(19)国家知识产权局

(12)发明专利申请

(10)申请公布号 CN 116325733 A
(43)申请公布日 2023.06.23

(21)申请号 202180067757.0

(22)申请日 2021.10.07

(30)优先权数据
63/088,938 2020.10.07 US
63/090,629 2020.10.12 US
63/091,821 2020.10.14 US
17/495,621 2021.10.06 US

(85)PCT国际申请进入国家阶段日
2023.03.31

(86)PCT国际申请的申请数据
PCT/US2021/054003 2021.10.07

(87)PCT国际申请的公布数据
WO2022/076708 EN 2022.04.14

(71)申请人 高通股份有限公司
地址 美国加利福尼亚州

(72)发明人 A·K·拉马苏布拉莫尼安 B·雷 G·范德奥韦拉 M·卡尔切维茨

(74)专利代理机构 北京市柳沈律师事务所 11105
专利代理师 安之斐

(51)Int.Cl.
H04N 19/13(2006.01)

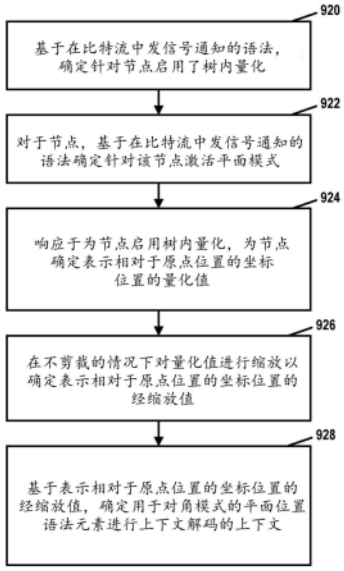
权利要求书6页 说明书61页 附图14页

(54)发明名称

几何点云压缩中的角模式和树内量化

(57)摘要

一种用于对包括点云数据的比特流进行解码的设备可以被配置为:基于在比特流中发信号通知的语法,确定为节点启用树内量化;对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文。



CN 116325733 A

1.一种用于对包括点云数据的比特流进行解码的设备,所述设备包括:

存储器,存储所述点云数据;以及

一个或多个处理器,耦合到所述存储器并在电路中实现,所述一个或多个处理器被配置为:

基于在所述比特流中发信号通知的语法,确定为节点启用树内量化;

对于所述节点,基于在所述比特流中发信号通知的所述语法确定为所述节点激活角模式;

响应于为所述节点启用树内量化,为所述节点确定表示相对于原点位置的坐标位置的量化值;

在不剪裁的情况下缩放所述量化值,以确定表示相对于所述原点位置的所述坐标位置的经缩放值;以及

基于表示相对于所述原点位置的所述坐标位置的所述经缩放值,确定用于对所述角模式的平面位置语法元素进行上下文解码的上下文。

2.根据权利要求1所述的设备,其中,为了基于表示相对于所述原点位置的所述坐标位置的所述经缩放值来确定用于对所述角模式的所述平面位置语法元素进行上下文解码的所述上下文,所述一个或多个处理器被配置为:

基于所述经缩放值和所述原点位置确定一个或多个激光特性;以及

基于所述激光特性对所述角模式的所述平面位置语法元素进行解码。

3.根据权利要求2所述的设备,其中,为了基于表示相对于所述原点位置的所述坐标位置的所述经缩放值来确定用于对所述角模式的所述平面位置语法元素进行上下文解码的所述上下文,所述一个或多个处理器还被配置为基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在所述第一距离阈值与第二距离阈值之间、在所述第二距离阈值与第三距离阈值之间还是低于所述第三距离阈值来确定上下文索引。

4.根据权利要求2所述的设备,其中,所述一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

5.根据权利要求1所述的设备,其中,所述平面位置语法元素指示垂直面位置。

6.根据权利要求1所述的设备,其中,所述一个或多个处理器还被配置为使用由所确定的上下文指示的上下文对所述角模式的所述平面位置进行算术解码。

7.根据权利要求1所述的设备,其中,为了在不剪裁的情况下对所述量化值进行缩放以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值,所述一个或多个处理器被配置为:

确定最高有效位(MSB)群组和最低有效位(LSB)群组;

在不剪裁的情况下对所述LSB进行缩放以确定缩放后的LSB;以及

将所述缩放后的LSB添加到所述MSB中,以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值。

8.根据权利要求1所述的设备,其中,为了在不剪裁的情况下对所述量化值进行缩放以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值,所述一个或多个处理器被配置为:

基于所述节点的量化参数确定所述MSB的移位量;

基于所述移位量对所述MSB进行移位;以及

将所述缩放后的LSB添加到移位后的MSB中,以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值。

9.根据权利要求6所述的设备,其中,所述一个或多个处理器还被配置为在所述比特流中发信号通知的所述语法中接收所述LSB群组中的比特数量的指示。

10.根据权利要求1所述的设备,其中,所述一个或多个处理器还被配置为从所述点云数据重建点云。

11.根据权利要求10所述的设备,其中,所述一个或多个处理器被配置为作为重建所述点云的一部分,基于所述平面位置确定所述点云的一个或多个点的位置。

12.根据权利要求11所述的设备,其中,所述一个或多个处理器还被配置为基于所述点云生成建筑物内部的地图。

13.根据权利要求11所述的设备,其中,所述一个或多个处理器还被配置为基于所述点云执行自主导航操作。

14.根据权利要求11所述的设备,其中,所述一个或多个处理器还被配置为基于所述点云生成计算机图形。

15.根据权利要求11所述的设备,其中,所述一个或多个处理器被配置为:

基于所述点云确定虚拟对象的位置;以及

生成其中所述虚拟对象处于所确定的位置的扩展现实(XR)可视化。

16.根据权利要求11所述的设备,还包括用于基于所述点云呈现图像的显示器。

17.根据权利要求1所述的设备,其中,所述设备是移动电话或平板计算机。

18.根据权利要求1所述的设备,其中,所述设备是车辆。

19.根据权利要求1所述的设备,其中,所述设备是扩展现实设备。

20.一种用于对包括点云数据的比特流进行解码的方法,所述方法包括:

基于在所述比特流中发信号通知的语法,确定为节点启用树内量化;

对于所述节点,基于在所述比特流中发信号通知的所述语法确定为所述节点激活角模式;

响应于为所述节点启用树内量化,为所述节点确定表示相对于原点位置的坐标位置的量化值;

在不剪裁的情况下对所述量化值进行缩放,以确定表示相对于所述原点位置的所述坐标位置的经缩放值;以及

基于表示相对于所述原点位置的所述坐标位置的所述经缩放值,确定用于对所述角模式的平面位置语法元素进行上下文解码的上下文。

21.根据权利要求20所述的方法,其中,基于表示相对于所述原点位置的所述坐标位置的所述经缩放值来确定用于对所述角模式的所述平面位置语法元素进行上下文解码的所述上下文包括:

基于所述经缩放值和所述原点位置确定一个或多个激光特性;以及

基于所述激光特性对所述角模式的所述平面位置语法元素进行解码。

22.根据权利要求21所述的方法,其中,基于表示相对于所述原点位置的所述坐标位置的所述经缩放值来确定用于对所述角模式的所述平面位置语法元素进行上下文解码的所

述上下文还包括:基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在所述第一距离阈值与第二距离阈值之间、在所述第二距离阈值与第三距离阈值之间还是低于所述第三距离阈值来确定上下文索引。

23.根据权利要求21所述的方法,其中,所述一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

24.根据权利要求20所述的方法,其中,所述平面位置语法元素指示垂直面位置。

25.根据权利要求20所述的方法,还包括使用由所确定的上下文指示的上下文对所述角模式的所述平面位置进行算术解码。

26.根据权利要求20所述的方法,其中,在不剪裁的情况下对所述量化值进行缩放以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值包括:

确定最高有效位(MSB)群组和最低有效位(LSB)群组;

在不剪裁的情况下对所述LSB进行缩放以确定缩放后的LSB;以及

将所述缩放后的LSB添加到所述MSB中,以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值。

27.根据权利要求26所述的方法,其中,在不剪裁的情况下对所述量化值进行缩放以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值包括:

基于所述节点的量化参数确定所述MSB的移位量;

基于所述移位量对所述MSB进行移位;以及

将所述缩放后的LSB添加到移位后的MSB中,以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值。

28.根据权利要求25所述的方法,还包括在所述比特流中发信号通知的所述语法中接收所述LSB群组中的比特数量的指示。

29.根据权利要求20所述的方法,还包括从所述点云数据重建点云。

30.根据权利要求29所述的方法,其中,重建所述点云包括基于所述平面位置确定所述点云的一个或多个点的位置。

31.根据权利要求29所述的方法,还包括基于所述点云生成建筑物内部的地图。

32.根据权利要求29所述的方法,还包括基于所述点云执行自主导航操作。

33.根据权利要求29所述的方法,还包括基于所述点云生成计算机图形。

34.根据权利要求29所述的方法,还包括:

基于所述点云确定虚拟对象的位置;以及

生成其中所述虚拟对象处于所确定的位置的扩展现实(XR)可视化。

35.一种用于对包括点云数据的比特流进行编码的设备,所述设备包括:

存储器,存储所述点云数据;以及

一个或多个处理器,耦合到所述存储器并在电路中实现,所述一个或多个处理器被配置为:

确定为节点启用树内量化;

确定为所述节点激活角模式;

响应于为所述节点启用树内量化,为所述节点确定表示相对于原点位置的坐标位置的量化值;

在不剪裁的情况下缩放所述量化值,以确定表示相对于所述原点位置的所述坐标位置的经缩放值;

基于表示相对于所述原点位置的所述坐标位置的所述经缩放值,确定用于对所述角模式的平面位置语法元素进行上下文编码的上下文。

36.根据权利要求35所述的设备,其中,为了基于表示相对于所述原点位置的所述坐标位置的所述经缩放值来确定用于对所述角模式的所述平面位置语法元素进行上下文编码的所述上下文,所述一个或多个处理器被配置为:

基于所述经缩放值和所述原点位置确定一个或多个激光特性;以及

基于所述激光特性对所述角模式的所述平面位置语法元素进行解码。

37.根据权利要求36所述的设备,其中,为了基于表示相对于所述原点位置的所述坐标位置的所述经缩放值来确定用于对所述角模式的所述平面位置语法元素进行上下文编码的所述上下文,所述一个或多个处理器还被配置为基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在所述第一距离阈值与第二距离阈值之间、在所述第二距离阈值与第三距离阈值之间还是低于所述第三距离阈值来确定上下文索引。

38.根据权利要求36所述的设备,其中,所述一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

39.根据权利要求35所述的设备,其中,所述平面位置语法元素指示垂直面位置。

40.根据权利要求35所述的设备,其中,所述一个或多个处理器还被配置为使用由所确定的上下文指示的上下文对所述角模式的所述平面位置进行算术编码。

41.根据权利要求35所述的设备,其中,为了在不剪裁的情况下对所述量化值进行缩放以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值,所述一个或多个处理器被配置为:

确定最高有效位(MSB)群组和最低有效位(LSB)群组;

在不剪裁的情况下对所述LSB进行缩放以确定缩放后的LSB;以及

将所述缩放后的LSB添加到所述MSB中,以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值。

42.根据权利要求41所述的设备,其中,为了在不剪裁的情况下对所述量化值进行缩放以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值,所述一个或多个处理器被配置为:

基于所述节点的量化参数确定所述MSB的移位量;

基于所述移位量对所述MSB进行移位;以及

将所述缩放后的LSB添加到移位后的MSB中,以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值。

43.根据权利要求35所述的设备,其中,所述一个或多个处理器还被配置为从所述点云数据重建点云。

44.根据权利要求35所述的设备,其中,为了重建所述点云,所述一个或多个处理器还被配置为基于所述平面位置确定所述点云的一个或多个点的位置。

45.根据权利要求35所述的设备,其中,所述设备是移动电话或平板计算机。

46.根据权利要求35所述的设备,其中,所述设备是车辆。

47.根据权利要求35所述的设备,其中,所述设备是扩展现实设备。

48.一种用于对包括点云数据的比特流进行编码的方法,所述方法包括:

确定为节点启用树内量化;

确定为所述节点激活角模式;

响应于为所述节点启用树内量化,为所述节点确定表示相对于原点位置的坐标位置的量化值;

在不剪裁的情况下对所述量化值进行缩放,以确定表示相对于所述原点位置的所述坐标位置的经缩放值;以及

基于表示相对于所述原点位置的所述坐标位置的所述经缩放值,确定用于对所述角模式的平面位置语法元素进行上下文编码的上下文。

49.根据权利要求48所述的方法,其中,基于表示相对于所述原点位置的所述坐标位置的所述经缩放值来确定用于对所述角模式的所述平面位置语法元素进行上下文编码的所述上下文包括:

基于所述经缩放值和所述原点位置确定一个或多个激光特性;以及

基于所述激光特性对所述角模式的所述平面位置语法元素进行解码。

50.根据权利要求49所述的方法,其中,基于表示相对于所述原点位置的所述坐标位置的所述经缩放值来确定用于对所述角模式的所述平面位置语法元素进行上下文编码的所述上下文还包括:基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在所述第一距离阈值与第二距离阈值之间、在所述第二距离阈值与第三距离阈值之间还是低于所述第三距离阈值来确定上下文索引。

51.根据权利要求49所述的方法,其中,所述一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

52.根据权利要求48所述的方法,其中,所述平面位置语法元素指示垂直面位置。

53.根据权利要求48所述的方法,还包括使用由所确定的上下文指示的上下文对所述角模式的所述平面位置进行算术编码。

54.根据权利要求48所述的方法,其中,在不剪裁的情况下对所述量化值进行缩放,以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值包括:

确定最高有效位(MSB)群组和最低有效位(LSB)群组;

在不剪裁的情况下对所述LSB进行缩放以确定缩放后的LSB;以及

将所述缩放后的LSB添加到所述MSB中,以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值。

55.根据权利要求54所述的方法,在不剪裁的情况下对所述量化值进行缩放,以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值包括:

基于所述节点的量化参数确定所述MSB的移位量;

基于所述移位量对所述MSB进行移位;以及

将所述缩放后的LSB添加到移位后的MSB中,以确定表示相对于所述原点位置的所述坐标位置的所述经缩放值。

56.根据权利要求48所述的方法,还包括从所述点云数据重建点云。

57.根据权利要求56所述的方法,其中,重建所述点云包括基于所述平面位置确定所述

点云的一个或多个点的位置。

58.一种存储指令的计算机可读存储介质,所述指令在由一个或多个处理器执行时使所述一个或多个处理器:

基于在所述比特流中发信号通知的语法,确定为节点启用树内量化;

对于所述节点,基于在所述比特流中发信号通知的所述语法确定为所述节点激活角模式;

响应于为所述节点启用树内量化,为所述节点确定表示相对于原点位置的坐标位置的量化值;

在不剪裁的情况下缩放所述量化值,以确定表示相对于所述原点位置的所述坐标位置的经缩放值;以及

基于表示相对于所述原点位置的所述坐标位置的所述经缩放值,确定用于对所述角模式的平面位置语法元素进行上下文解码的上下文。

59.一种用于对包括点云数据的比特流进行解码的设备,所述设备包括:

用于基于在所述比特流中发信号通知的语法,确定为节点启用树内量化的部件;

用于对于所述节点,基于在所述比特流中发信号通知的所述语法确定为所述节点激活角模式的部件;

用于响应于对所述节点启用树内量化来为所述节点确定表示相对于原点位置的坐标位置的量化值的部件;

用于在不剪裁的情况下对所述量化值进行缩放,以确定表示相对于所述原点位置的所述坐标位置的经缩放值的部件;以及

用于基于表示相对于所述原点位置的所述坐标位置的所述经缩放值,确定用于对所述角模式的平面位置语法元素进行上下文解码的上下文的部件。

几何点云压缩中的角模式和树内量化

[0001] 相关申请的交叉引用

[0002] 本申请要求以下申请的优先权：

[0003] 2021年10月6日提交的申请号为17/495,621的美国专利申请；

[0004] 2020年10月7日提交的申请号为63/088,938的美国临时专利申请；

[0005] 2020年10月12日提交的申请号为63/090,629的美国临时专利申请;以及

[0006] 2020年10月14日提交的申请号为63/091,821的美国临时专利申请,所有申请的全部内容通过引用并入本文。

[0007] 2021年10月6日提交的申请号为17/495,621的美国专利申请要求以下申请的权益：

[0008] 2020年10月7日提交的申请号为63/088,938的美国临时专利申请；

[0009] 2020年10月12日提交的申请号为63/090,629的美国临时专利申请;以及

[0010] 2020年10月14日提交的申请号为63/091,821的美国临时专利申请。

技术领域

[0011] 本公开涉及点云编码和解码。

背景技术

[0012] 点云是三维空间中点的集合。这些点可以对应于三维空间内对象上的点。因此,点云可以用于表示三维空间的物理内容。点云在很多情况下都有用处。例如,点云可以在自主车辆的上下文中用于表示道路上的对象的位置。在另一示例中,点云可以用于表示环境的物理内容的上下文中,以便在增强现实(AR)或混合现实(MR)应用中定位虚拟对象。点云压缩是对点云进行编码和解码的过程。对点云进行编码可以减少存储和传输点云所需的数据量。

发明内容

[0013] 在点云帧的常规译码中,角模式为译码效率提供了相当大的增益。然而,当启用树内量化时,角模式获得的增益会显著降低,在某些情况下,甚至会产生损耗。对于角模式,经量化比特被用于上下文推导,并且经量化比特位于不同的尺度空间中,并且与原始点不在同一域中。这降低了角模式和树内量化两者的有用性,因此,同时启用两者可能是不利的。

[0014] 本公开描述了当角模式和树内量化一起使用时,用于从点/位置坐标值x导出经缩放值xS以导出节点/点相对于激光雷达原点的位置的技术。更具体地,通过在不剪裁的情况下对表示坐标值的经量化值进行缩放,G-PCC解码器可以以足够的精度将经缩放值放入与原始点值相同的缩放空间中以结合树内量化实现来自角模式的译码增益的方式,来确定表示相对于原点位置的坐标位置的经缩放值。

[0015] 根据一个示例,一种用于对包括点云数据的比特流进行解码的设备,该设备包括:存储器,用于存储点云数据;以及一个或多个处理器,耦合到存储器并实现在电路中,该一

个或多个处理器被配置为:基于在比特流中发信号通知的语法,确定为节点启用树内量化;对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文。

[0016] 根据另一示例,一种用于对包括点云数据的比特流进行解码的方法,该方法包括:基于在比特流中发信号通知的语法,确定为节点启用树内量化;对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文。

[0017] 根据另一示例,一种用于对包括点云数据的比特流进行编码的设备,该设备包括:存储器,用于存储点云数据;以及一个或多个处理器,耦合到存储器并实现在电路中,该一个或多个处理器被配置为:确定为节点启用树内量化;确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文编码的上下文。

[0018] 根据另一示例,一种用于对包括点云数据的比特流进行编码的方法,该方法包括:确定为节点启用树内量化;确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文编码的上下文。

[0019] 根据另一示例,一种存储指令的计算机可读存储介质,该指令在由一个或多个处理器执行时使得一个或多个处理器:基于在比特流中发信号通知的语法,确定为节点启用树内量化;对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文。

[0020] 根据另一示例,一种用于对包括点云数据的比特流进行解码的设备,该设备包括:用于基于在比特流中发信号通知的语法,确定为节点启用树内量化的部件;用于对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式的部件;用于响应于对节点启用树内量化来为节点确定表示相对于原点位置的坐标位置的量化值的部件;用于在不剪裁的情况下对量化值进行缩放,以确定表示相对于原点位置的坐标位置的经缩放值的部件;以及用于基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文的部件。

[0021] 在附图和以下描述中阐述一个或多个示例的细节。根据说明书、附图和权利要求

书,其他特征、目的和优势将显而易见。

附图说明

[0022] 图1是示出可以执行本公开的技术的示例编码和解码系统的框图。

[0023] 图2是示出示例几何点云压缩(G-PCC)编码器的框图。

[0024] 图3是示出示例G-PCC解码器的框图。

[0025] 图4是示出垂直方向上的示例平面占用的概念图。

[0026] 图5是根据本公开的一种或多种技术的示例的概念图,其中基于激光束位置在节点的标记点之上或之下来确定上下文索引。

[0027] 图6是示出示例性三上下文(three-context)索引确定的概念图。

[0028] 图7是示出用于利用由细虚线分开的间隔、基于激光束位置来对平面模式的垂直面位置进行译码的示例上下文索引确定的概念图。

[0029] 图8A是示出用于对垂直面位置进行编码的示例操作的流程图。

[0030] 图8B是示出用于对垂直面位置进行解码的示例操作的流程图。

[0031] 图9A是示出根据本公开的一种或多种技术的用于对垂直面位置进行译码的示例操作的流程图。

[0032] 图9B是示出根据本公开的一种或多种技术的用于对垂直面位置进行译码的示例操作的流程图。

[0033] 图10是示出可以与本公开的一种或多种技术一起使用的示例测距系统的概念图。

[0034] 图11是示出其中可以使用本公开的一种或多种技术的示例基于车辆的场景的概念图。

[0035] 图12是示出其中可以使用本公开的一种或多种技术的示例扩展现实系统的概念图。

[0036] 图13是示出其中可以使用本公开的一种或多种技术的示例移动设备系统的概念图。

具体实施方式

[0037] ISO/IEC MPEG(JTC 1/SC 29/WG 11)和最近的ISO/IEC MPEG 3DG(JTC 1/SC 29/WG 7)已经研究了点云译码技术的标准化,其压缩能力可能超过现有方法。以前被公知为MPEG的小组,现在已经解散并拆分成单独的工作组,正在共同努力进行被公知为三维图形小组(3DG)的这项探索活动,以评估该领域专家提出的压缩技术设计。

[0038] 点云压缩活动通常分为两种不同的方法。第一种方法是“视频点云压缩”(V-PCC),它涉及对3D对象进行分段,并将分段投影在多个2D平面(在2D帧中表示为“分块(patch)”)中,这些平面进一步由传统二维视频编解码器(诸如HEVC)进行译码。第二种方法是“几何点云压缩”(G-PCC),它涉及直接压缩三维几何(即点的集合在三维空间中的位置)以及相关联的属性值(对于与三维几何相关联的每个点)。G-PCC解决了第一类点云(类别1)(静态点云)和第三类(类别3)点云(动态获取点云)的压缩问题。

[0039] 点云包含三维空间中的点的集合,并且可以具有与这些点相关联的属性。属性可以例如是诸如R/G/B、Y/Cb/Cr的颜色信息、反射率信息或其他属性。点云可以由各种相机或

传感器(诸如光探测和测距(LIDAR)扫描仪或3D扫描仪)来捕获,并且也可以是计算机生成的。点云数据可以被用于各种应用,包括但不限于建筑(建模)、图形(例如,用于可视化和动画的3D模型)以及汽车工业(例如,用于帮助导航的LIDAR传感器)。

[0040] 点云数据所占用的3D空间可以由虚拟边界框封闭。边界框中的点的位置可以用一定的精度来表示。因此,可以基于精度来量化一个或多个点的位置。在最小的级别上,边界框被分成体素(voxel),体素是由单位立方体表示的最小空间单位。边界框中的体素可以与零个、一个或多个点相关联。边界框可以被拆分成多个立方体/长方体区域,这些区域可以被称为图块(tile),并且每个图块可以被译码为一个或多个切片(slice)。将边界框拆分为切片和图块可以基于每个部分中的点的数量,或者基于其他考虑(诸如将特定区域译码为图块)。可以使用类似于视频编解码器中的拆分决策来进一步划分切片区域。

[0041] G-PCC编码器和解码器可以支持平面译码模式和角译码模式,它们也可以分别被称为平面模式和角模式。平面模式是一种可以改进哪些节点被占用的译码的技术。当节点的所有被占用的子节点邻近一平面并且在该平面的与正交于该平面的维度的增加的坐标值相关联的一侧上时,可以使用平面模式。例如,当节点的所有被占用的子节点位于穿过该节点的中心点的水平面之上或之下时,可以对该节点使用平面模式,或者当节点的所有被占用的子节点位于穿过该节点的中心点的垂直面的近侧或远侧时,可以对该节点使用平面模式。对于节点,G-PCC编码器可以对x、y和z维度中的每一个的语法元素进行编码,以指定该维度是否用平面模式进行译码,并且对于用平面模式进行译码的每个维度,平面位置语法元素(即,指示平面位置的语法元素)可以针对相应的维度被发信号通知。维度的平面位置语法元素指示与该维度正交的平面是在第一位置还是在第二位置。如果平面位于第一位置,则该平面对应于节点的边界。如果平面位于第二位置,则该平面穿过节点的3D中心。更一般地,如果平面在第一位置处,则节点中的点在节点的第一位置的那一侧,而不在第二位置的那一侧,并且如果平面在第二位置处,则节点中的点在节点的第二位置的那一侧,而不在该位置的那一侧。因此,对于z维度,G-PCC译码器可以对八叉树的节点中的平面模式的垂直面位置进行译码,该八叉树表示点云中的点的三维位置。

[0042] 点云通常可以使用LIDAR传感器或其他基于激光的传感器来捕获。角译码模式可选地与平面模式一起使用,并且通过采用典型LIDAR传感器中感测激光束的位置和仰角的知识来改进垂直(例如,z)平面位置语法元素的译码。此外,角译码模式可以可选地用于改进推断直接译码模式(IDCM)中垂直z位置比特的译码。

[0043] G-PCC译码器也可以支持树内量化。树内几何缩放提供了一种量化(编码器)和缩放(解码器)几何位置的手段,即使在构建编码树时也是如此。点云中的每个点位于特定的几何位置。在八叉树译码中,关于位置的信息可以不直接发信号通知,而是可以使用跨层次结构(从根节点到叶节点)的八叉树占用来指示点的几何位置。在编码器处,点,或者更确切地说,点的占用位置,被放置在八叉树的叶节点中。八叉树的尺寸取决于每个维度中位置的位深度。从根节点开始,对八个八位数中的每一个的占用进行译码(以不同的方式)。根节点上的占用有效地对三维中点位置的最高有效位(MSB)进行译码。这个过程一直持续到指示点的位置的叶节点。解码器遵循类似的过程来确定八叉树节点在每个级别的占用,直到叶节点,以确定点的位置。

[0044] 几何量化被应用于八叉树中的特定节点深度,该节点深度在比特流中被发信号通

知。节点深度通常是指八叉树解析中的特定级别。在只考虑八叉树(不考虑QTBT)的简单情况下,假设八叉树有12个级别。在根节点处,每个子节点的尺寸为$2^{11}\times2^{11}\times2^{11}$。这些节点中的每一个都可以被认为处于节点深度1。根节点的每个子节点的子节点的尺寸为$2^{10}\times2^{10}\times2^{10}$,并且这些被认为是节点深度为2的节点,依此类推。在一个简单的示例中,如果节点坐标是12比特并且要应用量化的深度是3,则节点坐标的前3个MSB(称为位置的MSB部分)不被量化;仅量化节点坐标的最后9个LSB(称为位置的LSB部分)。由于量化,9个LSB可以减少到更少的比特数,诸如N比特,其中N小于或等于9。这可能导致以重建精度为代价的比特率的一些降低。结果节点坐标尺寸变为N+3(即<=12)。同样,在解码器处,N个LSB被缩放并剪裁至1<<(9-1)的最大值,这确保经缩放值不超过原始点的9个LSB比特。通过连接3个MSB和经缩放LSB的9个比特来计算最终缩放位置。

[0045] 在点云帧的常规译码中,角模式为译码效率提供了相当大的增益。然而,当启用树内量化时,角模式的增益会显著降低,在某些情况下还会产生损耗。对于角模式(IDCM角和平面角),经量化比特被用于上下文推导,并且经量化比特位于不同的尺度空间中,并且与原始点不在同一域中。这降低了角模式和树内量化两者的有用性,因此,同时启用两者可能是不利的。

[0046] 本公开描述了当角模式和树内量化一起使用时,用于从点/位置坐标值x导出经缩放值xS以导出节点/点相对于激光雷达原点的位置的技术。更具体地,通过在不剪裁的情况下对表示坐标值的经量化值进行缩放,G-PCC解码器可以以足够的精度将经缩放值放入与原始点值相同的缩放空间中以实现来自角模式的译码增益的方式,来确定表示相对于原点位置的坐标位置的经缩放值。

[0047] 本公开使用术语G-PCC译码器来一般地指代G-PCC编码器和/或G-PCC解码器。此外,本公开中描述的关于解码的某些技术也可以应用于编码,反之亦然。例如,G-PCC编码器和G-PCC解码器经常被配置为执行相同的过程或相互的过程。而且,G-PCC编码器通常执行解码作为确定如何编码的过程的一部分。

[0048] 图1是示出可执行本公开的技术的示例编码和解码系统100的框图。本公开的技术通常涉及对点云数据进行译码(编码和/或解码),即支持点云压缩。一般地,点云数据包括用于处理点云的任何数据。该译码在对点云数据进行压缩和/或解压缩方面是有效的。

[0049] 如图1所示,系统100包括源设备102和目的地设备116。源设备102提供要由目的地设备116解码的经编码点云数据。特别地,在图1的示例中,源设备102经由计算机可读介质110向目的地设备116提供点云数据。源设备102和目的地设备116可以包括多种设备中的任何一种,包括台式计算机、笔记本(即膝上型)计算机、平板计算机、机顶盒、手持电话(诸如智能手机)、电视、照相机、显示设备、数字媒体播放器、视频游戏控制台、视频流式传输设备、陆地或海上运载工具、航天器、飞机、机器人、LIDAR设备、卫星等。在一些情况下,源设备102和目的地设备116可以被配备用于无线通信。

[0050] 在图1的示例中,源设备102包括数据源104、存储器106、G-PCC编码器200和输出接口108。目的地设备116包括输入接口122、G-PCC解码器300、存储器120和数据消费者118。根据本公开,源设备102的G-PCC编码器200和目的地设备116的G-PCC解码器300可以被配置为应用本公开的与G-PCC中的角模式和树内量化有关的技术。

[0051] 因而,源设备102表示编码设备的示例,而目的地设备116表示解码设备的示例。在

其他示例中，源设备102和目的地设备116可以包括其他组件或布置。例如，源设备102可以从内部或外部源接收数据(例如，点云数据)。同样，目的地设备116可以与外部数据消费者接口连接，而不是在同一设备中包括数据消费者。

[0052] 如图1所示的系统100仅是一个示例。一般地，其他数字编码和/或解码设备可以执行本公开的与G-PCC中的角模式和树内量化有关的技术。源设备102和目的地设备116仅仅是此类设备的示例，其中，源设备102生成用于传输到目的地设备116的经译码数据。本公开将“译码”设备表示为执行数据的译码(编码和/或解码)的设备。从而，G-PCC编码器200和G-PCC解码器300表示译码设备的示例，具体地，分别是编码器和解码器。在一些示例中，源设备102和目的地设备116可以以基本对称的方式操作，使得源设备102和目的地设备116中的每一个都包括编码和解码组件。因此，系统100可以支持源设备102与目的地设备116之间的单向或双向传输，例如用于流式传输、重放、广播、电话、导航和其他应用。

[0053] 一般地，数据源104表示数据(即，原始的、未编码的点云数据)的源，并且可以向G-PCC编码器200提供数据的“帧”的顺序系列，该编码器对帧的数据进行编码。源设备102的数据源104可以包括点云捕获设备，诸如各种相机或传感器中的任一个，例如3D扫描仪或光检测和测距(LIDAR)设备、一个或多个视频相机、包含先前捕获的数据的存档，和/或用于从数据内容提供商接收数据的数据馈送接口。可替代地或附加地，点云数据可以由计算机从扫描仪、相机、传感器或其他数据来生成。例如，数据源104可以生成基于计算机图形的数据作为源数据，或者产生实时数据、存档数据和计算机生成数据的组合。在每种情况下，G-PCC编码器200对捕获的、预捕获的或计算机生成的数据进行编码。G-PCC编码器200可以将帧从接收顺序(有时称为“显示顺序”)重新排列为用于译码的译码顺序。G-PCC编码器200可以生成包括经编码数据的一个或多个比特流。然后，源设备102可以经由输出接口108将经编码的数据输出到计算机可读介质110上，以供例如目的地设备116的输入接口122进行接收和/或取得。

[0054] 源设备102的存储器106和目的地设备116的存储器120可以表示通用存储器。在一些示例中，存储器106和存储器120可以存储原始数据，例如，来自数据源104的原始数据和来自G-PCC解码器300的原始的经解码的数据。附加地或可替代地，存储器106和存储器120可以分别存储可由例如G-PCC编码器200和G-PCC解码器300执行的软件指令。尽管在此示例中存储器106和存储器120与G-PCC编码器200和G-PCC解码器300分开示出，但是应当理解的是，G-PCC编码器200和G-PCC解码器300还可以包括实现功能上相似或等效目的的内部存储器。此外，存储器106和存储器120可以存储例如从G-PCC编码器200输出并输入到G-PCC解码器300的经编码的数据。在一些示例中，存储器106和存储器120的一些部分可以分配为一个或多个缓冲区，例如用来存储原始的、经解码的和/或经编码的数据。例如，存储器106和存储器120可以存储表示点云的数据。

[0055] 计算机可读介质110可以表示能够将经编码的数据从源设备102传输到目的地设备116的任何类型的介质或设备。在一个示例中，计算机可读介质110表示通信介质以使源设备102能够例如经由射频网络或基于计算机的网络将编码数据实时地直接传输到目标设备116。根据诸如无线通信协议的通信标准，输出接口108可以对包括经编码的数据的传输信号进行调制，并且输入接口122可以对接收到的传输信号进行解调。通信介质可以包括任何无线或有线通信介质，诸如射频(RF)频谱或者一条或多条物理传输线。通信介质可以形

成基于分组的网络的一部分，诸如局域网、广域网或诸如因特网的全球网络。通信介质可以包括路由器、交换机、基站或有助于从源设备102到目标设备116的通信的任何其他装备。

[0056] 在一些示例中，源设备102可以将经编码数据从输出接口108输出到存储设备112。类似地，目标设备116可以经由输入接口122访问来自存储设备112的编码的数据。存储设备112可以包括各种分布式或本地访问的数据存储介质中的任何一种，诸如硬盘、蓝光光盘、DVD、CD-ROM、闪存、易失性或非易失性存储器，或者用于存储编码的数据的任何其他合适的数字存储介质。

[0057] 在一些示例中，源设备102可以向文件服务器114或可存储由源设备102生成的经编码的数据的另一中间存储设备输出编码的数据。目的地设备116可以经由流式传输或下载来访问来自文件服务器114的存储的数据。文件服务器114可以是能够存储经编码的数据并将经编码的数据发送到目的地设备116的任何类型的服务器设备。文件服务器114可以表示(例如用于网站的)web服务器、文件传输协议(FTP)服务器、内容递送网络设备或网络附加存储(NAS)设备。目的地设备116可以通过包括因特网连接的任何标准数据连接来访问来自文件服务器114的经编码的数据。这可以包括无线信道(例如Wi-Fi连接)、有线连接(例如数字订户线路(DSL)、电缆调制解调器等)或者适合访问存储在文件服务器114上的经编码的数据的二者组合。文件服务器114和输入接口122可以配置为根据流式传输协议、下载传输协议或其组合来操作。

[0058] 输出接口108和输入接口122可以表示无线发射器/接收器、调制解调器、有线联网组件(例如以太网卡)、根据各种IEEE 802.11标准中的任何一种进行操作的无线通信组件，或者其他物理组件。在输出接口108和输入接口122包括无线组件的示例中，输出接口108和输入接口122可以配置为根据诸如4G、4G-LTE(长期演进)、LTE高级、5G或类似标准的蜂窝通信标准来传输诸如经编码数据的数据。在输出接口108包括无线发射器的某些示例中，输出接口108和输入接口122可以配置为根据其他无线标准，诸如IEEE 802.11规范、IEEE 802.15规范(例如ZigBee™)、蓝牙™标准等来传输诸如经编码数据的数据。在一些示例中，源设备102和/或目标设备116可以包括各自的片上系统(SoC)设备。例如，源设备102可以包括SoC设备来执行归于G-PCC编码器200和/或输出接口108的功能，并且目的地设备116可以包括SoC设备来执行归于G-PCC解码器300和/或输入接口122的功能。

[0059] 本公开的技术可以应用于支持各种应用中的任何应用的编码和解码，诸如自主车辆之间的通信，扫描仪、相机、传感器和诸如本地或远程服务器的处理设备之间的通信，地理测绘，或其他应用。

[0060] 目的地设备116的输入接口122从计算机可读介质110(例如通信介质、存储设备112、文件服务器114等)接收经编码比特流。经编码的比特流可以包括由G-PCC编码器200定义的、也由G-PCC解码器300使用的信令信息，诸如语法元素，该语法元素具有描述译码单元(例如切片、图片、图片组、序列等)的特性和/或处理的值。数据消费者118使用经解码的数据。例如，数据消费者118可以使用经解码的数据来确定物理对象的位置。在一些示例中，数据消费者118可以包括基于点云来呈现图像的显示器。

[0061] G-PCC编码器200和G-PCC解码器300可以各自实现为各种合适的编码器和/或解码器电路中的任何一种，诸如一个或多个微处理器、数字信号处理器(DSP)、专用集成电路(ASIC)、现场可编程门阵列(FPGA)、离散逻辑、软件、硬件、固件或其任何组合。当该技术部

分地在软件中实施时，设备可以将用于软件的指令存储在合适的非暂时计算机可读介质中，并使用一个或多个处理器在硬件中执行该指令来执行本公开的技术。G-PCC编码器200和G-PCC解码器300中的每一个可以被包括在一个或多个编码器或解码器中，这两者任一个都可以集成为各自设备中组合编码器/解码器(CODEC)的一部分。包括G-PCC编码器200和/或G-PCC解码器300的设备可以包括一个或多个集成电路、微处理器和/或其他类型的设备。

[0062] G-PCC编码器200和G-PCC解码器300可以根据译码标准操作(几何点云压缩(G-PCC)标准)。尽管编码器200和解码器300被描述为G-PCC编码器200和G-PCC解码器300，但编码器200和解码器300不应被认为限于根据G-PCC标准操作。在一些示例中，编码器200和解码器300可以根据视频点云压缩(V-PCC)标准来操作。本公开通常涉及图片的译码(例如编码和解码)，以包括对数据进行编码或解码的过程。经编码的比特流一般包括对于表示译码决策(例如译码模式)的语法元素的一系列值。

[0063] 一般地，本公开可以涉及“信令通知”某些信息，诸如语法元素。术语“信令通知”通常可以指对于语法元素的值和/或用于对经编码的数据进行解码的其他数据的通信。也就是说，G-PCC编码器200可以信令通知比特流中的语法元素的值。一般地，信令通知是指在比特流中生成值。如上所述，源设备102可以基本上实时地(或非实时地，诸如可能在将语法元素存储到存储设备112以供稍后由目标设备116检索时发生)将比特流传送到目标设备116。

[0064] 如上所述，ISO/IEC MPEG(JTC 1/SC 29/WG 11)正在研究压缩能力大大超过当前方法的点云译码技术的标准化的潜在需求，并将致力于创建标准。该小组正在合作开展这项探索活动，称为三维图形小组(3DG)，以评估该领域专家提出的压缩技术设计。

[0065] 点云压缩活动分为两种不同的方法。第一种方法是“视频点云压缩”(V-PCC)，它对3D对象进行分段，并将分段投影在多个2D平面(在2D帧中表示为“分块(patch)”)中，这些平面进一步由传统二维视频编解码器(诸如高效视频编码(HEVC)(ITU-T H.265)编解码器)进行译码。第二种方法是“基于几何的点云压缩”(G-PCC)，它直接压缩三维几何(即点的集合在三维空间中的位置)以及相关联的属性值(对于与三维几何相关联的每个点)。G-PCC解决了第一类点云(类别1)(静态点云)和第三类(类别3)点云(动态获取点云)的压缩问题。G-PCC标准的最近草案可以在2020年1月，比利时布鲁塞尔，G-PCC DIS，ISO/IEC JTC1/SC29/WG11 w19088中获得，并且编解码器的描述可以在2020年1月，比利时布鲁塞尔，G-PCC Codec Description v6，ISO/IEC JTC1/SC29/WG11 w19091中获得。

[0066] 点云包含三维空间中的点的集合，并且可以具有与这些点相关联的属性。属性可以是诸如R、G、B或Y、Cb、Cr的颜色信息、或反射率信息或其他属性。点云可以由各种相机或传感器(诸如LIDAR传感器和3D扫描仪)来捕获，并且也可以是计算机生成的。点云数据被用于各种应用，包括但不限于建筑(建模)、图形(用于可视化和动画的3D模型)以及汽车工业(用于帮助导航的LIDAR传感器)。

[0067] 点云数据所占用的3D空间可以由虚拟边界框封闭。边界框中的点的位置可以用一定的精度表示；因此，可以基于该精度来量化一个或多个点的位置。在最小的级别上，边界框被分成体素，体素是由单位立方体表示的最小空间单位。边界框中的体素可以与零个、一个或多个点相关联。边界框可以被拆分成多个立方体/长方体区域，这些区域可以被称为图块。每个图块可以被译码成一个或多个切片。将边界框拆分为切片和图块可以基于每个部分中的点的数量，或者基于其他考虑(例如，特定区域可以被译码为图块)。可以使用类似于

视频编解码器中的拆分决策来进一步划分切片区域。

[0068] 图2提供G-PCC编码器200的概述。图3提供G-PCC解码器300的概述。所示的模块是逻辑的，并不一定与G-PCC编解码器的参考实现中的实现代码一一对应，即ISO/IEC MPEG (JTC 1/SC 29/WG11)研究的TMC13测试模型软件。

[0069] 在G-PCC编码器200和G-PCC解码器300两者中，首先对点云位置进行译码。属性译码依赖于解码的几何图形。在图2和图3中，表面近似分析单元212、表面近似合成单元310和RAHT单元218和314表示通常用于类别1数据的选项，而LOD生成单元220和316、提升单元222和逆提升单元318表示通常用于类别3数据的选项。所有其他单元可以在类别1和类别3之间是通用的。

[0070] 对于类别3数据，经压缩几何通常表示为八叉树，从根一直向下到各个体素的叶级别。对于类别1数据，经压缩几何通常由修剪(prune)的八叉树(即，从根向下到大于体素的块(block)的叶级别的八叉树)加上近似修剪的八叉树的每个叶内的表面的模型来表示。这样，类别1和类别3的数据都共享八叉树译码机制，而类别1的数据还可以用表面模型来近似每个叶内的体素。所使用的表面模型是三角划分(triangulation)，其每个块包含1-10个三角形，得到三角形集合(triangle soup)。因此，类别1几何编解码器被称为三角形集合几何编解码器，而类别3几何编解码器被称为八叉树几何编解码器。

[0071] 在八叉树的每个节点处，它的一个或多个子节点(最多八个节点)的占用(occupancy)被发信令通知(如果没有推断)。指定多个邻域，其包括(a)与当前八叉树节点共享面的节点，(b)与当前八叉树节点共享面、边或顶点的节点等。在每个邻域内，节点和/或其子节点的占用可以被用于预测当前节点或其子节点的占用。对于稀疏地填充在八叉树的某些节点中的点，编解码器还支持直接译码模式，其中点的3D位置被直接编码。一个标志(flag)可以被发信令通知以指示直接模式被发信令通知。在最低级别处，还可以对与八叉树节点/叶节点相关联的点的数量进行译码。

[0072] 一旦几何被译码，对应于几何点的属性也被译码。当有多个属性点对应于一个经重建/解码的几何点时，可以导出代表该重建点的属性值。

[0073] G-PCC中有三种属性译码方法:区域自适应分层变换(RAHT)译码，基于内插的分层最近邻居预测(预测变换)，以及具有更新/提升步骤的基于内插的分层最近邻居预测(提升变换)。RAHT和提升变换(Lifting)通常用于类别1数据，而预测变换(Predicting)通常用于类别3数据。然而，任何一种方法都可以用于任何数据，并且，就像G-PCC中的几何编解码器，用于对点云进行译码的属性译码方法在比特流中指定。

[0074] 属性的译码可以在细节级别(LOD)中进行，其中对于每一个细节级别，可以获得点云属性的更精细的表示。每个细节级别可以基于与相邻节点的距离度量或基于采样距离来指定。

[0075] 在G-PCC编码器200处，作为用于属性的译码方法的输出而获得的残差被量化。可以使用上下文自适应算术译码来对经量化残差进行译码。

[0076] 在图2的示例中，G-PCC编码器200可以包括坐标变换单元202、颜色变换单元204、体素化单元206、属性传递单元208、八叉树分析单元210、表面近似分析单元212、算术编码单元214、几何重建单元216、RAHT单元218、LOD生成单元220、提升单元222、系数量化单元224和算术编码单元226。

[0077] 如图2的示例所示,G-PCC编码器200可以接收点位置集和属性集。位置可以包括点云中的点的坐标。属性可以包括关于点云中的点的信息,诸如与点云中的点相关联的颜色。

[0078] 坐标变换单元202可以对点的坐标应用变换,以将坐标从初始域变换到变换域。本公开可以将经变换的坐标称为变换坐标。颜色变换单元204可以应用变换以便将属性的颜色信息变换到不同的域。例如,颜色变换单元204可以将颜色信息从RGB颜色空间变换到YCbCr颜色空间。

[0079] 此外,在图2的示例中,体素化单元206可以对变换坐标进行体素化。变换坐标的体素化可以包括量化和移除点云的一些点。换句话说,点云的多个点可以被归入(subsum)到单个“体素”内,它们此后在某些方面可以被视为一个点。此外,八叉树分析单元210可以基于体素化变换坐标来生成八叉树。附加地,在图2的示例中,表面近似分析单元212可以分析点以潜在地确定点的集合的表面表示。算术编码单元214可以对语法元素进行熵编码,这些语法元素表示八叉树和/或由表面近似分析单元212确定的表面的信息。G-PCC编码器200可以在几何比特流中输出这些语法元素。

[0080] 几何重建单元216可以基于八叉树、表示由表面近似分析单元212确定的表面的数据和/或其他信息来重建点云中的点的变换坐标。由于体素化和表面近似,由几何重建单元216重建的变换坐标的数量可以不同于点云的原始点数。本公开可以将得到的点称为重建点。属性传递单元208可以将点云的原始点的属性传递到点云的重建点。

[0081] 此外,RAHT单元218可以对重建点的属性应用RAHT译码。可替代地或附加地,LOD生成单元220和提升单元222可以分别对重建点的属性应用LOD处理和提升。RAHT单元218和提升单元222可以基于属性生成系数。系数量化单元224可以量化由RAHT单元218或提升单元222生成的系数。算术编码单元226可以对表示经量化系数的语法元素应用算术译码。G-PCC编码器200可以在属性比特流中输出这些语法元素。

[0082] 在图3的示例中,G-PCC解码器300可以包括几何算术解码单元302、属性算术解码单元304、八叉树合成单元306、逆量化单元308、表面近似合成单元310、几何重建单元312、RAHT单元314、LoD生成单元316、逆提升单元318、逆变换坐标单元320和逆变换颜色单元322。

[0083] G-PCC解码器300可以获得几何比特流和属性比特流。G-PCC解码器300的几何算术解码单元302可以对几何比特流中的语法元素应用算术解码(例如,上下文自适应二进制算术译码(CABAC)或其他类型的算术解码)。类似地,属性算术解码单元304可以对属性比特流中的语法元素应用算术解码。

[0084] 八叉树合成单元306可以基于从几何比特流解析出的语法元素来合成八叉树。在几何比特流中使用表面近似的情况下,表面近似合成单元310可以基于从几何比特流解析出的语法元素并且基于八叉树来确定表面模型。

[0085] 此外,几何重建单元312可以执行重建以确定点云中的点的坐标。逆变换坐标单元320可以对重建坐标应用逆变换,以将点云中的点的重建坐标(位置)从变换域转换回初始域。

[0086] 附加地,在图3的示例中,逆量化单元308可以对属性值进行逆量化。属性值可以基于从属性比特流获得的语法元素(例如,包括由属性算术解码单元304解码的语法元素)。

[0087] 取决于如何对属性值进行编码,RAHT单元314可以执行RAHT译码以基于逆量化属

性值来确定点云的点的颜色值。可替代地,LOD生成单元316和逆提升单元318可以使用基于细节级别的技术来确定点云的点的颜色值。

[0088] 此外,在图3的示例中,逆变换颜色单元322可以对颜色值应用逆颜色变换。逆颜色变换可以是由G-PCC编码器200的颜色变换单元204应用的颜色变换的逆。例如,颜色变换单元204可以将颜色信息从RGB颜色空间变换到YCbCr颜色空间。相应的,逆颜色变换单元322可以将颜色信息从YCbCr颜色空间变换到RGB颜色空间。

[0089] 示出图2和图3的各种单元以帮助理解由G-PCC编码器200和G-PCC解码器300执行的操作。这些单元可以实现为固定功能电路、可编程电路或其组合。固定功能电路是指提供特定功能并预设在能够执行的操作上的电路。可编程电路是指可以被编程以执行各种任务并且在可以执行的操作中提供灵活功能的电路。例如,可编程电路可以执行软件或固件,软件或固件使可编程电路以由软件或固件的指令所定义的方式操作。固定功能电路可以执行软件指令(例如来接收参数或输出参数),但是固定功能电路执行的操作类型通常是不可变的。在一些示例中,一个或多个单元可以是不同的电路块(固定功能或可编程),并且在一些示例中,一个或多个单元可以是集成电路。

[0090] G-PCC编码器200和G-PCC解码器300可以被配置为使用平面、角和方位角译码模式对点云数据进行译码。在瑞士日内瓦举行的第128次MPEG会议上采用了平面译码模式。平面译码模式可以在每个节点位置处应用于三维x、y和z中的每一个。当针对节点处的特定维度(例如z)指定平面译码模式时,平面译码模式指示节点的所有子节点仅占用z半平面中的一个,如下面关于图4所描述的。类似的图示(例如,技术)应用于x和y维的平面模式。

[0091] 在比利时布鲁塞尔举行的第129次MPEG会议上采用了角译码模式。以下描述基于原始MPEG贡献文档:2019年10月,瑞士日内瓦,ISO/IEC JTC1/SC29/WG11 MPEG/M50642,Sébastien Lasserre、Jonathan Taquet的“[GPCC][CE 13.22related]An improvement of the planar coding mode”;以及w19088。角译码模式可选地与平面模式一起使用(例如,如2019年7月,瑞典哥德堡,ISO/IEC JTC1/SC29/WG11 MPEG/M48906,Sébastien Lasserre、David Flynn的“[GPCC]Planar mode in octree-based geometry coding”),并通过使用典型LIDAR传感器中感测激光束的位置和角度的知识来改进垂直(z)平面位置语法元素的译码(参见例如,2020年1月,比利时布鲁塞尔,ISO/IEC JTC1/SC29/WG11 MPEG/M51594,Sébastien Lasserre、Jonathan Taquet的“[GPCC]CE 13.22report on angular mode”)。

[0092] 在第130次MPEG电话会议上采用了方位角译码模式。方位角译码模式类似于角模式,并且将角模式扩展到平面模式的(x)和(y)平面位置语法元素的编解,并且改进了IDCM中x或y位置比特的译码。方位角模式使用每个激光器的方位角的采样信息(例如,激光器在一次旋转中可以获取的点的数量)。在本公开中,术语“角模式”也可以指如下所述的方位角模式。

[0093] 图4是示出垂直方向上的示例平面占用的概念图。在图4的示例中,节点400被划分为八个子节点。子节点402A-402H可以被占用或未被占用。在图4的示例中,被占用的子节点被阴影化。当一个或多个子节点402A-402D被占用并且没有子节点402E-402H被占用时,G-PCC编码器200可以用0的值发信令通知平面位置语法元素,以指示所有被占用的子节点相邻于节点400的最小z坐标平面的正侧(即,增加z坐标的一侧)。当一个或多个子节点402E-402H被占用并且没有子节点402A-402D被占用时,G-PCC编码器200可以用1的值发信令通知

平面位置语法元素,以指示所有被占用的子节点相邻于节点400的中点z坐标平面的正侧。以此方式,平面位置语法元素可以指示节点400中的平面模式的垂直面位置。

[0094] 角译码模式还可以可选地用于改进推断IDCM中垂直z位置比特的译码(2020年1月,比利时布鲁塞尔,ISO/IEC JTC1/SC29/WG11 MPEG/M51594,Sébastien Lasserre、Jonathan Taquet的“[GPCC]CE 13.22report on angular mode”)。IDCM是一种其中节点内的点的位置相对于节点内的点显式地(例如直接地)发信令通知的模式。在角译码模式中,点的位置可以相对于节点的原点发信号通知,并且例如使用激光特性的捕获点的关系用于有效地压缩位置。

[0095] 当基于诸如LIDAR系统的测距系统生成的数据来生成点云时,还可以使用角译码模式。LIDAR系统可以包括相对于原点以不同的角度排列在垂直面中的激光器集合。LIDAR系统可以围绕垂直轴旋转。LIDAR系统可以使用返回的激光来确定点云中的点的距离和位置。LIDAR系统的激光器发出的激光束可以用参数集来特征化。

[0096] 在以下描述中,激光、激光束、激光传感器或传感器或其他类似术语可以表示能返回距离测量和空间方位(包括潜在的时间指示)的任何传感器,例如典型的LIDAR传感器。

[0097] G-PCC编码器200或G-PCC解码器300可以通过从在诸如几何参数集的参数集中被发信令通知的激光候选集中选择激光索引来对节点中的平面模式的垂直面位置进行译码(例如,分别编码或解码),所选择的激光索引指示与节点相交或最靠近节点中心的激光束。激光束与节点的相交或接近确定用于对平面模式的垂直面位置进行算术译码的上下文索引(例如,下面的contextAngle、contextAnglePhiX或contextAnglePhiY)。

[0098] 因此,在一些示例中,G-PCC译码器(例如,G-PCC编码器200或G-PCC解码器300)可以对表示点云中的点的三维位置的八叉树的节点中的平面模式的垂直面位置进行译码。为了便于解释,本公开可以将G-PCC译码器正在进行译码的节点称为当前节点。作为对平面模式的垂直面位置进行译码的一部分,G-PCC译码器可以确定激光候选集中的激光候选的激光索引。所确定的激光索引指示最靠近当前节点或与当前节点相交的激光束。激光器候选集可以包括LIDAR阵列中的每一个激光器。在一些示例中,激光候选集可以在诸如几何参数集的参数集中指示。附加地,作为对垂直面位置进行译码的一部分,G-PCC译码器基于激光束与当前节点的相交或接近度来确定上下文索引。例如,G-PCC译码器可以基于激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间、还是低于第三距离阈值来确定上下文索引。此外,作为对垂直面定位进行译码的一部分,G-PCC译码器使用由所确定的上下文索引指示的上下文来对平面模式的垂直面位置进行算术译码。

[0099] 可以存在适格性条件来确定当前节点中的平面模式的垂直面位置是否具有使用角模式来进行译码的适格性。如果垂直面位置不具有使用角模式进行译码的适格性,则可以在不使用传感器信息的情况下对平面模式的垂直面位置进行译码。在一些示例中,适格性条件可以确定是否只有一个激光束与当前节点相交。换句话说,如果只有一束激光束(即,不是两个或更多个激光束)与当前节点相交,则当前节点的垂直面位置可以具有使用角模式来进行译码的适格性。在一些示例中,适格性条件可以确定候选激光集合当中的激光之间的最小角度差。换句话说,如果包络住当前节点的角度小于激光束之间的最小角度,则当前节点可以具有使用角模式来进行译码的适格性。包络住当前节点的角度是从激光原

点测量的穿过节点的远底角的线与穿过节点的近顶角的线之间的角度。当包络住当前节点的角度小于激光束之间的最小角度差时，最多一束激光束与该节点相交。在一些示例中，适格性条件是使得垂直节点维度小于(或等于)最小角差。换句话说，如果节点的垂直维度小于或等于在当前节点的最接近激光原点的垂直边处由最小角度差所分开的激光束之间的垂直距离，则当前节点可以具有使用角模式进行译码的适格性。

[0100] 如上所述，G-PCC译码器可以选择与当前节点相交或者最接近当前节点的激光束的激光索引。在一些示例中，G-PCC译码器可以通过选择最接近当前节点中的标记点的激光束来确定与当前节点相交或者最接近当前节点的激光的索引。在某些示例中，当前节点中的标记点可以是当前节点的中心点，其坐标位于当前节点所有三个维度的一半(例如，立方体或长方体维度)处。在其他示例中，当前节点中的标记点可以是作为当前节点一部分的任何其他点，诸如节点内或节点侧面上、或节点边缘上或节点拐角上的任何点。

[0101] G-PCC译码器可以基于比较候选激光的角度差来确定激光是否在标记点附近。例如，G-PCC译码器可以比较激光束的角度与标记点的角度之间的差异。激光束的角度可以定义为在水平面(z＝0)与激光束的方向之间。标记点的角度可以定义为在水平面与虚拟光束到标记点的方向之间。在这种情况下，原点可以与传感器或激光器的中心同位置。可替代地，在一些示例中，在比较之前，可以将数学函数或诸如正切的三角函数应用于这些角度。

[0102] 在一些示例中，G-PCC译码器可以基于垂直坐标差的比较来确定激光是否在标记点附近。例如，G-PCC译码器可以比较标记点相对于传感器原点的垂直坐标(例如，标记点的Z坐标)和激光与节点相交或接近节点的垂直坐标。G-PCC译码器可以通过将水平面与激光方向之间的角的正切乘以通过基于标记点的(x,y)坐标取欧几里得距离计算出的距离(三角学)，来获得激光与节点相交的垂直坐标。

[0103] 在一些示例中，G-PCC译码器可以基于激光束和标记点的相对位置来确定用于对垂直面位置语法元素进行译码的上下文索引。例如，如果激光束在标记点之上，G-PCC译码器可以确定上下文索引是第一上下文索引，并且如果激光束在标记点之下，G-PCC译码器可以确定上下文索引是第二上下文索引。G-PCC译码器可以以与确定与节点相交或接近节点的激光束索引类似的方式来确定激光束是在标记点之上还是之下，例如，通过比较角差、比较角值的正切差或比较垂直坐标差。

[0104] 在一些示例中，作为确定上下文索引的一部分，G-PCC译码器可以确定距离阈值。G-PCC译码器可以使用距离阈值来比较激光束与标记点之间的距离。距离阈值可以将当前节点内的距离范围划分为间隔。间隔可以是等长或不等长。如果激光束在由距离间隔确定的距离范围内，则这些间隔中的每个间隔可以对应于一个上下文索引。在一些示例中，有两个由标记点上方和下方的相等距离偏移确定的距离阈值，它们定义了对应于三个上下文索引的三个距离间隔。G-PCC译码器可以以与确定与节点相交的或者最接近的激光的激光束索引类似的方式，来确定激光束是否属于一个间隔(例如，通过比较角差、比较角值的正切差、或比较垂直坐标差)。

[0105] 采用传感器信息的上述原则不限于对节点内的平面模式的垂直(Z)平面位置语法元素进行译码，但类似的原则也可以应用于对节点内的平面模式的X或Y平面位置语法元素进行译码。如果平面模式的X或Y平面位置模式更适合对节点内的点分布进行译码，则可以由编码器选择这些模式。例如，如果被占用的子节点都在面向X方向的平面的一侧，则可以

使用X平面位置语法元素来对节点内的点分布进行译码。如果被占用的子节点都在面向Y方向的平面的一侧，则可以使用Y平面位置语法元素来对节点内的点分布进行译码。另外，在X、Y或Z方向上定向的两个或更多个平面的组合可以用于指示子节点的占用。

[0106] 作为一个示例，G-PCC译码器可以基于激光束是在标记点之上还是之下(即，定位在标记点之上还是之下)来从两个上下文中确定上下文。在本示例中，标记点是节点的中心。这在图5中示出。更具体地，图5是其中基于激光束位置500A、500B在节点504的标记点502之上或之下来确定上下文索引的示例的概念图。因此，在图5的示例中，如果与节点504相交的激光在标记点502之上，如关于激光束位置500A所示，则G-PCC译码器选择第一上下文索引(例如，Ctx＝0)。在图5的示例中，标记点502位于节点504的中心。如果与节点504相交的激光低于标记点502，如关于激光束位置500B所示，则G-PCC译码器选择第二上下文索引(例如，Ctx＝1)。

[0107] 在一些示例中，G-PCC译码器基于激光束位于两个距离阈值之上或之下，或者在距离阈值之间来确定三个上下文。在本示例中，标记点是节点的中心。这在图6中示出。更具体地，图6是示出节点600的示例三个上下文索引确定的概念图。在图6中，用细虚线602A、602B指示距离间隔阈值。激光束用实线604A、604B指示。这些激光束中的每一个都可以是激光候选。中心点606(标记点)用白色圆指示。因此，在图6的示例中，如果激光(诸如对应于线604A的激光)在线602A之上，则G-PCC译码器选择上下文索引ctx1。如果激光在线602A、602B之间，则G-PCC译码器可以选择上下文索引ctx0。如果激光(诸如对应于线604B的激光)在线602B之下，则G-PCC译码器可以选择上下文索引ctx2。

[0108] 在一些示例中，在使用角模式的情况下，G-PCC译码器使用四个上下文来对平面模式的垂直面位置进行译码。在这样的示例中，G-PCC译码器可以基于激光束在四个间隔内的位置来确定上下文索引。该示例在图7中示出。图7是示出利用细虚线分开的间隔、基于激光束位置(实心箭头)、用于对平面模式的垂直面位置(角模式)进行译码的示例上下文索引确定的概念图。在图7的示例中，线700A、700B和700C对应于距离间隔阈值。此外，在图7的示例中，标记点702位于节点704的中心。线706对应于与节点704相交的激光束。因为线706在线700A之上，因此G-PCC译码器可以选择上下文索引ctx2，用于对垂直面位置进行译码。

[0109] 图8A是示出用于对垂直面位置进行编码的示例操作的流程图。G-PCC编码器200可以执行图8A的操作作为对点云进行编码的一部分。

[0110] 在图8A的示例中，G-PCC编码器200(例如，G-PCC编码器200(图2)的算术编码单元214)可以对树(例如八叉树)的节点中的平面模式的垂直面位置进行编码(800)，该树表示由点云数据表示的点云中的点的三维位置。换句话说，G-PCC编码器200可以对垂直面位置进行编码。

[0111] 作为对平面模式的垂直面位置进行编码的一部分，G-PCC编码器200(例如，算术编码单元214)可以确定激光候选集中的激光候选的激光索引，其中所确定的激光索引指示与节点相交的激光束(802)。G-PCC编码器200可以根据本公开中其他地方提供的任何示例来确定激光索引。

[0112] 附加地，G-PCC编码器200(例如，算数编码单元214)可以基于激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间、还是低于第三距离阈值，来确定上下文索引(804)。例如，在图7的示例中，G-PCC编码器200可以

基于激光束是否高于第一距离阈值(对应于线700A)、是否在第一距离阈值与第二距离阈值(对应于线700B)之间、是否在第二距离阈值与第三距离阈值(对应于线700C)之间或是否低于第三距离阈值,来确定上下文索引。在一些示例中,为了确定激光束相对于第一、第二和第三距离阈值的位置,G-PCC编码器200可以通过从激光束的角度的正切减去穿过节点中心的线的角度的正切来确定激光差角度(例如,thetaLaserDelta),通过从激光差角度减去移位值来确定顶角差(例如,下面的DeltaTop);以及通过将移位值与激光差角度相加来确定底角差(例如,下面的DeltaBot)。

[0113] G-PCC编码器200可以执行确定激光差角度是否大于或等于0的第一比较(例如,thetaLaserDelta>=0)。G-PCC编码器200可以基于激光差角度是否大于或等于0而将所述上下文索引设置为0或1(例如,contextAngular[Child]=thetaLaserDelta>=0?0:1)。附加地,G-PCC编码器200可以执行确定顶角差是否大于或等于0的第二比较(例如,DeltaTop>=0)。当顶角差大于或等于0时,激光束在第一距离阈值之上。G-PCC编码器200还可以执行确定底角差是否小于0的第三比较(例如,DeltaBottom<0)。当底角差小于0时,激光束在第三距离阈值以下。G-PCC编码器200可以在如下情况将上下文索引增加2:基于顶角差大于或等于0(例如,if(DeltaTop>=0)contextAngular[Child]+=2),或基于底角差小于0(例如,else if(DeltaBottom<0)contextAngular[Child]+=2)。

[0114] G-PCC编码器200(例如,G-PCC编码器200的算术编码单元214)可以使用由所确定的上下文索引指示的上下文来对平面模式的垂直面位置进行算术编码(806)。例如,G-PCC编码器200可以对指示垂直面位置的语法元素执行CABAC编码。

[0115] 图8B是示出用于对垂直面位置进行解码的示例操作的流程图。G-PCC解码器300可以执行图8B的操作作为重建由点云数据表示的点云的一部分。在图8B的示例中,G-PCC解码器300(例如,图3的几何算术解码单元302)可以对树(例如,八叉树)的节点中的平面模式的垂直面位置进行解码(850),该树表示点云中的点的三维位置。换句话说,G-PCC解码器300可以对垂直面位置进行解码。

[0116] 作为对平面模式的垂直面位置进行解码的一部分,G-PCC解码器300(例如,几何算术解码单元302)可以确定激光候选集中的激光候选的激光索引,其中所确定的激光索引指示与节点相交或者最接近节点的激光束(852)。G-PCC解码器300可以根据本公开中其他地方提供的任何示例来确定激光索引。

[0117] 附加地,G-PCC解码器300可以基于激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间、还是低于第三距离阈值来确定上下文索引(contextAngular)(854)。如上所述,G-PCC解码器300可以以与G-PCC编码器200相同的方式来确定上下文索引。

[0118] G-PCC解码器300(例如,G-PCC解码器300的几何算术解码单元302)可以使用由所确定的上下文索引指示的上下文来对平面模式的垂直面位置进行算术解码(856)。例如,G-PCC解码器300可以对指示垂直面位置的语法元素执行CABAC解码。在一些示例中,G-PCC解码器300可以基于垂直面位置来确定点云中的一个或多个点的位置。例如,G-PCC解码器300可以基于垂直面位置来确定节点的被占用子节点的位置。然后,G-PCC解码器300可以处理被占用的子节点以确定被占用的子节点内的点的位置,并且可以不需要对未被占用的子节点执行进一步的处理。

[0119] G-PCC译码器(例如,G-PCC编码器200或G-PCC解码器300)可以部分地通过从激光候选集中选择激光索引来对节点内的IDCM的垂直点位置偏移进行译码(即,编码或解码)。可以在诸如几何参数集的参数集中发信令通知激光候选集,其中所选择的激光索引指示与该节点相交的激光束。激光器候选集可以对应于LIDAR阵列中的激光器。G-PCC译码器可以基于激光束与节点的相交来确定上下文索引,以便对来自IDCM的垂直点位置偏移的二进制数(比特)进行算术译码。

[0120] 如上所述,G-PCC编码器200和G-PCC解码器300可以被配置为执行角模式的上下文推导。下面描述三个坐标x、y和z的平面模式上下文推导。当LIDAR激光器在xy平面内旋转时,对x和y坐标的处理可以彼此相似并且不同于对z坐标的处理。在其他示例中,两个不同的平面可以与第三个不同的平面相似。

[0121] 为了确定节点的位置和节点的中点,G-PCC编码器200和G-PCC解码器300可以导出如下所述的变量absPos和midNode,其中child.pos和childSizeLog2分别参考当前节点的位置和尺寸。

```
{
  Vec3<int64_t> absPos = {child.pos[0] << childSizeLog2[0],
                          child.pos[1] << childSizeLog2[1],
                          child.pos[2] << childSizeLog2[2]};
```

[0122]

```
  // 合格
  Vec3<int64_t> midNode = {1 << (childSizeLog2[0] ? childSizeLog2[0] -
1 : 0),
                           1 << (childSizeLog2[1] ? childSizeLog2[1] - 1 : 0),
                           1 << (childSizeLog2[2] ? childSizeLog2[2] - 1 : 0)};
```

[0123] 当节点尺寸太大时(即当多于一个激光可能通过节点时),G-PCC编码器200以及因此G-PCC解码器300禁用角模式。G-PCC编码器200计算在节点中间的距离处的两个相邻激光之间的最小距离的估计(deltaAngle),并将该估计与节点尺寸进行比较。在当前节点大于阈值时,G-PCC编码器200禁用角模式。变量headPos指的是LIDAR头的位置,可用于如下查找相对于LIDAR原点的坐标xLidar/yLidar/zLidar:

```
uint64_t xLidar =
    std::abs(((absPos[0] - headPos[0] + midNode[0]) << 8) - 128);
uint64_t yLidar =
    std::abs(((absPos[1] - headPos[1] + midNode[1]) << 8) - 128);
```

[0124]

```
uint64_t rL1 = (xLidar + yLidar) >> 1;
uint64_t deltaAngleR = deltaAngle * rL1;
if (deltaAngleR <= (midNode[2] << 26))
    return -1;
```

[0125] 在下一步骤中,G-PCC编码器200计算节点的中间在z方向(zLidar)上的角度仰角theta32,并且如下估计最接近节点中心的激光的索引laserIndex:

```
// 确定 r (1/sqrt(r2) = irsqrt(r2))的逆
uint64_t r2 = xLidar * xLidar + yLidar * yLidar;
uint64_t rInv = irsqrt(r2);

// 确定未校正数据
int64_t zLidar = ((absPos[2] - headPos[2] + midNode[2]) << 1) - 1;
int64_t theta = zLidar * rInv;
```

[0126]

```
int theta32 = theta >= 0 ? theta >> 15 : -((-theta) >> 15);

// 确定激光
int laserIndex = int(child.laserIndex);
if (laserIndex == 255 || deltaAngleR <= (midNode[2] << (26 + 2))) {
    auto end = thetaLaser + numLasers - 1;
    auto it = std::upper_bound(thetaLaser + 1, end, theta32);
```

```
    if (theta32 - *std::prev(it) <= *it - theta32)
      --it;
```

[0127]

```
    laserIndex = std::distance(thetaLaser, it);
    child.laserIndex = uint8_t(laserIndex);
  }
```

[0128] 一旦估计了最接近的激光，G-PCC编码器200估计x和y坐标的平面模式的上下文。在一些示例中，G-PCC编码器200仅确定x和y中仅一个的上下文，并且当一个坐标以角平面模式译码时，另一个坐标不是平面译码的。对于每个激光，最后译码/预测的方位角被存储在phiBuffer中。基于激光的速度(输入参数)，G-PCC编码器200更新方位角的预测，并通过比较更新后的方位角预测、节点原点的方位角，然后检查最接近的激光穿过当前节点的哪个部分来确定上下文，其可以用于估计x/y维度的六个上下文之一，如下所示：

```
  // -- PHI    --
    //角度
    int posx = absPos[0] - headPos[0];
    int posy = absPos[1] - headPos[1];
    int phiNode = iatan2(posy + midNode[1], posx + midNode[0]);
    int phiNode0 = iatan2(posy, posx);

    // 找到预测器
```

[0129]

```
    int predPhi = phiBuffer[laserIndex];
    if (predPhi == 0x80000000)
      predPhi = phiNode;

    // 使用预测器
    if (predPhi != 0x80000000) {
      // 基本移位预测器
      int Nshift =
        ((predPhi - phiNode) * phiZi.invDelta(laserIndex) + 536870912) >>
```

```
30;
            predPhi -= phiZi.delta(laserIndex) * Nshift;

            // ctx 方位角 x 或 y
            int angleL = phiNode0 - predPhi;
            int angleR = phiNode - predPhi;
            int contextAnglePhi =
              (angleL >= 0 && angleR >= 0) || (angleL < 0 && angleR < 0) ? 2 : 0;
            angleL = std::abs(angleL);
            angleR = std::abs(angleR);
            if (angleL > angleR) {
              contextAnglePhi++;
              int temp = angleL;
              angleL = angleR;
              angleR = temp;
            }
            if (angleR > (angleL << 2))
              contextAnglePhi += 4;

            if (std::abs(posx) <= std::abs(posy))
              *contextAnglePhiX = contextAnglePhi;
            else
              *contextAnglePhiY = contextAnglePhi;
          }
```

[0130]

[0131] 对于z维的上下文,选择了三个阈值:theta32、theta32-zShift和theta32+zShift。通过将最接近的激光(thetaLaser[laserIndex])的仰角与三个阈值进行比较,G-PCC编码器200导出z坐标的平面模式的四个上下文之一,如下所示:

```
// -- THETA --
```

[0132]

```
int thetaLaserDelta = thetaLaser[laserIndex] - theta32;
int64_t hr = zLaser[laserIndex] * rInv;
```

```
    thetaLaserDelta += hr >= 0 ? -(hr >> 17) : ((-hr) >> 17);

    int64_t zShift = (rInv << childSizeLog2[2]) >> 20;
    int thetaLaserDeltaBot = thetaLaserDelta + zShift;
    int thetaLaserDeltaTop = thetaLaserDelta - zShift;
    int contextAngle = thetaLaserDelta >= 0 ? 0 : 1;
    if (thetaLaserDeltaTop >= 0)
      contextAngle += 2;
    else if (thetaLaserDeltaBot < 0)
      contextAngle += 2;

    return contextAngle;
  }
```

[0133]

[0134] 如上所述,G-PCC编码器200和G-PCC解码器300也可以被配置为确定IDCM模式上下文。IDCM指的是G-PCC中的一种模式,其中节点内的点的位置在不使用该节点内的八叉树结构的情况下被译码。例如,IDCM对于具有稀疏节点或异常值的点云可能是有益的,其中在这种情况下对点的坐标进行译码可能比用完整的八叉树结构对点进行译码更有利。在G-PCC中,节点中只有一个或两个不同的位置可以作为IDCM节点被译码。

[0135] 当使用角模式时,G-PCC编码器200可以基于激光参数对对应于位置的比特进行上下文译码。下面描述导出以角IDCM模式译码的比特的上下文的示例方式。以下说明描述可以由G-PCC编码器200和/或G-PCC解码器300执行的技术。

[0136] G-PCC编码器200和G-PCC解码器300可以被配置为确定nodePosition和激光索引。如上所述,在IDCM中,节点中多达两个不同的位置可以被编码。第一步骤可以是确定占用最多两个位置的点的数量,这可以如下完成:

[0137]

```
    int numPoints = 1;
      bool numPointsGt1 =
_arithmeticDecoder->decode(_ctxNumIdcmPointsGt1);
      numPoints += numPointsGt1;
```

```
int numDuplicatePoints = 0;
if (!geom_unique_points_flag && !numPointsGt1) {
  numDuplicatePoints
= !_arithmeticDecoder->decode(_ctxSinglePointPerBlock);
  if (numDuplicatePoints) {
    bool                              singleDup                              =
[0138] _arithmeticDecoder->decode(_ctxSingleIdcmDupPoint);
    if (!singleDup)
      numDuplicatePoints +=
        1          +          _arithmeticDecoder->decodeExpGolomb(0,
_ctxPointCountPerBlock);
  }
}
```

[0139] 对于下一步骤,G-PCC编码器200和G-PCC解码器300可以被配置为检查平面模式是否应用于当前节点。如果应用平面模式,则对应坐标的MSB可以被单独地发信号通知(平面),并且有效地仅对剩余的比特进行译码。使用平面模式的MSB的译码可以在IDCM译码之前执行。节点尺寸如下从effectiveNodeSizeLog2更新为nodeSizeLog2Rem:

```
//更新平面后的节点尺寸，并从平面确定位置的上部
Vec3<int32_t> deltaPlanar{0, 0, 0};
  Vec3<int> nodeSizeLog2Rem = nodeSizeLog2;
  for (int k = 0; k < 3; k++)
[0140]
    if (nodeSizeLog2Rem[k] > 0 && (planar.planarMode & (1 << k))) {
      deltaPlanar[k] |= (planar.planePosBits & (1 << k) ? 1 : 0);
      nodeSizeLog2Rem[k]--;
    }
```

[0141] 对于下一步骤,G-PCC编码器200和G-PCC解码器300计算节点相对于LIDAR原点头柱的位置posNodeLidar,如下所示。x和y位置被用于确定该节点中位置的x或y坐标是否要被上下文译码,而另一个位置被旁路译码。

```
Vec3<bool> directIdcm = !angularIdcm;
[0142]
  point_t posNodeLidar;
```

```
if (angularIdcm) {
  posNodeLidar =
    point_t(
      node.pos[0] << nodeSizeLog2[0], node.pos[1] << nodeSizeLog2[1],
      node.pos[2] << nodeSizeLog2[2])
    - headPos;
  bool codeXorY = std::abs(posNodeLidar[0]) <=
std::abs(posNodeLidar[1]);
  directIdcm.x() = !codeXorY;
  directIdcm.y() = codeXorY;
}
```

[0143]

[0144] 当对多于一个点被译码时,G-PCC编码器200和G-PCC解码器300对点进行排序,这进一步减少了对点位置进行译码所需的比特数。这是通过对x或y或z位置的一个或多个MSB进行译码来实现的;结果,可以进一步减少剩余待译码的比特数(nodeSizeLog2Rem)。

```
//对未排序的两个点进行解码
Vec3<int32_t> deltaPos[2];
  deltaPos[0] = deltaPlanar;
  deltaPos[1] = deltaPlanar;
  if (numPoints == 2 && joint_2pt_idcm_enabled_flag)
    decodeOrdered2ptPrefix(directIdcm, nodeSizeLog2Rem, deltaPos);
```

[0145]

[0146] 由于一些比特已经被译码(由于点的平面和联合译码),G-PCC编码器200和G-PCC解码器300可以计算更新后的节点位置的中点,并使用该中点来确定穿过节点中点的最接近的激光,如下所示:

```
if (angularIdcm) {
  for (int idx = 0; idx < 3; ++idx) {
    int N = nodeSizeLog2[idx] - nodeSizeLog2Rem[idx];
    for (int mask = N ? 1 << (N - 1) : 0; mask; mask >>= 1) {
      if (deltaPos[0][idx] & mask)
        posNodeLidar[idx] += mask << nodeSizeLog2Rem[idx];
```

[0147]

```
    }
    if (nodeSizeLog2Rem[idx])
      posNodeLidar[idx] += 1 << (nodeSizeLog2Rem[idx] - 1);
  }
  node.laserIndex = findLaser(posNodeLidar, thetaLaser, numLasers);
}
```

[0148]

[0149] 如果启用角模式,G-PCC编码器200和G-PCC解码器300可以对尚未译码的位置的比特进行上下文译码。例如,如下所述,x和y中的一个的比特可以是上下文译码的,而其他比特可以是旁路译码的,而z的比特是上下文译码的。否则,所有剩余的比特可以被旁路译码。

```
Vec3<int32_t> pos;
for (int i = 0; i < numPoints; i++) {
  if (angularIdcm) {
    *(outputPoints++) = pos = decodePointPositionAngular(
      nodeSizeLog2, nodeSizeLog2Rem, node, planar, headPos, zLaser,
      thetaLaser, deltaPos[i]);
  } else
    *(outputPoints++) = pos =
      decodePointPosition(nodeSizeLog2Rem, deltaPos[i]);
  }
```

[0150]

[0151] G-PCC编码器200和G-PCC解码器300可以被配置为确定用于角IDCM的上下文。现在将描述基于激光参数确定用于IDCM节点位置的上下文。

[0152] G-PCC编码器200和G-PCC解码器300可以计算相对于LIDAR头位置头柱的节点位置posXyz,其可以再次用于确定哪个坐标(x或y)将被上下文译码。如果posXyz[1]≥posXyz[0](即,绝对y坐标值更大),则G-PCC编码器200和G-PCC解码器300可以对y坐标比特进行旁路译码并且对x坐标比特进行上下文译码。否则,(即,绝对y坐标值小于绝对x坐标),G-PCC编码器200和G-PCC解码器300可以对x坐标比特进行旁路译码并且对y坐标比特进行上下文译码。当每个比特被译码时,计算更新后的节点posXyz。

```
{
  Vec3<int32_t> delta = deltaPlanar;
```

[0153]

```
Vec3<int> posXyz = {(child.pos[0] << nodeSizeLog2[0]) - headPos[0],
                    (child.pos[1] << nodeSizeLog2[1]) - headPos[1],
                    (child.pos[2] << nodeSizeLog2[2]) - headPos[2]};

// -- PHI --
// 对 x 或 y 直接译码并且计算节点的 phi
bool codeXorY = std::abs(posXyz[0]) <= std::abs(posXyz[1]);
if (codeXorY) {    // direct code y
  if (nodeSizeLog2AfterPlanar[1])
    for (int i = nodeSizeLog2AfterPlanar[1]; i > 0; i--) {
      delta[1] <<= 1;
      delta[1] |= _arithmeticDecoder->decode();
    }
  posXyz[1] += delta[1];
  posXyz[0] += delta[0] << nodeSizeLog2AfterPlanar[0];
} else {    //direct code x
  if (nodeSizeLog2AfterPlanar[0])
    for (int i = nodeSizeLog2AfterPlanar[0]; i > 0; i--) {
      delta[0] <<= 1;
      delta[0] |= _arithmeticDecoder->decode();
    }
  posXyz[0] += delta[0];
  posXyz[1] += delta[1] << nodeSizeLog2AfterPlanar[1];
}
```

[0154]

[0155] 在下一步骤中,G-PCC编码器200和G-PCC解码器300使用以上述方式相对于平面模式上下文导出的激光索引和包括在比特流中的激光索引率残差来计算该点的激光索引。

[0156] //找到预测器

[0157] int phiNode＝iatan2(posXyz[1],posXyz[0]);

[0158] int laserNode＝int(child.laserIndex);

[0159] //激光残差

[0160] int laserIndex＝laserNode+decodeThetaRes();

[0161] G-PCC编码器200和G-PCC解码器300使用这一更新后的laserIndex来确定存储在

buffer_phiBuffer中的方位角(phi)的预测。当激光近似均匀地对方位角范围内的点进行采样时,G-PCC编码器200和G-PCC解码器300可以使用phiZi.delta来指示方位角值中的最小差。因此,可能已经从预测的phi值跳过的点数被计算为nShift,并且预测的phi值被更新。

```
int predPhi = _phiBuffer[laserIndex];
  if (predPhi == 0x80000000)
    predPhi = phiNode;
```

[0162]

```
// 基本移位预测器
int nShift =
  ((predPhi - phiNode) * _phiZi.invDelta(laserIndex) + 536870912) >> 30;
predPhi -= _phiZi.delta(laserIndex) * nShift;
```

[0163] 对于进行上下文编码的x和y之间的坐标,G-PCC解码器300对剩余的nodeSizeLog2AfterPlanar[0](被上下文编码的)进行解码。对于每个比特,G-PCC解码器300通过在每个比特被解码时更新posXY来更新节点位置,从而确定更新后的上下文值。G-PCC解码器300还在节点位置被更新时重新计算预测的phi值。六个(或八个)上下文中的一个被选中比较节点位置的方位角、在上下文编码的方向(即x或y)上偏移半个节点尺寸的位置并且被预测的方位角。一旦位置被编码,节点phiNode的方位角被存储在缓冲器中。

```
// 选择 x 或 y
int* posXY = codeXorY ? &posXyz[0] : &posXyz[1];
  int idx = codeXorY ? 0 : 1;
```

[0164]

```
// 对 x 或 y 进行方位角译码
int mask2 = codeXorY
  ? (nodeSizeLog2AfterPlanar[0] > 0 ? 1 << (nodeSizeLog2AfterPlanar[0]
- 1)
                                    : 0)
```

```
        : (nodeSizeLog2AfterPlanar[1] > 0 ? 1 << (nodeSizeLog2AfterPlanar[1]
- 1)
                                                    : 0);
    for (; mask2; mask2 >>= 1) {
      // 角度左移或右移
      int phiR = codeXorY ? iatan2(posXyz[1], posXyz[0] + mask2)
                          : iatan2(posXyz[1] + mask2, posXyz[0]);
      int phiL = phiNode;

      // ctx 方位角
      int angleL = phiL - predPhi;
      int angleR = phiR - predPhi;
      int contextAnglePhi =
        (angleL >= 0 && angleR >= 0) || (angleL < 0 && angleR < 0) ? 2 : 0;
      angleL = std::abs(angleL);
[0165]  angleR = std::abs(angleR);
      if (angleL > angleR) {
        contextAnglePhi++;
        int temp = angleL;
        angleL = angleR;
        angleR = temp;
      }
      if (angleR > (angleL << 1))
        contextAnglePhi += 4;

      // 熵译码
      bool bit = _arithmeticDecoder->decode(
        _ctxPlanarPlaneLastIndexAngularPhiIDCM[contextAnglePhi]);
      delta[idx] <<= 1;
      if (bit) {
        delta[idx] |= 1;
```

```
                *posXY += mask2;
                phiNode = phiR;
                predPhi = _phiBuffer[laserIndex];
                if (predPhi == 0x80000000)
                  predPhi = phiNode;

                // 基本移位预测器
[0166]          int nShift =
                  ((predPhi - phiNode) * _phiZi.invDelta(laserIndex) + 536870912) >>
        30;
                  predPhi -= _phiZi.delta(laserIndex) * nShift;
              }
            }

            // 更新缓冲 phi
```

[0167] _phiBuffer[laserIndex]＝phiNode；

[0168] 一旦对x和y位置进行译码，就对z坐标进行译码，类似地，对z方向上的平面模式比特的上下文的推导(上面相对于平面模式上下文进行解释的)具有如上所述的多比特的泛化。当每个z坐标比特被译码时，G-PCC解码器300用更新后的节点位置和z值重新计算下一比特的上下文，该z值在z方向上偏离节点位置一半节点尺寸(posXyz[2]+maskz)。再次，G-PCC解码器300选择四个上下文中的一个来对z坐标比特进行译码。

```
          // -- THETA --
            int maskz =
              nodeSizeLog2AfterPlanar[2] > 0 ? 1 << (nodeSizeLog2AfterPlanar[2] -
        1) : 0;
[0169]      if (!maskz)
              return delta;

            int posz0 = posXyz[2];
            posXyz[2] += delta[2] << nodeSizeLog2AfterPlanar[2];
```

```
// 由于 x 和 y 已知，
// r 也是已知并且不依赖于 z 的比特
uint64_t xLidar = (int64_t(posXyz[0]) << 8) - 128;
uint64_t yLidar = (int64_t(posXyz[1]) << 8) - 128;
uint64_t r2 = xLidar * xLidar + yLidar * yLidar;
int64_t rInv = irsqrt(r2);

// 使用角模式对 z 的比特进行译码。资格是隐式的。激光已知。
int64_t hr = zLaser[laserIndex] * rInv;
int fixedThetaLaser =
  thetaLaser[laserIndex] + int(hr >= 0 ? -(hr >> 17) : ((-hr) >> 17));

int zShift = (rInv << nodeSizeLog2AfterPlanar[2]) >> 18;
for (int bitIdxZ = nodeSizeLog2AfterPlanar[2]; bitIdxZ > 0;
       bitIdxZ--, maskz >>= 1, zShift >>= 1) {
```

[0170]

```
  // 确定未校正数据
  int64_t zLidar = ((posXyz[2] + maskz) << 1) - 1;
  int64_t theta = zLidar * rInv;
  int theta32 = theta >= 0 ? theta >> 15 : -((-theta) >> 15);
  int thetaLaserDelta = fixedThetaLaser - theta32;

  int thetaLaserDeltaBot = thetaLaserDelta + zShift;
  int thetaLaserDeltaTop = thetaLaserDelta - zShift;
  int contextAngle = thetaLaserDelta >= 0 ? 0 : 1;
  if (thetaLaserDeltaTop >= 0)
    contextAngle += 2;
  else if (thetaLaserDeltaBot < 0)
    contextAngle += 2;

  delta[2] <<= 1;
```

```
            delta[2] |= _arithmeticDecoder->decode(
              _ctxPlanarPlaneLastIndexAngularIdcm[contextAngle]);
            posXyz[2] = posz0 + (delta[2] << (bitIdxZ - 1));
```

[0171] }

```
        return delta;
    }
```

[0172] G-PCC编码器200和G-PCC解码器300可以被配置为使用树内量化。树内几何缩放提供了一种量化(例如,在G-PCC编码器200处)和缩放(例如,在G-PCC解码300处器)几何位置的手段,即使在构建编码树时也是如此。在当前的文本草案中,量化步长和位置的缩放应用如下:

[0173] 移位值(sh)计算如下:

[0174] qpScaled=qp<<qpDivFactorLog2

[0175] sh=qpScaled>>3

[0176] 缩放过程指定如下:

[0177] scaled_x=(x*(8+qpScaled%8)<<sh+4)>>3

[0178] G-PCC编码器200和G-PCC解码器300在八叉树中的特定节点深度(在比特流中发信号通知的)处应用几何量化。在一个简单的示例中,如果节点坐标是12比特,并且要应用量化的深度是3,则节点坐标的前3个MSB(被称为位置的MSB部分)不被量化。只有节点坐标的最后9个LSB(被称为位置的LSB部分)被量化,。由于量化,9个LSB可以减少到更少的比特数,在本文称为N,其中N小于9。这一量化导致以重建精度为代价的比特率的一些降低。结果节点坐标尺寸变为N+3(即<=12)。

[0179] 类似地,在G-PCC解码器300处,N个LSB被缩放并剪裁至1<<(9-1)的最大值,这确保经缩放值不超过原始点的9个LSB比特。通过连接3个MSB和经缩放LSB的9个比特来计算最终缩放位置。

[0180] 另外,还发信号通知qp缩放因子,其确定步长每翻倍可以定义的QP的最小数量。在G-PCC中,该因子可以取值0、1、2和3。

[0181] 对于qpDivFactorLog2的各种值,步长每翻倍的QP点数量在下表中指定:

[0182]

qpDivFactorLog2	步长每翻倍的 QP 点大小

[0183]

0	8
1	4
2	2
3	1

[0184] 对于所有qpDivFactorLog2值,步长推导保持与缩放过程相同,针对步长翻倍8个

QP点,这是通过在计算移位比特和缩放过程之前调整QP值来完成的。

[0185] 现有的技术存在一些潜在的问题。例如,在点云帧的常规译码中,角模式为译码效率提供了相当大的增益。然而,当启用树内量化时,角模式的增益会显著降低,在某些情况下实际上会产生损耗。对于角模式(IDCM角和平面角),量化比特被用于上下文推导。量化比特与原始点不在同一域中,并且在不同的缩放空间中。这降低了角模式和树内量化两者的有用性,因为同时启用它们两者可能是不利的。

[0186] 本公开描述了可以解决上面介绍的一些问题的技术。本文描述的各种技术可以单独地或组合地执行。

[0187] 根据本公开的技术,G-PCC编码器200和G-PCC解码器300可以被配置为针对点/位置坐标值x导出节点/点位置坐标的经缩放值xS,以导出节点/点相对于激光雷达原点的位置。在一些示例中,缩放操作可以包括对位置的一个或多个比特进行缩放,并且可以附加地包括被缩放的比特值的最大数量,其中最大数量基于发信号通知的值,诸如从将要应用量化的最大深度导出的值。在一些示例中,对于y或z坐标,或者在一些情况下两个或更多个坐标,也可以导出类似的经缩放位置。

[0188] 根据本公开的技术,G-PCC编码器200和G-PCC解码器300使用经缩放值来确定节点相对于LIDAR头位置的相对位置,或者更一般地,相对于原点位置的相对位置。G-PCC编码器200和G-PCC解码器300可以例如基于是否已经在该节点位置/坐标上应用了任何量化来确定要使用的经缩放值。在一些示例中,当节点位置/坐标值在与头位置不同的域中时,G-PCC编码器200和G-PCC解码器300可以使用经缩放值。例如,如果在同一域(标度)中描述节点坐标x_0和头位置h_0,并且通过应用量化将x_0的值修改为x,则经缩放值被用于确定相对于LIDAR头的相对位置。

[0189] 根据本公开的技术,G-PCC编码器200和G-PCC解码器300使用一个或多个经缩放坐标值来计算激光特性。激光器特性可以包括例如仰角(激光相对于x-y平面形成的角度)或激光头偏移。激光特性可以包括方位角或方位预测。

[0190] 根据本公开的技术,G-PCC编码器200和G-PCC解码器300可以被配置为在没有剪裁的情况下执行缩放操作。例如,位置的MSB位可以与比特的LSB部分的经缩放值相加。在一些情况下,G-PCC编码器200和G-PCC解码器300可以应用剪裁,在这种情况下,位置的MSB比特与比特的LSB部分的经缩放值的剪裁后版本相加。在一些示例中,一些缩放操作可以包括剪裁,而其他缩放操作可以不应用剪裁。这可以由译码的坐标、应用QP的深度和QP值来确定。

[0191] 根据本公开的技术,G-PCC编码器200和G-PCC解码器300可以被配置为基于qpDivFactorLog2的值来限制角模式(全部或部分)。例如,当qpDivFactorLog2的值为0、1或2时,G-PCC编码器200和G-PCC解码器300可以被配置为禁用角模式(全部或部分)—即,仅当两个步长的幂被允许用于几何量化/缩放时。

[0192] 在一些示例中,G-PCC编码器200和G-PCC解码器300可以被配置为仅当参考参数集的所有节点的QP值对应于2的幂的步长时才启用角模式(例如在参数集(例如GPS)中指示)。在另一示例中,当树内几何缩放被启用时(例如,通过将geom_scaling_enabled_flag设置为等于1),G-PCC编码器200和G-PCC解码器300可以被配置为完全或部分地禁用角模式。在上面的示例中,角模式的部分禁用可以包括使用角模式禁用平面模式相关参数(或比特)的上下文推导和使用角模式禁用IDCM模式相关参数(或比特)的上下文推导中的一个或多个。

[0193] 根据本公开的技术,G-PCC编码器200和G-PCC解码器300可以被配置为基于特定节点的QP值来修改角模式的上下文推导。在一个示例中,可以基于QP值(诸如QP的函数)来指定在上下文推导中使用的一个或多个阈值。例如,阈值对于QP值0可以是不变的,但是当QP大于0时可以被修改。在其他示例中,对于导致步长为2的幂的QP值,阈值可以不变,但以其他方式修改。

[0194] 修改后的QP值可以用QP值和缩放因子和/或偏移的表来指定,以修改阈值。例如,当相对于QP进行修改时,可以使用固定倍数来缩放阈值(对于非零QP,将阈值加倍)。在另一示例中,阈值"z"可以被修改为z_mod＝func(z,QP),其中func()是预定函数,或者是比特流中指示的函数。例如,func(z,QP)＝a(QP)*z+b(QP),其中a(QP)和b(QP)是基于QP值的参数集(函数a()或b()可以是线性或非线性函数)。

[0195] 阈值的修改可以应用于三个分量x、y或z中的一个或多个。所述修改可以应用于与theta/激光角度相关联的上下文或与方位角相关联的上下文中的一个或多个。阈值的修改可以应用于平面角模式和角IDCM模式中的一个或多个。

[0196] 根据本公开的技术,G-PCC编码器200和G-PCC解码器300可以被配置为使用一个或多个阈值来确定是否启用角模式,并且可以使用用于上下文推导的一种或多种技术来基于QP修改一个或多个阈值。(例如,如果当节点尺寸大于阈值时角模式被禁用,则该阈值可以基于QP来修改,诸如将某些QP值的阈值加倍)。

[0197] 一个示例实现使用经缩放的点值来确定相对于LIDAR节点的相对位置。贯穿本公开中,在标识符<add>和</add>之间示出了添加,在标识符~~和~~之间示出了删除。

[0198] G-PCC编码器200和G-PCC解码器300可以被配置为确定平面模式上下文。

[0199] 定义了函数invQuantPositionAngular,该函数基于从qp导出的步长和quantMasks指示的比特,对点的三个坐标应用缩放操作。该函数可以定义如下:在该示例中,缩放操作类似于针对树内量化定义的逆缩放操作,但是可以应用任何通用缩放操作。quantMasks是1和0的集合,其指示节点位置的哪些比特要被缩放。例如,如果quantMasks＝00000111,则只缩放最后三个比特。在该示例中,在缩放操作之后,位置的未缩放部分和位置的缩放部分被相加以形成"经缩放"位置。未缩放的部分可以在添加到缩放的部分之前被移位。

[0200]
```
<add>
invQuantPositionAngular(int qp, Vec3<uint32_t> quantMasks, const Vec3<int32_t>& pos)
{
```

```
QuantizerGeom quantizer(qp);
int shiftBits = QuantizerGeom::qpShift(qp);
Vec3<int64_t> recon;
for (int k = 0; k < 3; k++) {
  int lowPart = (pos[k] & quantMasks[k]) >> shiftBits;
  int highPart = pos[k] & ~(quantMasks[k]);
```

[0201]
```
  int lowPartScaled = PCCClip(quantizer.scale(lowPart), 0, quantMasks[k]);
  recon[k] = highPart | lowPartScaled;
}

return recon;
</add>
```

[0202] 在一些示例中，仅当qp不为零时才应用该过程。在其他示例中，可能不执行PCCClip()操作，并且通过将highPart和lowPartSCaled相加来获得recon[]变量，如下所示

[0203] <add>int lowPartScaled＝quantizer.scale(lowPart)

[0204] recon[k]＝highPart+lowPartScaled;</add>

[0205] 在平面变量的上下文的确定中，G-PCC编码器200和G-PCC解码器300对子节点absPos、midNode(表示节点尺寸的一半)和childSize(表示节点尺寸)的位置应用缩放操作——在上下文推导过程中使用这些值的缩放版本。

```
{
  Vec3<int64_t> absPos = {child.pos[0] << childSizeLog2[0],
                          child.pos[1] << childSizeLog2[1],
                          child.pos[2] << childSizeLog2[2]};
```

[0206]
```
  // 资格
  Vec3<int64_t> midNode = {1 << (childSizeLog2[0] ? childSizeLog2[0] - 1 : 0),
                           1 << (childSizeLog2[1] ? childSizeLog2[1] - 1 : 0),
```

```
                                                    1 << (childSizeLog2[2] ? childSizeLog2[2]
- 1 : 0)};
      <add>
        Vec3<int64_t> childSize = {
          1 << childSizeLog2[0], 1 << childSizeLog2[1], 1 << childSizeLog2[2]};
        if (child.qp) {
          absPos = invQuantPositionAngular(child.qp, quantMasks, absPos);
          midNode = invQuantPositionAngular(child.qp, quantMasks, midNode);
          childSize = invQuantPositionAngular(child.qp, quantMasks, childSize);
        }
      </add>

        uint64_t xLidar =
          std::abs(((absPos[0] - headPos[0] + midNode[0]) << 8) - 128);
        uint64_t yLidar =
          std::abs(((absPos[1] - headPos[1] + midNode[1]) << 8) - 128);

        uint64_t rL1 = (xLidar + yLidar) >> 1;
        uint64_t deltaAngleR = deltaAngle * rL1;
        if (deltaAngleR <= (midNode[2] << 26))
          return -1;

        // 确定 r 的逆 (1/sqrt(r2) = irsqrt(r2))
        uint64_t r2 = xLidar * xLidar + yLidar * yLidar;
        uint64_t rInv = irsqrt(r2);

        // 确定未校正数据
        int64_t zLidar = ((absPos[2] - headPos[2] + midNode[2]) << 1) - 1;
        int64_t theta = zLidar * rInv;
        int theta32 = theta >= 0 ? theta >> 15 : -((-theta) >> 15);
```

[0207]

```
// 确定激光
int laserIndex = int(child.laserIndex);
if (laserIndex == 255 || deltaAngleR <= (midNode[2] << (26 + 2))) {
  auto end = thetaLaser + numLasers - 1;
  auto it = std::upper_bound(thetaLaser + 1, end, theta32);
  if (theta32 - *std::prev(it) <= *it - theta32)
    --it;

  laserIndex = std::distance(thetaLaser, it);
  child.laserIndex = uint8_t(laserIndex);
}

// -- PHI  --
//角度
int posx = absPos[0] - headPos[0];
int posy = absPos[1] - headPos[1];
int phiNode = iatan2(posy + midNode[1], posx + midNode[0]);
int phiNode0 = iatan2(posy, posx);

// 找到预测器
int predPhi = phiBuffer[laserIndex];
if (predPhi == 0x80000000)
  predPhi = phiNode;

// 使用预测器
if (predPhi != 0x80000000) {
  // 基本移位预测器
  int Nshift =
    ((predPhi - phiNode) * phiZi.invDelta(laserIndex) + 536870912) >>
30;
  predPhi -= phiZi.delta(laserIndex) * Nshift;
```

[0208]

```
    // ctx 方位角 x 或 y
    int angleL = phiNode0 - predPhi;
    int angleR = phiNode - predPhi;
    int contextAnglePhi =
      (angleL >= 0 && angleR >= 0) || (angleL < 0 && angleR < 0) ? 2 : 0;
    angleL = std::abs(angleL);
    angleR = std::abs(angleR);
    if (angleL > angleR) {
      contextAnglePhi++;
      int temp = angleL;
      angleL = angleR;
      angleR = temp;
    }
    if (angleR > (angleL << 2))
      contextAnglePhi += 4;
```

[0209]

```
    if (std::abs(posx) <= std::abs(posy))
      *contextAnglePhiX = contextAnglePhi;
    else
      *contextAnglePhiY = contextAnglePhi;
  }

  // -- THETA --
  int thetaLaserDelta = thetaLaser[laserIndex] - theta32;
  int64_t hr = zLaser[laserIndex] * rInv;
  thetaLaserDelta += hr >= 0 ? -(hr >> 17) : ((-hr) >> 17);

<del>  int64_t zShift = (rInv << childSizeLog2[2]) >> 20; </del>
  <add>int64_t zShift = (rInv * childSize[2]) >> 20; </add>
  int thetaLaserDeltaBot = thetaLaserDelta + zShift;
```

```
    int thetaLaserDeltaTop = thetaLaserDelta - zShift;
    int contextAngle = thetaLaserDelta >= 0 ? 0 : 1;
    if (thetaLaserDeltaTop >= 0)
      contextAngle += 2;
[0210]  else if (thetaLaserDeltaBot < 0)
      contextAngle += 2;

    return contextAngle;
  }
```

[0211] 在一些示例中,G-PCC编码器200和G-PCC解码器300可以被配置为仅当缩放步长(从QP导出的)不是2的幂时才用乘法实现zShift。否则,(步长是2的幂)zShift被实现为移位操作(如上所示)。

[0212] 本公开描述了IDCM角上下文的变化。

[0213] 逆缩放函数定义如下,其中缩放操作类似于前面定义的函数,函数的参数(noClip)确定是否应用剪裁。

```
class InvQuantizer {
  int qp;
  Vec3<uint32_t> quantMasks;

public:
  InvQuantizer(int qpVal, Vec3<uint32_t> quantMaskInp)
    : qp(qpVal), quantMasks(quantMaskInp)
  {}
[0214]
  int32_t invQuantPositionComp(
    int idx, const int32_t& pos, bool noClip = false)
  {
    if (!qp)
      return pos;
    QuantizerGeom quantizer(qp);
    int shiftBits = QuantizerGeom::qpShift(qp);
    int32_t recon;
```

```
        int lowPart = pos & (quantMasks[idx] >> shiftBits);
        int highPart = pos ^ lowPart;
        int lowPartScaled = quantizer.scale(lowPart);
        if (!noClip)
          lowPartScaled = PCCClip(lowPartScaled, 0, quantMasks[idx]);
[0215]  recon = (highPart << shiftBits) + lowPartScaled;

        return recon;
      }
    };
```

[0216] 在一些示例中,recon变量可以如下导出:

[0217] recon＝(highPart<<shiftBits)|lowPartScaled;

[0218] G-PCC编码器200和G-PCC解码器300可以被配置为例如以如上所述的类似方式来确定nodePosition和激光索引。

[0219] G-PCC解码器300可以用节点的QP和适用于点位置的量化掩码来初始化缩放操作(也称为逆量化),如下所示:

[0220] <add>InvQuantizer invQuantizerIDCM(child.qp,posQuantBitMasks);</add>

[0221] 在下面的代码中,只要要估计节点相对于LIDAR原点的位置,就使用位置的经缩放值。这确保了相对于LIDAR原点的相对位置被正确地估计,并且激光索引、仰角和方位角被正确地计算。当节点位置被更新时,通过添加掩码值或节点尺寸,添加的值在添加之前也被逆量化(或者在某些情况下,该值被添加到未量化的位置,然后被缩放)。

```
        int numPoints = 1;
          bool                              numPointsGt1                    =
    _arithmeticDecoder->decode(_ctxNumIdcmPointsGt1);
          numPoints += numPointsGt1;
[0222]
          int numDuplicatePoints = 0;
          if (!geom_unique_points_flag && !numPointsGt1) {
            numDuplicatePoints
    = !_arithmeticDecoder->decode(_ctxSinglePointPerBlock);
```

```
        if (numDuplicatePoints) {
          bool singleDup =
_arithmeticDecoder->decode(_ctxSingleIdcmDupPoint);
          if (!singleDup)
            numDuplicatePoints +=
              1 + _arithmeticDecoder->decodeExpGolomb(0,
_ctxPointCountPerBlock);
        }
      }

      // 更新平面后的节点尺寸，并从平面确定位置的上部
      Vec3<int32_t> deltaPlanar{0, 0, 0};
      Vec3<int> nodeSizeLog2Rem = nodeSizeLog2;
      for (int k = 0; k < 3; k++)
        if (nodeSizeLog2Rem[k] > 0 && (planar.planarMode & (1 << k))) {
[0223]    deltaPlanar[k] |= (planar.planePosBits & (1 << k) ? 1 : 0);
          nodeSizeLog2Rem[k]--;
        }

      // 指示哪些分量是直接译码的，或使用角译码的
      // 上下文化
      Vec3<bool> directIdcm = !angularIdcm;
      point_t posNodeLidar;

      if (angularIdcm) {
        posNodeLidar =
          point_t(
            node.pos[0] << nodeSizeLog2[0], node.pos[1] << nodeSizeLog2[1],
            node.pos[2] << nodeSizeLog2[2])
          <del>- headPos; </del>
<add>   if (node.qp)
```

```
            posNodeLidar =
              invQuantPosition(node.qp, posQuantBitMasks, posNodeLidar);
            posNodeLidar -= headPos; </add>
            bool codeXorY = std::abs(posNodeLidar[0]) <=
std::abs(posNodeLidar[1]);
            directIdcm.x() = !codeXorY;
            directIdcm.y() = codeXorY;
          }

          // 对未排序的两个点进行解码
          Vec3<int32_t> deltaPos[2];
          deltaPos[0] = deltaPlanar;
          deltaPos[1] = deltaPlanar;
          if (numPoints == 2 && joint_2pt_idcm_enabled_flag)
            decodeOrdered2ptPrefix(directIdcm, nodeSizeLog2Rem, deltaPos);
```

[0224]

```
          if (angularIdcm) {
           <add> InvQuantizer invQuantizerIDCM(node.qp, posQuantBitMasks);
</add>
            for (int idx = 0; idx < 3; ++idx) {
              <add> int delta = 0; </add>
              int N = nodeSizeLog2[idx] - nodeSizeLog2Rem[idx];
              for (int mask = N ? 1 << (N - 1) : 0; mask; mask >>= 1) {
                if (deltaPos[0][idx] & mask)
                  <add> delta</add> <del>posNodeLida[idx]</del> += mask <<
nodeSizeLog2Rem[idx];
              }
              if (nodeSizeLog2Rem[idx])
                <add> delta</add> <del>posNodeLidar[idx] </del> += 1 <<
(nodeSizeLog2Rem[idx] - 1);
              <add> posNodeLidar[idx] +=</add>
```

```
                <add>   invQuantizerIDCM.invQuantPositionComp(idx, delta, true);
        </add>
            }
            node.laserIndex = findLaser(posNodeLidar, thetaLaser, numLasers);
        }

        Vec3<int32_t> pos;
        for (int i = 0; i < numPoints; i++) {
          if (angularIdcm) {
            *(outputPoints++) = pos = decodePointPositionAngular(
              nodeSizeLog2, nodeSizeLog2Rem, node, planar, headPos, zLaser,
              thetaLaser, deltaPos[i]);
          } else
            *(outputPoints++) = pos =
              decodePointPosition(nodeSizeLog2Rem, deltaPos[i]);
        }

        for (int i = 0; i < numDuplicatePoints; i++)
          *(outputPoints++) = pos;

        return numPoints + numDuplicatePoints;
```

[0225]

[0226] G-PCC编码器200和G-PCC解码器300可以被配置为确定用于角IDCM的上下文。

[0227] 与上面类似，当节点的位置相对于LIDAR头部位置被比较时，使用节点尺寸和节点位置的缩放版本。

```
        {
          Vec3<int32_t> delta = deltaPlanar;
        <del>
          Vec3<int> posXyz = {(child.pos[0] << nodeSizeLog2[0]) - headPos[0],
                              (child.pos[1] << nodeSizeLog2[1]) - headPos[1],
                              (child.pos[2] << nodeSizeLog2[2]) - headPos[2]};
```

[0228]

```
        </del>
        <add>
Vec3<int> posXyz = {
            (child.pos[0] << nodeSizeLog2[0]), (child.pos[1] << nodeSizeLog2[1]),
            (child.pos[2] << nodeSizeLog2[2])};
          Vec3<int> posXyzBeforeQuant = posXyz;
          if (child.qp)
            posXyz = invQuantPosition(child.qp, posQuantBitMasks, posXyz);
          posXyz -= headPos;
        </add>
        // -- PHI --
          // 对 x 或 y 直接译码并且计算节点的 phi
          bool codeXorY = std::abs(posXyz[0]) <= std::abs(posXyz[1]);
          if (codeXorY) {    // direct code y
            if (nodeSizeLog2AfterPlanar[1])
[0229]        for (int i = nodeSizeLog2AfterPlanar[1]; i > 0; i--) {
                delta[1] <<= 1;
                delta[1] |= _arithmeticDecoder->decode();
              }
        <del>
            posXyz[1] += delta[1];
            posXyz[0] += delta[0] << nodeSizeLog2AfterPlanar[0];
        </del>
        <add>
        posXyz[1] =
              invQuantizerIDCM.invQuantPositionComp(1, posXyzBeforeQuant[1]
+ delta[1])
              - headPos[1];
            posXyz[0] += invQuantizerIDCM.invQuantPositionComp(
              0, delta[0] << nodeSizeLog2AfterPlanar[0], true);
        </add>
```

```
} else {   // 对 x 直接译码
    if (nodeSizeLog2AfterPlanar[0])
      for (int i = nodeSizeLog2AfterPlanar[0]; i > 0; i--) {
        delta[0] <<= 1;
        delta[0] |= _arithmeticDecoder->decode();
      }
  <del>
    posXyz[0] += delta[0];
    posXyz[1] += delta[1] << nodeSizeLog2AfterPlanar[1];
  </del>
  <add>
  posXyz[0] =
      invQuantizerIDCM.invQuantPositionComp(0, posXyzBeforeQuant[0]
+ delta[0])
      - headPos[0];
```

[0230]

```
    posXyz[1] += invQuantizerIDCM.invQuantPositionComp(
      1, delta[1] << nodeSizeLog2AfterPlanar[1], true);
  </add>
  }

  // 找到预测器
  int phiNode = iatan2(posXyz[1], posXyz[0]);
  int laserNode = int(child.laserIndex);

  // 激光残差
  int laserIndex = laserNode + decodeThetaRes();

  int predPhi = _phiBuffer[laserIndex];
  if (predPhi == 0x80000000)
    predPhi = phiNode;
```

```
// 基本移位预测器
int nShift =
  ((predPhi - phiNode) * _phiZi.invDelta(laserIndex) + 536870912) >> 30;
predPhi -= _phiZi.delta(laserIndex) * nShift;

// 选择 x 或 y
int* posXY = codeXorY ? &posXyz[0] : &posXyz[1];
int idx = codeXorY ? 0 : 1;

// 对 x 或 y 进行方位角译码
int mask2 = codeXorY
  ? (nodeSizeLog2AfterPlanar[0] > 0 ? 1 << (nodeSizeLog2AfterPlanar[0]
- 1)
                                    : 0)
  : (nodeSizeLog2AfterPlanar[1] > 0 ? 1 << (nodeSizeLog2AfterPlanar[1]
[0231] - 1)
                                    : 0);
for (; mask2; mask2 >>= 1) {
  // 角度左移和右移
<del>
  int phiR = codeXorY ? iatan2(posXyz[1], posXyz[0] + mask2)
                      : iatan2(posXyz[1] + mask2, posXyz[0]);
</del>
<add>
int32_t scaledMask =
    invQuantizerIDCM.invQuantPositionComp(codeXorY ? 0 : 1, mask2,
true);
  int phiR = codeXorY ? iatan2(posXyz[1], posXyz[0] + scaledMask)
                      : iatan2(posXyz[1] + scaledMask, posXyz[0]);
</add>
  int phiL = phiNode;
```

```
// ctx 方位角
int angleL = phiL - predPhi;
int angleR = phiR - predPhi;
int contextAnglePhi =
  (angleL >= 0 && angleR >= 0) || (angleL < 0 && angleR < 0) ? 2 : 0;
angleL = std::abs(angleL);
angleR = std::abs(angleR);
if (angleL > angleR) {
  contextAnglePhi++;
  int temp = angleL;
  angleL = angleR;
  angleR = temp;
}
if (angleR > (angleL << 1))
```

[0232]

```
  contextAnglePhi += 4;

// 熵译码
bool bit = _arithmeticDecoder->decode(
  _ctxPlanarPlaneLastIndexAngularPhiIDCM[contextAnglePhi]);
delta[idx] <<= 1;
if (bit) {
  delta[idx] |= 1;
  <del>*posXY += mask2; </del>
  <add>  *posXY += invQuantizerIDCM.invQuantPositionComp(idx,
mask2, true); </add>
  phiNode = phiR;
  predPhi = _phiBuffer[laserIndex];
  if (predPhi == 0x80000000)
    predPhi = phiNode;
```

```
                    // 基本移位预测器
                    int nShift =
                        ((predPhi - phiNode) * _phiZi.invDelta(laserIndex) + 536870912) >>
        30;
                    predPhi -= _phiZi.delta(laserIndex) * nShift;
                  }
                }

                // 更新缓冲 phi
                _phiBuffer[laserIndex] = phiNode;

                // -- THETA --
                int maskz =
                  nodeSizeLog2AfterPlanar[2] > 0 ? 1 << (nodeSizeLog2AfterPlanar[2] -
        1) : 0;
[0233]          if (!maskz)
                  return delta;

                int posz0 = posXyz[2];
              <del>   posXyz[2] += delta[2] << nodeSizeLog2AfterPlanar[2];</del>
              <add>
            posXyz[2] += invQuantizerIDCM.invQuantPositionComp(
                    2, delta[2] << nodeSizeLog2AfterPlanar[2], true);
              </add>

                // 由于 x 和 y 是已知的
                // r 也是已知的并且不依赖于 z 的比特
                uint64_t xLidar = (int64_t(posXyz[0]) << 8) - 128;
                uint64_t yLidar = (int64_t(posXyz[1]) << 8) - 128;
                uint64_t r2 = xLidar * xLidar + yLidar * yLidar;
                int64_t rInv = irsqrt(r2);
```

```
// 使用角模式对 z 的比特进行译码。资格是隐式的。激光已知。
int64_t hr = zLaser[laserIndex] * rInv;
int fixedThetaLaser =
  thetaLaser[laserIndex] + int(hr >= 0 ? -(hr >> 17) : ((-hr) >> 17));

<del>  int zShift = (rInv << nodeSizeLog2AfterPlanar[2]) >> 18; </del>
<add>
int zShift = (child.qp ? (
                    rInv
                    * invQuantizerIDCM.invQuantPositionComp(
                      2, 1 << nodeSizeLog2AfterPlanar[2], true))
                        : (rInv << nodeSizeLog2AfterPlanar[2]))
  >> 18;
</add>
for (int bitIdxZ = nodeSizeLog2AfterPlanar[2]; bitIdxZ > 0;
```

[0234]

```
      bitIdxZ--, maskz >>= 1, zShift >>= 1) {
  // 确定未校正 theta
<del>  int64_t zLidar = ((posXyz[2] + maskz) << 1) - 1; </del>
<add>  int scaledMaskZ = invQuantizerIDCM.invQuantPositionComp(2,
maskz, true); </add>
  <add> int64_t zLidar = ((posXyz[2] + scaledMaskZ) << 1) - 1; </add>
  int64_t theta = zLidar * rInv;
  int theta32 = theta >= 0 ? theta >> 15 : -((-theta) >> 15);
  int thetaLaserDelta = fixedThetaLaser - theta32;

  int thetaLaserDeltaBot = thetaLaserDelta + zShift;
  int thetaLaserDeltaTop = thetaLaserDelta - zShift;
  int contextAngle = thetaLaserDelta >= 0 ? 0 : 1;
  if (thetaLaserDeltaTop >= 0)
    contextAngle += 2;
```

```
        else if (thetaLaserDeltaBot < 0)
          contextAngle += 2;

        delta[2] <<= 1;
        delta[2] |= _arithmeticDecoder->decode(
          _ctxPlanarPlaneLastIndexAngularIdcm[contextAngle]);
```

[0235]
```
<del>posXyz[2] = posz0 + (delta[2] << (bitIdxZ - 1)); </del>
<add>  if (delta[2] & 1) </add>
        <add> posXyz[2] += scaledMaskZ; </add>
    }

    return delta;
  }
```

[0236] 推导的其余部分与上面描述的类似。

[0237] 以下示例示出了一种实现，其中平面角模式下z坐标的上下文阈值被更新(更新后的部分在**和**之间标识)，如下所示：

[0238]
```
Vec3<int64_t> childSize = {
    1 << childSizeLog2[0], 1 << childSizeLog2[1], 1 << childSizeLog2[2]};
  if (child.qp) {
    absPos = invQuantPositionAngular(child.qp, quantMasks, absPos);
    midNode = invQuantPositionAngular(child.qp, quantMasks, midNode);
    childSize = invQuantPositionAngular(child.qp, quantMasks, childSize);
    **childSize[2] = 2*childSize[2];**
  }
```

[0239] 在本示例中，上述示例中示出的其余更改也可以保留。

[0240] 本公开的各个方面中的示例可以单独使用或以任何组合使用。

[0241] 图9A是示出根据本公开的一种或多种技术的G-PCC编码器200的示例操作的流程图。G-PCC编码器200确定针对节点启用了树内量化(900)。G-PCC编码器200确定针对节点激活平面模式(902)。响应于针对节点启用树内量化，G-PCC编码器200为节点确定表示相对于原点位置的坐标位置的量化值(904)。G-PCC编码器200在不剪裁的情况下对量化值进行缩放，以确定表示相对于原点位置的坐标位置的经缩放值(906)。

[0242] 为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值，G-PCC编码器200可以被配置为：确定最高有效位(MSB)群组和最低有效位

(LSB)群组;在不剪裁的情况下对LSB进行缩放以确定缩放后的LSB;以及将缩放后的LSB添加到MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值,G-PCC编码器200可以被配置为:基于节点的量化参数确定MSB的移位量;基于移位量对MSB进行移位;以及将缩放后的LSB添加到移位后的MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。G-PCC编码器200可以例如在比特流中发信号通知的语法中接收LSB群组中的比特数量的指示。该指示可以是显式指示或隐式指示。作为一个示例,基于接收到的节点深度,G-PCC编码器200可以被配置为导出MSB和LSB的数量。

[0243] G-PCC编码器200基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文编码的上下文(908)。平面位置语法元素可以例如指示垂直面位置。为了基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文编码的上下文,G-PCC编码器200可以被配置为:基于经缩放值和原点位置确定一个或多个激光特性;以及基于激光特性对角模式的平面位置语法元素进行编码。为了基于激光特性来确定用于对角模式的平面位置语法元素进行上下文编码的上下文,G-PCC编码器200可以被配置为基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间还是低于第三距离阈值来确定上下文索引。

[0244] 例如,上面的步骤904、906和908可以作为G-PCC编码器200执行的解码操作的一部分来执行。G-PCC编码器200可以执行解码作为编码的一部分。例如,为了确定特定编码方案是否提供期望的速率-失真折衷,G-PCC编码器200可以对点云数据进行编码,然后对编码的点云数据进行解码,使得解码的点云数据可以与原始点云数据进行比较,以确定失真量。G-PCC编码器200还可以对点云数据进行编码,然后对编码后的点云数据进行解码,使得当使用各种预测译码工具时,G-PCC编码器200可以基于G-PCC解码器可用的相同点云数据来执行预测。

[0245] 图9B是示出根据本公开的一种或多种技术的G-PCC解码器300的示例操作的流程图。G-PCC解码器300可以基于在比特流中发信号通知的语法,确定针对节点启用了树内量化(920)。G-PCC解码300对于节点,基于在比特流中发信号通知的语法确定针对该节点激活平面模式(922)。响应于为节点启用树内量化,G-PCC解码器300为节点确定表示相对于原点位置的坐标位置的量化值(924)。

[0246] G-PCC解码器300在不剪裁的情况下对量化值进行缩放,以确定表示相对于原点位置的坐标位置的经缩放值(906)。

[0247] 为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值,G-PCC解码器300可以被配置为:确定最高有效位(MSB)群组和最低有效位(LSB)群组;在不剪裁的情况下对LSB进行缩放以确定缩放后的LSB;以及将缩放后的LSB添加到MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值,G-PCC解码器300可以被配置为:基于节点的量化参数确定MSB的移位量;基于移位量对MSB进行移位;以及将缩放后的LSB添加到移位后的MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。G-PCC解码器300可以例如在比特流中发信号通知的语法中接收LSB群组中的比特数量的指示。该

指示可以是显式指示或隐式指示。作为一个示例，基于接收到的节点深度，G-PCC解码器300可以被配置为导出MSB和LSB的数量。

[0248] G-PCC解码器300基于表示相对于原点位置的坐标位置的经缩放值，确定用于对角模式的平面位置语法元素进行上下文解码(例如，算术解码)的上下文(928)。平面位置语法元素可以例如指示垂直面位置。为了基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文解码的上下文，G-PCC解码器300可以被配置为：基于经缩放值和原点位置确定一个或多个激光特性；以及基于激光特性对角模式的平面位置语法元素进行解码。为了基于激光特性来确定用于对角模式的平面位置语法元素进行上下文解码的上下文，G-PCC解码器300可以被配置为基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间还是低于第三距离阈值来确定上下文索引。

[0249] G-PCC解码器300可以被配置为通过例如基于平面位置确定点云的一个或多个点的位置来确重建点云。

[0250] 图10是示出可以与本公开的一种或多种技术一起使用的示例测距系统1000的概念图。在图10的示例中，测距系统1000包括照明器1002和传感器1004。照明器1002可以发出光1006。在一些示例中，照明器1002可以发出作为一个或多个激光束的光1006。光1006可以是一个或多个波长，诸如红外波长或可见光波长。在其他示例中，光1006不是相干的激光。当光1006遇到诸如对象1008的对象时，光1006产生返回光1010。返回光1010可以包括后向散射和/或反射光。返回光1010可以通过透镜1011，透镜1411引导返回光1010以在传感器1004上创建对象1008的图像1012。传感器1004基于图像1012生成信号1018。图像1012可以包括点集(例如，如由图10的图像1012中的点表示)。

[0251] 在一些示例中，照明器1002和传感器1004可以安装在旋转结构上，使得照明器1002和传感器1004捕获环境的360度视图。在其他示例中，测距系统1000可以包括一个或多个光学组件(例如，反射镜、准直器、衍射光栅等)，其使照明器1002和传感器1004能够检测特定范围内(例如，高达360度)的对象。尽管图10的示例仅示出单个照明器1002和传感器1004，但测距系统1000可以包括多组照明器和传感器。

[0252] 在一些示例中，照明器1002生成结构光图案。在这些示例中，测距系统1000可以包括多个传感器1004，在这些传感器上形成结构光图案的相应图像。测距系统1000可以使用结构光图案的图像之间的差异来确定到对象1008的距离，结构光图案是从该对象反向散射的。当对象1008相对靠近传感器1004(例如0.2米至2米)时，基于结构光的测距系统可以具有高水平的精度(例如，在亚毫米范围内的精度)。这种高精度水平可以用于面部识别应用，诸如解锁移动设备(例如，移动电话、平板计算机等)以及用于安全应用。

[0253] 在一些示例中，测距系统1000是基于飞行时间(ToF)的系统。在测距系统1000是基于TOF的系统的一些示例中，照明器1002生成光脉冲。换句话说，照明器1002可以调制发射光1006的振幅。在这些示例中，传感器1004从由照明器1002生成的光脉冲1006中检测返回光1010。然后，测距系统1000可以基于当光1006被发射和检测到之间的延迟以及空气中已知的光速来确定到光1006从其反向散射的对象1008的距离。在一些示例中，照明器1002可以调制发射光1006的相位，而代替于(或者附加于)调制发射光1006的振幅。在这些示例中，传感器1004可以检测来自对象1008的返回光1010的相位，并且使用光速并基于当照明器

1002在特定相位处生成光1006时与当传感器1004在特定相位处检测到返回光1010时之间的时间差,来确定到对象1008上的点的距离。

[0254] 在其他示例中,可以在不使用照明器1002的情况下生成点云。例如,在一些示例中,测距系统1000的传感器1004可以包括两个或更多个光学相机。在这样的示例中,测距系统1000可以使用光学相机来捕获环境的立体图像(包括对象1008)。测距系统1000(例如,点云生成器1020)可以然后计算立体图像中位置之间的差异。然后,测距系统1000可以使用该差异来确定到立体图像中所示位置的距离。根据这些距离,点云生成器1020可以生成点云。

[0255] 传感器1004还可以检测对象1008的其他属性,诸如颜色和反射率信息。在图10的示例中,点云生成器1020可以基于由传感器1004生成的信号1018来生成点云。测距系统1000和/或点云发生器1020可以形成数据源104(图1)的一部分。

[0256] 图11是示出其中可以使用本公开的一种或多种技术的示例基于车辆的场景的概念图。在图11的示例中,车辆1100包括激光封装1102,诸如LIDAR系统。尽管未在图11的示例中示出,但车辆1100还可以包括数据源和G-PCC编码器(诸如G-PCC编码器200(图1))。在图11的示例中,激光封装1102发出激光束1104,激光束1504从行人1106或道路中的其他对象反射。车辆1100的数据源可以基于由激光封装1102生成的信号来生成点云。车辆1100的G-PCC编码器可以对点云进行编码以生成比特流1108。比特流1108可以包括比G-PCC编码器获得的未编码点云少得多的比特。车辆1100的输出接口(例如,输出接口108(图1))可以将比特流1108发送到一个或多个其他设备。因此,车辆1100能够比未编码点云数据更快地将比特流1108发送到其他设备。附加地,比特流1108可能需要较少的数据存储容量。

[0257] 在图11的示例中,车辆1100可以向另一车辆1110发送比特流1108。车辆1110可以包括G-PCC解码器(诸如G-PCC解码器300(图1))。车辆1110的G-PCC解码器可以对比特流1108进行解码以重建点云。车辆1110可以将重建的点云用于各种目的。例如,车辆1110可以基于重建的点云来确定行人1106在车辆1100之前的道路中,并且因此例如在车辆1110的驾驶员意识到行人1106在道路中之前就开始减速。因此,在一些示例中,车辆1110可以基于重建的点云来执行自主导航操作、生成通知或警告、或执行另一动作。

[0258] 附加地或可替代地,车辆1100可以向服务器系统1112发送比特流1108。服务器系统1112可将比特流1108用于各种目的。例如,服务器系统1112可以存储比特流1108,用于点云的后续重建。在该示例中,服务器系统1112可以使用点云以及其他数据(例如,由车辆1100生成的车辆遥测数据)来训练自主驾驶系统。在其他示例中,服务器系统1112可以存储比特流1108,用于随后法医碰撞调查的重建(例如,如果车辆1100与行人碰撞1106),或者可以向车辆1100或车辆1110发送用于导航的通知或指令。

[0259] 图12是示出其中可以使用本公开的一种或多种技术的示例扩展现实系统的概念图。扩展现实(XR)是一个术语,用于涵盖一系列技术,包括增强现实(AR)、混合现实(MR)和虚拟现实(VR)。在图12的示例中,第一用户1200位于第一位置1202。用户1200佩戴XR头戴式设备(headset)1204。作为XR头戴式设备1204的替代方案,用户1200可以使用移动设备(例如,移动电话、平板计算机等)。XR头戴式设备1204包括深度检测传感器(诸如LIDAR系统),其检测在第一位置1202处的对象1206上的点的位置。XR头戴式设备1204的数据源可以使用由深度检测传感器生成的信号来生成位置1202处的对象1206的点云表示。XR头戴式设备1204可以包括G-PCC编码器(例如,图1的G-PCC编码器200),其被配置为对点云进行编码以

生成比特流1208。

[0260] XR头戴式设备1204可以向第二位置1214处的用户1212佩戴的XR头戴式设备1210发送比特流1208(例如,经由诸如因特网的网络)。XR头戴式设备1210可以对比特流1208进行解码以重建点云。XR头戴式设备1210可以使用点云来生成表示位置1202处的对象1206的XR可视化(例如,AR、MR、VR可视化)。因此,在一些示例中,例如当XR头戴式设备1210生成VR可视化时,位置1214处的用户1212可以具有位置1202的3D沉浸式体验。在一些示例中,XR头戴式设备1210可以基于重建的点云来确定虚拟对象的位置。例如,XR头戴式设备1210可以基于重建的点云来确定环境(例如,位置1202)包括平坦表面,然后确定虚拟对象(例如,卡通人物)将被定位在该平坦表面上。XR头戴式设备1210可以生成XR可视化,其中虚拟对象位于所确定的位置。例如,XR头戴式设备1210可以示出坐在平坦表面上的卡通人物。

[0261] 图13是示出其中可以使用本公开的一种或多种技术的示例移动设备系统的概念图。在图13的示例中,诸如移动电话或平板计算机的移动设备1300包括诸如LIDAR系统的深度检测传感器,其检测移动设备1300的环境中的对象1302上的点的位置。移动设备1300的数据源可以使用由深度检测传感器生成的信号来生成对象1302的点云表示。移动设备1300可以包括G-PCC编码器(例如,图1的G-PCC编码器200),其被配置为对点云进行编码以生成比特流1304。在图13的示例中,移动设备1300可以向远程设备1306(诸如服务器系统或其他移动设备)发送比特流。远程设备1306可以对比特流1304进行解码以重建点云。远程设备1306可以将点云用于各种目的。例如,远程设备1306可以使用点云来生成移动设备1300的环境的地图。例如,远程设备1306可以基于重建的点云来生成建筑物内部的地图。在另一示例中,远程设备1306可以基于点云来生成图像(例如,计算机图形)。例如,远程设备1306可以使用点云的点作为多边形的顶点,并且使用点的颜色属性作为对多边形着色的基础。在一些示例中,远程设备1306可以使用点云来执行面部识别。

[0262] 以下编号的条款示出了本公开中描述的设备和技术的一个或多个方面。

[0263] 条款1A.一种处理点云数据的方法,该方法包括:接收表示点云的数据;以及根据本公开的任何一种或多种技术处理表示点云的数据以生成点云。

[0264] 条款2A.一种用于处理点云的设备,该设备包括用于接收表示点云的数据并且根据本公开的任何一种或多种技术处理表示点云的数据以生成点云的一个或多个部件。

[0265] 条款3A.如条款1A的方法或者条款2A的设备,其中,根据本公开的任何一种或多种技术处理表示所述点云的数据以生成点云包括完全或部分禁用角模式。

[0266] 条款4A.根据条款3A的设备,其中一个或多个部件包括在电路中实现的一个或多个处理器。

[0267] 条款5A.根据条款2A-条款4A的任一项的设备,还包括用于存储表示点云的数据的存储器。

[0268] 条款6A.根据条款2A-条款5A的任一项的设备,其中设备包括解码器。

[0269] 条款7A.根据条款2A-条款5A的任一项的设备,其中设备包括编码器。

[0270] 条款8A.根据条款2A-条款7A的任一项的设备,还包括生成点云的设备。

[0271] 条款9A.根据条款2A-条款8A的任一项的设备,还包括用于基于点云呈现图像的显示器。

[0272] 条款10A.一种计算机可读存储介质,其上存储有指令,当指令被执行时使一个或

多个处理器接收表示点云的数据并且根据本公开的任何一种或多种技术处理表示点云的数据以生成点云。

[0273] 条款1B.一种用于对包括点云数据的比特流进行解码的设备,该设备包括:存储器,用于存储点云数据;以及一个或多个处理器,耦合到存储器并实现在电路中,该一个或多个处理器被配置为:基于在比特流中发信号通知的语法,确定为节点启用树内量化;对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文。

[0274] 条款2B.根据条款1B的设备,其中,为了基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文解码的上下文,该一个或多个处理器被配置为:基于经缩放值和原点位置确定一个或多个激光特性;以及基于激光特性对角模式的平面位置语法元素进行解码。

[0275] 条款3B.根据条款2B的设备,其中,为了基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文解码的上下文,一个或多个处理器还被配置为基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间还是低于第三距离阈值来确定上下文索引。

[0276] 条款4B.根据条款2B的设备,其中,平面位置语法元素指示垂直面位置。

[0277] 条款5B.根据条款2B的设备,其中,一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

[0278] 条款6B.根据条款1B的设备,其中,一个或多个处理器还被配置为使用由所确定的上下文指示的上下文对角模式的平面位置进行算术解码。

[0279] 条款7B.根据条款1B的设备,其中为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值,一个或多个处理器被配置为:确定最高有效位(MSB)群组和最低有效位(LSB)群组;在不剪裁的情况下对LSB进行缩放以确定缩放后的LSB;以及将缩放后的LSB添加到MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0280] 条款8B.根据条款7B的设备,其中为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值,一个或多个处理器被配置为:基于节点的量化参数确定MSB的移位量;基于移位量对MSB进行移位;以及将缩放后的LSB添加到移位后的MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0281] 条款9B.根据条款6B的设备,其中,一个或多个处理器还被配置为在比特流中发信号通知的语法中接收LSB群组中的比特数量的指示。

[0282] 条款10B.根据条款1B的设备,其中,一个或多个处理器还被配置为重建点云。

[0283] 条款11B.根据条款10B的设备,并且其中,一个或多个处理器被配置为作为重建点云的一部分,基于平面位置确定点云的一个或多个点的位置。

[0284] 条款12B.根据条款11B的设备,其中,一个或多个处理器还被配置为基于重建点云生成建筑物内部的地图。

[0285] 条款13B.根据条款11B的设备,其中,一个或多个处理器还被配置为基于重建点云执行自主导航操作。

[0286] 条款14B.根据条款11B的设备,其中,一个或多个处理器还被配置为基于重建点云生成计算机图形。

[0287] 条款15B.根据条款11B的设备,其中,一个或多个处理器被配置为:基于重建的点云来确定虚拟对象的位置;以及生成其中虚拟对象处于所确定的位置的扩展现实(XR)可视化。

[0288] 条款16B.根据条款11B的设备,还包括用于基于重建点云呈现图像的显示器。

[0289] 条款17B.根据条款1B的设备,其中,该设备是移动电话或平板计算机之一。

[0290] 条款18B.根据条款1B的设备,其中,设备是车辆。

[0291] 条款19B.根据条款1B的设备,其中,设备是扩展现实设备。

[0292] 条款20B.一种用于对包括点云数据的比特流进行解码的方法,该方法包括:基于在比特流中发信号通知的语法,确定为节点启用树内量化;对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文。

[0293] 条款21B.根据条款20B的方法,其中基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文解码的上下文包括:基于经缩放值和原点位置确定一个或多个激光特性;以及基于激光特性对角模式的平面位置语法元素进行解码。

[0294] 条款22B.根据条款21B的方法,其中,基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文解码的上下文还包括基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间还是低于第三距离阈值来确定上下文索引。

[0295] 条款23B.根据条款21B的方法,其中,平面位置语法元素指示垂直面位置。

[0296] 条款24B.根据条款21B的方法,其中,一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

[0297] 条款25B.根据条款20B的方法,还包括:使用由所确定的上下文指示的上下文对角模式的平面位置进行算术解码。

[0298] 条款26B.根据条款20B的方法,其中,在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值包括:确定最高有效位(MSB)群组和最低有效位(LSB)群组;在不剪裁的情况下对LSB进行缩放以确定缩放后的LSB;以及将缩放后的LSB添加到MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0299] 条款27B.根据条款26B的方法,其中,在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值包括:基于节点的量化参数确定MSB的移位量;基于移位量对MSB进行移位;以及将缩放后的LSB添加到移位后的MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0300] 条款28B.根据条款25B的方法,还包括:在比特流中发信号通知的语法中接收LSB群组中的比特数量的指示。

[0301] 条款29B.根据条款20B的设备,还包括:重建点云。

[0302] 条款30B.根据条款29B的方法,其中,重建点云包括基于平面位置确定点云的一个或多个点的位置。

[0303] 条款31B.根据条款29B的方法,还包括:基于重建点云生成建筑物内部的地图。

[0304] 条款32B.根据条款29B的方法,还包括:基于重建点云执行自主导航操作。

[0305] 条款33B.根据条款29B的方法,还包括基于重建点云生成计算机图形。

[0306] 条款34B.根据条款29B的方法,还包括:基于重建的点云来确定虚拟对象的位置;以及生成其中虚拟对象处于所确定的位置的扩展现实(XR)可视化。

[0307] 条款35B.一种用于对包括点云数据的比特流进行编码的设备,该设备包括:存储器,用于存储点云数据;以及一个或多个处理器,耦合到存储器并实现在电路中,该一个或多个处理器被配置为:确定为节点启用树内量化;确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文编码的上下文。

[0308] 条款36B.根据条款35B的设备,其中,为了基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文编码的上下文,该一个或多个处理器被配置为:基于经缩放值和原点位置确定一个或多个激光特性;以及基于激光特性对角模式的平面位置语法元素进行解码。

[0309] 条款37B.根据条款36B的设备,其中,为了基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文编码的上下文,一个或多个处理器还被配置为基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间还是低于第三距离阈值来确定上下文索引。

[0310] 条款38B.根据条款36B的设备,其中,平面位置语法元素指示垂直面位置。

[0311] 条款39B.根据条款36B的设备,其中,一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

[0312] 条款40B.根据条款35B的设备,其中,一个或多个处理器还被配置为使用由所确定的上下文指示的上下文对角模式的平面位置进行算术编码。

[0313] 条款41B.根据条款35B的设备,其中为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值,一个或多个处理器被配置为:确定最高有效位(MSB)群组和最低有效位(LSB)群组;在不剪裁的情况下对LSB进行缩放以确定缩放后的LSB;以及将缩放后的LSB添加到MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0314] 条款42B.根据条款41B的设备,其中为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值,一个或多个处理器被配置为:基于节点的量化参数确定MSB的移位量;基于移位量对MSB进行移位;以及将缩放后的LSB添加到移位

后的MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0315] 条款43B.根据条款35B的设备,其中,一个或多个处理器还被配置为重建点云。

[0316] 条款44B.根据条款35B的设备,其中,为了重建点云,一个或多个处理器还被配置为基于平面位置确定点云的一个或多个点的位置。

[0317] 条款45B.根据条款35B的设备,其中,该设备是移动电话或平板计算机之一。

[0318] 条款46B.根据条款35B的设备,其中,设备是车辆。

[0319] 条款47B.根据条款35B的设备,其中,设备是扩展现实设备。

[0320] 条款48B.一种用于对包括点云数据的比特流进行编码的方法,该方法包括:确定为节点启用树内量化;确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文编码的上下文。

[0321] 条款49B.根据条款48B的方法,其中基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文编码的上下文包括:基于经缩放值和原点位置确定一个或多个激光特性;以及基于激光特性对角模式的平面位置语法元素进行解码。

[0322] 条款50B.根据条款49B的方法,其中,基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文编码的上下文还包括基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间还是低于第三距离阈值来确定上下文索引。

[0323] 条款51B.根据条款49B的方法,其中,平面位置语法元素指示垂直面位置。

[0324] 条款52B.根据条款49B的方法,其中,一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

[0325] 条款53B.根据条款48B的方法,还包括:使用由所确定的上下文指示的上下文对角模式的平面位置进行算术编码。

[0326] 条款54B.根据条款48B的方法,其中,在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值包括:确定最高有效位(MSB)群组和最低有效位(LSB)群组;在不剪裁的情况下对LSB进行缩放以确定缩放后的LSB;以及将缩放后的LSB添加到MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0327] 条款55B.根据条款54B的方法,在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值包括:基于节点的量化参数确定MSB的移位量;基于移位量对MSB进行移位;以及将缩放后的LSB添加到移位后的MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0328] 条款56B.根据条款48B的方法,还包括:重建点云。

[0329] 条款57B.根据条款56B的方法,其中,重建点云包括基于平面位置确定点云的一个或多个点的位置。

[0330] 条款58B.一种存储指令的计算机可读存储介质,该指令在由一个或多个处理器执行时使得一个或多个处理器:基于在比特流中发信号通知的语法,确定为节点启用树内量

化;对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文。

[0331] 条款59B.一种用于对包括点云数据的比特流进行解码的设备,该设备包括:用于基于在比特流中发信号通知的语法,确定为节点启用树内量化的部件;用于对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式的部件;用于响应于对节点启用树内量化来为节点确定表示相对于原点位置的坐标位置的量化值的部件;用于在不剪裁的情况下对量化值进行缩放,以确定表示相对于原点位置的坐标位置的经缩放值的部件;以及用于基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文的部件。

[0332] 条款1C.一种用于对包括点云数据的比特流进行解码的设备,该设备包括:存储器,用于存储点云数据;以及一个或多个处理器,耦合到存储器并实现在电路中,该一个或多个处理器被配置为:基于在比特流中发信号通知的语法,确定为节点启用树内量化;对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文。

[0333] 条款2C.根据条款1C的设备,其中,为了基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文解码的上下文,该一个或多个处理器被配置为:基于经缩放值和原点位置确定一个或多个激光特性;以及基于激光特性对角模式的平面位置语法元素进行解码。

[0334] 条款3C.根据条款2C的设备,其中,为了基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文解码的上下文,一个或多个处理器还被配置为基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间还是低于第三距离阈值来确定上下文索引。

[0335] 条款4C.根据条款1C-条款3C的任一项的设备,其中,平面位置语法元素指示垂直面位置。

[0336] 条款5C.根据条款2C-条款4C的任一项的设备,其中,一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

[0337] 条款6C.根据条款1C-条款5C的任一项的设备,其中,一个或多个处理器还被配置为使用由所确定的上下文指示的上下文对角模式的平面位置进行算术解码。

[0338] 条款7C.根据条款1C-条款6C的任一项的设备,其中,为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值,一个或多个处理器被配置为:确定最高有效位(MSB)群组和最低有效位(LSB)群组;在不剪裁的情况下对LSB进行缩放以确定缩放后的LSB;以及将缩放后的LSB添加到MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0339] 条款8C.根据条款7C的设备,其中,为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值,一个或多个处理器被配置为:基于节点的量化参数确定MSB的移位量;基于移位量对MSB进行移位;以及将缩放后的LSB添加到移位后的MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0340] 条款9C.根据条款6C-条款8C的任一项的设备,其中,一个或多个处理器还被配置为在比特流中发信号通知的语法中接收LSB群组中的比特数量的指示。

[0341] 条款10C.根据条款1C-条9C的任一项的设备,其中,一个或多个处理器还被配置为重建点云。

[0342] 条款11C.根据条款10C的设备,并且其中,一个或多个处理器被配置为作为重建点云的一部分,基于平面位置确定点云的一个或多个点的位置。

[0343] 条款12C.根据条款11C的设备,其中,一个或多个处理器还被配置为基于重建点云生成建筑物内部的地图。

[0344] 条款13C.根据条款11C的设备,其中,一个或多个处理器还被配置为基于重建点云执行自主导航操作。

[0345] 条款14C.根据条款11C的设备,其中,一个或多个处理器还被配置为基于重建点云生成计算机图形。

[0346] 条款15C.根据条款11C的设备,其中,一个或多个处理器被配置为:基于重建的点云来确定虚拟对象的位置;以及生成其中虚拟对象处于所确定的位置的扩展现实(XR)可视化。

[0347] 条款16C.根据条款11C-条款15C的任一项的设备,还包括用于基于重建点云呈现图像的显示器。

[0348] 条款17C.根据条款1C-条款16C的任一项的设备,其中,该设备是移动电话或平板计算机之一。

[0349] 条款18C.根据条款1C-条款16C的任一项的设备,其中,设备是车辆。

[0350] 条款19C.根据条款1C-条款16C的任一项的设备,其中,设备是扩展现实设备。

[0351] 条款20C.一种用于对包括点云数据的比特流进行解码的方法,该方法包括:基于在比特流中发信号通知的语法,确定为节点启用树内量化;对于节点,基于在比特流中发信号通知的语法确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文解码的上下文。

[0352] 条款21C.根据条款20C的方法,其中,基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文解码的上下文包括:基于经缩放值和原点位置确定一个或多个激光特性;以及基于激光特性对角模式的平面位置语法元素进行解码。

[0353] 条款22C.根据条款21C的方法,其中,基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文解码的上下文还包括基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间还是低于第三距离阈值来确定上下文索

引。

[0354] 条款23C.根据条款20C-条款22C的任一项的方法,其中,平面位置语法元素指示垂直面位置。

[0355] 条款24C.根据条款21C-条款23C的任一项的方法,其中,一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

[0356] 条款25C.根据条款20C-条款24C的任一项的方法,还包括:使用由所确定的上下文指示的上下文对角模式的平面位置进行算术解码。

[0357] 条款26C.根据条款20C-条款25C的任一项的方法,其中,在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值包括:确定最高有效位(MSB)群组和最低有效位(LSB)群组;在不剪裁的情况下对LSB进行缩放以确定缩放后的LSB;以及将缩放后的LSB添加到MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0358] 条款27C.根据条款26C的方法,其中,在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值包括:基于节点的量化参数确定MSB的移位量;基于移位量对MSB进行移位;以及将缩放后的LSB添加到移位后的MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0359] 条款28C.根据条款25C-条款27C的任一项的方法,还包括:在比特流中发信号通知的语法中接收LSB群组中的比特数量的指示。

[0360] 条款29C.根据条款20C-条款28C的任一项的方法,还包括:重建点云。

[0361] 条款30C.根据条款29C的方法,其中,重建点云包括基于平面位置确定点云的一个或多个点的位置。

[0362] 条款31C.根据条款29C的方法,还包括:基于重建点云生成建筑物内部的地图。

[0363] 条款32C.根据条款29C的方法,还包括:基于重建点云执行自主导航操作。

[0364] 条款33C.根据条款29C的方法,还包括基于重建点云生成计算机图形。

[0365] 条款34C.根据条款29C的方法,还包括:基于重建的点云来确定虚拟对象的位置;以及生成其中虚拟对象处于所确定的位置的扩展现实(XR)可视化。

[0366] 条款35C.一种用于对包括点云数据的比特流进行编码的设备,该设备包括:存储器,用于存储点云数据;以及一个或多个处理器,耦合到存储器并实现在电路中,该一个或多个处理器被配置为:确定为节点启用树内量化;确定为节点激活角模式;响应于为节点启用树内量化,为节点确定表示相对于原点位置的坐标位置的量化值;在不剪裁的情况下缩放量化值,以确定表示相对于原点位置的坐标位置的经缩放值;以及基于表示相对于原点位置的坐标位置的经缩放值,确定用于对角模式的平面位置语法元素进行上下文编码的上下文。

[0367] 条款36C.根据条款35C的设备,其中,为了基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文编码的上下文,该一个或多个处理器被配置为:基于经缩放值和原点位置确定一个或多个激光特性;以及基于激光特性对角模式的平面位置语法元素进行解码。

[0368] 条款37C.根据条款36C的设备,其中,为了基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文编码的上下文,一个或多个

处理器还被配置为基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间还是低于第三距离阈值来确定上下文索引。

[0369] 条款38C.根据条款35C-条款37C的任一项的设备，其中，平面位置语法元素指示垂直面位置。

[0370] 条款39C.根据条款36C-条款38C的任一项的设备，其中，一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

[0371] 条款40C.根据条款35C-条款39C的任一项的设备，其中，一个或多个处理器还被配置为使用由所确定的上下文指示的上下文对角模式的平面位置进行算术编码。

[0372] 条款41C.根据条款35C-条款40C的任一项的设备，其中，为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值，一个或多个处理器被配置为：确定最高有效位(MSB)群组和最低有效位(LSB)群组；在不剪裁的情况下对LSB进行缩放以确定缩放后的LSB；以及将缩放后的LSB添加到MSB中，以确定表示相对于原点位置的坐标位置的经缩放值。

[0373] 条款42C.根据条款41C的设备，其中为了在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值，一个或多个处理器被配置为：基于节点的量化参数确定MSB的移位量；基于移位量对MSB进行移位；以及将缩放后的LSB添加到移位后的MSB中，以确定表示相对于原点位置的坐标位置的经缩放值。

[0374] 条款43C.根据条款35C-条42C的任一项的设备，其中，一个或多个处理器还被配置为重建点云。

[0375] 条款44C.根据条款35C-条款43C的任一项的设备，其中，为了重建点云，一个或多个处理器还被配置为基于平面位置确定点云的一个或多个点的位置。

[0376] 条款45C.根据条款35C-条款44C的任一项的设备，其中，该设备是移动电话或平板计算机之一。

[0377] 条款46C.根据条款35C-条款44C的任一项的设备，其中，设备是车辆。

[0378] 条款47C.根据条款35C-条款44C的任一项的设备，其中，设备是扩展现实设备。

[0379] 条款48C.一种用于对包括点云数据的比特流进行编码的方法，该方法包括：确定为节点启用树内量化；确定为节点激活角模式；响应于为节点启用树内量化，为节点确定表示相对于原点位置的坐标位置的量化值；在不剪裁的情况下缩放量化值，以确定表示相对于原点位置的坐标位置的经缩放值；以及基于表示相对于原点位置的坐标位置的经缩放值，确定用于对角模式的平面位置语法元素进行上下文编码的上下文。

[0380] 条款49C.根据条款48C的方法，其中基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文编码的上下文包括：基于经缩放值和原点位置确定一个或多个激光特性；以及基于激光特性对角模式的平面位置语法元素进行解码。

[0381] 条款50C.根据条款49C的方法，其中，基于表示相对于原点位置的坐标位置的经缩放值来确定用于对角模式的平面位置语法元素进行上下文编码的上下文还包括基于具有所确定的一个或多个激光特性的激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间还是低于第三距离阈值来确定上下文索

引。

[0382] 条款51C.根据条款48C-条款50C的任一项的方法,其中,平面位置语法元素指示垂直面位置。

[0383] 条款52C.根据条款49C-条款51C的任一项的方法,其中,一个或多个激光特性包括仰角、激光头偏移或激光的方位角。

[0384] 条款53C.根据条款48C-条款52C的任一项的方法,还包括:使用由所确定的上下文指示的上下文对角模式的平面位置进行算术编码。

[0385] 条款54C.根据条款48C-条款53C的任一项的方法,其中,在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值包括:确定最高有效位(MSB)群组和最低有效位(LSB)群组;在不剪裁的情况下对LSB进行缩放以确定缩放后的LSB;以及将缩放后的LSB添加到MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0386] 条款55C.根据条款54C的方法,在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值包括:基于节点的量化参数确定MSB的移位量;基于移位量对MSB进行移位;以及将缩放后的LSB添加到移位后的MSB中,以确定表示相对于原点位置的坐标位置的经缩放值。

[0387] 条款56C.根据条款48C-条款55C的任一项的方法,还包括:重建点云。

[0388] 条款57C.根据条款56C的方法,其中,重建点云包括基于平面位置确定点云的一个或多个点的位置。

[0389] 应该认识到,根据示例,本文中描述的任何技术的某些动作或事件可以以不同的序列执行,可以被一起添加、合并或省去(例如,不是所有描述的动作或事件是技术实践所必须的)。此外,在某些示例中,动作或事件可以,例如通过多线程处理、中断处理或多个处理器并发地执行而不是顺序地执行。

[0390] 在一个或多个示例中,可以在硬件、软件、固件或其任意组合中来实现所描述的功能。如果在软件中实现,则功能可以作为一个或多个指令或代码存储在计算机可读介质上或通过计算机可读介质传输,并由基于硬件的处理单元执行。计算机可读介质可以包括计算机可读存储介质,其对应于诸如数据存储介质之类的有形介质,或者通信介质,包括例如根据通信协议来促进将计算机程序从一个地方转移到另一个地方的任何介质。以这种方式,计算机可读介质通常可以对应于(1)非暂时性的有形计算机可读存储介质,或者(2)诸如信号或载波的通信介质。数据存储介质可以是可以由一个或多个计算机或一个或多个处理器访问以检索指令、代码和/或数据结构以实现本公开中描述的技术的任何可用介质。计算机程序产品可以包括计算机可读介质。

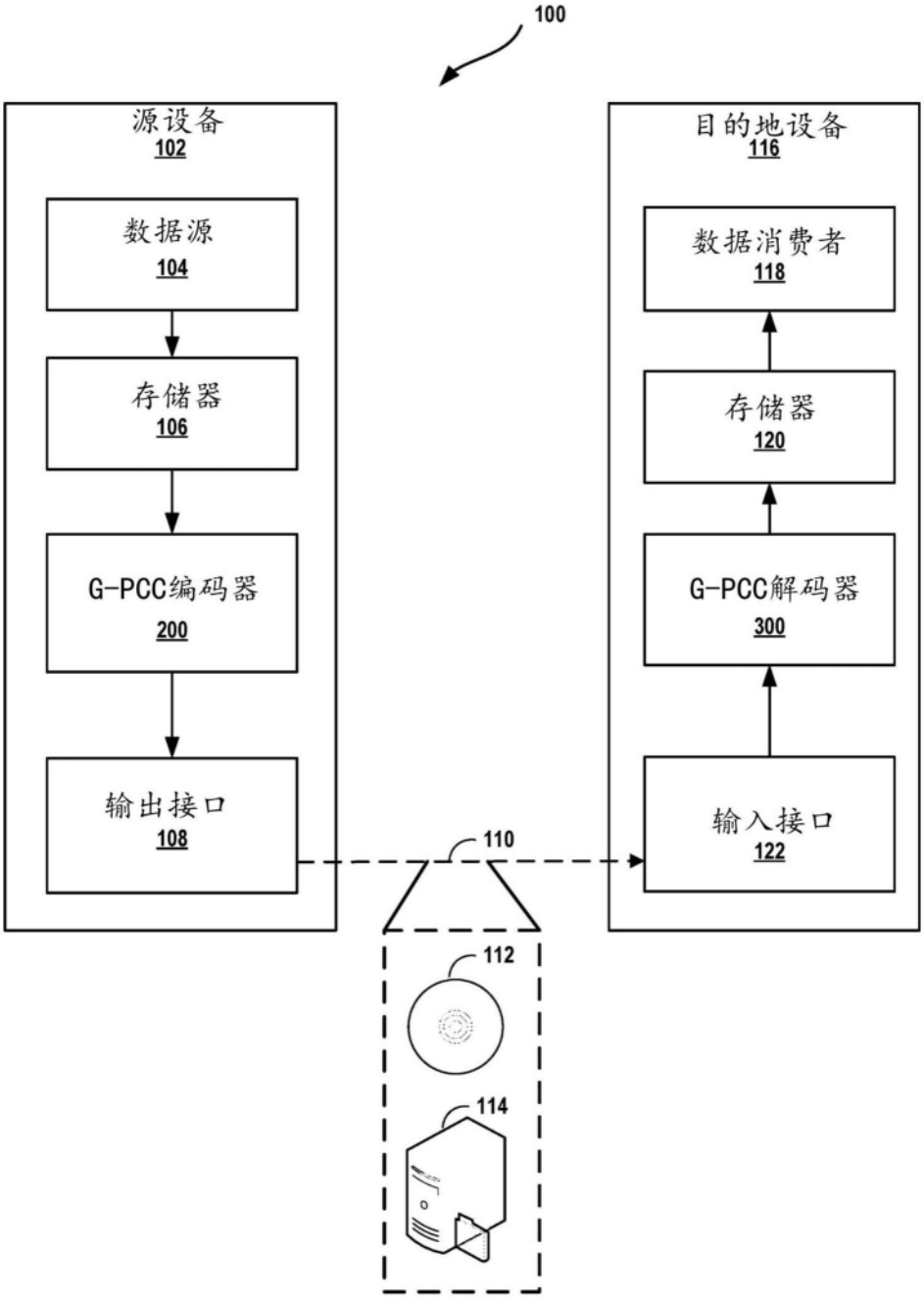
[0391] 作为示例而非限制,这种计算机可读存储介质可以包括RAM、ROM、EEPROM、CD-ROM或其他光盘存储、磁盘存储或其他磁性存储设备、闪存或可以用于以指令或数据结构形式存储所需程序代码并且可以由计算机访问的任何其他介质。而且,任何连接都适当地称为计算机可读介质。例如,如果使用同轴电缆、光纤电缆、双绞线、数字订户线(DSL)或无线技术(诸如红外、无线电和微波)从网站、服务器或其他远程源发送指令,则介质的定义包括同轴电缆、光纤电缆、双绞线、DSL或无线技术(诸如红外、无线电和微波)。然而,应当理解,计算机可读存储介质和数据存储介质不包括连接、载波、信号或其他暂时性介质,而是针对非

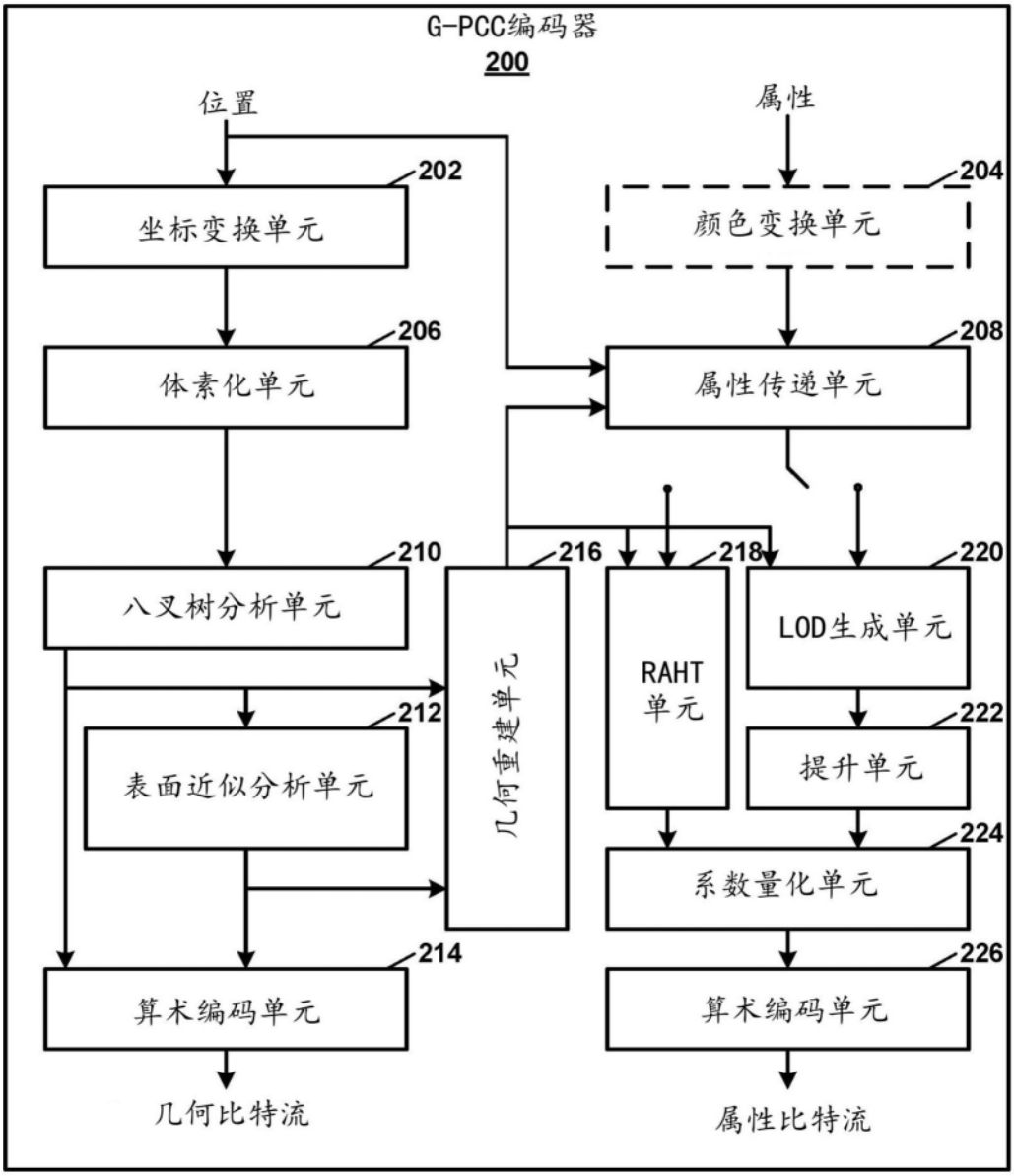
暂时性有形存储介质。如本文使用的，磁盘和光盘包括压缩光盘(CD)、激光光盘、光学光盘、数字多功能光盘(DVD)、软盘和蓝光光盘，其中，磁盘通常以磁性方式重现数据，而光盘用激光光学地重现数据。上述的组合也应包括在计算机可读介质的范围内。

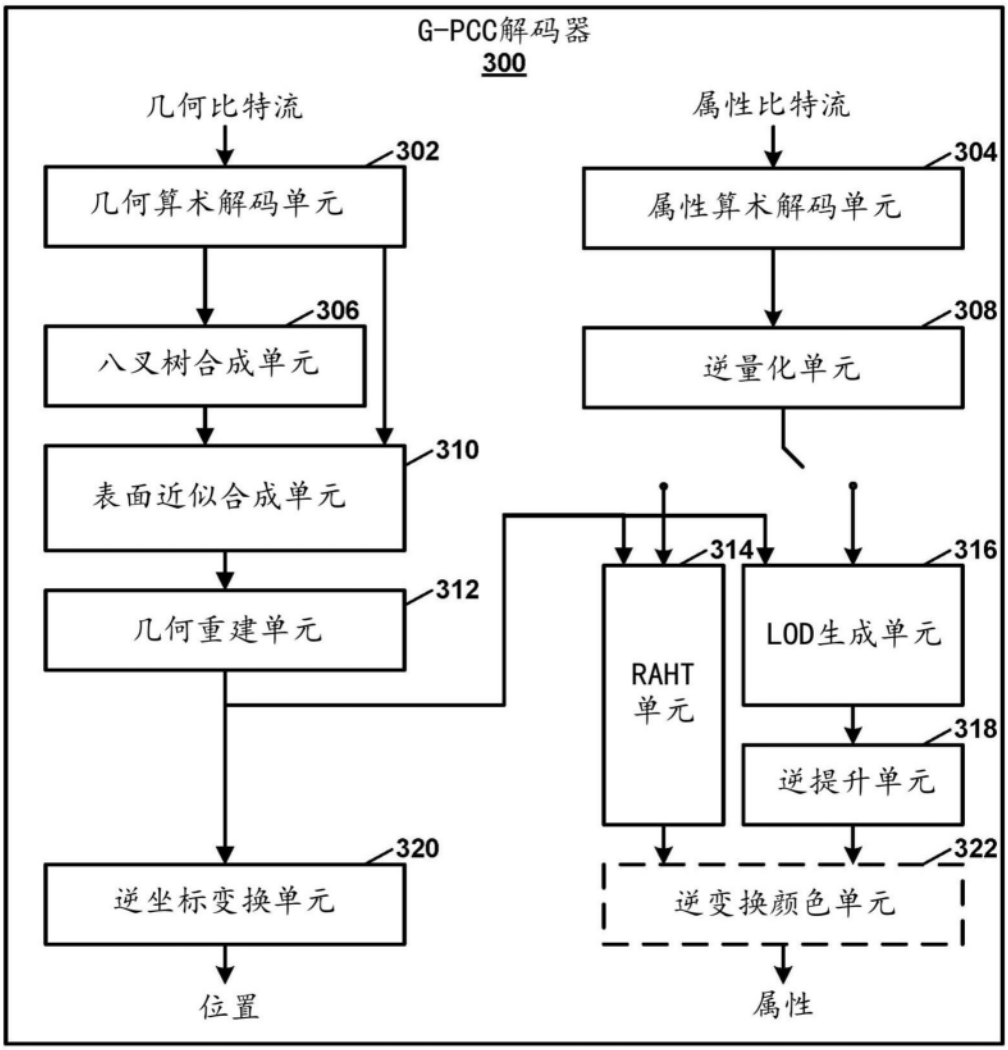
[0392] 指令可以由一个或多个处理器执行，诸如一个或多个DSP、通用微处理器、ASIC、FPGA或其他等效集成的或分立逻辑电路。因此，如本申请中所使用的术语“处理器”和“处理电路”可以是指任何前述结构或适合于实现本申请中描述的技术的任何其他结构。另外，在一些方面，本申请中描述的功能可以在被配置用于编码和解码的专用硬件和/或软件模块内提供，或合并在组合编解码器中。同样，该技术可以在一个或多个电路或逻辑元件中完全实现。

[0393] 本公开的技术可以在包括无线手机、集成电路(IC)或一组IC(例如，芯片集)的多种设备或装置中实现。在本公开中描述各种组件、模块或单元以强调被配置为执行所公开技术的设备的功能方面，但不一定需要由不同硬件单元来实现。相反，如上所述，各种单元可以组合在编解码器硬件单元中，或者由互操作硬件单元的集合来提供，包括与合适的软件和/或固件结合的如上所述的一个或多个处理器。

[0394] 已经描述了各种示例。这些和其他示例在所附权利要求的范围内。


100
源设备
102
数据源
104
存储器
106
G-PCC编码器
200
输出接口
108
110
112
114
目的地设备
116
数据消费者
118
存储器
120
G-PCC解码器
300
输入接口
122


G-PCC编码器
200
位置
属性
202
坐标变换单元
204
颜色变换单元
206
体素化单元
208
属性传递单元
210
八叉树分析单元
216
几何重建单元
218
RAHT
单元
220
LOD生成单元
212
表面近似分析单元
222
提升单元
224
系数量化单元
214
算术编码单元
226
算术编码单元
几何比特流
属性比特流


G-PCC解码器
300
几何比特流
属性比特流
302
几何算术解码单元
304
属性算术解码单元
306
八叉树合成单元
308
逆量化单元
310
表面近似合成单元
312
几何重建单元
314
RAHT
单元
316
LOD生成单元
318
逆提升单元
320
逆坐标变换单元
322
逆变换颜色单元
位置
属性

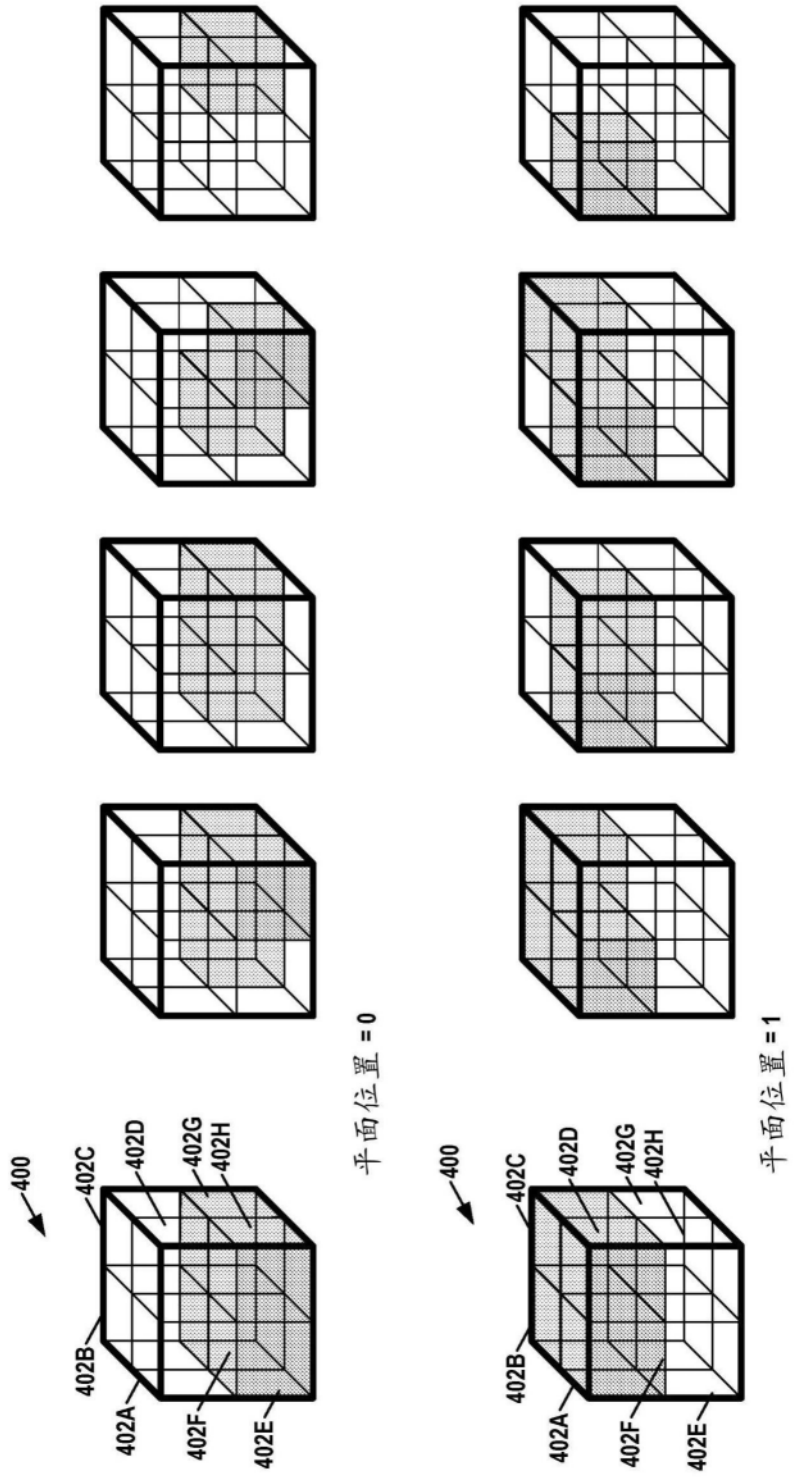

400
402C
402D
402G
402H
402B
402A
402F
402E
平面位置 = 0
400
402C
402D
402G
402H
402B
402A
402F
402E
平面位置 = 1

图4

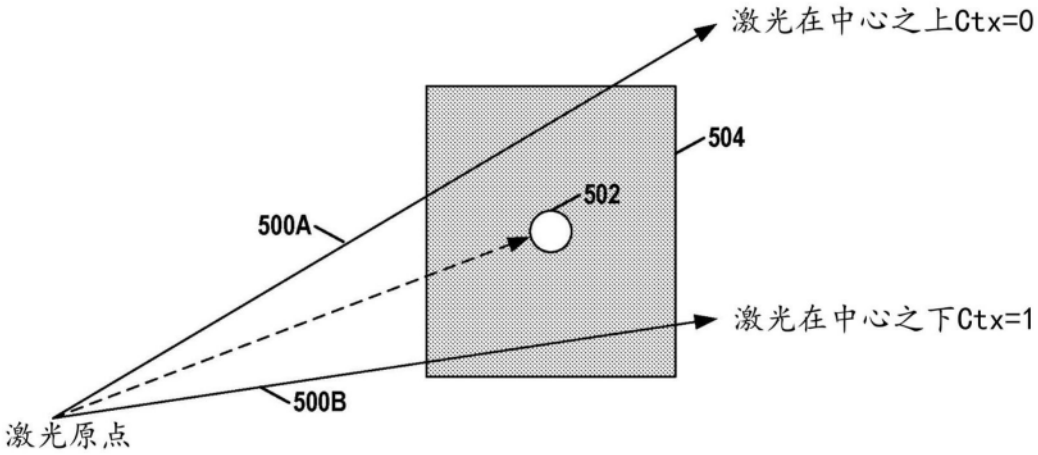

激光在中心之上Ctx=0
504
502
500A
激光在中心之下Ctx=1
500B
激光原点

图5

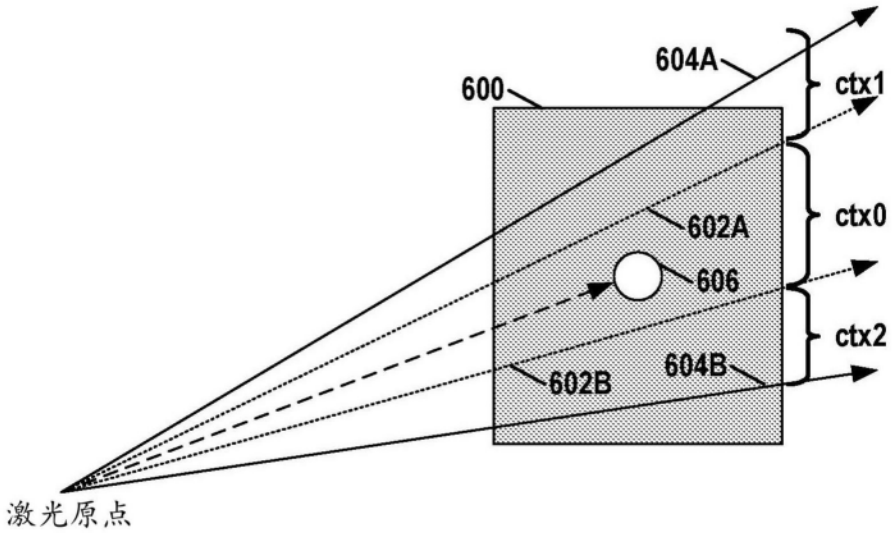

604A
600
ctx1
ctx0
602A
606
ctx2
602B
604B
激光原点

图6

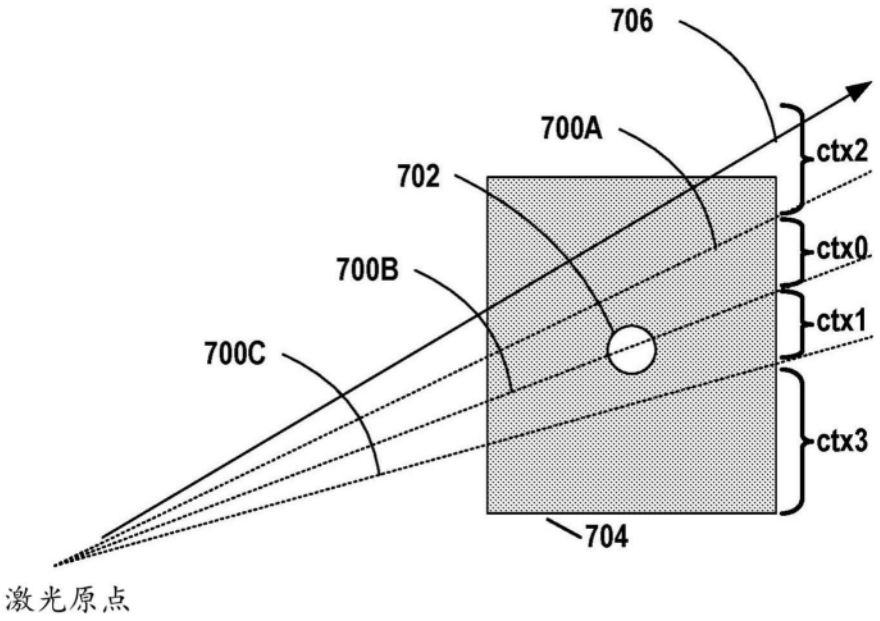


图7

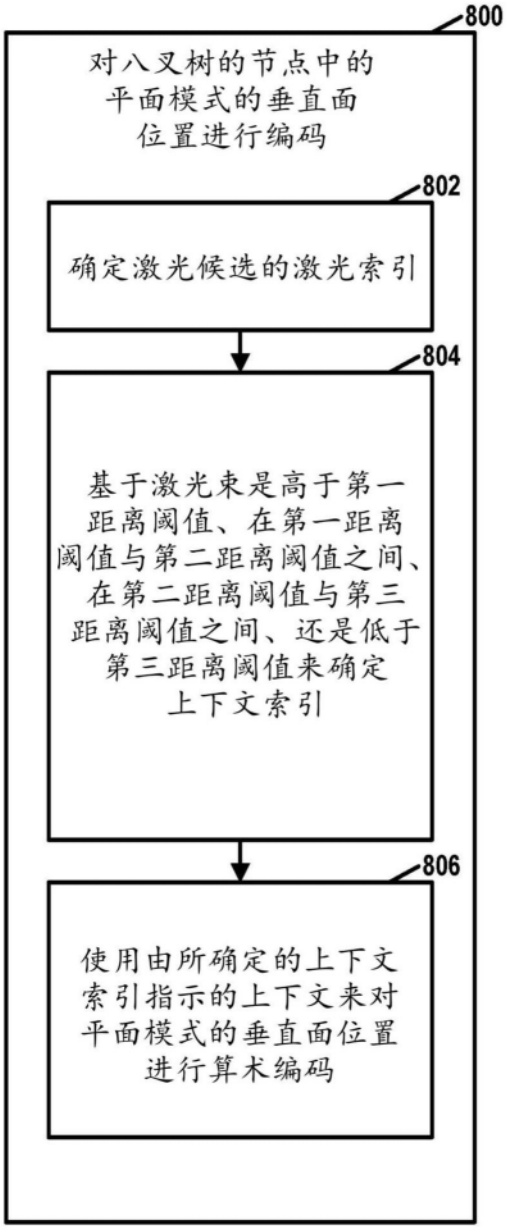

800
对八叉树的节点中的平面模式的垂直面位置进行编码
802
确定激光候选的激光索引
804
基于激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间、还是低于第三距离阈值来确定上下文索引
806
使用由所确定的上下文索引指示的上下文来对平面模式的垂直面位置进行算术编码

图8A

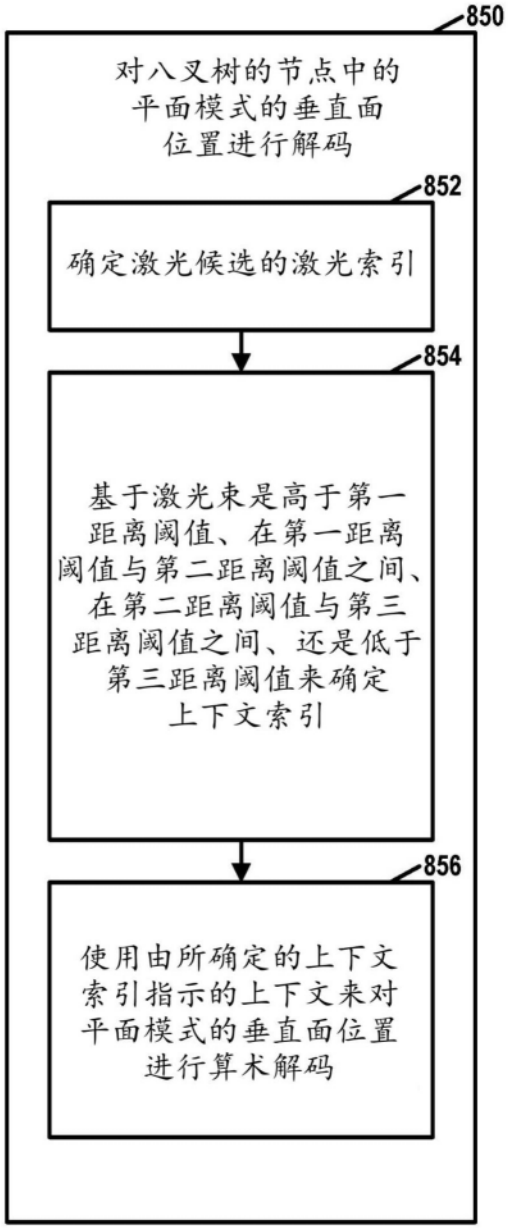

850
对八叉树的节点中的平面模式的垂直面位置进行解码
852
确定激光候选的激光索引
854
基于激光束是高于第一距离阈值、在第一距离阈值与第二距离阈值之间、在第二距离阈值与第三距离阈值之间、还是低于第三距离阈值来确定上下文索引
856
使用由所确定的上下文索引指示的上下文来对平面模式的垂直面位置进行算术解码

图8B

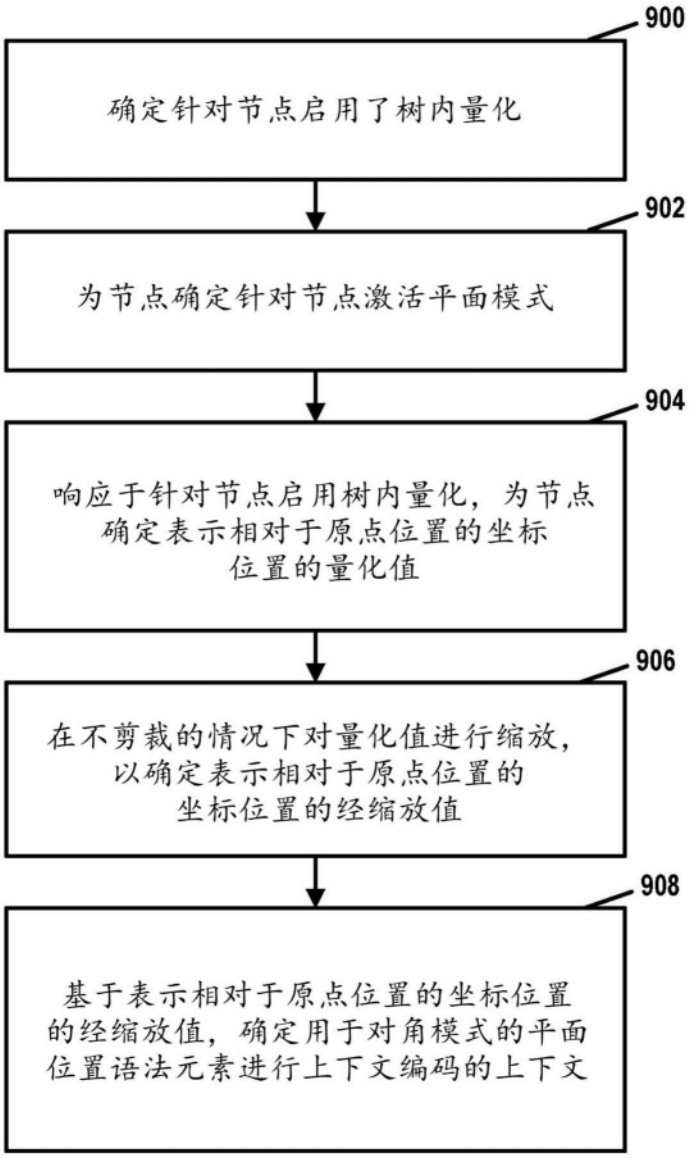

900
确定针对节点启用了树内量化
902
为节点确定针对节点激活平面模式
904
响应于针对节点启用树内量化，为节点确定表示相对于原点位置的坐标位置的量化值
906
在不剪裁的情况下对量化值进行缩放，以确定表示相对于原点位置的坐标位置的经缩放值
908
基于表示相对于原点位置的坐标位置的经缩放值，确定用于对角模式的平面位置语法元素进行上下文编码的上下文

图9A

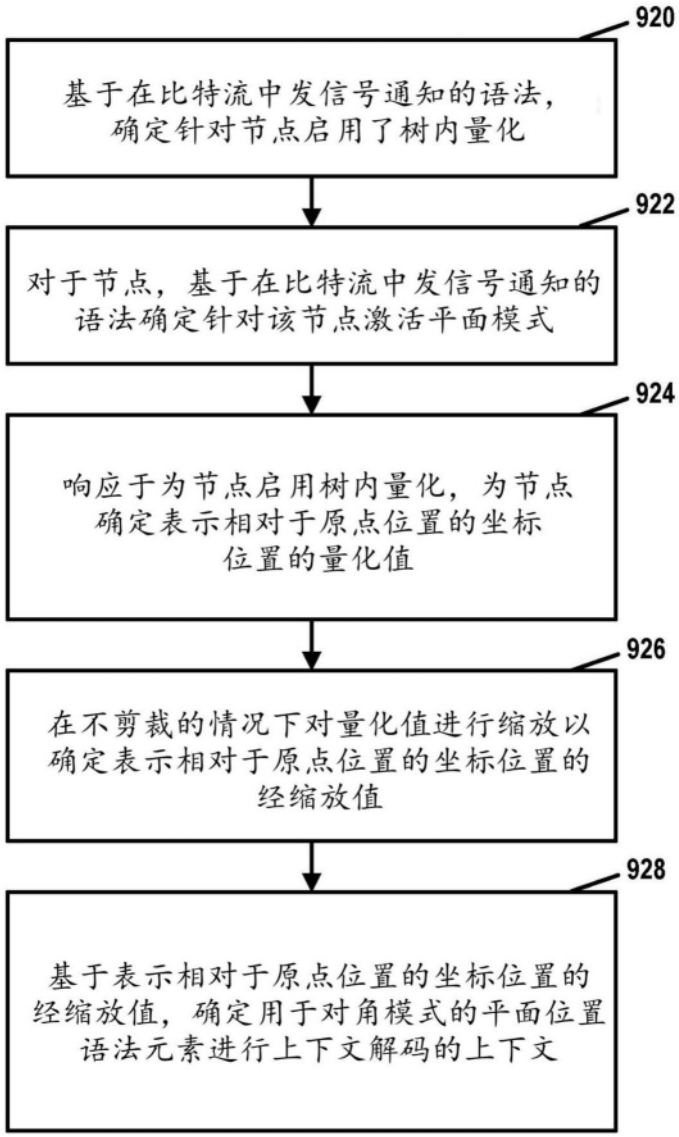

920
基于在比特流中发信号通知的语法，确定针对节点启用了树内量化
922
对于节点，基于在比特流中发信号通知的语法确定针对该节点激活平面模式
924
响应于为节点启用树内量化，为节点确定表示相对于原点位置的坐标位置的量化值
926
在不剪裁的情况下对量化值进行缩放以确定表示相对于原点位置的坐标位置的经缩放值
928
基于表示相对于原点位置的坐标位置的经缩放值，确定用于对角模式的平面位置语法元素进行上下文解码的上下文

图9B

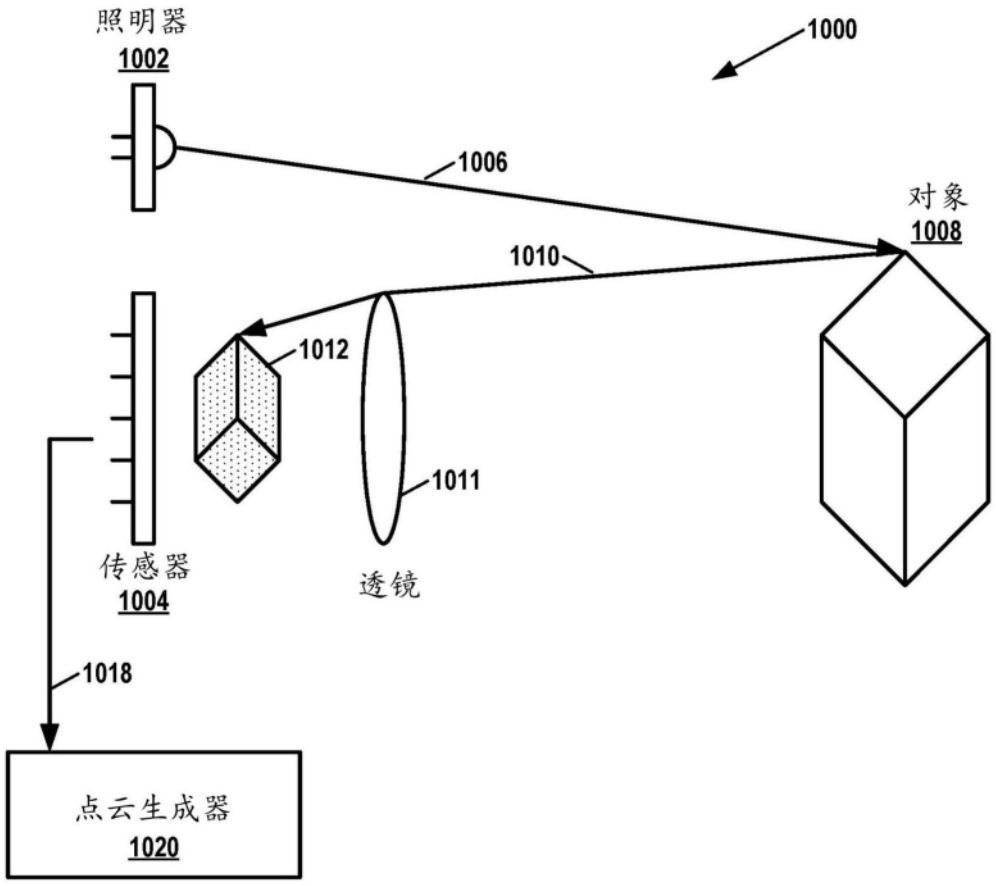

照明器
1002
1000
1006
对象
1008
1010
1012
1011
传感器
1004
透镜
1018
点云生成器
1020

图10

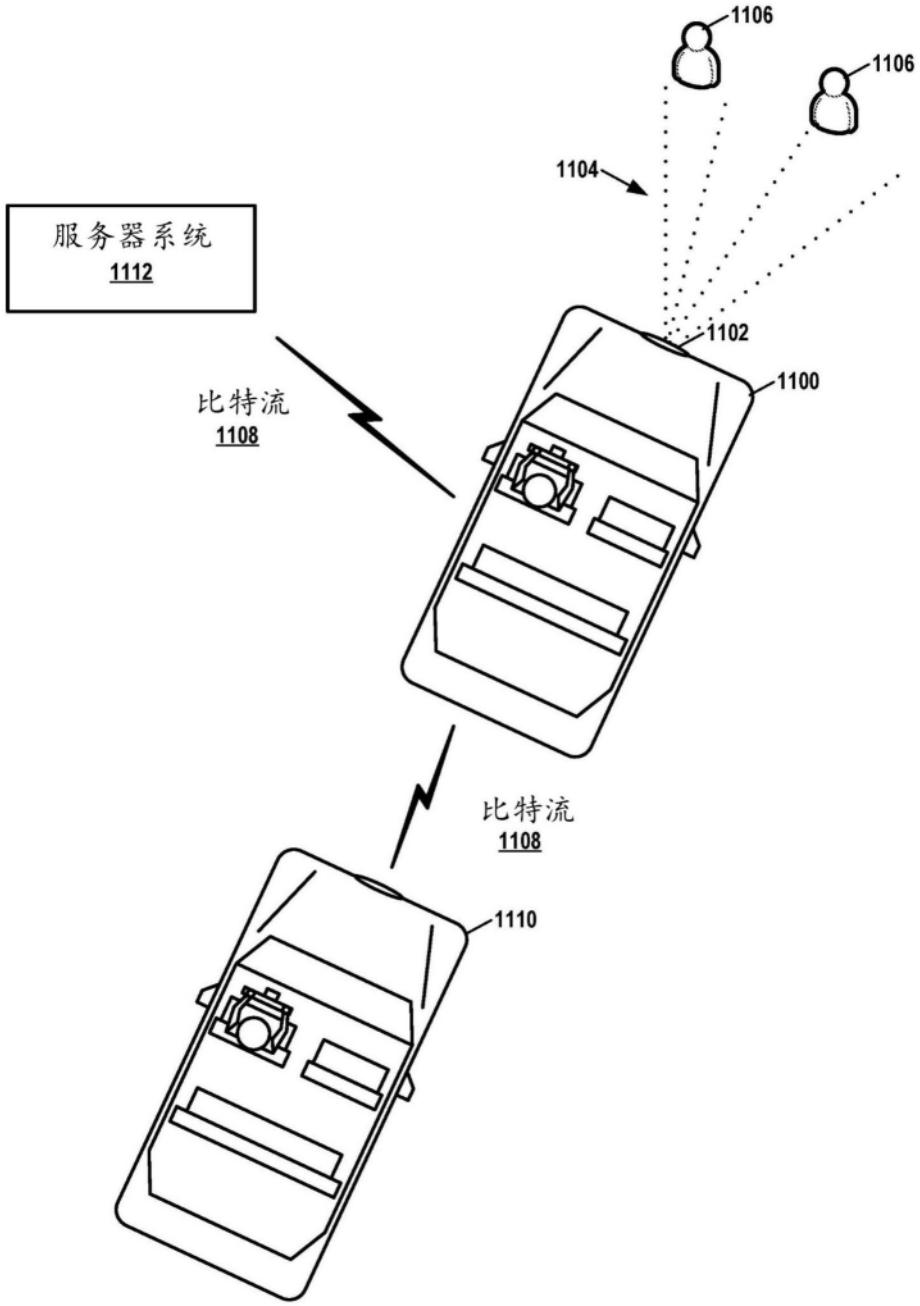


图11

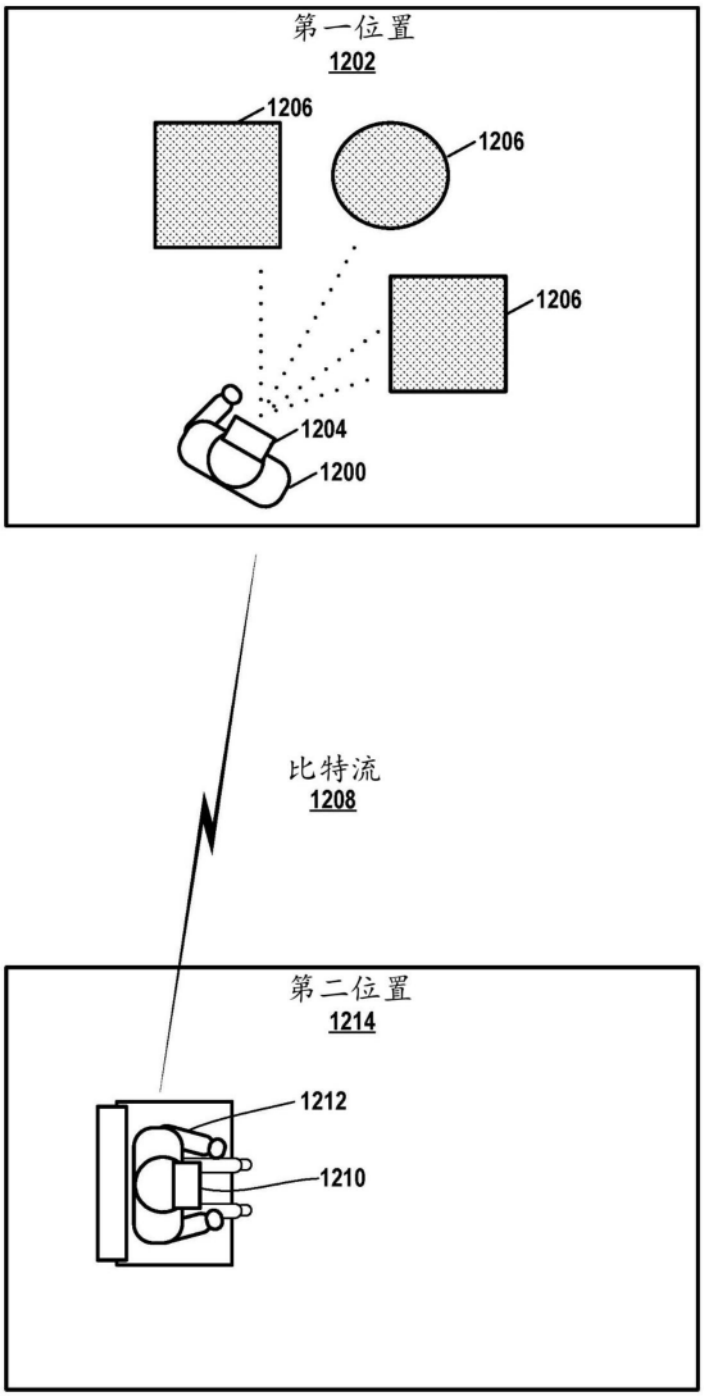

第一位置
1202
1206
1206
1206
1204
1200
比特流
1208
第二位置
1214
1212
1210

图12

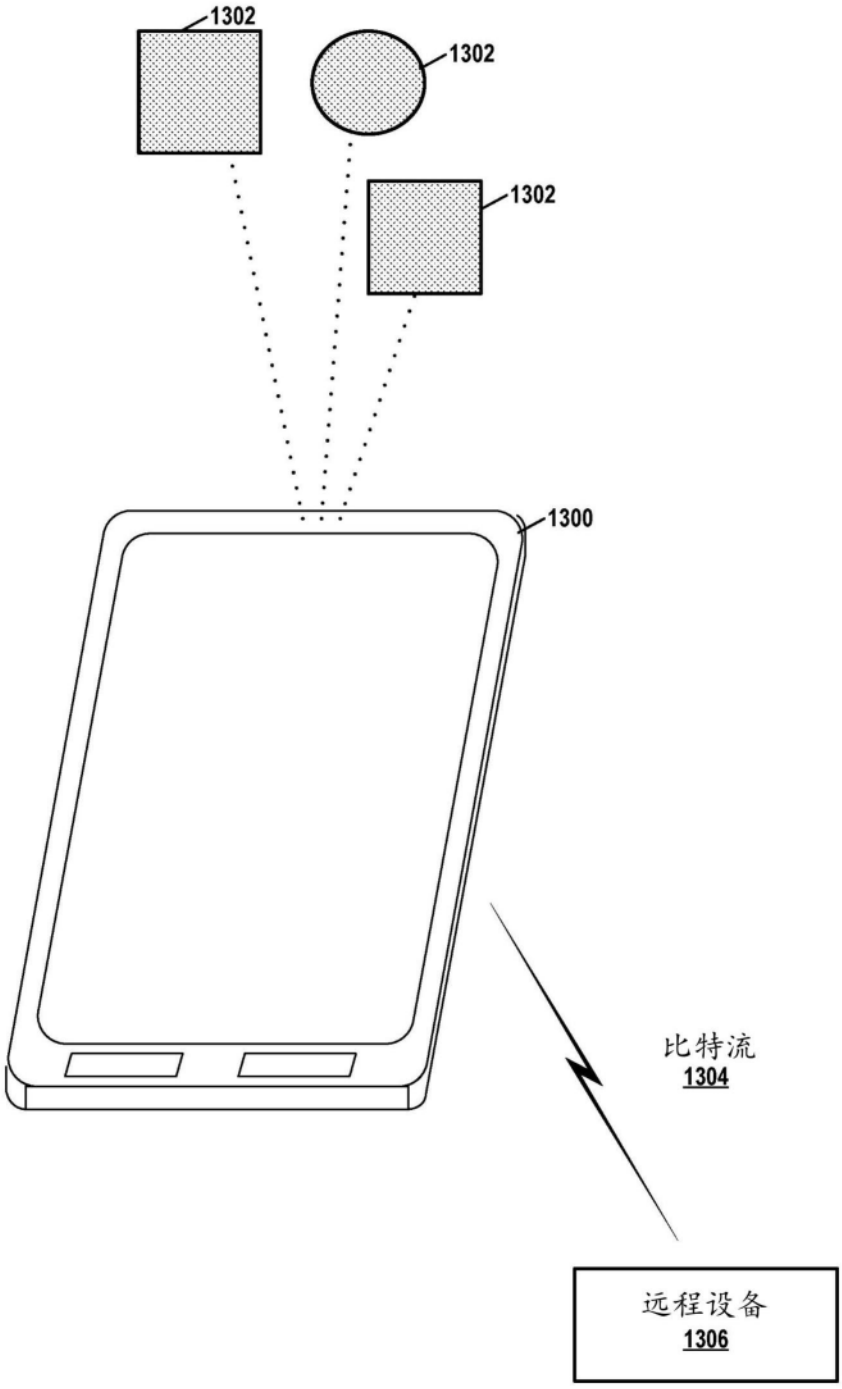


图13