

(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 16 日 (16.01.2025)



(10) 国际公布号
WO 2025/010604 A1

(51) 国际专利分类号:
H04N 19/103 (2014.01)

(21) 国际申请号: PCT/CN2023/106659

(22) 国际申请日: 2023 年 7 月 10 日 (10.07.2023)

(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)。

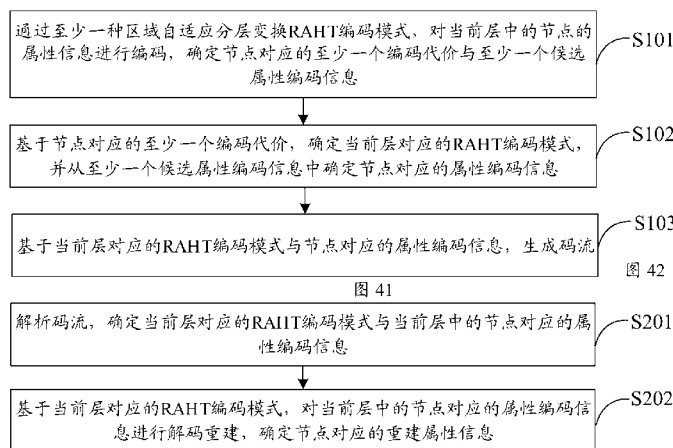
(72) 发明人: 孙泽星(SUN, Zexing); 中国广东省东莞市长安镇乌沙海滨路 18 号, Guangdong 523860 (CN)。

(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。

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

(54) 发明名称: 点云编解码方法、编码器、解码器、码流以及存储介质



S201 Parse a code stream and determine an RAHT encoding mode corresponding to a current layer and attribute encoding information corresponding to a node in the current layer

S202 On the basis of the RAHT encoding mode corresponding to the current layer, decode and reconstruct the attribute encoding information corresponding to the node in the current layer, and determine reconstruction attribute information corresponding to the node

(57) **Abstract:** Embodiments of the present application provide a point cloud encoding method and a point cloud decoding method, which can improve the attribute encoding and decoding efficiency of a point cloud, thereby improving the encoding and decoding performance of the point cloud. The point cloud decoding method comprises: parsing a code stream and determining a region-adaptive hierarchical transform (RAHT) encoding mode corresponding to a current layer and attribute encoding information corresponding to a node in the current layer; an encoder encoding attribute information of the node in the current layer by means of at least one RAHT encoding mode, determining at least one encoding cost corresponding to the node, and on the basis of the at least one encoding cost corresponding to the node, determining an RAHT encoding mode corresponding to the current layer; and on the basis of the RAHT



(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))。

encoding mode corresponding to the current layer, decoding and reconstructing the attribute encoding information corresponding to the node in the current layer, and determining reconstruction attribute information corresponding to the node.

(57) 摘要: 本申请实施例提供了一种点云编解码方法, 能够提高点云的属性编解码效率, 进而提升点云的编解码性能。其中, 点云解码方法包括: 解析码流, 确定当前层对应的RAHT编码模式与当前层中的节点对应的属性编码信息; 当前层对应的RAHT编码模式是编码器通过至少一种区域自适应分层变换RAHT编码模式对当前层的节点属性信息进行编码, 确定节点对应的至少一个编码代价, 并基于节点对应的至少一个编码代价确定的; 基于当前层对应的RAHT编码模式, 对当前层中的节点对应的属性编码信息进行解码重建, 确定节点对应的重建属性信息。

点云编解码方法、编码器、解码器、码流以及存储介质

技术领域

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

5 背景技术

在运动图像专家组（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）编解码框架中，点云的几何信息和属性信息是分开进行编码的。

10 目前，属性信息编码主要针对颜色信息的编码，在颜色信息编码中，主要有三种变换方法，包括：依赖于细节层次（Level of Detail, LOD）划分的预测变换(Predicting Transform, PT)与提升变换(Lifting Transform, LT),以及依据八叉树的区域自适应分层变换(Region Adaptive Hierarchical Transform, RAHT)。

15 其中，对于 RAHT 的编解码，在满足 RAHT 预测变换编解码条件时，在待编解码的点云中指定采用 RAHT 帧间预测变换模式进行编解码的 RAHT 编解码层，对于之下的 RAHT 编解码层，仅仅采用 RAHT 帧内预测编解码方式。在当前的 RAHT 编码方案中，根据当前节点的邻域节点数量来确定是否满足 RAHT 预测变换编解码条件，如果满足 RAHT 预测变换编解码条件，对当前节点的属性信息进行 RAHT 预测变换编码，否则，仅仅会对当前节点的属性信息进行 RAHT 变换。这种编解码方法仅仅利用了每个节点的空域相关性，尤其是邻域几何空间相关性，来确定对每个节点的属性信息的 RAHT 编解码方式，从而导致对属性信息的编解码效率较低。

发明内容

20 本申请实施例提供一种点云编解码方法、编码器、解码器、码流以及存储介质，能够提高点云的属性编解码效率，进而提升点云的编解码性能。

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，所述方法包括：

25 解析码流，确定当前层对应的 RAHT 编码模式与所述当前层中的节点对应的属性编码信息；所述当前层对应的 RAHT 编码模式是编码器通过至少一种区域自适应分层变换 RAHT 编码模式对当前层的节点的属性信息进行编码，确定所述节点对应的至少一个编码代价，并基于节点对应的至少一个编码代价确定的；

基于所述当前层对应的 RAHT 编码模式，对所述当前层中的节点对应的属性编码信息进行解码重建，确定所述节点对应的重建属性信息。

30 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，所述方法包括：

通过至少一种区域自适应分层变换 RAHT 编码模式，对当前层中的节点的属性信息进行编码，确定所述节点对应的至少一个编码代价与至少一个候选属性编码信息；

基于所述节点对应的至少一个编码代价，确定所述当前层对应的 RAHT 编码模式，并从所述至少一个候选属性编码信息中确定所述节点对应的属性编码信息；

35 基于所述当前层对应的 RAHT 编码模式与所述节点对应的属性编码信息，生成码流。

第三方面，本申请实施例提供了一种解码器，包括：

40 解析部分，被配置为解析码流，确定当前层对应的 RAHT 编码模式与所述当前层中的节点对应的属性编码信息；所述当前层对应的 RAHT 编码模式是编码器通过至少一种区域自适应分层变换 RAHT 编码模式对当前层的节点的属性信息进行编码，确定所述节点对应的至少一个编码代价，并基于节点对应的至少一个编码代价确定的；

解码重建部分，被配置为基于所述当前层对应的 RAHT 编码模式，对所述当前层中的节点对应的属性编码信息进行解码重建，确定所述节点对应的重建属性信息。

第四方面，本申请实施例提供了一种编码器，包括：

编码部分，被配置为通过至少一种区域自适应分层变换 RAHT 编码模式，对当前层中的节点的属

性信息进行编码, 确定所述节点对应的至少一个编码代价与至少一个候选属性编码信息;

确定部分, 被配置为基于所述节点对应的至少一个编码代价, 确定所述当前层对应的 RAHT 编码模式, 并从至少一个候选属性编码信息中确定所述节点对应的属性编码信息;

生成部分, 被配置为基于所述当前层对应的 RAHT 编码模式与所述节点对应的属性编码信息, 生成码流。

第四方面, 本申请实施例提供了一种编码器, 所述编码器包括第一存储器和第一处理器;

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, 执行如第二方面所述的方法。

第五方面, 本申请实施例提供了一种解码器, 所述解码器包括

第六方面, 本申请实施例提供了一种解码器, 所述解码器包括第二存储器和第二处理器;

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, 执行如第一方面所述的方法。

第七方面, 本申请实施例提供了一种码流, 所述码流是根据待编码信息进行比特编码生成的; 其中, 所述待编码信息至少包括: 当前层对应的 RAHT 编码模式与所述当前层中的节点对应的属性编码信息;

所述当前层对应的 RAHT 编码模式基于所述当前层中的节点对应的至少一个编码代价确定; 所述当前层中的节点对应的至少一个编码代价是通过至少一种 RAHT 编码模式, 对所述当前层中的节点的属性信息进行编码确定的。

第八方面, 本申请实施例提供了一种存储介质, 该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点云编解码方法、编码器、解码器、码流以及存储介质, 本申请实施例中, 解码器根据编码器传递的 RAHT 编码模式对当前层中的节点对应的属性编码信息进行解码重建, 确定节点对应的重建属性信息。由于当前层对应的 RAHT 编码模式是编码器综合考虑了每个节点的属性分布特性以及空间分布特性, 通过至少一种 RAHT 编码方式对当前层中的每个节点进行编码, 确定每个节点对应的至少一个编码代价, 根据每个节点对应的至少一个编码代价确定出的对当前层最佳的编码模式, 解码器根据当前层对应的 RAHT 编码模式对应选择解码模式进行解码, 实现了根据编码器传递的当前层对应的最佳 RAHT 编码模式来解码当前层的属性信息。这样, 通过在多种编解码模式中自适应选择最佳编解码模式, 提升了 RAHT 属性编码效率与点云属性信息的解码效率, 进而提升了点云解码性能。

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;

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;

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;

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;

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；
图 19A 为一种子块包括的三个交点示意图；
5 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；
图 22 为一种属性预测的编码流程示意图；
10 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；
图 27A 为一种共面的空间关系示意图；
15 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；
图 30 为一种基于快速查找的层内预测示意图；
20 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；
图 35A 为一种 RAHT 正变换的过程示意图；
25 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；
图 39 为一种属性变换系数的计算过程示意图；
30 图 40 为一种 RAHT 属性帧间预测编码结构示意图；
图 41 为本申请实施例提供的一种点云编码方法的流程示意图；
图 42 为本申请实施例提供的一种点云解码方法的流程示意图；
图 43 为本申请实施例提供的一种编码器的组成结构示意图；
图 44 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；
35 图 45 为本申请实施例提供的一种解码器的组成结构示意图；
图 46 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 47 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

40 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。

45 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)。

50 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。

二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，点云数据通常包括三维位置信息所组成的几何信息，三维颜色信息，以及一维反射率信息所组成的属性信息；点云中的点可以包括点的位置信息和点的属性信息。例如，点的位置信息可以是点的三维坐标信息（x， y， z）。点的位置信息也可称为点的几何信息。例如，点的属性信息可以包括颜色信息（三维颜色信息）和/或反射率（一维反射率信息 r）等等。例如，颜色信息可以是任意一种色彩空间上的信息。例如，颜色信息可以是 RGB 信息。其中，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。

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，点云中的点可以可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，点云中的点可以可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；

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；

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。

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，每秒可以获取百万级点云；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，每秒可以获取千万级点云。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。

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。

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)。

5 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。

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

10 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)。

下面以 G-PCC 编解码框架为例进行相关技术的说明。

15 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。

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 Bounding Box 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 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 Bounding Box 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，首先需要对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（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：

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

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，local_node_density 的更新具体如下：

$$\text{local_node_density}_{\text{new}} = \text{local_node_density} + 4 * \text{numSiblings}$$

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)。

5 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。

15 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。

20 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;
- (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 ($i=0, 1, 2$)。

30 需要说明的是, 在本申请实施例中, 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)。

40 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。

进一步地, 在本申请实施例中, 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。
- 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。

50 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。

在本申请实施例中, 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。

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。

5 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。

10 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:

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

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:

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

20 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。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:

25 (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。

30 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。

40 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:

● 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)。

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

45 1) 首先编码当前节点的 numPoints 是否大于 1;

2) 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。

● 当前节点含有的点数 numPoints 大于 2,则编码过程如下:

- 1) 首先编码当前节点的 numPoints 小于或等于 1;
- 2) 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。

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);

(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])
```

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:

```

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

```

在编码完成优先编码的坐标轴 directAxis 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)
30      encodePosBit(!!(pointPos[axisIdx]&mask)).

```

(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 directAxis,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])
```

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,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 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:

```

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

```

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

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。

5 示例性地, 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) 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 $\theta(i)$ 。例如, 在 i 等于 1 时, 这时候可以得到 $\theta(1)$, 对应的扫描角度为 -15° ; 在 i 等于 2 时, 这时候可以得到 $\theta(2)$, 对应的扫描角度为 -13° ; 在 i 等于 10 时, 这时候可以得到 $\theta(10)$, 对应的扫描角度为 $+13^\circ$; 在 i 等于 9 时, 这时候可以得到 $\theta(19)$, 对应的扫描角度为 $+15^\circ$ 。

10 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 Distoration=INT_MAX;
For(int LaserIdx=0; LaserIdx<numLaser;++ LaserIdx){
    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 $\times$ invRadius
    if(std::abs(tanTheta-tan $\theta_i$ )< Distoration){
        Distoration= std::abs(tanTheta-tan $\theta_i$ );
        bestLaserIdx= LaserIdx;
    }
}
```

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。

30 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:

$$\phi = \arctan(\text{nodePos}[1]/\text{nodePos}[0])$$

35 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:

$$\text{deltaPhi} = \frac{2\pi}{\text{numPoints}}$$

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 $\phi_{\text{predPoint}}$, 即如图 17A 和图 17B 所示的水平方位角的预测值。40 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 $\phi_{\text{predPoint}}$, 计算方式如下:

$$\phi_{\text{predPoint}} = \frac{\phi_{\text{pred}} - \phi_{\text{node}}}{\text{deltaPhi}} \times \text{deltaPhi} + \phi_{\text{pred}}$$

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 $\phi_{\text{predPoint}}$ 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:

```
int angLel =  $\phi_{\text{left}} - \phi_{\text{pred}}$ ;
```

```

int angLeR =  $\phi_{\text{right}} - \phi_{\text{pred}}$ ;
int context = (angLeL  $\geq 0$  & angLeR  $\geq 0$ ) || (angLeL  $< 0$  & angLeR  $< 0$ ) ? 0 : 2;
int minAngle = std :: min(abs(angLeL), abs(angLeR));
int maxAngle = std :: max(abs(angLeL), abs(angLeR));
5 context += maxAngle > minAngle ? 0 : 1;
context += maxAngle > minAngle ? 0 : 4.

```

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：

```

10 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_pred = radius  $\times$  tanTheta - zOffset.

```

15 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。

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。

20 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。

30 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。

35 (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。

40 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；
- 45 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）。

50 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 directAxis，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：

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

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 directAxis，其次按照如下方式首先对优先解码的坐标轴 directAxis 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：

```

10      Bool sameBit=true;
      while(nodeSizeLog2&& sameBit){
          pointPos[0][ directAxis]<<1;
          pointPos[1][ directAxis]<<1;
          --nodeSizeLog2;
15      int bit=0;
          deEntropyCodeSameBit(sameBits); ///

```

在解码完成优先解码的坐标轴 directAxis 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：

```

30      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 directAxis，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]) (11)

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 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。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 pointPos[0]和 pointPos[1]。具体编码过程如下：

```

45      Bool sameBit=true;
      while(nodeSizeLog2&& sameBit){
          pointPos[0][ directAxis]<<1;
          pointPos[1][ directAxis]<<1;
          --nodeSizeLog2;
          int bit=0;
          deEntropyCodeSameBit(sameBits); ///

```

在优先编码轴的方向上进行排序，所以可以保证 $\text{pointPos}[0][\text{directAxis}] < \text{pointPos}[1][\text{directAxis}]$ ，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 为 1。

}

5 在解码完优先解码的坐标轴 directAxis 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：

```

10 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。

15 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：

$$\text{PointLaserIdx} = \text{nodeLaserIdx} + \text{ResLaserIdx}$$

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：

20 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：

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

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：

$$\text{deltaPhi} = \frac{2\pi}{\text{numPoints}}$$

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：

$$\varphi_{\text{predPoint}} = \frac{\varphi_{\text{pred}} - \varphi_{\text{node}}}{\text{deltaPhi}} \times \text{deltaPhi} + \varphi_{\text{pred}}$$

30 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);
35 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：

```

40 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 =  $Z_{\text{laserIdx}}$ ;
45  $Z_{\text{pred}} = \text{radius} \times \tan\Theta - z\text{Offset}$ ;

```

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。

5 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。

10 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。

20 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:

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

30 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:

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

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:

35 (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。

40 (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。

45 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 $D < d_l$,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。

$$\text{Attr}_i' = \text{Round}\left(\frac{1}{N} \sum_{m \in p_i} \frac{\frac{1}{D_m^2}}{\sum_{m \in p_i} \frac{1}{D_m^2}} \text{Attr}_m\right)$$

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 P2 的属性值时, 将最近邻居点 P4 属性值的预测变量索引设为 1; 将次近邻点 P5 和三近邻点 P0 的属性预测变量索引分别设为 2 和 3; 将点 P0、P5 和 P4 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:

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

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

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

$\hat{a}_i$ 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	P4 (一近邻的属性值)
2	P5 (二近邻的属性值)
3	P0 (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:

$$r_i = a_i - \hat{a}_i$$

进一步对预测残差进行量化:

$$Q_i = \frac{r_i}{Q_s}$$

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:

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:

$$\hat{r}_i = Q_i \times Q_s$$

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

$$\tilde{a}_i = \hat{r}_i + \hat{a}_i$$

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:

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。

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。

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 $L(k)$;

5 (3) 进行下一次迭代时 $I \leftarrow O(k)$ 。

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 $L(k)$ 集中的点在 $O(k)$ 集中进行最近邻查找, 具体的查找算法如下:

10 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。

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。

15 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。

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:

20 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。

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。

25 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 $\text{EnableRefferingSameLoD}=1$, 那么可以允许在层内最近邻查找, 如对于 LOD1 层, 当前点 $P6$ 的最近邻点可以为 $P1$, 其他层不允许; 如果语法元素 $\text{EnableRefferingSameLoD}=0$, 那么允许在其他层进行层间查找, 如对于 LOD1 层, 当前点 $P6$ 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)。

30 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 $[i+1, i+\text{searchRange}]$ 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:

35 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 $[j-\text{searchRange}, j+\text{searchRange}]$ 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:

40 • 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;

45 • 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:

50 • 第一层: $\text{BucketSize}_0=2^5=32$;

• 第二层: $\text{BucketSize}_1=2^5=32 \times \text{BucketSize}_0=1024$;

• 第三层: $\text{BucketSize}_2=2^5=32 \times \text{BucketSize}_1=32768$ 。

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:

$$\text{idx_2} = \text{index} / \text{BucketSize_2}$$

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:

$$\begin{aligned} \text{startIdx1} &= \text{idx_2} \times \text{BucketSize_1} \\ \text{endIdx} &= \text{idx_2} \times \text{BucketSize_1} + \text{BucketSize_1} - 1 \end{aligned}$$

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:

$$\begin{aligned} \text{int dx} &= \text{int}(\text{std::max}(\text{std::max}(\text{minPos}[0] - \text{point}[0], 0), \text{point}[0] - \text{maxPos}[0])); \\ \text{int dy} &= \text{int}(\text{std::max}(\text{std::max}(\text{minPos}[1] - \text{point}[1], 0), \text{point}[1] - \text{maxPos}[1])); \\ \text{int dz} &= \text{int}(\text{std::max}(\text{std::max}(\text{minPos}[2] - \text{point}[2], 0), \text{point}[2] - \text{maxPos}[2])); \\ D &= \text{dx} + \text{dy} + \text{dz}; \end{aligned}$$

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 $(\text{LOD}_l)_{l=0,1,2}$, 经过分割后, LOD_2 为高 LOD 层, 记为 H(N), $(\text{LOD}_l)_{l=0,1}$ 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:

$$D(N) = H(N) - P(N)$$

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:

$$L'(N) = L(N) + U(N)$$

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:

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

其中, T_{w_0,w_1} 为变换矩阵:

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

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的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。

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。

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:

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

(e) 表示属性预测值, 对应的属性变换系数如下:

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

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:

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

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, 可以包括两种区域自适应分

层帧间预测变换编码方案，如下：

一、区域自适应分层帧间预测变换编码方案一

在区域自适应分层帧间预测变换编码方案一中，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属性编码结构具体参见图 40。

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

其中，根据如下两种不同的方式得到当前节点的属性预测值：

- 1、当前节点的帧间预测节点有效：即同位节点存在，则将同位节点的属性直接作为当前节点的属性预测值；
- 2、当前节点的帧间预测节点无效：即同位节点不存在，则利用当前节点的帧内相邻节点的属性预测值作为当前节点的属性预测值。

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

二、区域自适应分层帧间预测变换编码方案二

在区域自适应分层帧间预测变换编码方案一中，与帧内预测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，则首先会基于当前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，再根据 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，得到每个子节点 DC 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，则将同位节点的属性直接作为当前节点的属性预测值。

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，则会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。

如果子节点对应的同位节点位置上的 AC 系数不存在，就不能启动预测。

a. 如果同位节点包含的子节点的 AC 系数不为零：则将该子节点的 AC 系数直接作为当前节点对应位置上的子节点的预测值。

b. 如果同位节点包含的子节点的 AC 系数为零，则会将帧内预测对应子节点的 AC 系数作为当前节点对应位置上的子节点的预测值。

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

目前，在常见的 G-PCC 的 RAHT 属性编解码时，通过在高层语法元素，如点云序列级的语法元素中决定是否采用 RAHT 预测变换编码来对当前点云序列中的每个点云中的节点进行帧间/帧内预测的编解码。并且，通过一个语法元素，如 treeDepth 来决定帧间预测编解码的启动层数，在之下的 RAHT 编解码层，仅仅采用 RAHT 帧内预测编解码。在当前的 RAHT 编码方案中，首先会利用当前节点的邻域节点数目 N 是否大于一定阈值，当邻域节点数目大于一定阈值时，才会对当前节点的 AC 系数进行预测（帧内或者帧间预测）。当当前节点的邻域节点数目 N 小于一定阈值，则认为当前节点不满足预测编码的条件，仅仅会对当前节点进行属性变换。目前的编解码方案主要是为了利用当前节点的空域相关性，特别是当前节点的邻域几何空间相关性来提升编码效率。然而，这种编码方案并没有考虑到每个节点的属性信息的本身分布特性，而是直接在序列级中决定对整个点云序列的 RAHT 编码方案，其次在启动预测变换编码方案的同时，仅仅根据当前节点的邻域几何空间相关性决定当前节点的编码方式，并没有有效结合当前节点的属性分布特性，从而导致属性信息的编码效率较低。

为了解决上述问题，本申请的实施例综合考虑了每个节点的属性分布特性以及空间分布特性，由编码器通过至少一种 RAHT 编码方式对当前层中的每个节点进行编码，确定每个节点对应的至少一个编码代价，根据每个节点对应的至少一个编码代价确定出对当前层最佳的编码模式传递给解码器，解码器根据当前层对应的 RAHT 编码模式对应选择解码模式进行解码。这样，通过在多种编解码模式中自适应选择最佳编解码模式，提升了 RAHT 属性编码效率与点云属性信息的解码效率，进而提升了点云解码性能。

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点云编码方法的流程示意图。如图 41 所示，该方法可以包括：

S101、通过至少一种区域自适应分层变换 RAHT 编码模式，对当前层中的节点的属性信息进行编

码, 确定节点对应的至少一个编码代价与至少一个候选属性编码信息。

需要说明的是, 本申请实施例的编码方法具体是指点云编码方法, 该方法可以应用于点云编码器(也可简称为“编码器”)。

5 相应的, 在本申请的实施例中, 当前层可以为待编码的 RAHT 变换层。编码器可以基于点云中节点的几何信息, 构建点云对应的 RAHT 属性变换编码结构。

在本申请的实施例中, 需要先基于点云中的点的几何信息来构建 RAHT 属性变换编码结构, 具体地, 可以基于点云对应的八叉树结构, 由体素级别不断进行合并直至得到根节点, 从而完成整个属性的变换编码分层结构, 得到 RAHT 属性变换编码结构。

10 在一些实施例中, 编码器可以基于 RAHT 属性变换编码结构, 从根节点开始沿每个空间坐标轴方向进行下采样, 确定点云对应的至少一个 RAHT 变换层。这里, 至少一个 RAHT 变换层包含当前层。

示例性地, 可以定义对 RAHT 属性变换编码结构, 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)。

需要说明的是, 在本申请的实施例中, 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。

15 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。

本申请实施例中, 通过至少一种区域自适应分层变换 RAHT 编码模式, 对当前层中的节点的属性信息进行编码, 确定节点对应的至少一个编码代价与至少一个候选属性编码信息。

20 本申请实施例中, 至少一种区域自适应分层变换 RAHT 编码模式可以包括上述的 RAHT 变换编码模式、RAHT 帧内预测变换编码模式以及 RAHT 帧间预测变换编码模式。编码器通过至少一种区域自适应分层变换 RAHT 编码模式, 对当前层中每个节点的属性信息进行编码, 确定每种 RAHT 编码模式对应的编码代价以及每种 RAHT 编码模式对应的编码信息。编码器将每种 RAHT 编码模式对应的编码信息作为一个候选属性编码信息, 从而得到每个节点对应的至少一个编码代价与至少一个候选属性编码信息。

需要说明的是, 在本申请的实施例中, RAHT 变换编码模式可以是基于八叉树层级的顺序来执行属性变换编码的。编码器可以基于八叉树的层级顺序, 由体素级别不断进行变换编码直至得到根节点, 从而完成对点云中每个节点的属性信息的 RAHT 变换编码。而在 RAHT 预测变换编码模式, 如 RAHT 帧内预测变换编码模式与 RAHT 帧间预测变换编码模式中, 同样可以基于八叉树的层级顺序, 但是从根节点不断进行编码直至到体素级别。

30 在一些实施例中, 编码器可以预先设置至少一种 RAHT 编码模式对应的缓存空间(buffer), 将每种 RAHT 编码模式对应的候选属性编码信息存储在对应的缓存空间中。示例性地, 编码器可以设置 RAHT 变换编码模式对应的缓存 1、RAHT 帧内预测变换编码模式对应的缓存 2、以及 RAHT 帧间预测变换编码模式对应的缓存 3。编码器分别通过 RAHT 变换编码模式、RAHT 帧内预测变换编码模式与 RAHT 帧间预测变换编码模式对当前节点的属性信息进行编码, 将 RAHT 变换编码模式对应的候选属性编码信息存储在缓存 1 中, 将 RAHT 帧内预测变换编码模式对应的候选属性编码信息存储在缓存 2 中, 将 RAHT 帧间预测变换编码模式对应的候选属性编码信息存储在缓存 3 中。

35 S102、基于节点对应的至少一个编码代价, 确定当前层对应的 RAHT 编码模式, 并从至少一个候选属性编码信息中确定节点对应的属性编码信息。

本申请实施例中, 编码器可以基于节点对应的至少一个编码代价, 评估至少一种 RAHT 编码模式针对当前节点的编码性能, 从至少一个候选属性编码信息中确定节点对应的属性编码信息。

40 在一些实施例中, 编码器可以基于当前层中每个节点对应的至少一个编码代价, 确定出每种 RAHT 编码模式对于当前层中所有节点的编码代价总和。根据每种 RAHT 编码模式的编码代价总和, 确定出编码代价总和最小的 RAHT 编码模式作为当前层对应的 RAHT 编码模式。进而, 对于当前层中的每个节点, 编码器在每个节点所对应的至少一个候选属性编码信息中, 确定出与当前层对应的 RAHT 编码模式对应的候选属性编码信息, 作为每个节点对应的属性编码信息。

50 也就是说, 编码器可以根据每个节点对应的至少一个编码代价, 确定出对当前层的最佳编码模式, 作为当前层对应的 RAHT 编码模式。

在一些实施例中, 编码器可以在至少一个 RAHT 编码模式对应的至少一个缓存中, 将与当前层对

应的 RAHT 编码模式对应的缓存中存储的每个节点的候选属性编码信息，确定为每个节点对应的属性编码信息。

S103、基于当前层对应的 RAHT 编码模式与节点对应的属性编码信息，生成码流。

本申请实施例中，编码器基于当前层中的每个节点对应的属性编码信息，可以确定当前层对应的属性编码信息。编码器根据当前层对应的 RAHT 编码模式，可以确定当前层对应的 RAHT 编码模式标志位。其中，RAHT 编码模式标志位用于表征 RAHT 变换编码模式或 RAHT 预测变换编码模式；RAHT 预测变换编码模式包括：帧间预测变换编码模式或帧内预测变换编码模式。

编码器对点云中的每个 RAHT 变换层进行同样的处理，可以确定每个 RAHT 变换层对应的编码模式标志位与每个 RAHT 变换层对应的属性编码信息。编码器基于每个 RAHT 变换层对应的编码模式标志位与每个 RAHT 变换层对应的属性编码信息，生成码流。

这样，编码器将码流发送至解码器，解码器可以根据码流中点云的每个 RAHT 变换层对应的编码模式标志位，确定对每个 RAHT 变换层的解码方式；并按照对每个 RAHT 变换层的解码方式，对每个 RAHT 变换层对应的属性编码信息进行解码。

在一些实施例中，编码器可以将点云中每一个 RAHT 变换层的 RAHT 预测编码模式添加到 Attribute Brick Header (ABH) 参数集中。由解码器通过 ABH 来得到每一个 RAHT 变换层的 RAHT 预测编码模式。对于 ABH 参数集以何种形式进行编码，本申请实施例不限定。

可以理解的是，本申请实施例中，编码器通过至少一种 RAHT 编码方式对当前层中的每个节点进行编码，确定每个节点对应的至少一个编码代价与至少一个候选属性编码信息，从而可以根据每个节点对应的至少一个编码代价确定出对当前层最佳的编码模式，利用该编码模式确定当前层中每个节点对应的属性编码信息，并将当前层对应的 RAHT 编码模式与节点对应的属性编码信息传输至解码端，使得解码端可以根据当前层对应的 RAHT 编码模式来自适应地解码当前层的属性信息。这样，通过引入多种编码模式并结合编码代价进行最佳模式的自适应选择，综合考虑了每个节点的属性分布特性以及空间分布特性，提升了 RAHT 属性编码效率与点云属性信息的编码效率，进而提升了点云编码性能。

在一些实施例中，至少一个候选属性编码信息包括：变换编码信息与预测变换编码信息；上述 S101 中的过程可以通过 S1011-S1013 来实现，包括：

S1011、通过对点云中每个节点对应的属性信息进行 RAHT 变换编码，确定当前层中的每个节点对应的属性变换信息与变换编码信息。

在一些实施例中，编码器可以基于点云对应的 RAHT 属性变换编码结构，从体素级别的节点开始，递归地对点云中每个节点对应的属性信息进行 RAHT 变换与编码直至根节点，确定点云中每个节点对应的属性变换信息与变换编码信息，从而确定当前层中的每个节点对应的变换编码信息；属性变换信息通过对属性信息进行 RAHT 变换确定；变换编码信息通过对属性变换信息进行编码确定。

示例性地，编码器基于八叉树结构，从体素级别的节点开始，递归地对点云中每个节点对应的属性信息进行 RAHT 变换，可以得到每个节点的 DC 系数与 AC 系数，其中，DC 系数被传递到下一层继续进行变换，AC 系数作为每个节点对应的属性变换信息对其进行编码，确定点云中每个节点对应的变换编码信息。这样，根据当前层中包含的节点，确定当前层中的每个节点对应的属性变换信息与变换编码信息。

在一些实施例中，编码器也可以基于点云对应的 RAHT 属性变换编码结构，由根节点开始，确定点云中每个节点对应的 N 个子节点；对于每个节点对应的 N 个子节点，对 N 个子节点对应的属性信息进行 RAHT 属性变换与编码，确定 N 个子节点对应的属性变换信息与变换编码信息，从而确定点云中每个节点对应的属性变换信息与变换编码信息；N 大于 0 且不大于预设子节点数量阈值；根据点云中每个节点对应的属性变换信息与变换编码信息，确定当前层中的每个节点对应的属性变换信息与变换编码信息。示例性地，预设子节点数量阈值可以是 8，也可以根据实际情况设置为其他数值，具体的根据实际情况进行选择，本申请实施例不限定。

示例性地，编码器可以基于八叉树结构，从根节点开始划分，可以得到每个节点对应的 N 个子节点，其中，N 大于 0，且小于或等于 8。编码器从根节点开始，针对每个节点对应的 N 个子节点进行独立正交变换，得到每个子节点的 DC 系数与 AC 系数，DC 系数同样传递到下一层继续进行变换，将每个子节点的 AC 系数作为属性变换信息对其进行编码，确定每个子节点对应的变换编码信息。这样，根据父节点和子节点的对应关系，对八叉树结构中的每个节点进行同样的处理，得到点云中每个节点对应的属性变换信息与变换编码信息。根据当前层中包含的节点，确定当前层中的每个节点对应的属性变换信息与变换编码信息。

S1012、对当前层中的每个节点对应的属性变换信息进行预测编码，确定当前层中的每个节点对应的预测变换编码信息。

S1012 中，编码器对当前层中的每个节点对应的属性变换信息进行预测编码。这里，在帧间预测未

使能的情况下,编码器可以对当前层中的每个节点进行帧内预测编码,确定每个节点对应的帧内预测编码信息。在帧间预测使能的情况下,编码器可以对当前层中的每个节点进行帧内预测编码与帧间预测编码,确定每个节点对应的帧内预测编码信息与帧间预测编码信息。

需要说明的是,编码器在进行 RAHT 帧内/帧间预测变换编码之前,会先确定当前节点是否满足 RAHT 帧内/帧间预测变换编码条件。编码器确定当前层中的每个节点的父节点的邻域节点数量是否大于预设第一数量阈值;在大于预设第一数量阈值的情况下,确定当前层中的每个节点的邻域节点数量是否大于预设第二数量阈值;在大于预设第二数量阈值的情况下,对当前层中的每个节点对应的属性变换信息进行预测编码,确定当前层中的每个节点对应的预测变换编码信息。

在前层中的每个节点的父节点的邻域节点数量不大于预设第一数量阈值,或者,当前层中的每个节点的邻域节点数量不大于预设第二数量阈值的情况下,编码器对当前节点进行 RAHT 变换编码。

在一些实施例中,对于帧内预测编码,可以根据上述的帧内预测编码方式,当前层中的每个节点的邻域节点的属性信息预测值,确定当前层中的每个节点对应的属性信息预测值。进而根据每个节点对应的属性信息预测值进行编码,确定当前层中的每个节点对应的帧内预测变换编码信息。示例性地,属性信息预测值可以是 AC 系数预测值。

在一些实施例中,对于帧间预测编码,可以通过上述的帧间预测编码方案一和帧间预测编码方案二中的任一种来实现。

示例性地,如通过帧间预测编码方案一来实现,编码器可以确定当前层中的每个节点在点云对应的参考点云中位置对应的同位节点;基于同位节点的重建属性变换信息,确定当前层中的每个节点对应的属性信息预测值;基于属性信息预测值,对当前层中的每个节点对应的属性变换信息进行编码,确定当前层中的每个节点对应的预测变换编码信息。

在一些实施例中,在参考点云中不存在节点对应的同位节点的情况下,通过帧内预测,根据节点的邻域节点的重建属性变换信息,确定节点对应的属性信息预测值。

示例性地,如通过帧间预测编码方案二来实现,编码器可以确定当前层中的每个节点在点云对应的参考点云中对应的同位节点;在当前层中的每个节点为当前层中的父节点的情况下,基于同位节点的重建属性变换信息,确定当前层中的每个节点对应的属性信息预测值;在当前层中的每个节点为当前层中的子节点的情况下,基于同位节点在参考点云中对应的 M 个子节点的重建属性变换信息,确定当前层中的每个节点对应的属性信息预测值;M 大于 0 且不大于预设子节点数量阈值;基于属性信息预测值,对当前层中的每个节点对应的属性变换信息进行编码,确定当前层中的每个节点对应的帧间预测变换编码信息。

这里,对于当前层中的父节点,编码器直接将该父节点在参考点云中的同位节点的重建属性变换信息,作为该父节点对应的属性信息预测值。对于父节点对应的 N 个子节点,编码器根据父节点在参考点云对应的同位节点,将该同位节点在参考点云中对应的 M 个子节点的重建属性变换信息,作为该父节点的 N 个子节点对应的属性信息预测值。

需要说明的是,上述参考点云为属性信息已编码重建的点云,示例性地,参考点云可以是与当前层所在点云相邻的,属性信息已编码重建的点云。在一些实施例中,重建属性变换信息可以是重建的 AC 系数。

在一些实施例中,上述基于同位节点在参考点云中对应的 M 个子节点的重建属性变换信息,确定当前层中的每个节点对应的属性信息预测值,可以通过以下过程来实现:

确定每个子节点在 M 个子节点中位置对应的同位子节点;在同位子节点的重建属性变换信息大于预设信息值阈值的情况下,根据同位子节点的重建属性变换信息确定当前层中的每个节点的属性信息预测值;在同位子节点的重建属性变换信息小于或等于预设信息值阈值的情况下,基于当前层中的每个节点的邻域节点的重建属性变换信息,确定当前层中的每个节点对应的属性信息预测值。

这里,对于父节点中的 N 个子节点,编码器确定每个子节点在同位节点的 M 个子节点中,位置相同的子节点,作为每个子节点对应的同位子节点。在同位子节点的重建属性变换信息大于预设信息值阈值的情况下,如同位子节点的重建 AC 系数大于零的情况下,将同位子节点的重建属性变换信息,如同位子节点的建 AC 系数直接作为每个子节点的属性信息预测值。在同位子节点的重建属性变换信息小于或等于预设信息值阈值的情况下,如同位子节点的重建 AC 系数为零的情况下,则会利用帧内预测,根据每个子节点的邻域节点的重建属性变换信息,确定每个子节点对应的属性信息预测值。

需要说明的是,上述过程中,如果子节点对应的同位子节点的 AC 系数不存在,则不启动帧内预测或帧间预测。

在一些实施例中,在参考点云中不存在父节点对应的同位节点的情况下,通过帧内预测,根据父节点的邻域节点的重建属性变换信息,确定父节点对应的属性信息预测值。

在一些实施例中,在参考点云中不存在子节点对应的同位子节点的情况下,通过帧内预测,根据子节点的邻域节点的重建属性变换信息,确定子节点对应的属性信息预测值。

需要说明的是,本申请实施例中,RAHT 变换编码模式包括两种实现方式,RAHT 预测变换编码模式包括 RAHT 帧内预测变换编码模式与 RAHT 帧间预测变换编码模式;其中 RAHT 帧间预测变换编码模式包含两种实现方式。故,编码器在根据至少一种 RAHT 编码模式对节点的属性信息进行编码时,可以将上述各种编码模式的各种实现方式进行组合,具体的根据实际情况进行选择,本申请实施例不作限定。

S1013、基于变换编码信息与预测变换编码信息,确定节点对应的至少一个编码代价。

S1013 中,在帧内预测未使能的情况下,编码器可以基于变换编码信息与帧内预测变换编码信息进行解码重建与编码代价计算,确定节点对应的至少一个编码代价。

或者,在帧内预测使能的情况下,编码器可以基于变换编码信息、帧内预测变换编码信息与帧间预测变换编码信息进行解码重建与编码代价计算,确定节点对应的至少一个编码代价。

在一些实施例中,编码器可以基于变换编码信息进行解码重建,确定当前层中的每个节点对应的第一率失真代价;基于帧内预测变换编码信息进行解码重建,确定当前层中的每个节点对应的第二率失真代价;基于帧间预测变换编码信息进行解码重建,确定当前层中的每个节点对应的第三率失真代价。

这样,编码器可以基于当前层中的每个节点对应的第一率失真代价,确定当前层对应的第一编码代价;第一编码代价对应 RAHT 变换编码模式;基于当前层中的每个节点对应的第二率失真代价,确定当前层对应的第二编码代价;第二编码代价对应 RAHT 帧内预测变换编码模式;基于当前层中的每个节点对应的第三率失真代价,确定当前层对应的第三编码代价;第三编码代价对应 RAHT 帧间预测变换编码模式;通过对比第一编码代价、第二编码代价、以及第三编码代价,确定当前层对应的 RAHT 编码模式。

示例性地,率失真优化算法中,首先计算每种 RAHT 编码模式对应的重建属性信息与节点的原始属性信息之间的失真 D ,其次得到每种 RAHT 编码模式所需要编码的码率 R ,通过以下公式计算每种 RAHT 编码模式对应的率失真代价 J :

$$J = D + \lambda \times R$$

其中, λ 可以通过属性量化参数进行计算得到,示例性地, $\lambda = 2^{\frac{QP-4}{6}} \times N$;其中,参数 N 可以根据反射率和颜色预设为不同的数值。

在上述方案中,是通过对当前层中的节点属性信息编码引入至少一种编码模式:RAHT 变换编码、RAHT 帧内预测变换编码以及 RAHT 帧间预测变换编码,通过至少一种编码模式的编码代价对比来对当前层所有的节点进行自适应选择 RAHT 编码模式。在一些实施例中,编码器还可以通过对当前层中节点的父节点邻域节点的重建属性分布特性进行综合分析,确定对当前层中的节点属性编码模式。

在一些实施例中,编码器可以基于当前层的节点的父节点属性信息,以及父节点的邻域节点的重建属性信息,确定当前层对应的 RAHT 编码模式;利用当前层对应的 RAHT 编码模式,对当前层中的节点属性信息进行编码,确定节点对应的属性编码信息。

示例性地,在当前层中的节点的父节点属性信息,与父节点的邻域节点的重建属性信息之间的误差大于预设误差阈值的情况下;将 RAHT 变换编码模式确定为当前层对应的 RAHT 编码模式;变换编码模式表征对属性信息进行 RAHT 变换与编码;在当前层中的节点的父节点属性信息,与父节点的邻域节点的重建属性信息之间的误差小于或等于预设误差阈值的情况下,将 RAHT 预测变换编码模式确定为当前层对应的 RAHT 编码模式;预测变换编码模式表征对属性信息进行 RAHT 变换确定的属性变换系数进行预测编码。

也就是说,如果当前节点父节点属性与当前节点父节点邻域节点的重建属性误差在一定的范围内,则认为当前节点的邻域属性分布特性较为平缓,基于这样的分布特性,则可以隐式推导当前节点采用 RAHT 预测变换编码模式;否则,认为当前节点的邻域范围的属性分布较为抖动,则会采用 RAHT 变换编码模式。编码器以上述方式得到对当前层的每个节点进行处理,从而可以确定当前层对应的 RAHT 编码模式。

可以理解的是,编码器通过父节点邻域节点的重建属性分布特性,确定出对当前层中节点的编码模式,实现了综合考虑节点属性分布特性以及空间分布特性,从而提升了 RAHT 属性编码效率与点云属性信息的编码效率,进而提升了点云编码性能。

在一些实施例中,编码器可以在点云级别,指示对该点云中的每一个 RAHT 变换层是否使能预测变换编码模式。编码器可以根据点云中每一个 RAHT 变换层对应确定的 RAHT 编码模式,确定点云对应的预测变换使能标志位。这里,预测变换使能标志位表征是否对点云中的 RAHT 变换层使能预测变换编码模式。

在一些实施例中，编码器可以根据点云中确定了 RAHT 编码模式的至少一个 RAHT 变换层，确定点云对应的编码模式标志位的数量。这里，编码模式标志位的数量用于告知解码器需要解码的编码模式标志位的数量。编码器将预测使能变换标志位与编码模式标志位的数量中的至少一个写入码流发送至解码器。

5 在一些实施例中，编码器可以将当前层中的节点划分为至少两个节点组；于至少两个节点组中的每个节点组，基于每个节点组中的每个节点对应的至少一个编码代价，确定每个节点组对应的至少一个编码代价；基于每个节点组对应的至少一个编码代价，确定每个节点组对应的 RAHT 编码模式；基于每个节点组对应的 RAHT 编码模式，确定每个节点组中的每个节点对应的属性编码信息，从而确定每个节点组对应的属性编码信息；根据每个节点组对应的 RAHT 编码模式以及每个节点组对应的属性编码信息，确定当前层对应的至少两组属性编码信息与每组属性编码信息对应的 RAHT 编码模式，并生成码流。

示例性地，编码器通过将当前层中的节点划分为至少两个节点组，可以对当前层的节点实现属性信息，如 AC 系数组的划分，得到当前层对应的至少两个 AC 系数组。编码器针对每个系数组选取最佳的编码模式，向解码器传递每个系数组对应的编码模式，以在解码端针对每个系数组的属性信息进行重建。

15 可以理解的是，编码器通过将当前层中节点划分为至少两个节点组，对每个节点组进行至少一种 RAHT 编码方式的编码与编码代价对比，针对每个节点组确定出对最佳的编码模式传递给解码器，进一步提升了 RAHT 属性编码效率与点云属性信息的编码效率，进而提升了点云编码性能。

在本申请的一实施例中，参见图 42，其示出了本申请实施例提供的一种点云解码方法的流程示意图。如图 42 所示，该方法可以包括：

20 S201、解析码流，确定当前层对应的 RAHT 编码模式与当前层中的节点对应的属性编码信息。

需要说明的是，本申请实施例的解码方法具体是指点云解码方法，该方法可以应用于点云解码器（也可简称为“解码器”）。

本申请实施例中，如上述编码器的方法描述，当前层对应的 RAHT 编码模式是编码器通过至少一种区域自适应分层变换 RAHT 编码模式对当前层的节点的属性信息进行编码，确定节点对应的至少一个编码代价，并基于节点对应的至少一个编码代价确定的。

25 在一些实施例中，解码器可以通过解析码流，得到当前层对应的 RAHT 编码模式标志位，根据 RAHT 编码模式标志位确定当前层对应的 RAHT 编码模式；其中，RAHT 编码模式标志位用于表征 RAHT 变换编码模式或 RAHT 预测变换编码模式。

30 本申请实施例中，解码器可以通过解析点云对应的码流，确定点云中每个 RAHT 变换层的 RAHT 预测编码模式。这样，在对当前层进行解码时，即可确定当前层对应的 RAHT 预测编码模式。示例性地，解码器可以通过解码码流中的 ABH 参数集，来得到点云中每个 RAHT 变换层的 RAHT 预测编码模式。

35 在一些实施例中，解码器通过解析码流，确定点云对应的预测变换使能标志位；预测变换使能标志位表征是否对点云使能预测变换编码模式；在预测变换使能标志位表征使能预测变换编码模式的情况下，确定点云对应的编码模式标志位的数量。

在一些实施例中，解码器可以通过解析码流，确定点云对应的编码模式标志位的数量；点云为当前层所在的点云；根据编码模式标志位的数量，对码流中的 RAHT 编码模式标志位进行解析，确定点云中每一层对应的 RAHT 编码模式标志位。

40 示例性地，预测变换使能标志位可以通过码流中点云级别的 disableAttrInterPred 标志位确定。编码模式标志位的数量可以通过码流中的 attr_code_mode_cnt 字段确定。如下：

attribute_data_unit_header() {	Descriptor	Semantics
adu_attr_parameter_set_id	u(4)	7.4.4.2
adu_reserved_zero_3bits	u(3)	7.4.4.2
adu_sps_attr_idx	ue(v)	7.4.4.2
adu_slice_id	ue(v)	7.4.4.2
if(lod_dist_log2_offset_present)		
lod_dist_log2_offset	se(v)	10.6.2
if(last_comp_pred_enabled && AttrDim == 3)		
for(dpth = 0; dpth ≤ lod_max_levels_minus1; dpth++)		
last_comp_pred_coeff_diff[dpth]	se(v)	10.6.10.1
if(inter_comp_pred_enabled)		
for(dpth = 0; dpth ≤ lod_max_levels_minus1; dpth++)		
for(c = 1; c < AttrDim; c++)		
inter_comp_pred_coeff_diff[dpth][c]	se(v)	10.6.10.1
if(attr_qp_offsets_present)		
for(qc = 0; qc < Min(2, AttrDim); qc++)		
attr_qp_offset[qc]	se(v)	10.7.1
attr_qp_layers_present	u(1)	10.7.1
if(attr_qp_layers_present) {		
attr_qp_layer_cnt_minus1	ue(v)	10.7.1
for(dpth = 0; dpth ≤ attr_qp_layer_cnt_minus1; dpth++)		
for(qc = 0; qc < Min(2, AttrDim); qc++)		
attr_qp_layer_offset[dpth][qc]	se(v)	10.7.1
}		
attr_qp_region_cnt	ue(v)	10.7.1
if(attr_qp_region_cnt)		
attr_qp_region_bits_minus1	ue(v)	10.7.1
for(i = 0; i < attr_qp_region_cnt; i++) {		
if(¬attr_coord_conv_enabled) {		
for(k = 0; k < 3; k++)		
attr_qp_region_origin_xyz[i][k]	u(v)	10.7.1
for(k = 0; k < 3; k++)		
attr_qp_region_size_minus1_xyz[i][k]	u(v)	10.7.1
} else {		
for(k = 0; k < 3; k++)		
attr_qp_region_origin_rpi[i][k]	u(v)	10.7.1

for($k = 0; k < 3; k++$)		
attr_qp_region_size_minus1_rpi [i][k]	$u(v)$	10.7.1
}		
for($ps = 0; ps < \text{Min}(2, \text{AttrDim}); ps++$)		
attr_qp_region_offset [i][ps]	$se(v)$	10.7.1
}		
disableAttrInterPred	$u(1)$	
if(attr_coding_type == 0 && !disableAttrInterPred)		
if(raht_prediction_enabled){		
attr_code_mode_cnt	$ue(v)$	
for($i = 0; i < \text{attr_code_mode_cnt}; i++$)		
attr_code_mode [i]	$u(1)$	
}		
byte_alignment()		
}		

S202、基于当前层对应的 RAHT 编码模式，对当前层中的节点对应的属性编码信息进行解码重建，确定节点对应的重建属性信息。

本申请实施例中，当前层可以为待解码的 RAHT 变换层。

在本申请的实施例中，需要先基于点云中的点的几何信息来构建 RAHT 属性变换编码结构，具体地，可以基于点云对应的八叉树结构，由体素级别不断进行合并直至得到根节点，从而完成整个属性的变换编码分层结构，得到 RAHT 属性变换编码结构。

在本申请的实施例中，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)。

需要说明的是，在本申请的实施例中，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。

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。

在一些实施例中，RAHT 编码模式可以包括：RAHT 变换编码模式与 RAHT 预测变换编码模式。

在 RAHT 编码模式为 RAHT 变换编码模式的情况下，解码器确定 RAHT 变换解码模式为对应的解码模式，对属性编码信息进行 RAHT 变换解码，确定节点对应的重建属性信息；其中，RAHT 变换编码模式表征编码器对节点的属性信息进行 RAHT 变换与编码。

示例性地，在 RAHT 编码模式为 RAHT 变换编码模式的情况下，解码器对当前层中每个节点对应的属性编码信息进行熵解码，得到每个节点对应的属性变换信息，如每个节点对应的 AC 系数。解码器对每个节点对应的属性变换信息，如 AC 系数进行 RAHT 逆变换，得到重建属性信息。

在 RAHT 编码模式为 RAHT 预测变换编码模式的情况下，解码器对当前层中的节点对应的属性变换信息进行预测，确定当前层中的节点对应的属性信息预测值；基于属性信息预测值，对属性编码信息进行 RAHT 变换解码，确定当前层中的节点对应的重建属性信息；其中，RAHT 预测变换编码模式表征编码器对节点的属性信息进行 RAHT 变换，并对 RAHT 变换确定的属性变换信息进行预测编码。

示例性地，在 RAHT 编码模式为 RAHT 预测变换编码模式的情况下，解码器对节点的属性编码信息解码得到残差信息，并对节点的属性变换信息进行帧内/帧间预测，确定属性信息预测值。示例性地，属性信息预测值可以是 AC 系数预测值。解码器确定根据残差与属性信息预测值得到重建属性变换信息，如重建 AC 系数，对重建属性变换信息进行 RAHT 逆变换，得到重建属性信息。

在一些实施例中, RAHT 预测变换编码模式包括: 帧间预测变换编码模式或帧内预测变换编码模式。编码器根据帧间预测变换编码模式或帧内预测变换编码模式, 确定相应的 RAHT 预测变换解码模式, 对当前层中的每个节点对应的属性编码信息进行解码。

5 可以理解的是, 本申请实施例中, 解码器根据编码器传递的 RAHT 编码模式对当前层中的节点对应的属性编码信息进行解码重建, 确定节点对应的重建属性信息。由于当前层对应的 RAHT 编码模式是编码器综合考虑了每个节点属性分布特性以及空间分布特性, 通过至少一种 RAHT 编码方式对当前层中的每个节点进行编码, 确定每个节点对应的至少一个编码代价, 根据每个节点对应的至少一个编码代价确定出的对当前层最佳的编码模式, 解码器根据当前层对应的 RAHT 编码模式对应选择解码模式进行解码, 实现了根据当前层对应的 RAHT 编码模式来自适应地解码当前层的属性信息。这样, 通过
10 在多种编解码模式中自适应选择最佳编解码模式, 提升了 RAHT 属性编码效率与点云属性信息的解码效率, 进而提升了点云解码性能。

在一些实施例中, RAHT 预测变换编码模式包括: RAHT 帧内预测变换编码模式; 解码器在 RAHT 编码模式为帧内预测变换编码模式的情况下, 根据当前层中的节点的邻域节点对应的属性信息预测值,
15 确定当前层中的节点对应的属性信息预测值。

在一些实施例中, 在 RAHT 编码模式为帧间预测变换编码模式的情况下, 解码器可以确定当前层中的节点在参考点云中位置对应的同位节点; 基于同位节点的重建属性变换信息, 确定当前层中的节点对应的属性信息预测值。在不存在同位节点的情况下, 根据当前层中的节点的邻域节点对应的属性信息预测值, 确定当前层中的节点对应的属性信息预测值。

20 在一些实施例中, RAHT 预测变换编码模式包括: RAHT 帧内预测变换编码模式; 解码器在 RAHT 编码模式为帧内预测变换编码模式的情况下, 根据当前层中的节点的邻域节点对应的属性信息预测值, 确定当前层中的节点对应的属性信息预测值。

这里, 对解码器

需要说明的是, 在一些实施例中, 解码器在采用 RAHT 帧内/帧间预测解码模式对节点进行解码之前, 先确定当前层中的每个节点的父节点的邻域节点数量是否大于预设第一数量阈值; 在大于预设第一数量阈值的情况下, 确定当前层中的每个节点的邻域节点数量是否大于预设第二数量阈值; 在大于预设第二数量阈值的情况下, 说明可以启用 RAHT 帧内/帧间预测解码模式对节点进行解码。解码器根据当前层对应的 RAHT 编码模式, 确定当前层中每个节点对应的解码模式。否则, 也即当前层中的每个节点的父节点的邻域节点数量不大于预设第一数量阈值, 或者当前层中的每个节点的邻域节点数量不大于
30 预设第二数量阈值的情况下, 解码器对当前层中的每个节点进行 RAHT 变换解码。

在一些实施例中, 与上述编码器对当前层中的节点分组确定 RAHT 编码模式相对应, 解码器通过解析码流, 确定当前层对应的至少两组属性编码信息, 以及至少两组属性编码信息中每组属性编码信息对应的 RAHT 编码模式; 至少两组属性编码信息对应至少两个节点组; 至少两个节点组通过对当前层中的节点划分得到; 每组属性编码信息对应的 RAHT 编码模式基于每个节点组对应的至少一个编码代价确定; 至少一个编码代价为至少一种 RAHT 编码模式对应的编码代价; 根据每组属性编码信息对应的 RAHT 编码模式, 对每组属性编码信息进行解码重建, 确定每组属性编码信息对应的重建属性信息, 从而确定当前层中的节点对应的重建属性信息。

可以理解的是, 解码器根据编码器传递当前层中每个节点组对应的 RAHT 编码模式, 对每个节点组对应的属性编码信息进行解码, 得到每个节点组对应的重建属性信息。由于每个节点组对应的 RAHT 编码模式是编码器通过对每个节点组进行至少一种 RAHT 编码方式的编码与编码代价对比, 针对每个
40 节点组确定出对最佳的编码模式, 从而进一步提升了 RAHT 属性编码效率与点云属性信息的编码效率, 进而提升了点云编码性能。

基于上述实施例, 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, 图 43 为编码器的组成结构示意图, 如图 43 所示, 编码器 20 可以包括:

45 编码部分 211, 被配置为通过至少一种区域自适应分层变换 RAHT 编码模式, 对当前层中的节点的属性信息进行编码, 确定所述节点对应的至少一个编码代价与至少一个候选属性编码信息;

确定部分 212, 被配置为基于所述节点对应的至少一个编码代价, 确定所述当前层对应的 RAHT 编码模式, 并从至少一个候选属性编码信息中确定所述节点对应的属性编码信息;

50 生成部分 213, 被配置为基于所述当前层对应的 RAHT 编码模式与所述节点对应的属性编码信息, 生成码流。

在一些实施例中, 所述至少一个候选属性编码信息包括: 变换编码信息与预测变换编码信息; 所述编码部分 211, 还被配置为通过对点云中每个节点属性信息进行 RAHT 变换编码, 确定所述当前层中

的每个节点对应的属性变换信息与变换编码信息;对所述当前层中的每个节点对应的属性变换信息进行预测编码,确定所述当前层中的每个节点对应的预测变换编码信息;基于所述变换编码信息与所述预测变换编码信息,确定所述节点对应的至少一个编码代价。

5 在一些实施例中,所述编码部分 211,还被配置为确定所述当前层中的每个节点的父节点的邻域节点数量是否大于预设第一数量阈值;在大于所述预设第一数量阈值的情况下,确定所述当前层中的每个节点的邻域节点数量是否大于预设第二数量阈值;在大于所述预设第二数量阈值的情况下,对所述当前层中的每个节点对应的属性变换信息进行预测编码,确定所述当前层中的每个节点对应的预测变换编码信息。

10 在一些实施例中,所述编码部分 211,还被配置为基于点云对应的 RAHT 属性变换编码结构,从体素级别的节点开始,递归地对所述点云中每个节点的属性信息进行 RAHT 变换与编码直至根节点,确定所述点云中每个节点对应的属性变换信息与变换编码信息,从而确定所述当前层中的每个节点对应的变换编码信息;所述属性变换信息通过对属性信息进行 RAHT 变换确定;所述变换编码信息通过对所述属性变换信息进行编码确定。

15 在一些实施例中,所述预测变换编码信息包括:帧间预测变换编码信息;所述编码部分 211,还被配置为对所述当前层中的每个节点对应的属性变换信息进行帧间预测编码,确定所述当前层中的每个节点对应的帧间预测变换编码信息。

在一些实施例中,所述预测变换编码信息包括:帧内预测变换编码信息;所述确定部分 212,还被配置为对所述当前层中的每个节点对应的属性变换信息进行帧内预测编码,确定所述当前层中的每个节点对应的帧内预测变换编码信息。

20 在一些实施例中,所述编码部分 211,还被配置为基于点云对应的 RAHT 属性变换编码结构,由根节点开始,确定所述点云中每个节点对应的 N 个子节点;对于所述每个节点对应的 N 个子节点,对所述 N 个子节点的属性信息进行 RAHT 属性变换与编码,确定所述 N 个子节点对应的属性变换信息与变换编码信息,从而确定所述点云中每个节点对应的属性变换信息与变换编码信息;N 大于 0 且不大于预设子节点数量阈值;根据所述点云中每个节点对应的属性变换信息与变换编码信息,确定所述当前层中的每个节点对应的属性变换信息与变换编码信息。

25 在一些实施例中,所述编码部分 211,还被配置为确定所述当前层中的每个节点在所述点云对应的参考点云中位置对应的同位节点;基于所述同位节点的重建属性变换信息,确定所述当前层中的每个节点对应的属性信息预测值;基于所述属性信息预测值,对所述当前层中的每个节点对应的属性变换信息进行编码,确定所述当前层中的每个节点对应的帧间预测变换编码信息。

30 在一些实施例中,所述编码部分 211,还被配置为确定所述当前层中的每个节点在所述点云对应的参考点云中对应的同位节点;在所述当前层中的每个节点为所述当前层中的父节点的情况下,基于所述同位节点的重建属性变换信息,确定所述当前层中的每个节点对应的属性信息预测值;在所述当前层中的每个节点为所述当前层中的子节点的情况下,基于所述同位节点在所述参考点云中对应的 M 个子节点的重建属性变换信息,确定所述当前层中的每个节点对应的属性信息预测值;M 大于 0 且不大于预设子节点数量阈值;基于所述属性信息预测值,对所述当前层中的每个节点对应的属性变换信息进行编码,确定所述当前层中的每个节点对应的帧间预测变换编码信息。

35 在一些实施例中,所述当前层中的每个节点包括:当前层中的子节点;所述编码部分 211,还被配置为确定每个子节点在所述 M 个子节点中位置对应的同位子节点;在所述同位子节点的重建属性变换信息大于预设信息值阈值的情况下,根据所述同位子节点的重建属性变换信息确定所述当前层中的每个节点的属性信息预测值;在所述同位子节点的重建属性变换信息小于或等于所述预设信息值阈值的情况下,基于所述每个子节点的邻域节点的重建属性变换信息,确定所述每个子节点对应的属性信息预测值。

40 在一些实施例中,所述编码部分 211,还被配置为在所述当前层中的每个节点在所述点云对应的参考点云中不存在位置对应的同位节点的情况下,根据所述当前层中的每个节点的邻域节点的属性信息预测值,确定所述当前层中的每个节点对应的属性信息预测值。

45 在一些实施例中,所述预测变换编码信息包括:帧内预测变换编码信息与帧间预测变换编码信息;所述至少一个编码代价包括:第一率失真代价、第二率失真代价与第三率失真代价,所述编码部分 211,还被配置为基于所述变换编码信息进行解码重建,确定所述当前层中的每个节点对应的第一率失真代价;基于所述帧内预测变换编码信息进行解码重建,确定所述当前层中的每个节点对应的第二率失真代价;基于所述帧间预测变换编码信息进行解码重建,确定所述当前层中的每个节点对应的第三率失真代价。

50 在一些实施例中,所述确定部分 212,还被配置为基于所述当前层中的每个节点对应的第一率失真代价,确定所述当前层对应的第一编码代价;所述第一编码代价对应 RAHT 变换编码模式;基于所述当前层中的每个节点对应的第二率失真代价,确定所述当前层对应的第二编码代价;所述第二编码代价

对应 RAHT 帧内预测变换编码模式；基于所述当前层中的每个节点对应的第三率失真代价，确定所述当前层对应的第三编码代价；所述第三编码代价对应 RAHT 帧间预测变换编码模式；通过对比所述第一编码代价、所述第一编码代价、以及所述第三编码代价，确定所述当前层对应的 RAHT 编码模式。

在一些实施例中，所述确定部分 212，还被配置为基于所述当前层的节点的父节点的属性信息，以及所述父节点的邻域节点的重建属性信息，确定所述当前层对应的 RAHT 编码模式；利用所述当前层对应的 RAHT 编码模式，对所述当前层中的节点的属性信息进行编码，确定节点对应的属性编码信息。

在一些实施例中，所述确定部分 212，还被配置为在所述当前层中的节点的父节点的属性信息，与
所述父节点的邻域节点的重建属性信息之间的误差大于预设误差阈值的情况下；将 RAHT 变换编码模式确定为所述当前层对应的 RAHT 编码模式；所述变换编码模式表征对属性信息进行 RAHT 变换与编码；在所述当前层中的节点的父节点的属性信息，与
所述父节点的邻域节点的重建属性信息之间的误差小于或等于所述预设误差阈值的情况下，将 RAHT 预测变换编码模式确定为所述当前层对应的 RAHT 编码模式；所述预测变换编码模式表征对属性信息进行 RAHT 变换确定的属性变换系数进行预测编码。

在一些实施例中，所述生成部分 213，还被配置为根据所述当前层对应的 RAHT 编码模式，确定所述当前层对应的 RAHT 编码模式标志位；所述 RAHT 编码模式标志位用于表征 RAHT 变换编码模式或 RAHT 预测变换编码模式；所述 RAHT 预测变换编码模式包括：帧间预测变换编码模式或帧内预测变换编码模式；基于所述节点对应的属性编码信息，确定所述当前层对应的属性编码信息；通过对所述点云中的每个 RAHT 变换层进行处理，确定所述每个 RAHT 变换层对应的编码模式标志位与
所述每个 RAHT 变换层对应的属性编码信息；基于所述每个 RAHT 变换层对应的编码模式标志位与
所述每个 RAHT 变换层对应的属性编码信息，生成所述码流。

在一些实施例中，所述编码部分 211，还被配置为基于所述 RAHT 属性变换编码结构，从根节点开始沿每个空间坐标轴方向进行下采样，确定所述点云对应的至少一个 RAHT 变换层；所述至少一个 RAHT 变换层包含所述当前层。

在一些实施例中，所述确定部分 2121，还被配置为确定所述点云对应的预测变换使能标志位；所述预测变换使能标志位表征是否对所述点云中的 RAHT 变换层使能预测变换编码模式；和/或，确定所述点云对应的编码模式标志位的数量；

所述生成部分 213，还配置为将所述预测使能变换标志位与
所述编码模式标志位的数量中的至少一个写入所述码流。

在一些实施例中，所述编码部分 211，还被配置为将所述当前层中的节点划分为至少两个节点组；

对于所述至少两个节点组中的每个节点组，基于所述每个节点组中的每个节点对应的至少一个编码代价，确定所述每个节点组对应的至少一个编码代价；基于所述每个节点组对应的至少一个编码代价，确定所述每个节点组对应的 RAHT 编码模式；基于所述每个节点组对应的 RAHT 编码模式，确定所述每个节点组中的每个节点对应的属性编码信息，从而确定所述每个节点组对应的属性编码信息；根据所述每个节点组对应的 RAHT 编码模式以及每个节点组对应的属性编码信息，确定所述当前层对应的至少两组属性编码信息与
每组属性编码信息对应的 RAHT 编码模式，并生成码流。

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。

需要说明的是，以上编码器装置实施例的描述，与上述编码方法实施例的描述是类似的，具有同方法实施例相似的有益效果。对于本发明装置实施例中未披露的技术细节，请参照本发明方法实施例的描述而理解。

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存储介质（即计算机可读存储介质），应用于编码器 20，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。

基于上述编码器 20 的组成以及计算机可读存储介质，图 44 为编码器的组成结构示意图二，如图

44 所示, 编码器 20 可以包括: 第一存储器 221 和第一处理器 222, 第一通信接口 223 和第一总线系统 224。第一存储器 221、第一处理器 222、第一通信接口 223 通过第一总线系统 224 耦合在一起。可理解, 第一总线系统 224 用于实现这些组件之间的连接通信。第一总线系统 224 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, 将各种总线都标为第一总线系统 224。

其中, 第一通信接口 223, 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;

所述第一存储器 221, 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 222, 用于确定当前层的节点的第一节点数量和所述当前层的节点对应的子节点的第二节点数量; 其中, 所述第一节点数量和所述第二节点数量用于确定是否对所述当前层的节点进行 RAHT 变换; 根据所述第一节点数量和所述第二节点数量, 确定所述当前层的节点对应的子节点的重建值。

可以理解, 本申请实施例中的第一存储器 221 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 221 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 222 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 222 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 222 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)、现场可编程门阵列 (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 221, 第一处理器 222 读取第一存储器 221 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 222 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。

图 45 为解码器的组成结构示意图, 如图 45 所示, 解码器 30 可以包括:

解析部分 311, 被配置为解析码流, 确定当前层对应的 RAHT 编码模式与所述当前层中的节点对应的属性编码信息; 所述当前层对应的 RAHT 编码模式是编码器通过至少一种区域自适应分层变换 RAHT 编码模式对当前层的节点属性信息进行编码, 确定所述节点对应的至少一个编码代价, 并基于节点对应的至少一个编码代价确定的;

解码重建部分 312, 被配置为基于所述当前层对应的 RAHT 编码模式, 对所述当前层中的节点对应的属性编码信息进行解码重建, 确定所述节点对应的重建属性信息。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。

在一些实施例中,所述解码重建部分 312,还被配置为在所述 RAHT 编码模式为 RAHT 变换编码模式的情况下,对所述属性编码信息进行 RAHT 变换解码,确定所述节点对应的重建属性信息;其中,所述 RAHT 变换编码模式表征编码器对节点属性信息进行 RAHT 变换与编码。

在一些实施例中,所述解码重建部分 312,还被配置为在所述 RAHT 编码模式为 RAHT 预测变换编码模式的情况下,对所述当前层中的节点对应的属性变换信息进行预测,确定所述当前层中的节点对应的属性信息预测值;基于所述属性信息预测值,对所述属性编码信息进行 RAHT 变换解码,确定所述当前层中的节点对应的重建属性信息;其中,所述 RAHT 预测变换编码模式表征编码器对节点属性信息进行 RAHT 变换,并对 RAHT 变换确定的属性变换信息进行预测编码。

在一些实施例中,所述 RAHT 预测变换编码模式包括: RAHT 帧内预测变换编码模式;所述解码重建部分 312,还被配置为在所述 RAHT 编码模式为帧内预测变换编码模式的情况下,根据所述当前层中的节点的邻域节点对应的属性信息预测值,确定所述当前层中的节点对应的属性信息预测值。

在一些实施例中,所述 RAHT 预测变换编码模式包括: RAHT 帧间预测变换编码模式,所述解码重建部分 312,还被配置为在所述 RAHT 编码模式为帧间预测变换编码模式的情况下,确定所述当前层中的节点在参考点云中位置对应的同位节点;基于所述同位节点的重建属性变换信息,确定所述当前层中的节点对应的属性信息预测值。

在一些实施例中,所述解码重建部分 312,还被配置为在不存在所述同位节点的情况下,根据所述当前层中的节点的邻域节点对应的属性信息预测值,确定所述当前层中的节点对应的属性信息预测值。

在一些实施例中,所述解析部分 311,还被配置为根据所述当前层对应的 RAHT 编码模式标志位,确定所述当前层对应的 RAHT 编码模式;其中,所述 RAHT 编码模式标志位用于表征 RAHT 变换编码模式或 RAHT 预测变换编码模式;所述 RAHT 预测变换编码模式包括:帧间预测变换编码模式或帧内预测变换编码模式。

在一些实施例中,所述解析部分 311,还被配置为确定点云对应的编码模式标志位的数量;所述点云为所述当前层所在的点云;根据所述编码模式标志位的数量,对所述码流中的 RAHT 编码模式标志位进行解析,确定所述点云中每一层对应的 RAHT 编码模式标志位。

在一些实施例中,所述解码重建部分 312,还被配置为通过解析所述码流,确定所述当前层对应的至少两组属性编码信息,以及所述至少两组属性编码信息中每组属性编码信息对应的 RAHT 编码模式;所述至少两组属性编码信息对应至少两个节点组;所述至少两个节点组通过对所述当前层中的节点划分得到;所述每组属性编码信息对应的 RAHT 编码模式基于每个节点组对应的至少一个编码代价确定;所述至少一个编码代价为至少一种 RAHT 编码模式对应的编码代价;根据所述每组属性编码信息对应的 RAHT 编码模式,对所述每组属性编码信息进行解码重建,确定所述每组属性编码信息对应的重建属性信息,从而确定所述当前层中的节点对应的重建属性信息。

在一些实施例中,所述解析部分 311,还被配置为通过解析所述码流,确定所述点云对应的预测变换使能标志位;所述预测变换使能标志位表征是否对所述点云使能预测变换编码模式;在所述预测变换使能标志位表征使能预测变换编码模式的情况下,确定所述点云对应的编码模式标志位的数量。

在一些实施例中,所述解码重建部分 312,还被配置为确定所述当前层中的每个节点的父节点的邻域节点数量是否大于预设第一数量阈值;在大于所述预设第一数量阈值的情况下,确定所述当前层中的每个节点的邻域节点数量是否大于预设第二数量阈值;在大于所述预设第二数量阈值的情况下,根据所述当前层对应的 RAHT 编码模式,确定所述当前层中每个节点对应的解码模式。

需要说明的是,以上编码器装置实施例的描述,与上述编码方法实施例的描述是类似的,具有同方法实施例相似的有益效果。对于本发明装置实施例中未披露的技术细节,请参照本发明方法实施例的描述而理解。

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存储介质,即计算机可读存储介质,应用于解码器 30,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。

基于上述解码器 30 的组成以及计算机可读存储介质,图 46 为解码器的组成结构示意图二,如图

46 所示, 解码器 30 可以包括: 第二存储器 321 和第二处理器 322, 第二通信接口 323 和第二总线系统 324。第二存储器 321 和第二处理器 322, 第二通信接口 323 通过第二总线系统 324 耦合在一起。可理解, 第二总线系统 324 用于实现这些组件之间的连接通信。第二总线系统 324 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, 将各种总线都标为第二总线系统 324。其中,

第二通信接口 323, 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;

所述第二存储器 321, 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 322, 用于确定当前层的节点的第一节点数量和所述当前层的节点对应的子节点的第二节点数量; 其中, 所述第一节点数量和所述第二节点数量用于确定是否对所述当前层的节点进行 RAHT 变换; 根据所述第一节点数量和所述第二节点数量, 确定所述当前层的节点对应的子节点的属性重建值。

可以理解, 本申请实施例中的第二存储器 321 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二存储器 321 旨在包括但不限于这些和任意其它适合类型的存储器。

而第二处理器 322 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二处理器 322 中的硬件的集成逻辑电路或者软件形式的指令完成。上述的第二处理器 322 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)、现场可编程门阵列 (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。该存储介质位于第二存储器 321, 第二处理器 322 读取第二存储器 321 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。

在本申请的再一实施例中, 参见图 47, 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 47 所示, 编解码系统 230 可以包括编码器 2301 和解码器 2302。

在本申请实施例中, 编码器 2301 可以为前述实施例中任一项所述的编码器, 解码器 2302 可以为前述实施例中任一项所述的解码器。

在本申请的又一实施例中, 本申请实施例还提供一种码流, 该码流是根据待编码信息进行比特编码生成的; 其中, 所述待编码信息至少包括: 当前层对应的 RAHT 编码模式与所述当前层中的节点对应的属性编码信息; 所述当前层对应的 RAHT 编码模式基于所述当前层中的节点对应的至少一个编码代价确定; 所述当前层中的节点对应的至少一个编码代价是通过至少一种 RAHT 编码模式, 对所述当前层中的节点的属性信息进行编码确定的。

需要说明的是, 在本申请的实施例中, 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所揭露的方法，在不冲突的情况下可以任意组合，得到新的方法实施例。

5 本申请所提供的几个产品实施例中所揭露的特征，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所揭露的特征，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。

10 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

本申请实施例提供了一种点云编解码方法、编码器、解码器、码流以及存储介质，编码器通过至少一种 RAHT 编码方式对当前层中的每个节点进行编码，确定每个节点对应的至少一个编码代价与至少一个候选属性编码信息，从而可以根据每个节点对应的至少一个编码代价确定出对当前层最佳的编码模式，利用该编码模式确定当前层中每个节点对应的属性编码信息，并将当前层对应的 RAHT 编码模式与节点对应的属性编码信息传输至解码端。解码器根据编码器传递的 RAHT 编码模式对当前层中的节点对应的属性编码信息进行解码重建，确定节点对应的重建属性信息。这样，通过引入多种编码模式并结合编码代价进行最佳编解码模式的自适应选择，综合考虑了每个节点的属性分布特性以及空间分布特性，提升了 RAHT 属性编解码效率与点云属性信息的编解码效率，进而提升了点云编解码性能。

15

20

权利要求书

1、一种点云解码方法，应用于解码器，所述方法包括：

解析码流，确定当前层对应的 RAHT 编码模式与所述当前层中的节点对应的属性编码信息；所述当前层对应的 RAHT 编码模式是编码器通过至少一种区域自适应分层变换 RAHT 编码模式对当前层的节点的属性信息进行编码，确定所述节点对应的至少一个编码代价，并基于节点对应的至少一个编码代价确定的；

基于所述当前层对应的 RAHT 编码模式，对所述当前层中的节点对应的属性编码信息进行解码重建，确定所述节点对应的重建属性信息。

2、根据权利要求 1 所述的方法，其中，所述基于所述当前层对应的 RAHT 编码模式，对所述当前层中的节点对应的属性编码信息进行解码重建，确定所述节点对应的重建属性信息，包括：

在所述 RAHT 编码模式为 RAHT 变换编码模式的情况下，对所述属性编码信息进行 RAHT 变换解码，确定所述节点对应的重建属性信息；

其中，所述 RAHT 变换编码模式表征编码器对节点的属性信息进行 RAHT 变换与编码。

3、根据权利要求 1 所述的方法，其中，所述基于所述当前层对应的 RAHT 编码模式，对所述当前层中的节点对应的属性编码信息进行解码重建，确定所述节点对应的重建属性信息，包括：

在所述 RAHT 编码模式为 RAHT 预测变换编码模式的情况下，对所述当前层中的节点对应的属性变换信息进行预测，确定所述当前层中的节点对应的属性信息预测值；

基于所述属性信息预测值，对所述属性编码信息进行 RAHT 变换解码，确定所述当前层中的节点对应的重建属性信息；

其中，所述 RAHT 预测变换编码表征编码器对节点的属性信息进行 RAHT 变换，并对 RAHT 变换确定的属性变换信息进行预测编码。

4、根据权利要求 3 所述的方法，其中，所述 RAHT 预测变换编码模式包括：RAHT 帧内预测变换编码模式；所述在所述 RAHT 编码模式为 RAHT 预测变换编码模式的情况下，对所述当前层中的节点对应的属性变换信息进行预测，确定所述当前层中的节点对应的属性信息预测值，包括：

在所述 RAHT 编码模式为帧内预测变换编码模式的情况下，根据所述当前层中的节点的邻域节点对应的属性信息预测值，确定所述当前层中的节点对应的属性信息预测值。

5、根据权利要求 3 所述的方法，其中，所述 RAHT 预测变换编码模式包括：RAHT 帧间预测变换编码模式，所述在所述 RAHT 编码模式为 RAHT 预测变换编码模式的情况下，对所述当前层中的节点对应的属性变换信息进行预测，确定所述当前层中的节点对应的属性信息预测值，包括：

在所述 RAHT 编码模式为帧间预测变换编码模式的情况下，确定所述当前层中的节点在参考点云中位置对应的同位节点；

基于所述同位节点的重建属性变换信息，确定所述当前层中的节点对应的属性信息预测值。

6、根据权利要求 5 所述的方法，其中，所述方法还包括：

在不存在所述同位节点的情况下，根据所述当前层中的节点的邻域节点对应的属性信息预测值，确定所述当前层中的节点对应的属性信息预测值。

7、根据权利要求 1-6 任一项所述的方法，其中，所述确定当前层对应的 RAHT 编码模式，包括：根据所述当前层对应的 RAHT 编码模式标志位，确定所述当前层对应的 RAHT 编码模式；其中，

所述 RAHT 编码模式标志位用于表征 RAHT 变换编码模式或 RAHT 预测变换编码模式；所述 RAHT 预测变换编码模式包括：帧间预测变换编码模式或帧内预测变换编码模式。

8、根据权利要求 7 所述的方法，其中，所述方法还包括：

确定点云对应的编码模式标志位的数量；所述点云为所述当前层所在的点云；

根据所述编码模式标志位的数量，对所述码流中的 RAHT 编码模式标志位进行解析，确定所述点云中每一层对应的 RAHT 编码模式标志位。

9、根据权利要求 1 所述的方法，其中，所述方法还包括：

通过解析所述码流，确定所述当前层对应的至少两组属性编码信息，以及所述至少两组属性编码信息中每组属性编码信息对应的 RAHT 编码模式；所述至少两组属性编码信息对应至少两个节点组；所述至少两个节点组通过对所述当前层中的节点划分得到；所述每组属性编码信息对应的 RAHT 编码模式基于每个节点组对应的至少一个编码代价确定；所述至少一个编码代价为至少一种 RAHT 编码模式对应的编码代价；

根据所述每组属性编码信息对应的 RAHT 编码模式，对所述每组属性编码信息进行解码重建，确

定所述每组属性编码信息对应的重建属性信息，从而确定所述当前层中的节点对应的重建属性信息。

10、根据权利要求 8 所述的方法，其中，所述方法还包括：

通过解析所述码流，确定所述点云对应的预测变换使能标志位；所述预测变换使能标志位表征是否对所述点云使能预测变换编码模式；

5 在所述预测变换使能标志位表征使能预测变换编码模式的情况下，确定所述点云对应的编码模式标志位的数量。

11、根据权利要求 7 所述的方法，其中，所述方法还包括：

确定所述当前层中的每个节点的父节点的邻域节点数量是否大于预设第一数量阈值；

10 在大于所述预设第一数量阈值的情况下，确定所述当前层中的每个节点的邻域节点数量是否大于预设第二数量阈值；

在大于所述预设第二数量阈值的情况下，根据所述当前层对应的 RAHT 编码模式，确定所述当前层中每个节点对应的解码模式。

12、一种点云编码方法，应用于编码器，所述方法包括：

15 通过至少一种区域自适应分层变换 RAHT 编码模式，对当前层中的节点的属性信息进行编码，确定所述节点对应的至少一个编码代价与至少一个候选属性编码信息；

基于所述节点对应的至少一个编码代价，确定所述当前层对应的 RAHT 编码模式，并从所述至少一个候选属性编码信息中确定所述节点对应的属性编码信息；

基于所述当前层对应的 RAHT 编码模式与所述节点对应的属性编码信息，生成码流。

20 13、根据权利要求 12 所述的方法，其中，所述至少一个候选属性编码信息包括：变换编码信息与预测变换编码信息；所述通过至少一种区域自适应分层变换 RAHT 编码模式，对当前层中的节点的属性信息进行编码，确定所述节点对应的至少一个编码代价与至少一个候选属性编码信息，包括：

通过对点云中每个节点的属性信息进行 RAHT 变换编码，确定所述当前层中的每个节点对应的属性变换信息与变换编码信息；

25 对所述当前层中的每个节点对应的属性变换信息进行预测编码，确定所述当前层中的每个节点对应的预测变换编码信息；

基于所述变换编码信息与所述预测变换编码信息，确定所述节点对应的至少一个编码代价。

14、根据权利要求 13 所述的方法，其中，所述对所述当前层中的每个节点对应的属性变换信息进行预测编码，确定所述当前层中的每个节点对应的预测变换编码信息，包括：

确定所述当前层中的每个节点的父节点的邻域节点数量是否大于预设第一数量阈值；

30 在大于所述预设第一数量阈值的情况下，确定所述当前层中的每个节点的邻域节点数量是否大于预设第二数量阈值；

在大于所述预设第二数量阈值的情况下，对所述当前层中的每个节点对应的属性变换信息进行预测编码，确定所述当前层中的每个节点对应的预测变换编码信息。

35 15、根据权利要求 13 或 14 所述的方法，其中，所述通过对点云中每个节点的属性信息进行 RAHT 变换编码，确定所述当前层中的每个节点对应的属性变换信息与变换编码信息，包括：

基于点云对应的 RAHT 属性变换编码结构，从体素级别的节点开始，递归地对所述点云中每个节点的属性信息进行 RAHT 变换与编码直至根节点，确定所述点云中每个节点对应的属性变换信息与变换编码信息，从而确定所述当前层中的每个节点对应的变换编码信息；所述属性变换信息通过对属性信息进行 RAHT 变换确定；所述变换编码信息通过对所述属性变换信息进行编码确定。

40 16、根据权利要求 13 或 14 所述的方法，其中，所述预测变换编码信息包括：帧间预测变换编码信息；所述对所述当前层中的每个节点对应的属性变换信息进行预测编码，确定所述当前层中的每个节点对应的预测变换编码信息，包括：

对所述当前层中的每个节点对应的属性变换信息进行帧间预测编码，确定所述当前层中的每个节点对应的帧间预测变换编码信息。

45 17、根据权利要求 13 或 14 所述的方法，其中，所述预测变换编码信息包括：帧内预测变换编码信息；对所述当前层中的每个节点对应的属性变换信息进行预测编码，确定所述当前层中的每个节点对应的预测变换编码信息，包括：

对所述当前层中的每个节点对应的属性变换信息进行帧内预测编码，确定所述当前层中的每个节点对应的帧内预测变换编码信息。

50 18、根据权利要求 13 或 14 所述的方法，其中，所述通过对点云中每个节点的属性信息进行 RAHT 变换编码，确定所述当前层中的每个节点对应的属性变换信息与变换编码信息，包括：

基于点云对应的 RAHT 属性变换编码结构，由根节点开始，确定所述点云中每个节点对应的 N 个

子节点;

对于所述每个节点对应的N个子节点,对所述N个子节点的属性信息进行RAHT属性变换与编码,确定所述N个子节点对应的属性变换信息与变换编码信息,从而确定所述点云中每个节点对应的属性变换信息与变换编码信息;N大于0且不大于预设子节点数量阈值;

5 根据所述点云中每个节点对应的属性变换信息与变换编码信息,确定所述当前层中的每个节点对应的属性变换信息与变换编码信息。

19、根据权利要求16所述的方法,其中,所述对所述当前层中的每个节点对应的属性变换系数进行帧间预测编码,确定所述当前层中的每个节点对应的帧间预测变换编码信息,包括:

确定所述当前层中的每个节点在所述点云对应的参考点云中位置对应的同位节点;

10 基于所述同位节点的重建属性变换信息,确定所述当前层中的每个节点对应的属性信息预测值;

基于所述属性信息预测值,对所述当前层中的每个节点对应的属性变换信息进行编码,确定所述当前层中的每个节点对应的帧间预测变换编码信息。

20、根据权利要求16所述的方法,其中,所述对所述当前层中的每个节点对应的属性变换信息进行帧间预测编码,确定所述当前层中的每个节点对应的帧间预测变换编码信息,包括:

15 确定所述当前层中的每个节点在所述点云对应的参考点云中对应的同位节点;

在所述当前层中的每个节点为所述当前层中的父节点的情况下,基于所述同位节点的重建属性变换信息,确定所述当前层中的每个节点对应的属性信息预测值;

20 在所述当前层中的每个节点为所述当前层中的子节点的情况下,基于所述同位节点在所述参考点云中对应的M个子节点的重建属性变换信息,确定所述当前层中的每个节点对应的属性信息预测值;M大于0且不大于预设子节点数量阈值;

基于所述属性信息预测值,对所述当前层中的每个节点对应的属性变换信息进行编码,确定所述当前层中的每个节点对应的帧间预测变换编码信息。

21、根据权利要求20所述的方法,其中,所述当前层中的每个节点包括:当前层中的子节点;所述基于所述同位节点在所述参考点云中对应的M个子节点的重建属性变换信息,确定所述当前层中的每个节点对应的属性信息预测值,包括:

25 确定每个子节点在所述M个子节点中位置对应的同位子节点;

在所述同位子节点的重建属性变换信息大于预设信息值阈值的情况下,根据所述同位子节点的重建属性变换信息确定所述当前层中的每个节点对应的属性信息预测值;

30 在所述同位子节点的重建属性变换信息小于或等于所述预设信息值阈值的情况下,基于所述每个子节点的邻域节点的重建属性变换信息,确定所述每个子节点对应的属性信息预测值。

22、根据权利要求19-21任一项所述的方法,其中,所述方法还包括:

在所述当前层中的每个节点在所述点云对应的参考点云中不存在位置对应的同位节点的情况下,根据所述当前层中的每个节点的邻域节点的属性信息预测值,确定所述当前层中的每个节点对应的属性信息预测值。

35 23、根据权利要求13、14、19-21任一项所述的方法,其中,所述预测变换编码信息包括:帧内预测变换编码信息与帧间预测变换编码信息;所述至少一个编码代价包括:第一率失真代价、第二率失真代价与第三率失真代价,所述基于所述变换编码信息与所述预测变换编码信息,确定所述当前层中的每个节点对应的至少一个编码代价,包括:

基于所述变换编码信息进行解码重建,确定所述当前层中的每个节点对应的第一率失真代价;

40 基于所述帧内预测变换编码信息进行解码重建,确定所述当前层中的每个节点对应的第二率失真代价;

基于所述帧间预测变换编码信息进行解码重建,确定所述当前层中的每个节点对应的第三率失真代价。

45 24、根据权利要求23所述的方法,其中,所述基于所述节点对应的至少一个编码代价,确定所述当前层对应的RAHT编码模式,包括:

基于所述当前层中的每个节点对应的第一率失真代价,确定所述当前层对应的第一编码代价;所述第一编码代价对应RAHT变换编码模式;

基于所述当前层中的每个节点对应的第二率失真代价,确定所述当前层对应的第二编码代价;所述第二编码代价对应RAHT帧内预测变换编码模式;

50 基于所述当前层中的每个节点对应的第三率失真代价,确定所述当前层对应的第三编码代价;所述第三编码代价对应RAHT帧间预测变换编码模式;

通过对比所述第一编码代价、所述第二编码代价、以及所述第三编码代价,确定所述当前层对应的

RAHT 编码模式。

25、根据权利要求 12 所述的方法，其中，所述方法还包括：

基于所述当前层的节点的父节点的属性信息，以及所述父节点的邻域节点的重建属性信息，确定所述当前层对应的 RAHT 编码模式；

5 利用所述当前层对应的 RAHT 编码模式，对所述当前层中的节点的属性信息进行编码，确定节点对应的属性编码信息。

26、根据权利要求 25 所述的方法，其中，所述基于所述当前层的节点的父节点的属性信息，以及所述父节点的邻域节点的重建属性信息，确定所述当前层对应的 RAHT 编码模式，包括：

10 在所述当前层中的节点的父节点的属性信息，与所述父节点的邻域节点的重建属性信息之间的误差大于预设误差阈值的情况下；将 RAHT 变换编码模式确定为所述当前层对应的 RAHT 编码模式；所述变换编码模式表征对属性信息进行 RAHT 变换与编码；

在所述当前层中的节点的父节点的属性信息，与所述父节点的邻域节点的重建属性信息之间的误差小于或等于所述预设误差阈值的情况下，将 RAHT 预测变换编码模式确定为所述当前层对应的 RAHT 编码模式；所述预测变换编码模式表征对属性信息进行 RAHT 变换确定的属性变换系数进行预测编码。

15 27、根据权利要求 12-14、19-21、24-26 中任一项所述的方法，其中，所述基于所述当前层对应的 RAHT 编码模式与所述节点对应的属性编码信息，生成码流，包括：

根据所述当前层对应的 RAHT 编码模式，确定所述当前层对应的 RAHT 编码模式标志位；所述 RAHT 编码模式标志位用于表征 RAHT 变换编码模式或 RAHT 预测变换编码模式；所述 RAHT 预测变换编码模式包括：帧间预测变换编码模式或帧内预测变换编码模式；

20 基于所述节点对应的属性编码信息，确定所述当前层对应的属性编码信息；

通过对所述点云中的每个 RAHT 变换层进行处理，确定所述每个 RAHT 变换层对应的编码模式标志位与所述每个 RAHT 变换层对应的属性编码信息；

基于所述每个 RAHT 变换层对应的编码模式标志位与所述每个 RAHT 变换层对应的属性编码信息，生成所述码流。

25 28、根据权利要求 27 所述的方法，其中，所述方法还包括：

基于所述 RAHT 属性变换编码结构，从根节点开始沿每个空间坐标轴方向进行下采样，确定所述点云对应的至少一个 RAHT 变换层；所述至少一个 RAHT 变换层包含所述当前层。

29、根据权利要求 27 所述的方法，其中，所述方法还包括：

30 确定所述点云对应的预测变换使能标志位；所述预测变换使能标志位表征是否对所述点云中的 RAHT 变换层使能预测变换编码模式；和/或，

确定所述点云对应的编码模式标志位的数量；

将所述预测使能变换标志位与所述编码模式标志位的数量中的至少一个写入所述码流。

30、根据权利要求 12-14、19-21、24-26、28、29 中任一项所述的方法，其中，所述方法还包括：

将所述当前层中的节点划分为至少两个节点组；

35 对于所述至少两个节点组中的每个节点组，基于所述每个节点组中的每个节点对应的至少一个编码代价，确定所述每个节点组对应的至少一个编码代价；

基于所述每个节点组对应的至少一个编码代价，确定所述每个节点组对应的 RAHT 编码模式；

基于所述每个节点组对应的 RAHT 编码模式，确定所述每个节点组中的每个节点对应的属性编码信息，从而确定所述每个节点组对应的属性编码信息；

40 根据所述每个节点组对应的 RAHT 编码模式以及每个节点组对应的属性编码信息，确定所述当前层对应的至少两组属性编码信息与每组属性编码信息对应的 RAHT 编码模式，并生成码流。

31、一种解码器，包括：

45 解析部分，被配置为解析码流，确定当前层对应的 RAHT 编码模式与所述当前层中的节点对应的属性编码信息；所述当前层对应的 RAHT 编码模式是编码器通过至少一种区域自适应分层变换 RAHT 编码模式对当前层的节点的属性信息进行编码，确定所述节点对应的至少一个编码代价，并基于节点对应的至少一个编码代价确定的；

解码重建部分，被配置为基于所述当前层对应的 RAHT 编码模式，对所述当前层中的节点对应的属性编码信息进行解码重建，确定所述节点对应的重建属性信息。

32、一种编码器，包括：

50 编码部分，被配置为通过至少一种区域自适应分层变换 RAHT 编码模式，对当前层中的节点的属性信息进行编码，确定所述节点对应的至少一个编码代价与至少一个候选属性编码信息；

确定部分，被配置为基于所述节点对应的至少一个编码代价，确定所述当前层对应的 RAHT 编码

模式，并从至少一个候选属性编码信息中确定所述节点对应的属性编码信息；

生成部分，被配置为基于所述当前层对应的 RAHT 编码模式与所述节点对应的属性编码信息，生成码流。

33、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 12-30 中任一项所述的方法。

34、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，

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 1-11 中任一项所述的方法。

35、一种码流，所述码流是根据待编码信息进行比特编码生成的；其中，所述待编码信息至少包括：当前层对应的 RAHT 编码模式与所述当前层中的节点对应的属性编码信息；所述当前层对应的 RAHT 编码模式基于所述当前层中的节点对应的至少一个编码代价确定；所述当前层中的节点对应的至少一个编码代价是通过至少一种 RAHT 编码模式，对所述当前层中的节点的属性信息进行编码确定的。

36、一种存储介质，其中，所述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 1-11 中任一项所述的方法、或者实现如权利要求 12-30 中任一项所述的方法。

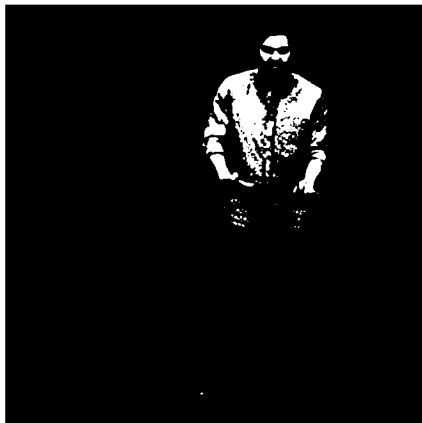


图 1A



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

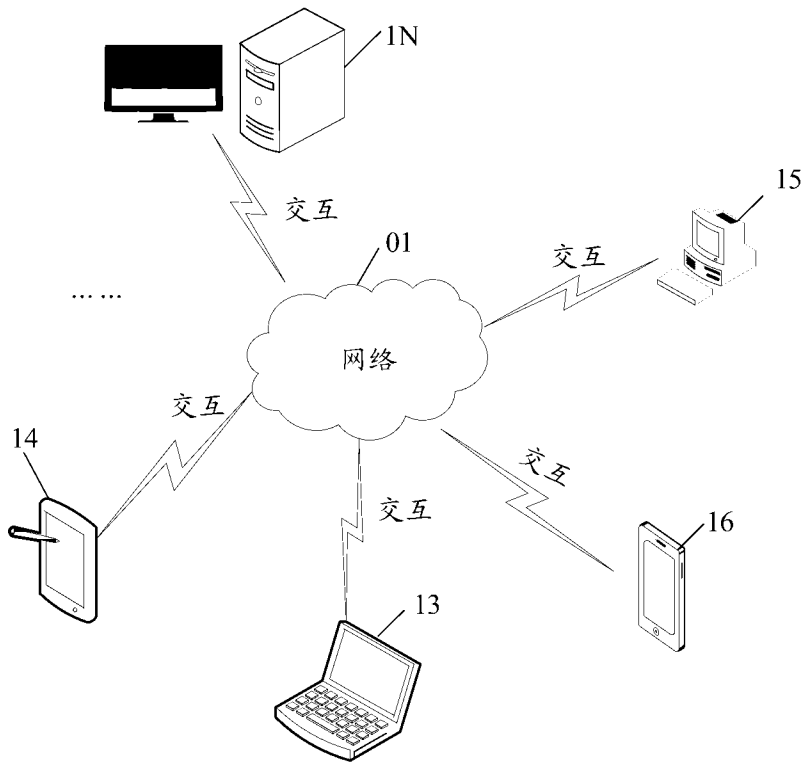


图 3

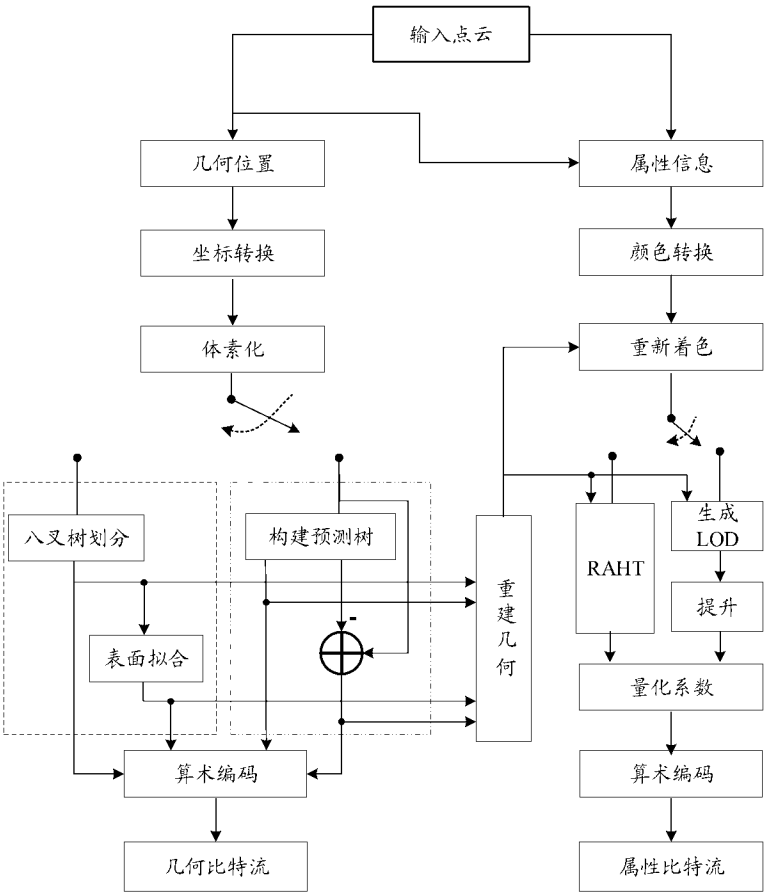


图 4A

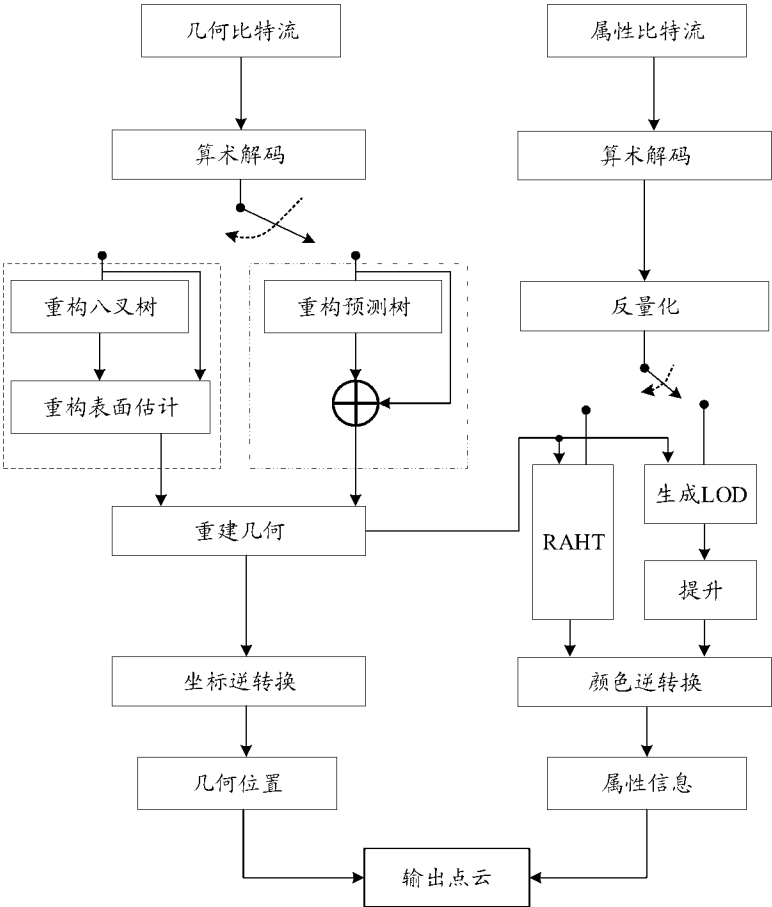
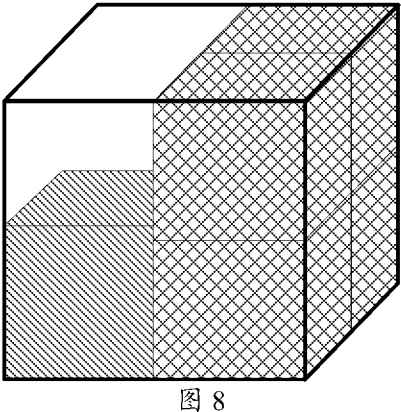
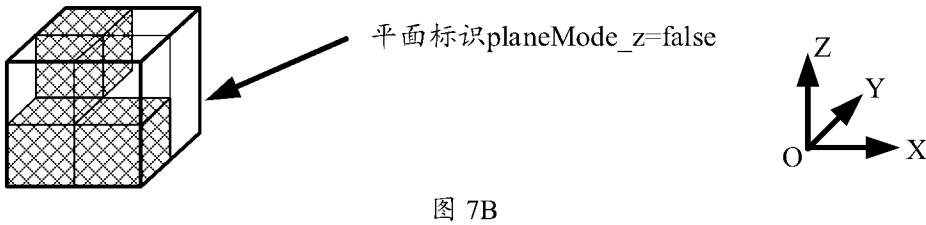
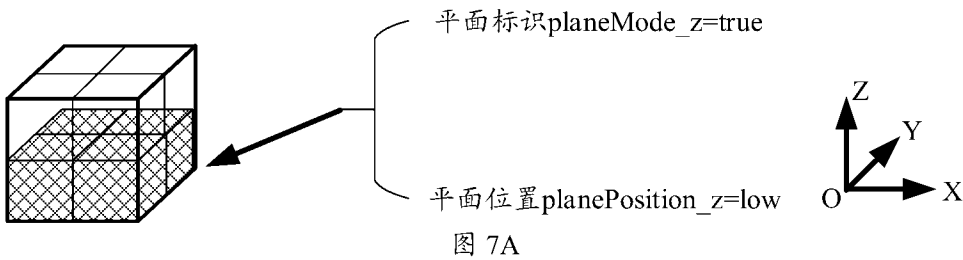
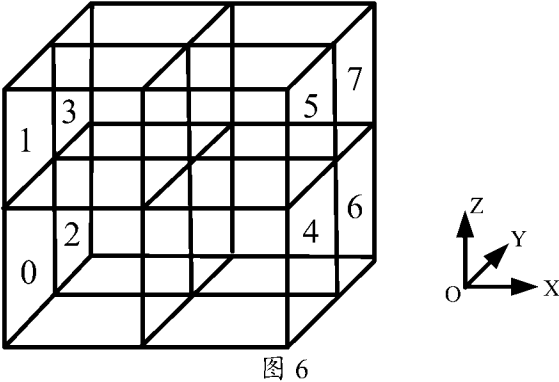
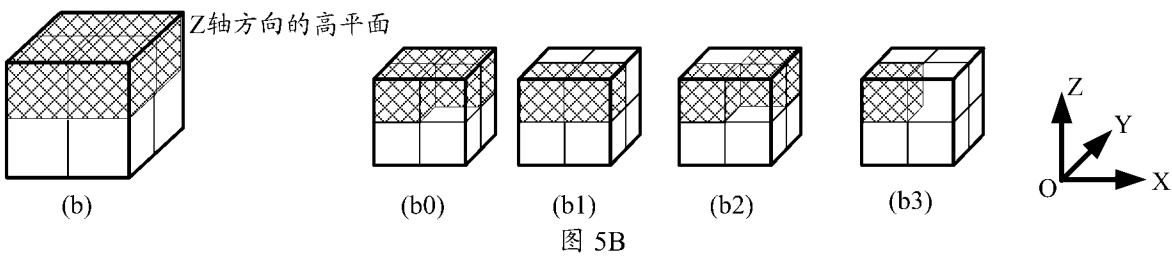
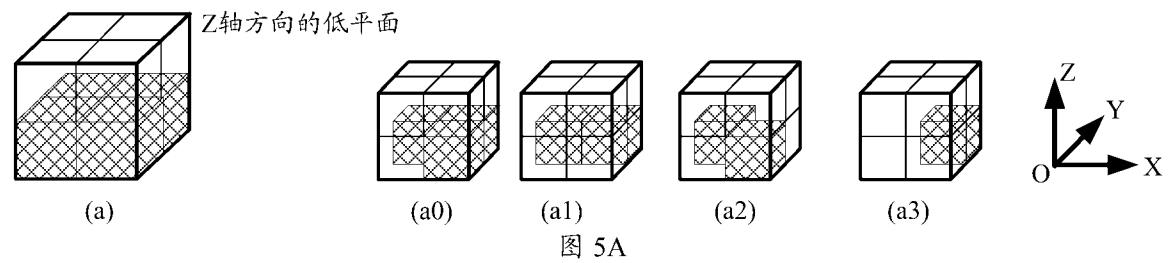


图 4B



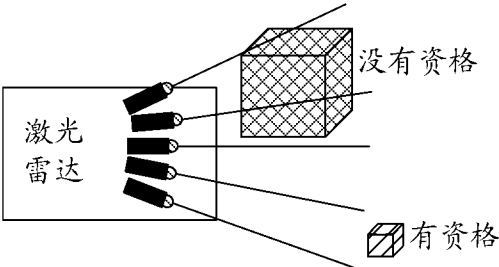


图 9

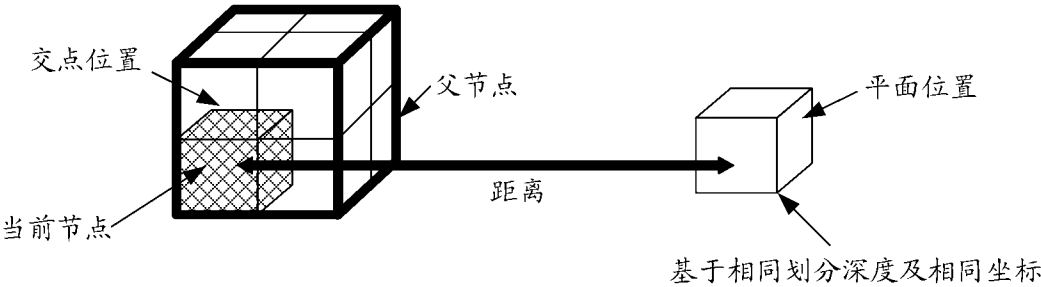


图 10

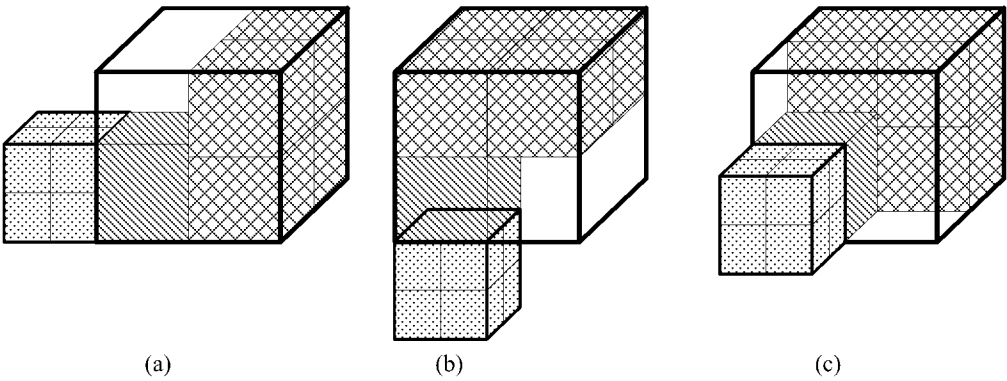


图 11

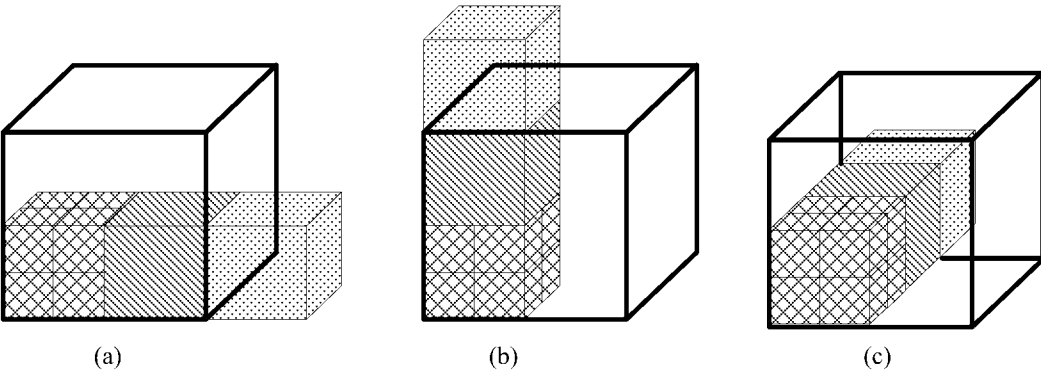


图 12

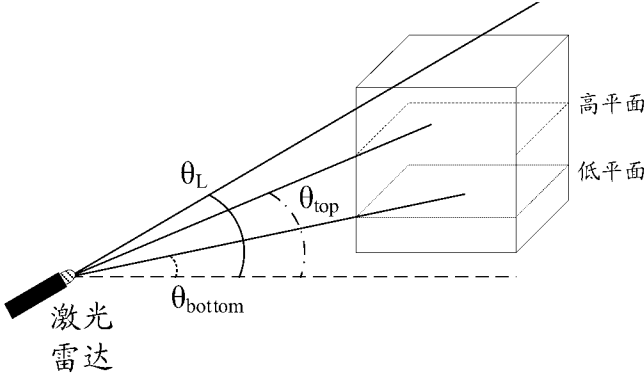


图 13

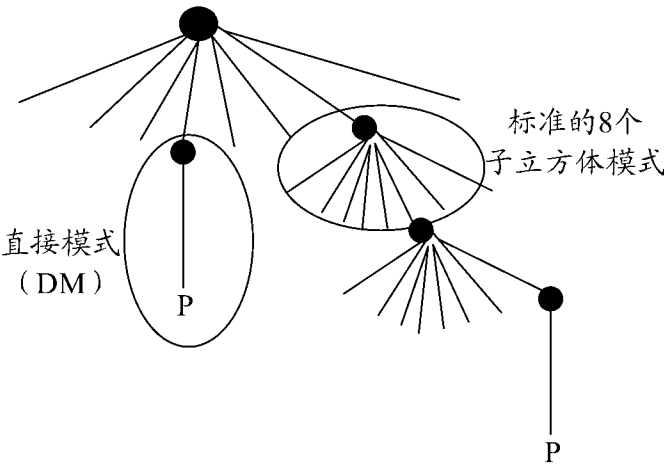


图 14

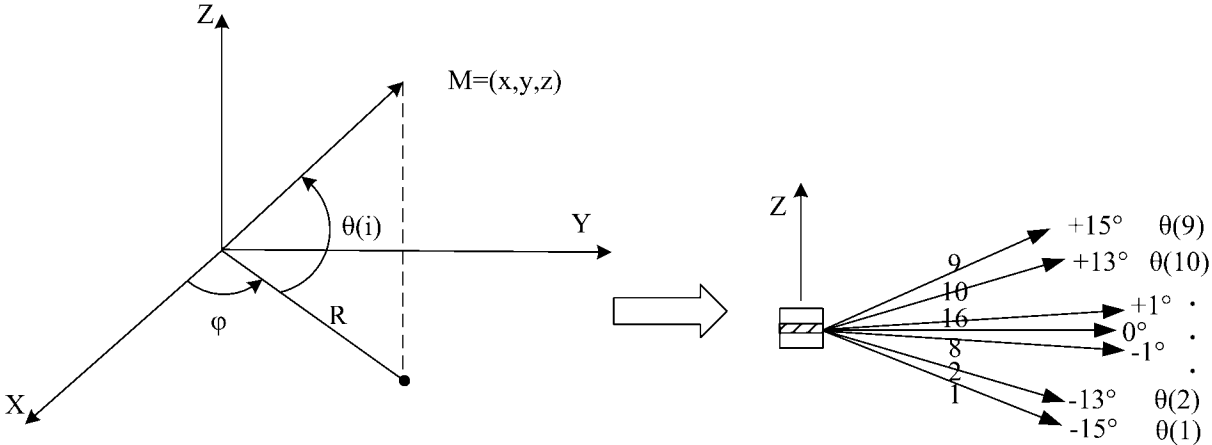


图 15

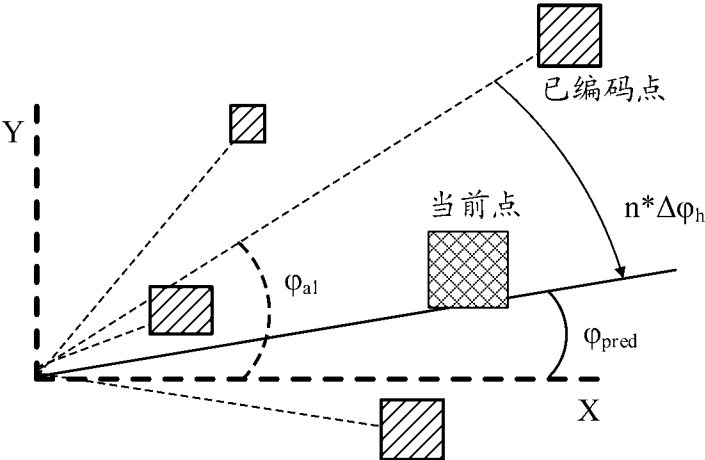


图 16

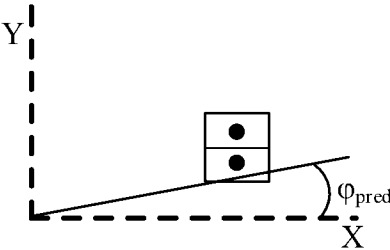


图 17A

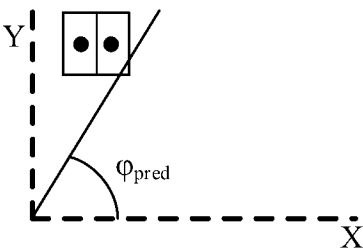


图 17B

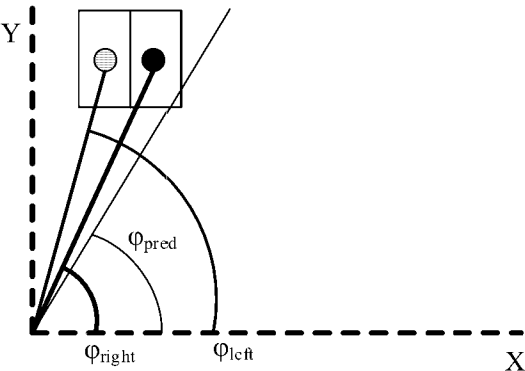


图 18

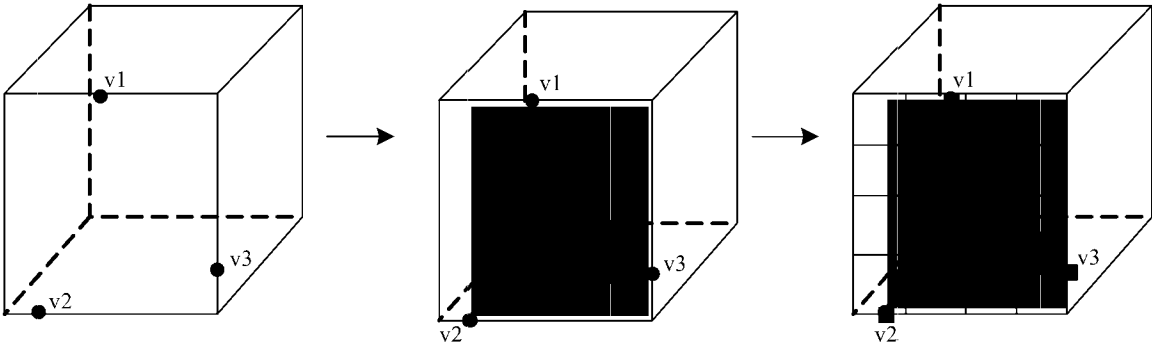
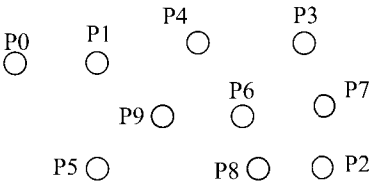


图 19A

图 19B

图 19C



原始顺序

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

LOD层顺序

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

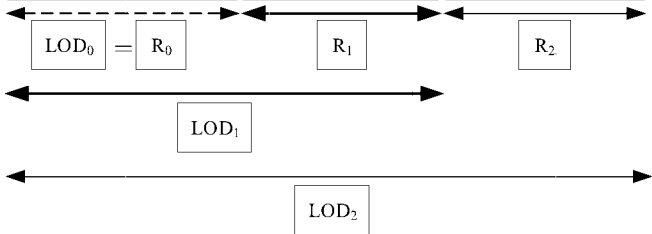


图 20

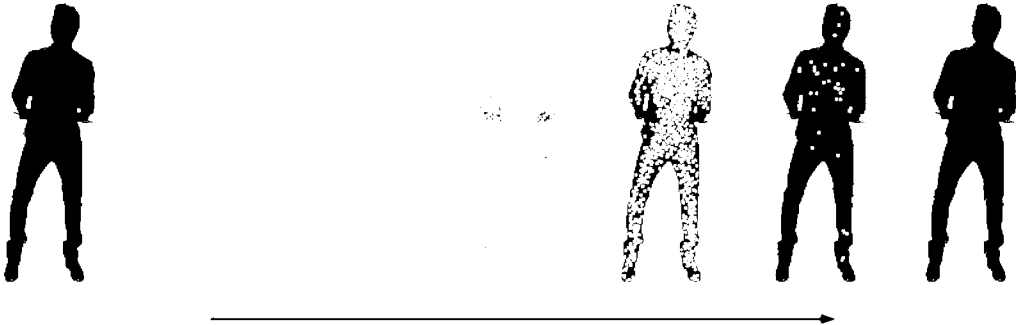


图 21

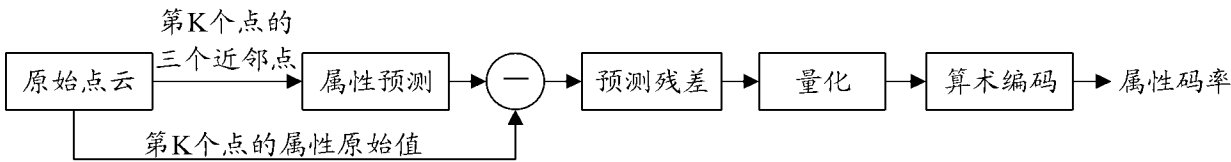


图 22

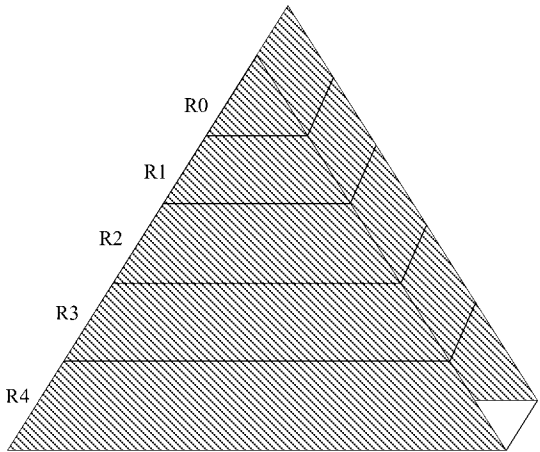


图 23

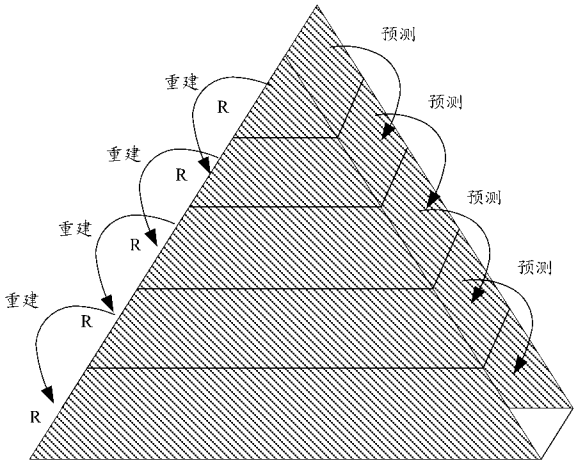


图 24

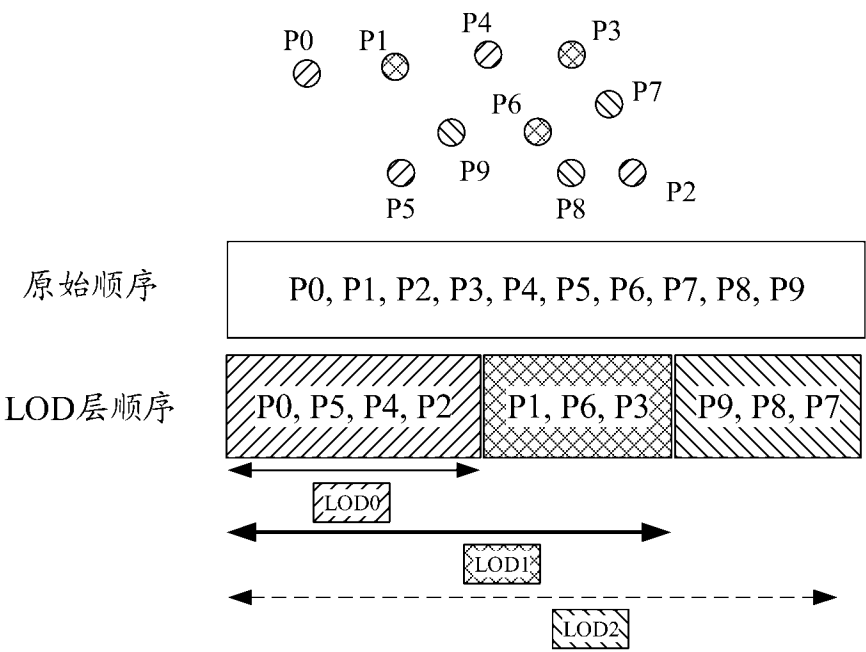


图 25

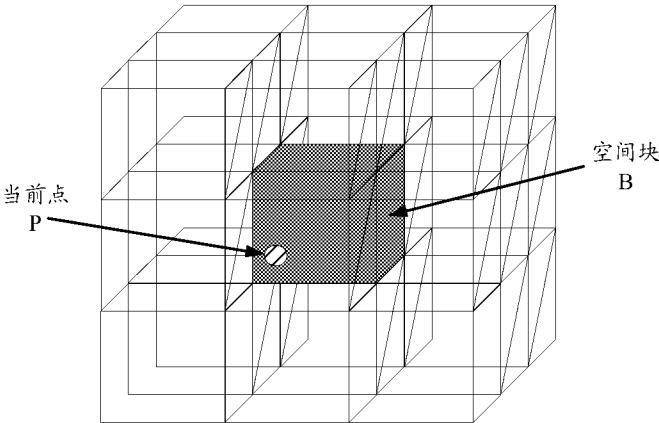


图 26

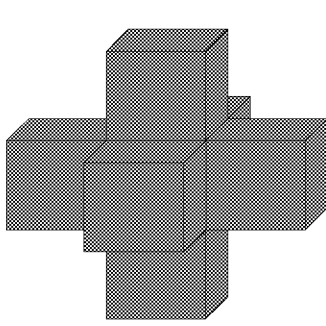


图 27A

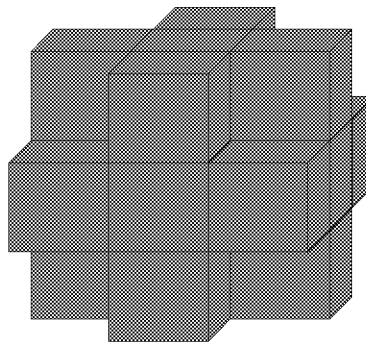


图 27B

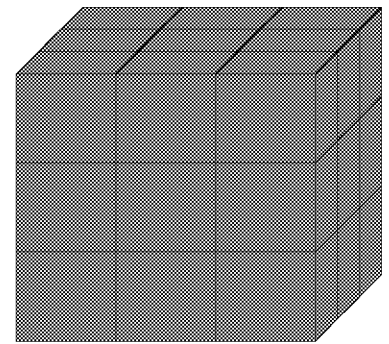


图 27C

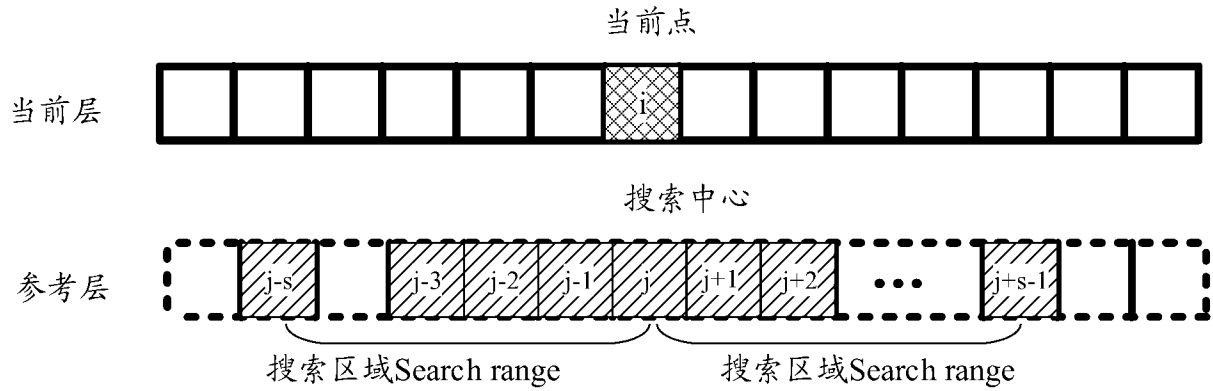


图 28

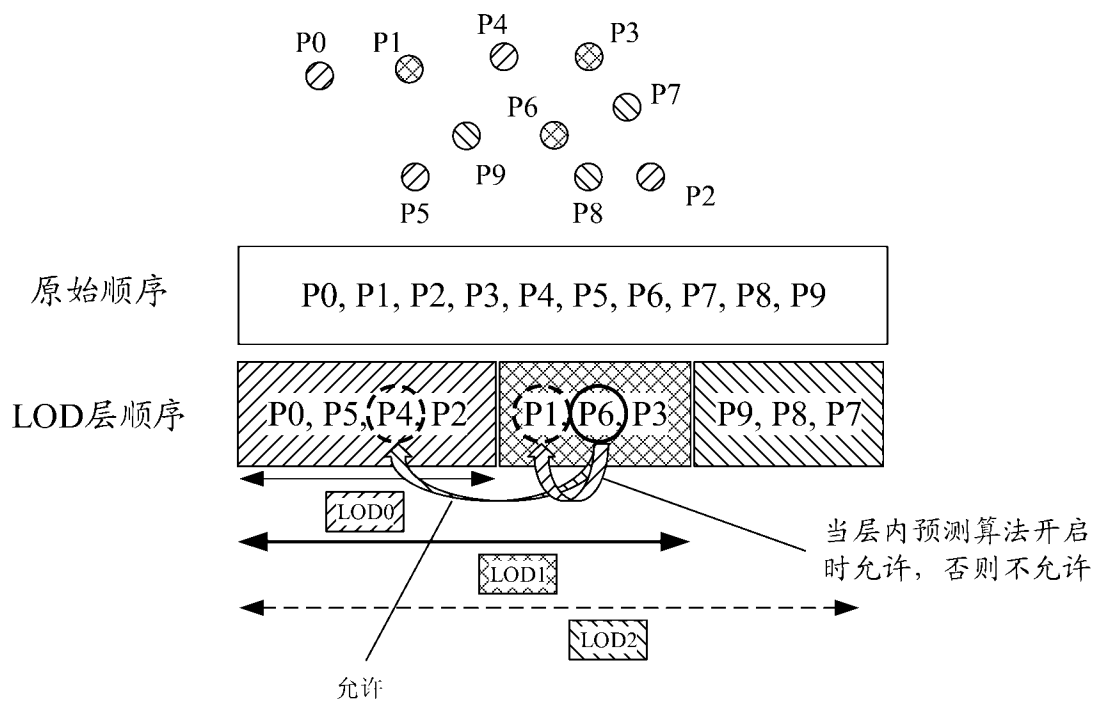


图 29

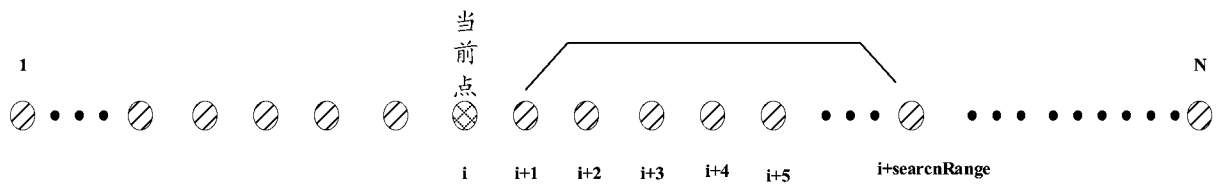


图 30

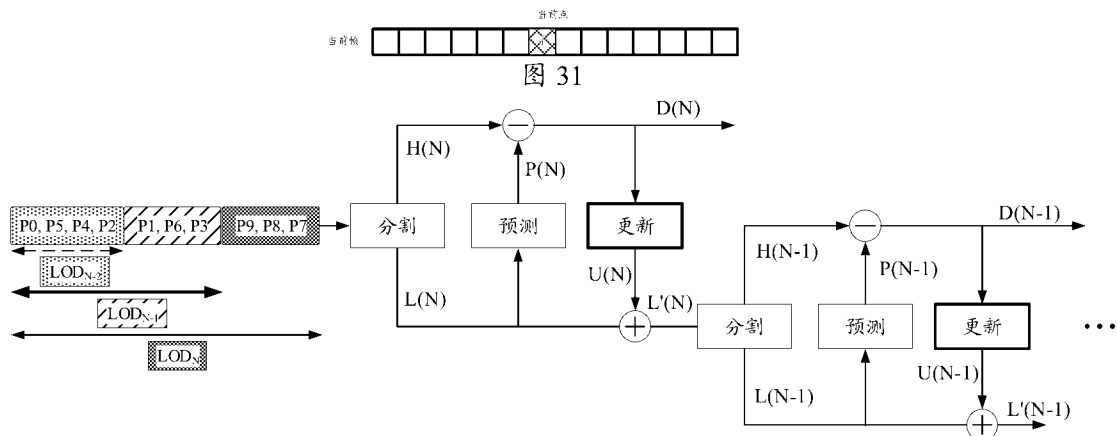
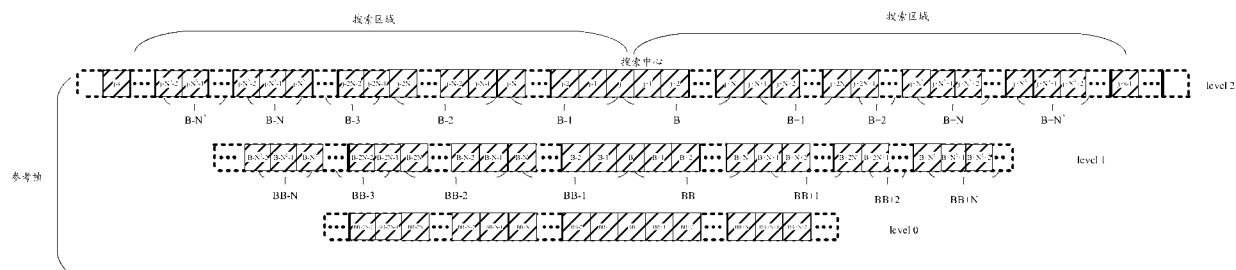


图 32

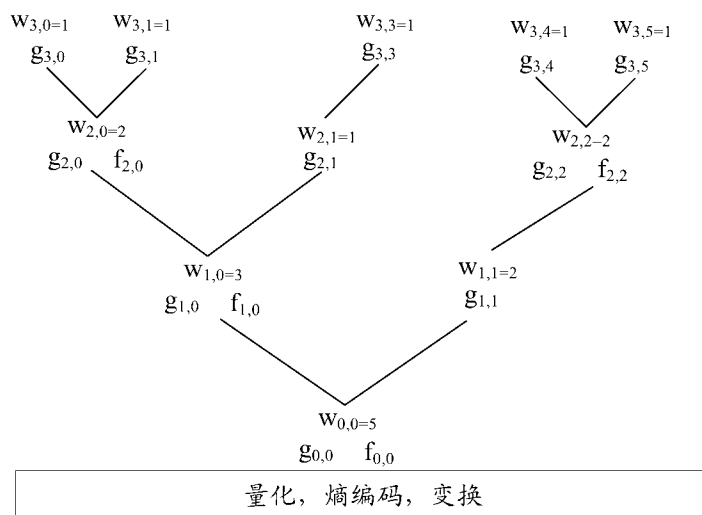


图 33

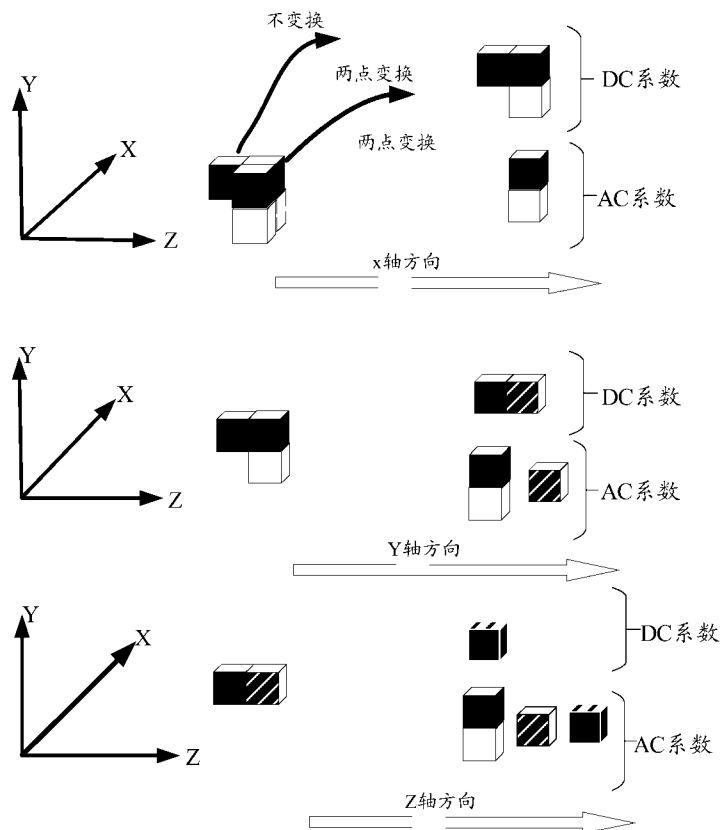


图 34

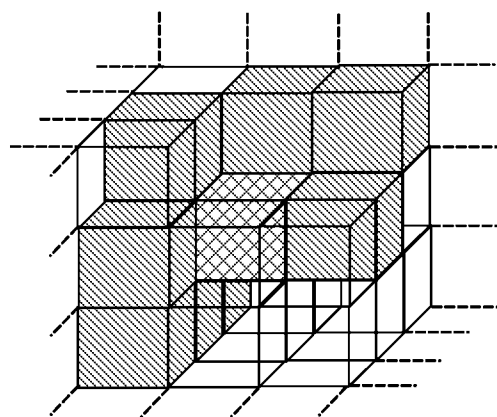
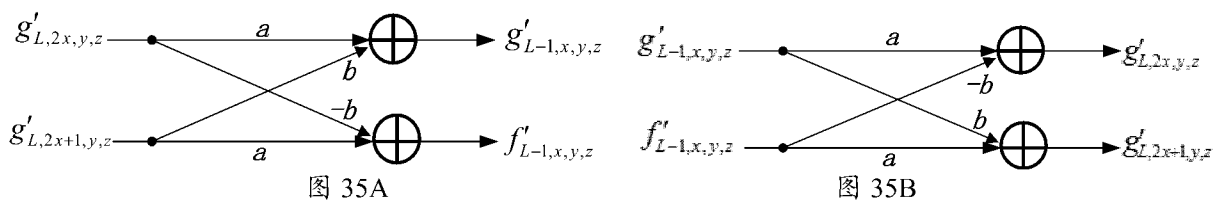


图 36

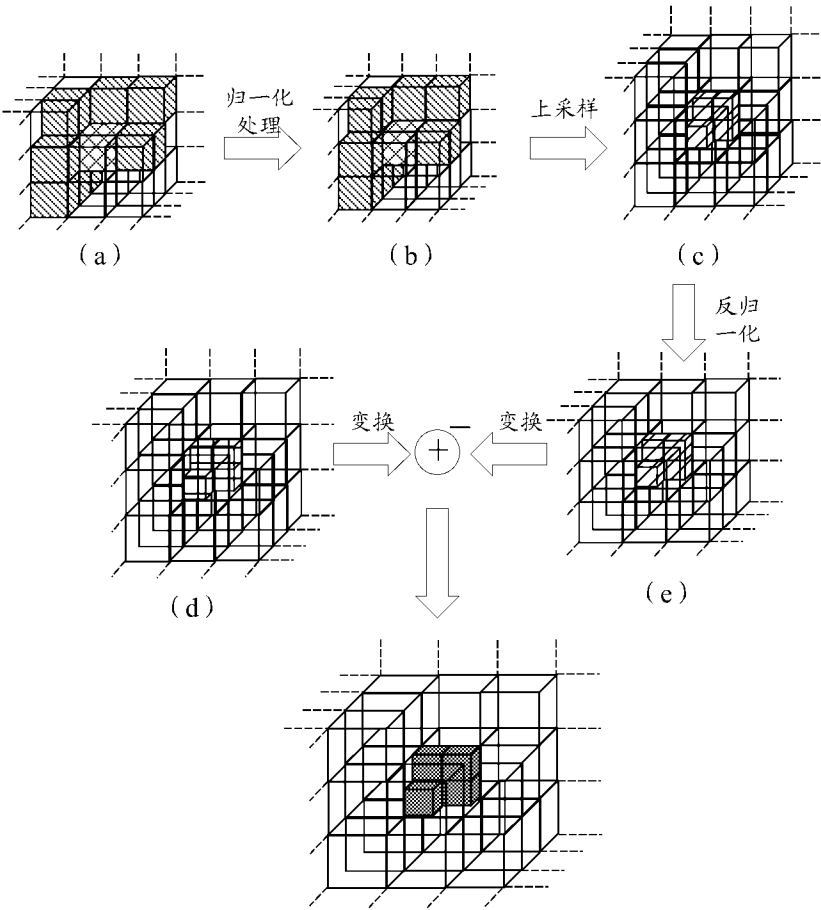


图 37

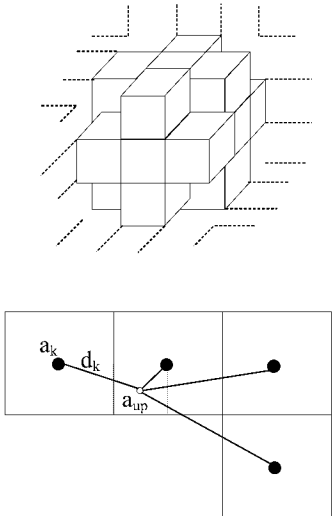
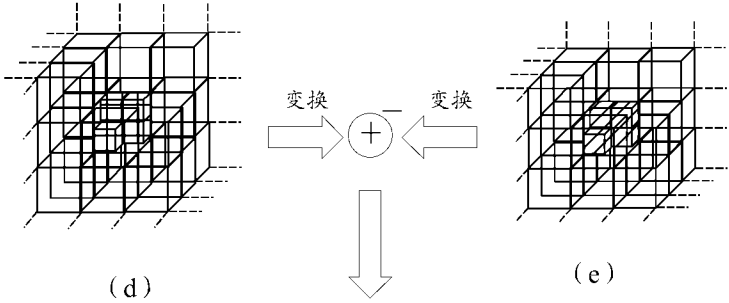


图 38



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

图 39

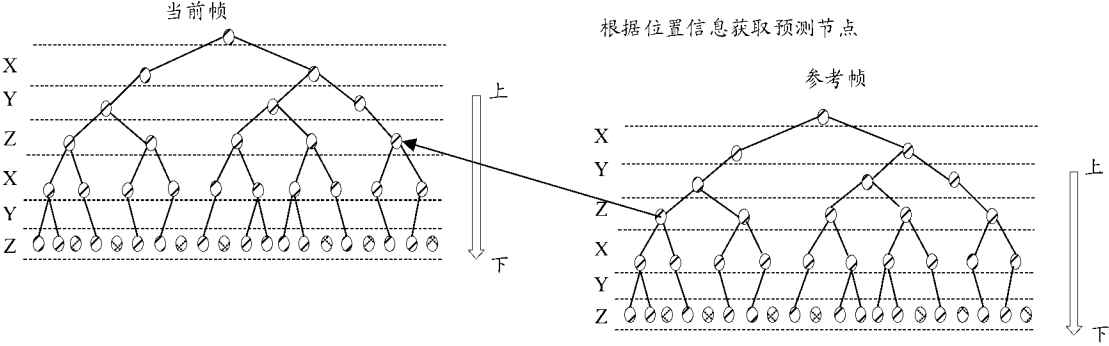


图 40

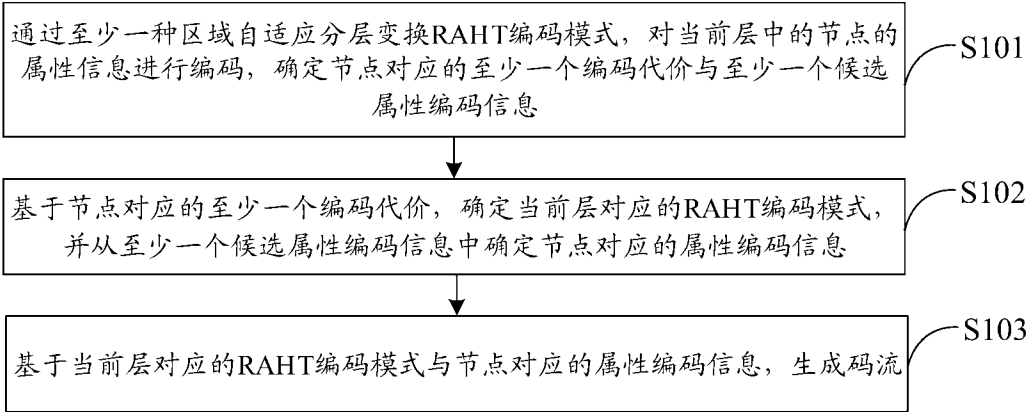


图 41

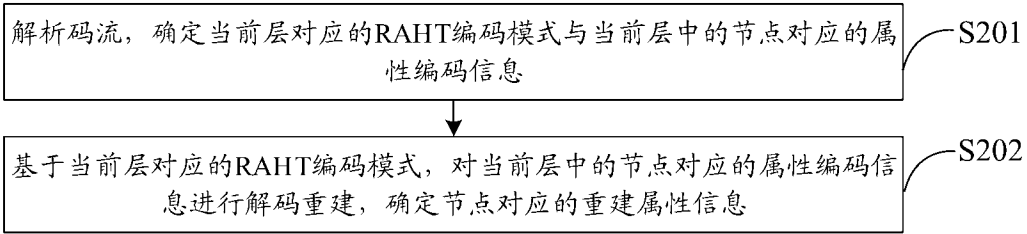


图 42

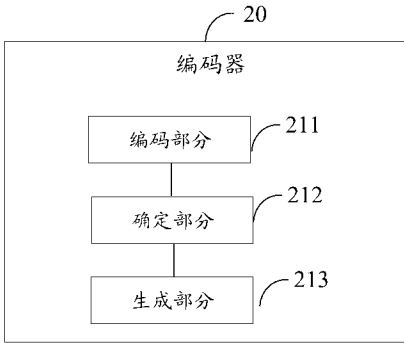


图 43

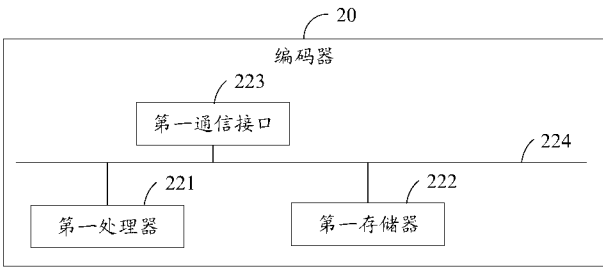


图 44

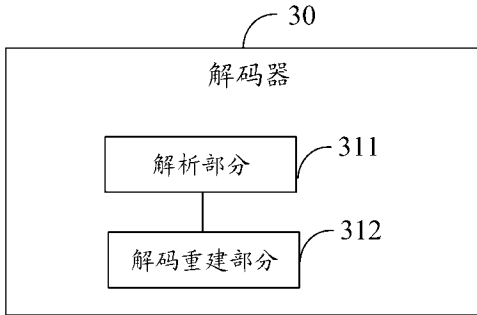


图 45

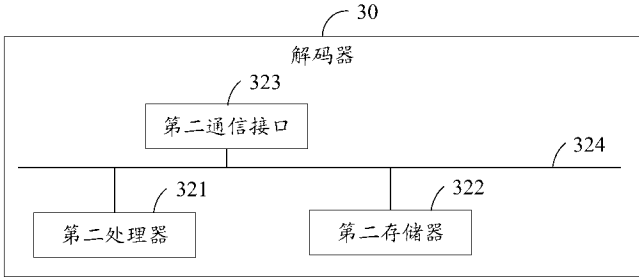


图 46

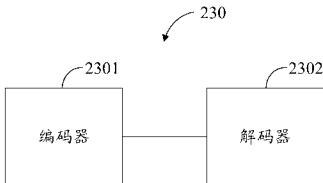


图 47

INTERNATIONAL SEARCH REPORT

International application No.

PCT/CN2023/106659

A. CLASSIFICATION OF SUBJECT MATTER

H04N 19/103(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, WPABSC, ENTXTC, VCN, IEEE, JVET: 点云, 编码, RAHT, 区域自适应分层变换, 节点, 属性, 颜色, 代价, 模式, 候选, point cloud, coding, region-adaptive hierarchical transform, node, attribute, color, cost, mode, candidate

C. DOCUMENTS CONSIDERED TO BE RELEVANT

Category*	Citation of document, with indication, where appropriate, of the relevant passages	Relevant to claim No.
X	CN 108632607 A (PEKING UNIVERSITY SHENZHEN GRADUATE SCHOOL) 09 October 2018 (2018-10-09) description, paragraphs [0009]-[0045]	1-36
A	CN 113678466 A (BLACKBERRY LTD.) 19 November 2021 (2021-11-19) entire document	1-36
A	CN 115714864 A (PENG CHENG LABORATORY) 24 February 2023 (2023-02-24) entire document	1-36
A	CN 114600464 A (BEIJING XIAOMI MOBILE SOFTWARE CO., LTD. et al.) 07 June 2022 (2022-06-07) entire document	1-36
A	WO 2023075453 A1 (LG ELECTRONICS INC.) 04 May 2023 (2023-05-04) entire document	1-36
A	WO 2022042538 A1 (PEKING UNIVERSITY SHENZHEN GRADUATE SCHOOL) 03 March 2022 (2022-03-03) entire document	1-36



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

14 December 2023

Date of mailing of the international search report

20 December 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/CN2023/106659

Patent document cited in search report			Publication date (day/month/year)	Patent family member(s)			Publication date (day/month/year)
CN	108632607	A	09 October 2018	US	2020137399	A1	30 April 2020
				US	10939123	B2	02 March 2021
				WO	2019213986	A1	14 November 2019
CN	113678466	A	19 November 2021	US	2020302651	A1	24 September 2020
				US	10964068	B2	30 March 2021
				WO	2020187710	A1	24 September 2020
				EP	3942830	A1	26 January 2022
CN	115714864	A	24 February 2023	None			
CN	114600464	A	07 June 2022	WO	2022073156	A1	14 April 2022
				IN	202347031853	A	16 June 2023
WO	2023075453	A1	04 May 2023	None			
WO	2022042538	A1	03 March 2022	CN	114095735	A	25 February 2022

A. 主题的分类 H04N 19/103(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)) CNTXT,CNABS,WPABSC,ENTXTC,VCN,IEEEJVET: 点云, 编码, RAHT, 区域自适应分层变换, 节点, 属性, 颜色, 代价, 模式, 候选, point cloud, coding, region-adaptive hierarchical transform, node, attribute, color, cost, mode, candidate		
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 108632607 A (北京大学深圳研究生院) 2018年10月9日 (2018 - 10 - 09) 说明书第9-45段	1-36
A	CN 113678466 A (黑莓有限公司) 2021年11月19日 (2021 - 11 - 19) 全文	1-36
A	CN 115714864 A (鹏城实验室) 2023年2月24日 (2023 - 02 - 24) 全文	1-36
A	CN 114600464 A (北京小米移动软件有限公司 等) 2022年6月7日 (2022 - 06 - 07) 全文	1-36
A	WO 2023075453 A1 (LG ELECTRONICS INC.) 2023年5月4日 (2023 - 05 - 04) 全文	1-36
A	WO 2022042538 A1 (PEKING UNIVERSITY SHENZHEN GRADUATE SCHOOL) 2022年3月3日 (2022 - 03 - 03) 全文	1-36
☐ 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12月14日	国际检索报告邮寄日期 2023年12月20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 盛建军 电话号码 (+86) 010-53961819	

国际检索报告
关于同族专利的信息

国际申请号

PCT/CN2023/106659

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	108632607	A	2018年10月9日	US	2020137399	A1	2020年4月30日
				US	10939123	B2	2021年3月2日
				WO	2019213986	A1	2019年11月14日
CN	113678466	A	2021年11月19日	US	2020302651	A1	2020年9月24日
				US	10964068	B2	2021年3月30日
				WO	2020187710	A1	2020年9月24日
				EP	3942830	A1	2022年1月26日
CN	115714864	A	2023年2月24日	无			
CN	114600464	A	2022年6月7日	WO	2022073156	A1	2022年4月14日
				IN	202347031853	A	2023年6月16日
WO	2023075453	A1	2023年5月4日	无			
WO	2022042538	A1	2022年3月3日	CN	114095735	A	2022年2月25日