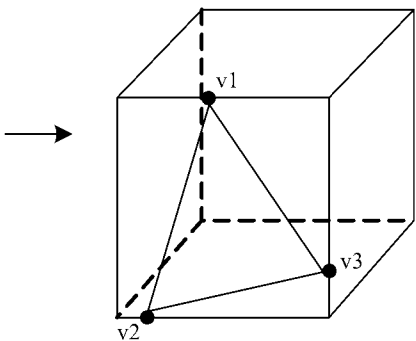


图 19B

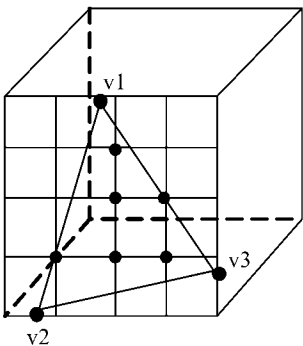
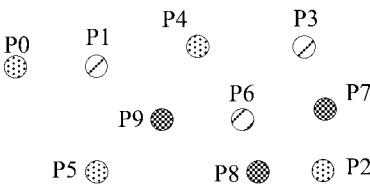


图 19C



原始顺序

P0, P1, P2, P3, P4, P5, P6, P7, P8, P9

LOD层顺序

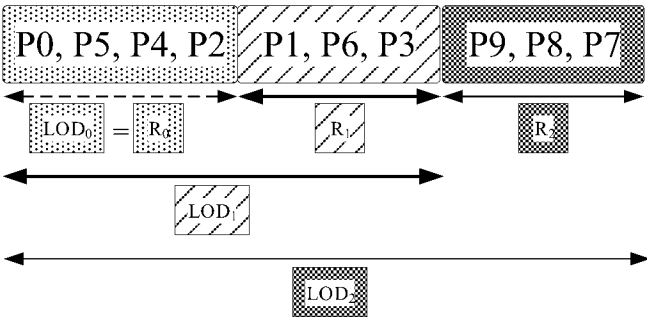


图 20

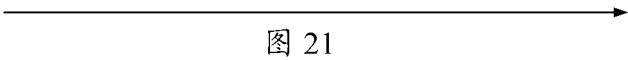


图 21

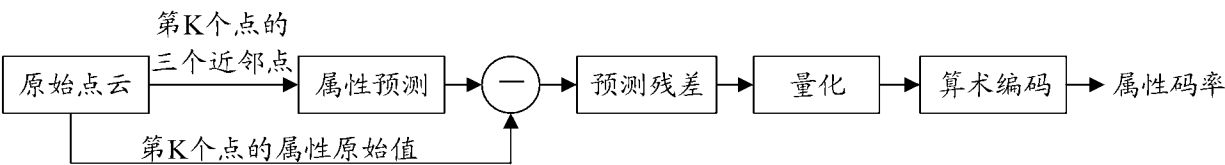


图 22

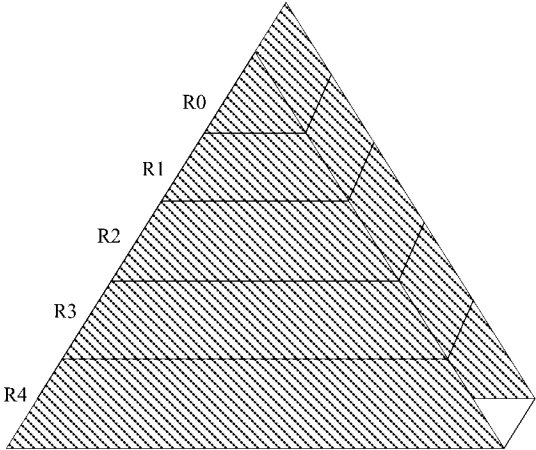


图 23

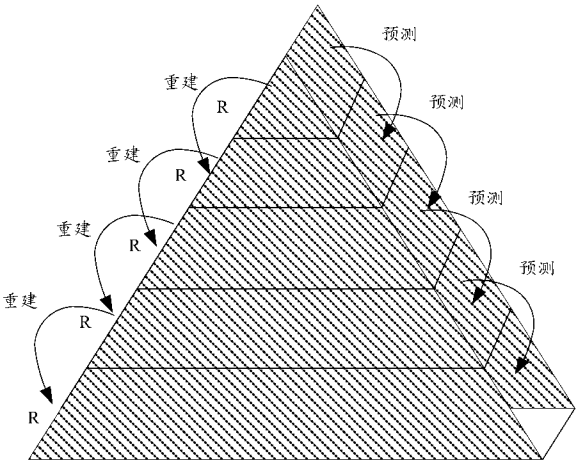


图 24

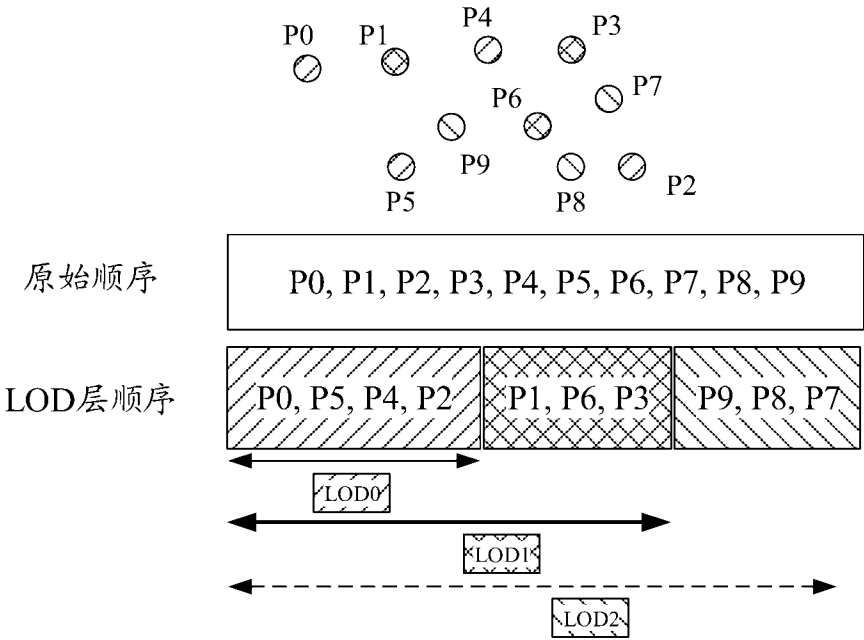


图 25

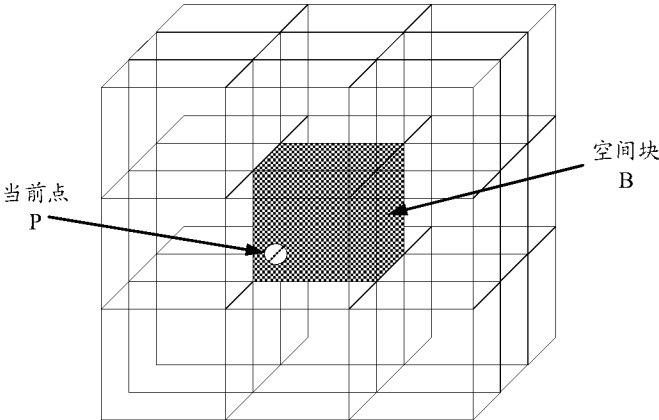


图 26

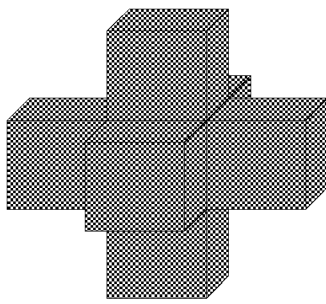


图 27A

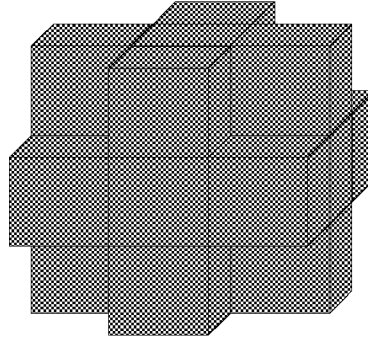


图 27B

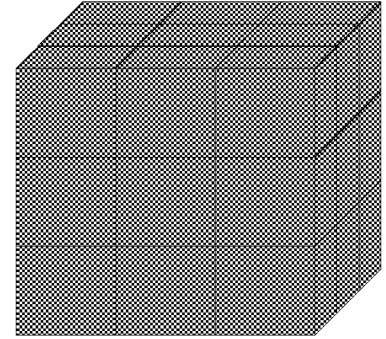


图 27C

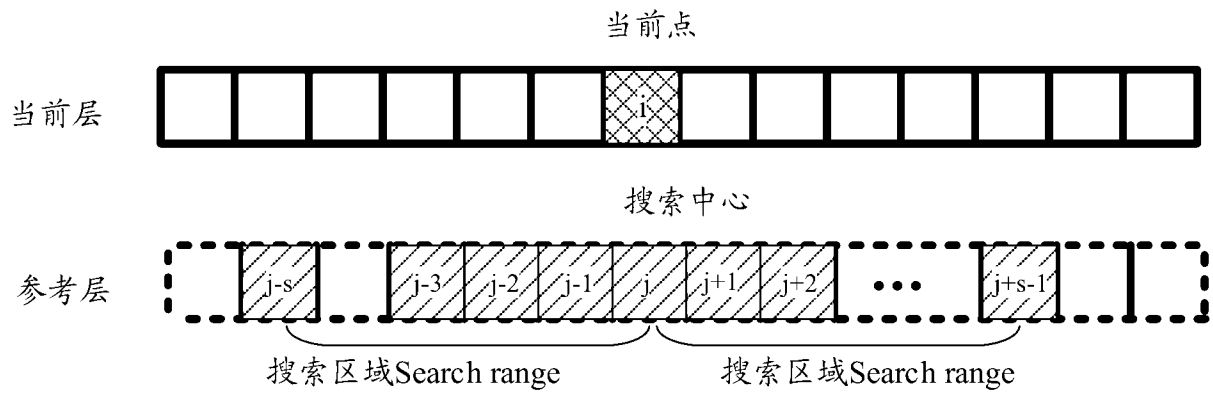


图 28

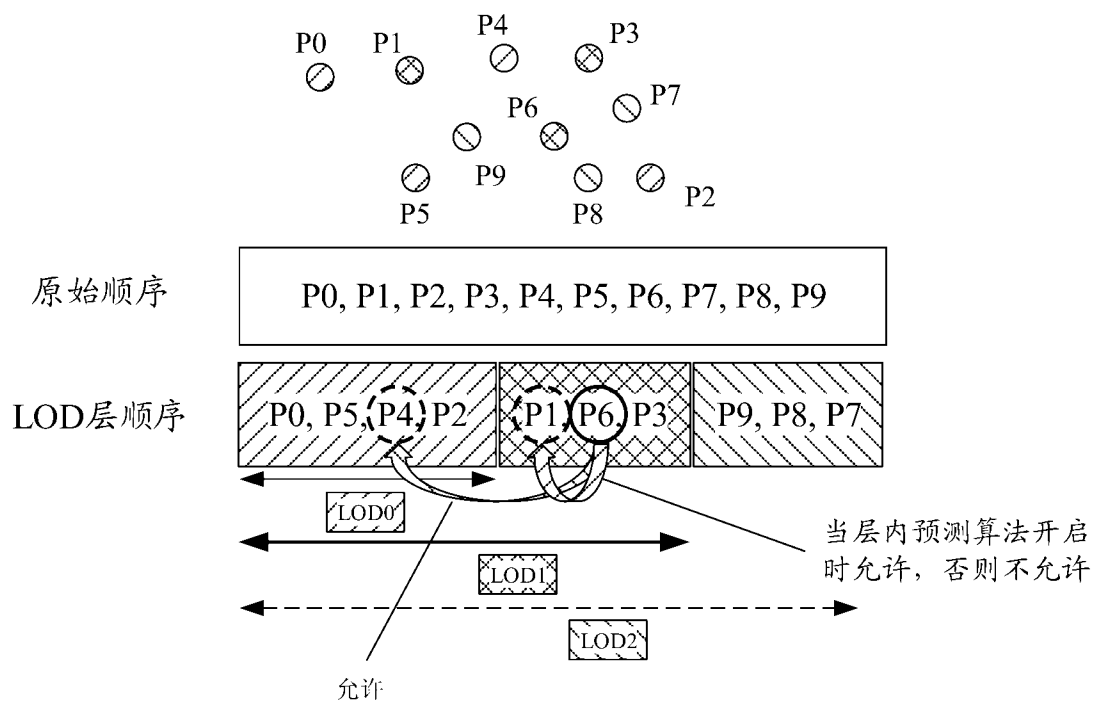


图 29

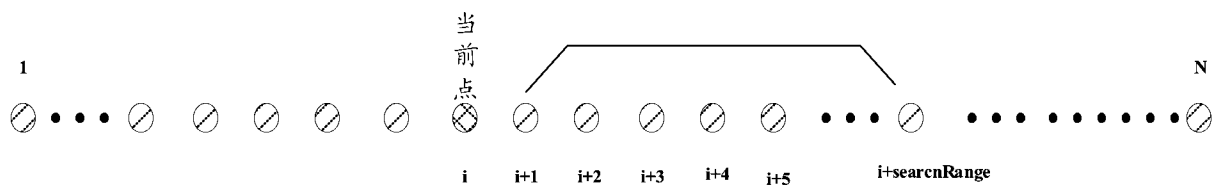


图 30

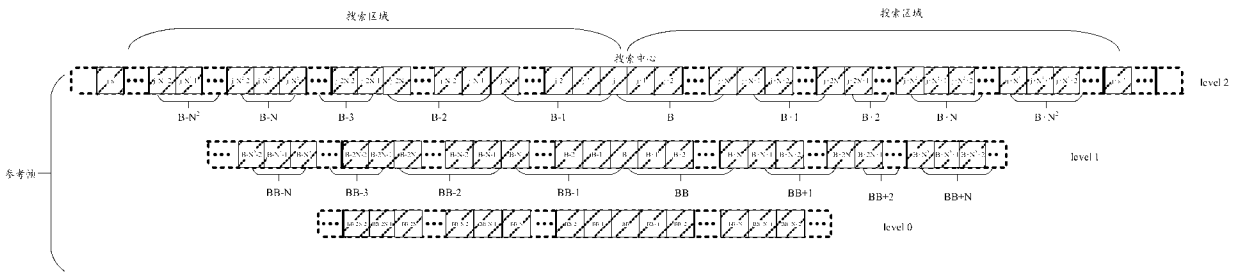


图 31

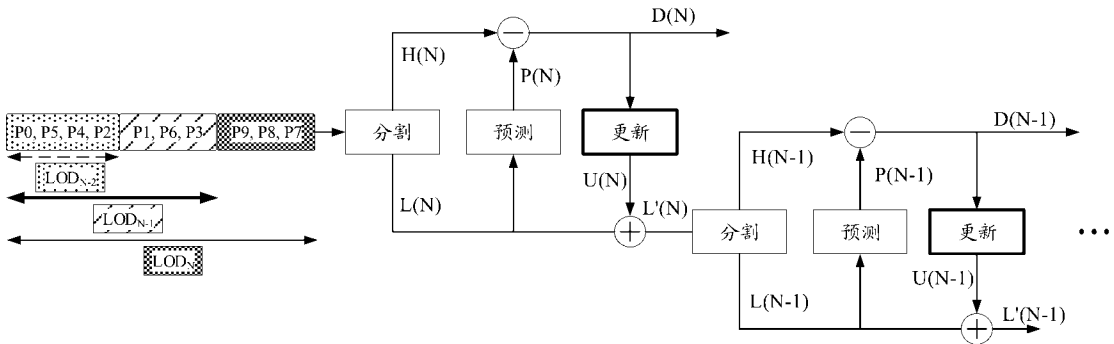


图 32

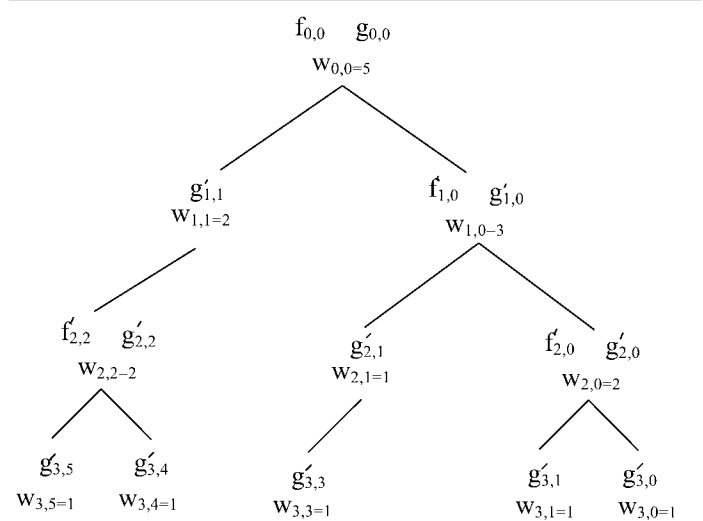
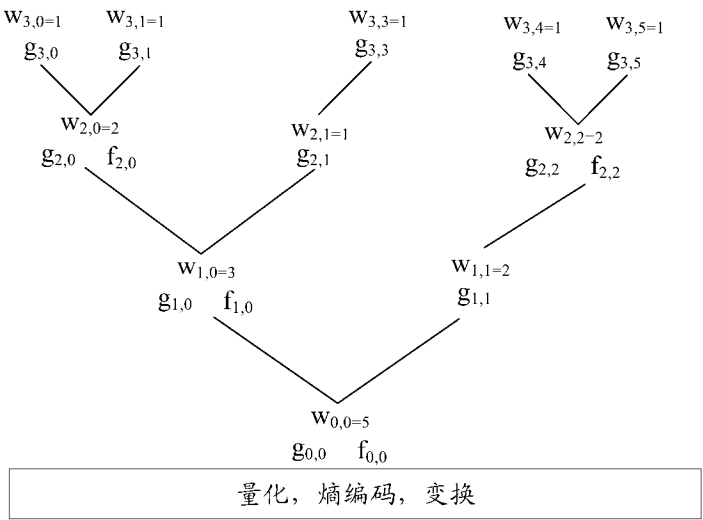


图 33

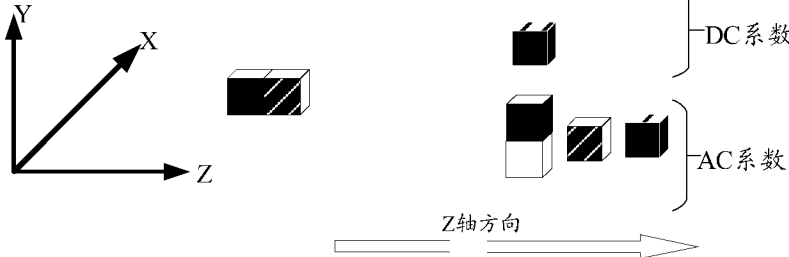
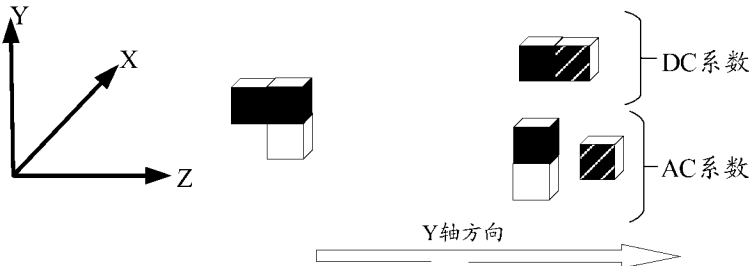
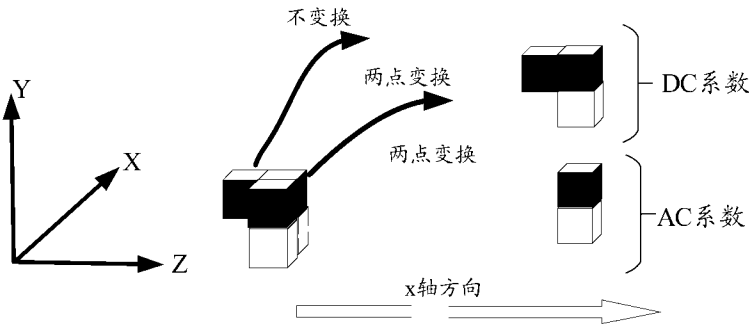


图 34

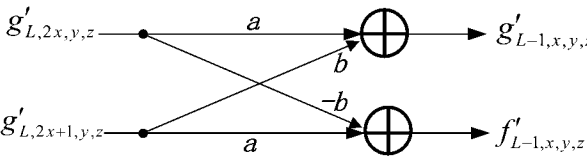


图 35A

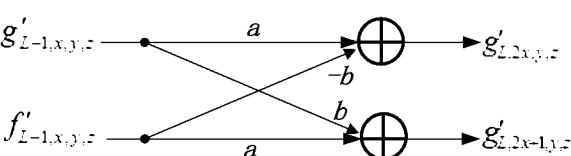


图 35B

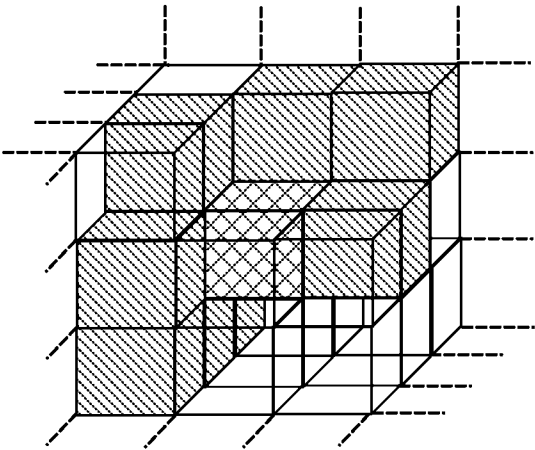


图 36

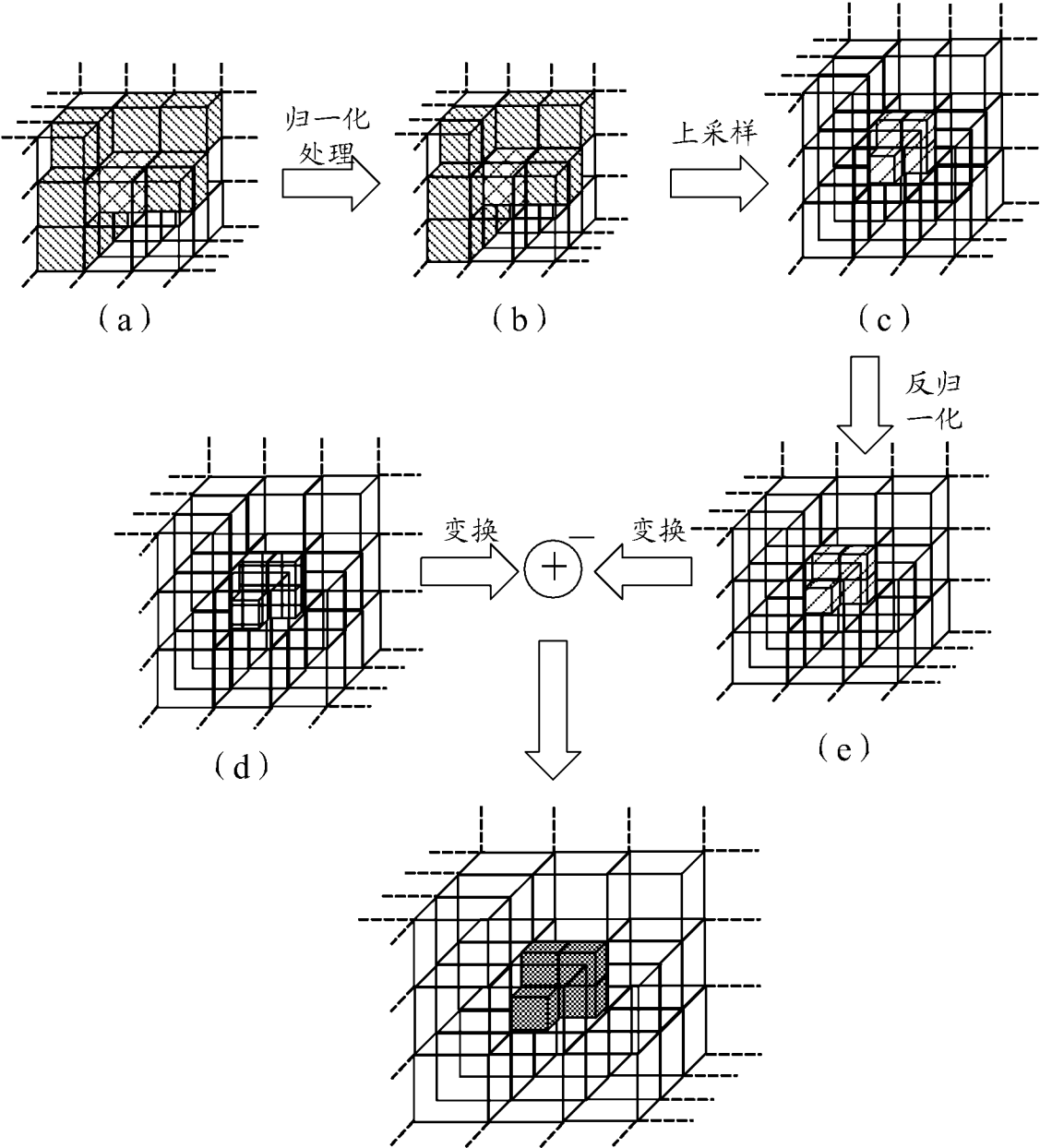


图 37

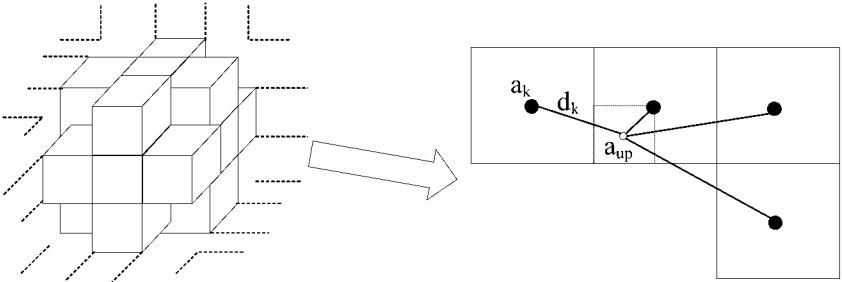
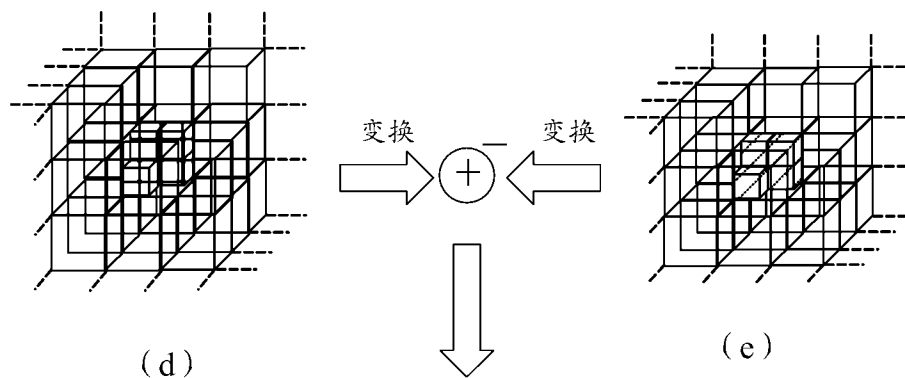


图 38



$$\begin{bmatrix} DC_{\text{depth } d-1} \\ AC_{1,\text{res}} \\ \vdots \\ AC_{k-1,\text{res}} \end{bmatrix} = \begin{bmatrix} DC_{\text{depth } d-1} \\ AC_{1,\text{orig}} \\ \vdots \\ AC_{k-1,\text{orig}} \end{bmatrix} - \begin{bmatrix} 0 \\ AC_{1,\text{up}} \\ \vdots \\ AC_{k-1,\text{up}} \end{bmatrix}$$

图 39

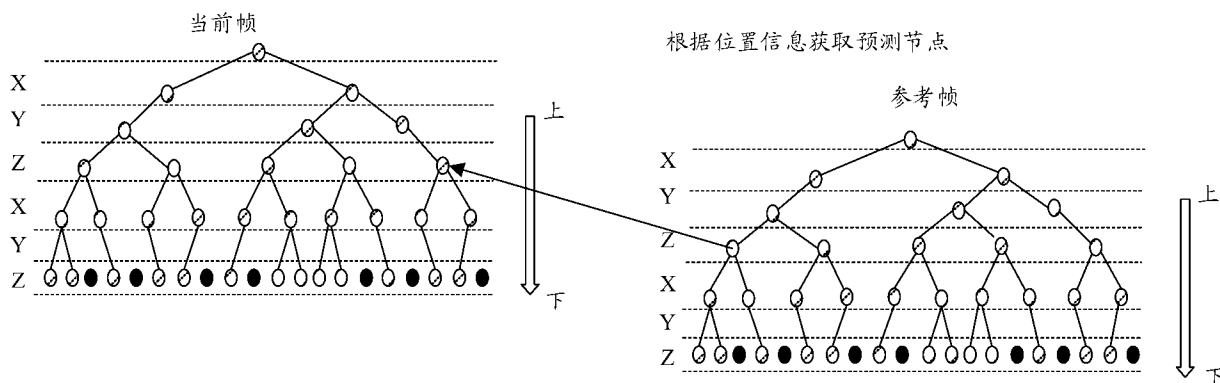


图 40

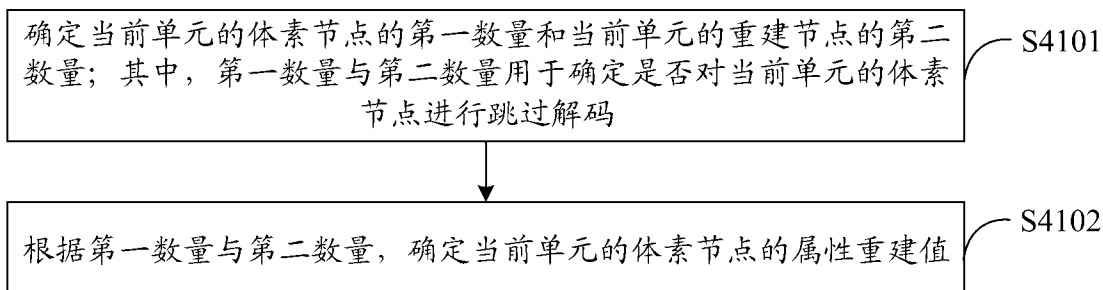


图 41

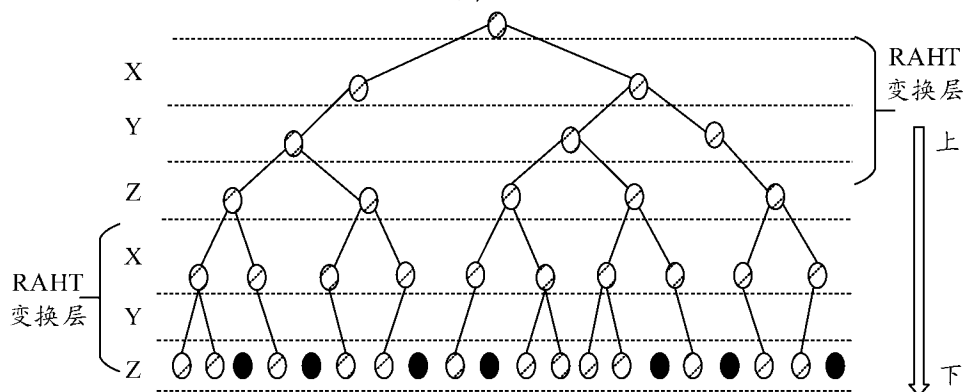
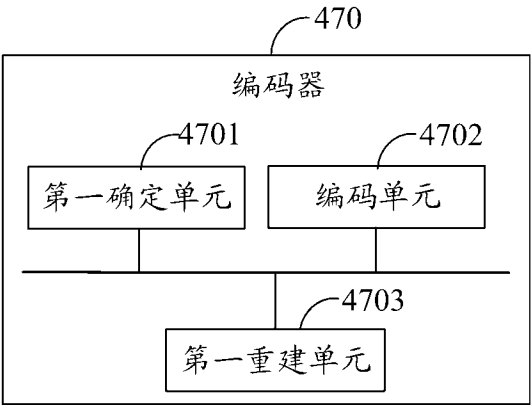
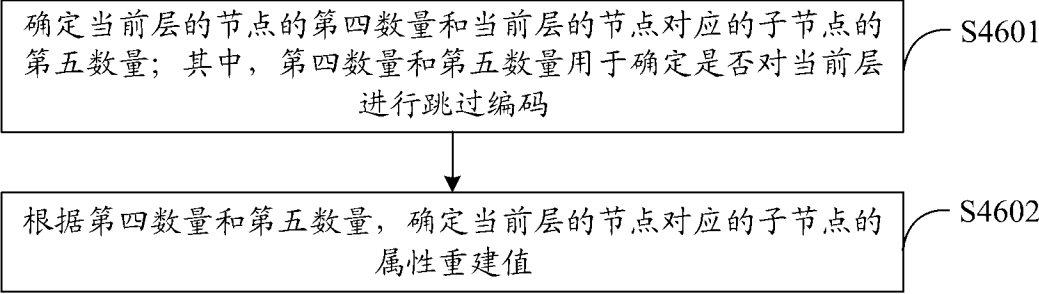
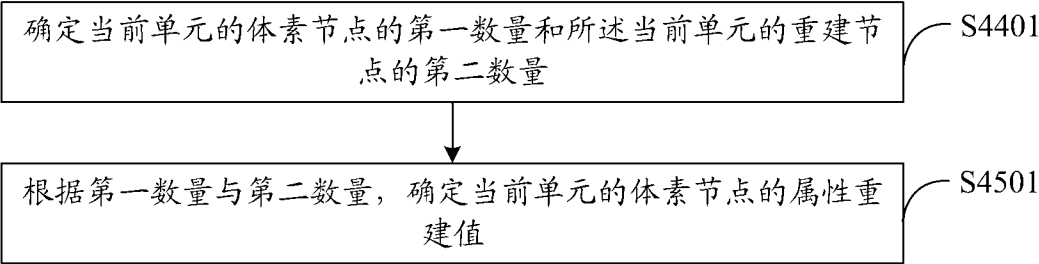
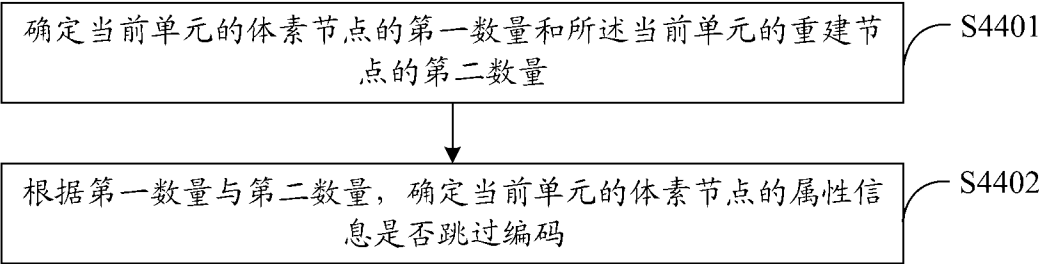
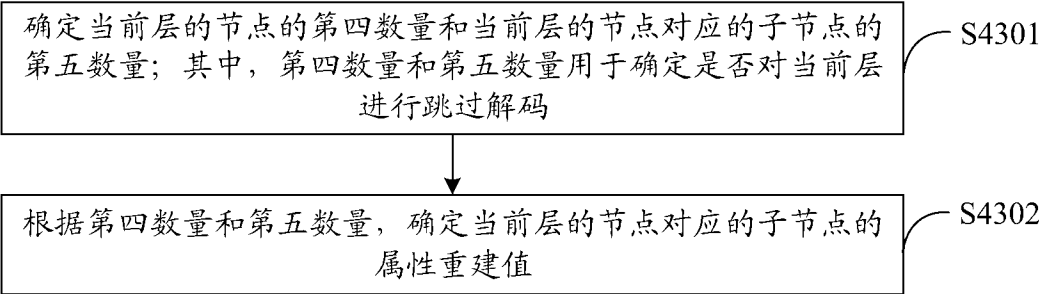


图 42



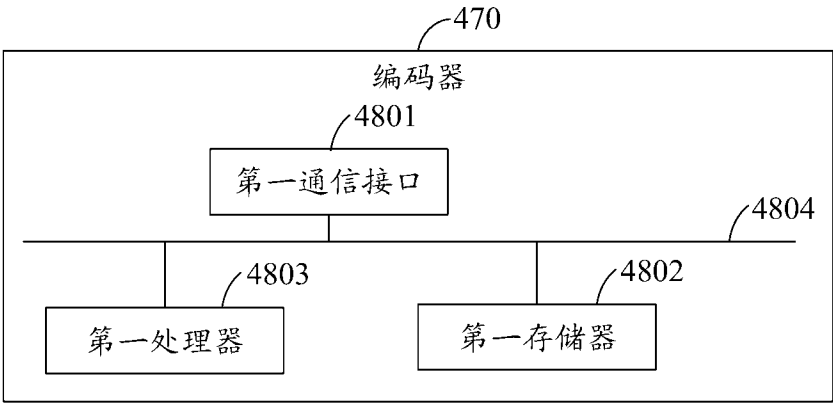


图 48

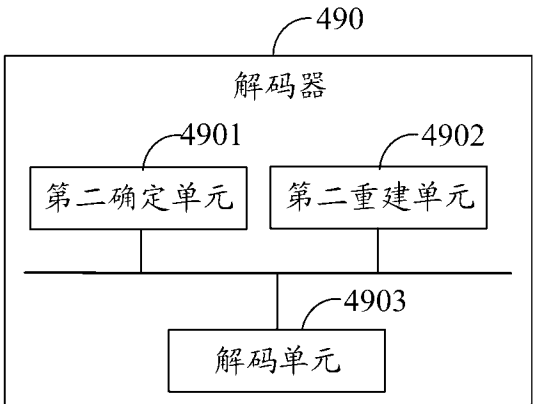


图 49

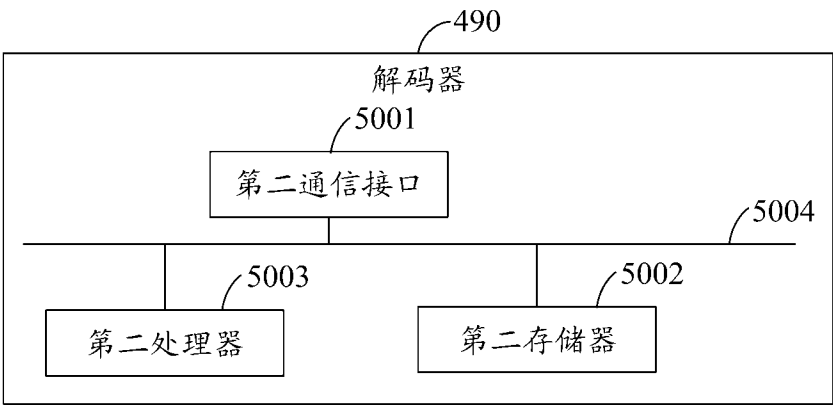


图 50

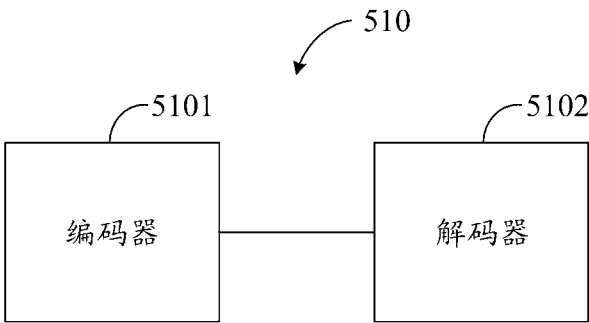


图 51

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2023/106178

A. CLASSIFICATION OF SUBJECT MATTER

H04N19/597(2014.01)i

According to International Patent Classification (IPC) or to both national classification and IPC

B. FIELDS SEARCHED

Minimum documentation searched (classification system followed by classification symbols)

IPC: H04N

Documentation searched other than minimum documentation to the extent that such documents are included in the fields searched

Electronic data base consulted during the international search (name of data base and, where practicable, search terms used)

CNTXT, CNABS, WPABS, WPABSC, ENTXT, ENTXTC, DWPI, CNKI, IEEE: 点云, 编码, 解码, 译码, 体素节点, 叶结点, 跳过, 绕过, 旁路, 重建, 数目, 数量, 重复, 属性, 层, 高频, 低频, 码流, 预测模式, point, cloud, encod+, coding, decod+, leaf, node, bypass, reconstruct+, count, attribute, layer, high, low, frequency, predict+, mode

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 115023741 A (PANASONIC INTELLECTUAL PROPERTY CORP. OF AMERICA) 06 September 2022 (2022-09-06) description, paragraphs [0755]-[0770]	44
A	CN 115380537 A (LG ELECTRONICS INC.) 22 November 2022 (2022-11-22) entire document	1-49
A	CN 116033186 A (TENCENT TECHNOLOGY (SHENZHEN) CO., LTD.) 28 April 2023 (2023-04-28) entire document	1-49
A	WO 202217215 A1 (QUALCOMM INC.) 13 October 2022 (2022-10-13) entire document	1-49
A	WO 2022126326 A1 (GUANGDONG OPPO MOBILE TELECOMMUNICATIONS CORP., LTD.) 23 June 2022 (2022-06-23) entire document	1-49



Further documents are listed in the continuation of Box C.



See patent family annex.

* Special categories of cited documents:

“A” document defining the general state of the art which is not considered to be of particular relevance

“D” document cited by the applicant in the international application

“E” earlier application or patent but published on or after the international filing date

“L” document which may throw doubts on priority claim(s) or which is cited to establish the publication date of another citation or other special reason (as specified)

“O” document referring to an oral disclosure, use, exhibition or other means

“P” document published prior to the international filing date but later than the priority date claimed

“T” later document published after the international filing date or priority date and not in conflict with the application but cited to understand the principle or theory underlying the invention

“X” document of particular relevance; the claimed invention cannot be considered novel or cannot be considered to involve an inventive step when the document is taken alone

“Y” document of particular relevance; the claimed invention cannot be considered to involve an inventive step when the document is combined with one or more other such documents, such combination being obvious to a person skilled in the art

“&” document member of the same patent family

Date of the actual completion of the international search

11 October 2023

Date of mailing of the international search report

19 October 2023

Name and mailing address of the ISA/CN

China National Intellectual Property Administration (ISA/
CN)
China No. 6, Xitucheng Road, Jimenqiao, Haidian District,
Beijing 100088

Authorized officer

Telephone No.

INTERNATIONAL SEARCH REPORT
Information on patent family members

International application No.

PCT/CN2023/106178

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	115023741	A	06 September 2022	WO	2021141094	A1	15 July 2021
				US	2022343551	A1	27 October 2022
CN	115380537	A	22 November 2022	US	2021329052	A1	21 October 2021
				EP	4131961	A1	08 February 2023
				EP	4131961	A4	13 September 2023
				WO	2021210837	A1	21 October 2021
				IN	202217062646	A	03 February 2023
CN	116033186	A	28 April 2023	None			
WO	2022217215	A1	13 October 2022	TW	202245476	A	16 November 2022
				US	2022321913	A1	06 October 2022
WO	2022126326	A1	23 June 2022	None			

A. 主题的分类

H04N19/597(2014.01)i

按照国际专利分类(IPC)或者同时按照国家分类和IPC两种分类

B. 检索领域

检索的最低限度文献(标明分类系统和分类号)

IPC: H04N

包含在检索领域中的除最低限度文献以外的检索文献

在国际检索时查阅的电子数据库(数据库的名称, 和使用的检索词(如使用))

CNTEXT,CNABS,WPABS,WPABSC,ENTXT,ENTXTC,DWPI,CNKI,IEEE: 点云, 编码, 解码, 译码, 体素节点, 叶节点, 跳过, 绕过, 旁路, 重建, 数目, 数量, 重复, 属性, 层, 高频, 低频, 码流, 预测模式, point, cloud, encode+, coding, decode+, leaf, node, bypass, reconstruct+, count, attribute, layer, high, low, frequency, predict+, mode

C. 相关文件

类型*	引用文件, 必要时, 指明相关段落	相关的权利要求
X	CN 115023741 A (松下电器(美国)知识产权公司) 2022年9月6日 (2022 - 09 - 06) 说明书第[0755]-[0770]段	44
A	CN 115380537 A (LG电子株式会社) 2022年11月22日 (2022 - 11 - 22) 全文	1-49
A	CN 116033186 A (腾讯科技(深圳)有限公司) 2023年4月28日 (2023 - 04 - 28) 全文	1-49
A	WO 2022217215 A1 (QUALCOMM INC.) 2022年10月13日 (2022 - 10 - 13) 全文	1-49
A	WO 2022126326 A1 (GUANGDONG OPPO MOBILE TELECOMMUNICATIONS CO- RP., LTD.) 2022年6月23日 (2022 - 06 - 23) 全文	1-49

☐ 其余文件在C栏的续页中列出。

☒ 见同族专利附件。

* 引用文件的具体类型:

“A” 认为不特别相关的表示了现有技术一般状态的文件

“D” 申请人在国际申请中引证的文件

“E” 在国际申请日的当天或之后公布的在先申请或专利

“L” 可能对优先权要求构成怀疑的文件, 或为确定另一篇引用文件的公布日而引用的或者因其他特殊理由而引用的文件(如具体说明的)

“O” 涉及口头公开、使用、展览或其他方式公开的文件

“P” 公布日先于国际申请日但迟于所要求的优先权日的文件

“T” 在申请日或优先权日之后公布, 与申请不相抵触, 但为了理解发明之理论或原理的在后文件

“X” 特别相关的文件, 单独考虑该文件, 认定要求保护的发明不是新颖的或不具有创造性

“Y” 特别相关的文件, 当该文件与另一篇或者多篇该类文件结合并且这种结合对于本领域技术人员为显而易见时, 要求保护的发明不具有创造性

“&” 同族专利的文件

国际检索实际完成的日期

2023年10月11日

国际检索报告邮寄日期

2023年10月19日

ISA/CN的名称和邮寄地址

中国国家知识产权局
中国北京市海淀区蓟门桥西土城路6号 100088

授权官员

王从雷

电话号码 (+86) 010-53961717

PCT/ISA/210 表(第2页) (2022年7月)

国际检索报告
关于同族专利的信息

国际申请号

PCT/CN2023/106178

检索报告引用的专利文件			公布日 (年/月/日)	同族专利			公布日 (年/月/日)
CN	115023741	A	2022年9月6日	WO	2021141094	A1	2021年7月15日
				US	2022343551	A1	2022年10月27日
CN	115380537	A	2022年11月22日	US	2021329052	A1	2021年10月21日
				EP	4131961	A1	2023年2月8日
				EP	4131961	A4	2023年9月13日
				WO	2021210837	A1	2021年10月21日
				IN	202217062646	A	2023年2月3日
CN	116033186	A	2023年4月28日	无			
WO	2022217215	A1	2022年10月13日	TW	202245476	A	2022年11月16日
				US	2022321913	A1	2022年10月6日
WO	2022126326	A1	2022年6月23日	无			