摘要

三维模型提供了空间的体积表示，这对于包括飞行机器人和装有机械手的机器人在内的各种机器人应用非常重要。在本文中，我们提出了一个开源框架来生成体积 3D 环境模型。我们的映射方法基于八叉树，使用概率占用估计。它明确地表示不仅占用的空间，而且自由和未知的区域。此外，我们提出一种八叉树地图压缩方法，以保持三维模型紧凑。我们的框架可以作为一个开源的 C++库，并且已经成功地应用在多个机器人项目中。我们提出了一系列用真实机器人和公开可用的真实世界数据集进行的实验结果。结果表明，我们的方法能够有效地更新表示，并在保持最小内存需求的同时对数据进行统一建模。

Keywords 3D · Probabilistic · Mapping · Navigation

1 .导言

一些机器人应用需要环境的 3D 模型。这些包括空中、水下、室外或外层任务。然而，3D 模型也与许多家庭场景相关，例如，用于移动操作和导航任务。

尽管 3D 映射是许多机器人系统的一个组成部分，但是几乎没有现成的、可靠的和有效的实现。缺乏这样的实现导致重新创建基本的软件组件，因此，可以看作是机器人学研究的瓶颈。因此，我们相信，开发开源的 3D 映射框架将极大地促进需要三维几何环境表示的机器人系统的开发。

大多数机器人应用需要概率表示、自由区域、占用区域和未映射区域的建模，以及运行时和内存使用方面的额外效率。现在我们将详细讨论这三个要求。

概率表示：为了创建 3D 地图，移动机器人通过三维距离测量来感知环境。这种测量受到不确定度的影响：通常，距离测量的误差是厘米的量级。但是也可能存在由反射或动态障碍物引起的看似随机的测量。当任务是从这些噪声测量中建立环境的精确模型时，必须概率地考虑潜在的不确定性。然后可以将多个不确定测量融合到环境的真实状态的鲁棒估计中。另一个重要方面是概率传感器融合允许来自多个传感器和多个机器人的数据集成。

–未映射区域的建模：在自主导航任务中，机器人只能为传感器测量覆盖并检测为自由的区域规划无碰撞路径。相反，需要避免未映射区域，并且由于这个原因，地图必须表示这些区域。此外，在勘探过程中，关于未绘制地图的区域的知识是必不可少的。当地图是自主创建的，机器人必须计划其行动，以便测量采取在以前未映射的区域。

–效率：地图是任何自治系统的中心组件，因为它在行动规划和执行期间使用。由于这个原因，映射在访问时间方面需要是有效的，但是在内存消耗方面也是有效的。从实用的角度来看，内存消耗往往是 3D 映射系统的主要瓶颈。因此，重要的是该模型在存储器中是紧凑的，以便能够映射大的环境，机器人可以将该模型保存在其主存储器中，并且可以在多个机器人之间有效地传输。

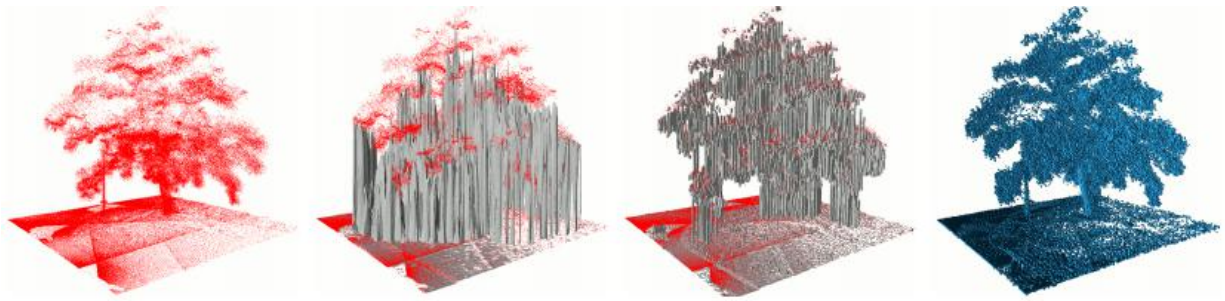

Fig. 1 3D representations of a tree scanned with a laser range sensor (from left to right): Point cloud, elevation map, multi-level surface map, and our volumetric (voxel) representation. Please note that our volumetric representation explicitly models free space but that for clarity only occupied volumes are visualized.

已经提出了几种方法来模拟机器人中的 3D 环境。作为示例，我们将我们的方法与三种常见的映射方法进行比较——结果的可视化在图 1 中给出。在该示例中，使用点云、海拔图(Hebert et al.，1989)、

多级表面图(Triebel et al.，2006)以及使用我们的框架以体积方式表示 3D 测量。前面的方法都不能满足上面提出的所有要求。点云存储了大量的测量点，因此不具有记忆性。此外，它们不允许区分无障碍区和未映射区，也不提供概率融合多个测量的手段。高程图和多级表面图是有效的，但也不代表未映射区。最重要的是，这些方法不能表示任意的 3D 环境，比如示例中的树枝。

在本文中，我们提出了一个基于八叉树的三维环境表示集成框架 **OctoMap**。在我们的框架中，我们结合了以前 3D 环境建模方法的优点，以满足上面讨论的需求。我们的方法的一个核心特性是，它允许对占用空间和空闲空间进行高效和概率的更新，同时将内存消耗保持在最小值。占用空间由诸如激光测距仪的距离传感器的端点获得，而自由空间对应于传感器与端点之间的观察区域。作为我们方法的一个关键贡献，我们引入了一种压缩方法，该方法通过局部组合映射空闲区域和占用空间中的相干映射卷来减少内存需求。我们实现了我们的方法，并使用各种公开可用的室内和室外环境的真实世界机器人数据集对其进行了彻底的评估。

我们的开源实现是以一个独立的 C++库的形式自由提供的。它是在 BSD 许可下发布的，可以从 http://octomap.github.com.获得。该库支持多个平台，如 Linux、Mac OS 和 Windows。它已经集成到机器人操作系统（ROS）中，并且可以直接用于其他软件框架。自从 2010 年首次引入（Wurm 等人，2010）以来，OctoMap 框架不断改进，并在越来越多的机器人研究项目中使用。

本文的组织如下。在下一节中，我们将详细讨论 3D 数据结构和映射方法领域的相关工作，然后在 Sect. 三。在 SCT 中给出了实现细节。4，随后对所提出的框架进行评价。5。最后，通过对 OctoMap 在机器人领域应用的案例研究，说明了 Sect.6。

2 相关工作

环境的三维模型是许多机器人系统必不可少的重要条件，因此它们已成为二十多年的研究课题。在 3D 中建模环境的一种流行方法是使用等大小的立方体网格（称为体素）来离散映射区域。Roth-Tabak 和 Jain（1989）以及 Moravec（1996）都用这种表现形式展示了早期的作品。刚性网格的一个主要缺点是其大的内存需求。网格地图需要初始化，以便至少与映射区域的边界框一样大，而不管卷中地图单元的实际分布。在大规模户外场景中，或者当需要精细分辨率时，内存消耗可能变得令人望而却步。此外，需要预先知道映射区域的范围，或者每次扩展映射区域都需要执行昂贵的复制操作。

通过直接存储三维距离测量，可以避免环境的离散化。然后，通过由诸如激光测距仪或立体相机之类的距离传感器返回的 3D 点云来对环境中所占用的空间进行建模。这种点云方法已经在一些 3DSLAM 系统中使用，例如 Cole 和 Newman（2006）提出的那些系统，以及 Nuchter 等人的 SLAM 方法中。（2007）。这种方法的缺点是既不模拟自由空间也不模拟未知区域，不能直接处理传感器噪声和动态对象。因此，点云仅适用于静态环境中的高精度传感器，并且当不需要表示未知区域时。此外，这种表示的内存消耗随着测量次数的增加而增加，这是有问题的，因为没有上限。

如果能够对映射区域作出某些假设，那么 2.5D 映射就足以对环境进行建模。通常，2D 网格用于存储每个单元的测量高度。在最基本的形式中，这导致一个高程图，其中地图精确地存储每个单元格一个值（Hebert 等人，1989）。已经证明这种地图是足够的一种方法是由 Hadsell 等人描述的户外地形导航方法。（2009）。只要机器人使用单个表面进行导航，高程图就足以对环境进行建模，因为可以安全地忽略高于车辆的悬空障碍，例如树木、桥梁或地下通道。通过允许每个单元有多个表面（Triebel 等人，2006；Pfaff 等人，2007），或者通过使用对应于不同类型结构的单元类（Gutmann 等人，2008），可以放松对单个表面的严格假设。大多数 2.5D 地图的一个普遍缺点是它们不以体积的方式表示环境，而是基于机器人的高度在垂直维度上对其进行离散化。虽然这对于具有固定机器人形状的路径规划和导航是足够的，但是该地图不代表实际环境，例如用于定位。

为了解决这个问题，赖德和胡（2010）提出了一个相关的方法。该方法为 2D 网格中的每个单元存储占用的体素列表。虽然这种表示是容量的，但是它不区分自由卷和未知卷。德拉扬诺夫斯基等人。（2010）在多卷占用网格方法中存储每个 2D 单元的占用和空闲体素的列表。然而，与我们的方法相

比，需要预先知道地图范围，地图更新在计算上更复杂，并且没有多分辨率能力。另一个潜在的问题是后续的映射更新不能细分现有卷，从而导致环境的模型不正确。类似地，DouiLad 等人。(2010)将用于背景结构的粗略高程图与具有较高分辨率的对象体素图结合。与我们的工作相反，这种方法专注于单个测量的 3D 分割，并且没有将多个测量集成到环境的模型中。

在机器人映射中，八叉树避免了固定网格结构的主要缺点之一：它们延迟了映射卷的初始化，直到需要集成测量为止。通过这种方式，映射环境的范围不需要预先知道，并且映射只包含已经测量的卷。如果树的内部节点被适当地更新，那么树也可以被用作多分辨率表示，因为它可以在任何级别被切割以获得更粗的细分。八叉树映射的使用最初是由 MaGHER（1982）提出的。早期的工作主要集中于对布尔属性进行建模，如占用（Wilhelms 和 Van Gelder，1992）。（1997）使用八叉树来适应从 2D 到 3D 的占用网格映射，从而引入了一种对占用空间和自由空间进行建模的概率方法。福涅尔等人使用了类似的方法。（2007）和 PATAK 等（2007）。然而，与本文提供的方法相比，作者没有明确地解决映射压缩或映射中的有界置信度问题。

费尔菲尔德等人提出了一种基于八叉树的三维地图表示方法。（2007）。它们的映射结构称为延迟引用计数八叉树（Deferred Reference Counting Octree），它被设计成允许高效的映射更新，尤其是在粒子过滤器 SLAM 的上下文中。为了实现地图紧凑性，周期性地执行有损最大似然压缩。与我们方法中使用的压缩技术相比，这种方法丢弃了用于将来更新的概率信息。此外，没有解决过度自信映射和多分辨率查询的问题。作为一种数据结构，八叉树被应用于各种应用中，最显著的是用于有效绘制的计算机图形领域(Botsch 等人，2002；Surmann 等人，2003；Laine 和 Karras，2010)以及用于存储和处理大点云的摄影测量领域(GirardeauMonta)UT 等人，2005；ELSBER 等人，2011）。另一个流行的用例是静态点云压缩（Schnabel 和 Klein，2006）或点云流（Kammerl 等人，2012）。虽然我们的框架足够通用，可以存储原始点云，但是它的主要目的是将这些点云集成到一个内存高效的、体积占用地图中，因为点云作为机器人学中的环境表示有许多缺点，如本文开头所述。是章节，YGEL 等。（2007 年 B）提出了一种基于 Haar 小波数据结构的三维地图。这种表示也是多分辨率和概率的。然而，作者没有深入评价 3D 建模的应用。在它们的评估中，未对未知区域进行建模，并且仅使用单个模拟 3D 数据集。如果没有公开可用的实现，则很难评估这种映射结构是否像八叉树那样节省内存。

表面表示，如 3D 正态分布变换（ManguSon 等人，2007）或 Surfels（Habbecke 和 KabBand，2007）最近被用于 3D 路径规划（Stoyayv 等人，2010）和对象建模（Wee 等人，2009；Krin 等人，2011）。类似地，纽康等人提出了一种基于低成本深度相机和 GPU 处理的精确实时三维 SLAM 系统。（2011）重建室内场景中的密集曲面。最近，这项工作已经扩展到工作在更大的室内环境（惠兰等，2012）。然而，表面表示无法区分自由空间和未知空间，可能需要大的存储器，特别是户外，并且通常基于关于相应环境的强假设。在移动操作场景中，例如，能够区分未知的空间对于安全导航来说是必不可少的。

最后，据我们所知，没有开放源代码实现的 3D 占用映射框架满足引言中概述的要求是免费的。

3 OctoMAP 映射框架

本文提出的方法使用基于树的表示来提供映射面积和分辨率的最大灵活性。它执行概率占用估计，以确保可更新性和应付传感器噪声。此外，压缩方法确保所得到的模型的紧凑性。

3.1 八叉树

八叉树是用于三维空间细分的分层数据结构（Meagher，1982；Wilhelms 和 Van Gelder，1992）。八叉树中的每个节点表示立方体卷中包含的空间，通常称为体素。该卷被递归地细分为八个子卷，直到达到给定的最小体素大小，如图 2 所示。最小体素大小决定八叉树的分辨率。由于八叉树是分层数据结构，因此如果内部节点被相应地维护，那么可以在任何级别上切割树以获得更粗的分割。图 3 显示了一个八叉树映射的示例，该映射以几种分辨率查询占用的体素。

在其最基本的形式中，八叉树可以用来建模布尔属性。在机器人映射的上下文中，这通常是卷的占用。如果某个卷被测量为占用，则八叉树中的对应节点被初始化。在这个布尔设置中，任何未初始化的节点都可以是空闲的或未知的。为了解决这种模糊性，我们明确地表示树中的自由卷。这些是在传感器与测量端点之间的区域内产生的，例如，沿着通过射线扫描确定的射线。未初始化的区域隐式地模拟未知空间。如图 4 所示，一个八叉树包含来自真实激光传感器数据的自由和占用的节点。使用布尔占用状态或离散标签允许八叉树的紧凑表示：如果节点的所有子节点具有相同的状态（占用或空闲），则可以对其进行修剪。这导致需要维护在树中的节点数量显著减少。

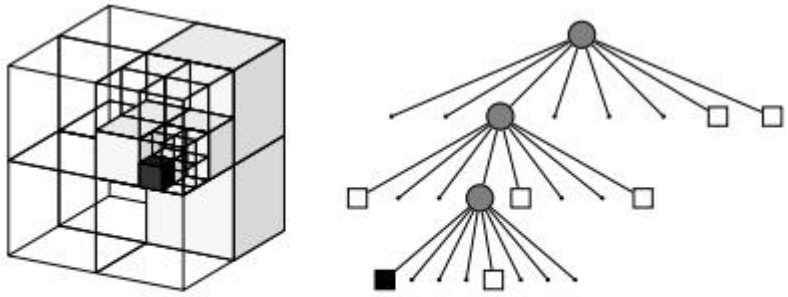

Fig. 2 Example of an octree storing free (shaded white) and occupied (black) cells. The volumetric model is shown on the left and the corresponding tree representation on the right.

Fig. 3 By limiting the depth of a query, multiple resolutions of the same map can be obtained at any time. Occupied voxels are displayed in resolutions 0.08 m, 0.64 , and 1.28 m.

在机器人系统中，人们通常必须应对传感器噪声和暂时或永久变化的环境。在这种情况下，离散占用标签是不够的。取而代之的是，必须对占用进行概率建模，例如通过应用占用网格映射（Moravec 和 Elfes，1985）。然而，这样的概率模型缺乏通过修剪进行无损压缩的可能性。
本文提出的方法提供了结合使用离散标签的八叉树的紧凑性和概率建模的可更新性和灵活性的方法，我们将在第三节讨论。3.4。

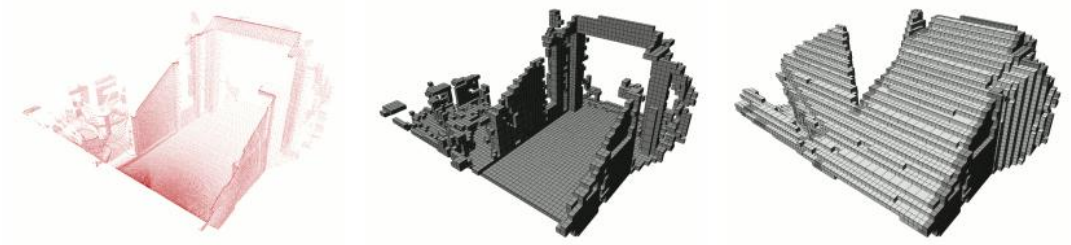

Fig. 4 An octree map generated from example data. *Left:* Point cloud recorded in a corridor with a tilting laser range finder. *Center:* Octree generated from the data, showing occupied voxels only. *Right:* Visualization of the octree showing occupied voxels (dark) and free voxels (white). The free areas are obtained by clearing the space on a ray from the sensor origin to each end point. Lossless pruning results in leaf nodes of different sizes, mostly visible in the free areas on the right.

在数据访问复杂性方面，由于树结构，八叉树与固定大小的 3D 网格相比需要开销。对包含具有树深度为 d 的 n 个节点的树数据结构，可以以 O(d)=O(logn)的复杂度执行单个随机查询。以深度优先的方式遍历整个树需要 O(n)的复杂性。请注意，在实践中，我们的八叉树被限制为固定的最大深度 DMAX。这导致 O(dmax)的随机节点查找复杂度为常数。因此，对于固定深度 dmax，与对应的 3D 网格相比的开销是恒定的。注意，在我们的所有实验中，使用的最大深度为 16，这足以覆盖体积为(655:36m)3、分辨率为 1cm 的立方体。在 SECT 中提供了该设置的精确定时。5.5。

3.2 概率传感器融合

在我们的方法中，传感器读数使用 Moravec 和 Elfes（1985）介绍的占用网格映射进行集成。给定传感器测量值 z1:t，估计叶节点 n 被占用的概率 P(n j z1:t)，其依据是

$$P(n \mid z_{1:t}) = \left[1 + \frac{1-P(n \mid z_t)}{P(n \mid z_t)} \frac{1-P(n \mid z_{1:t-1})}{P(n \mid z_{1:t-1})} \frac{P(n)}{1-P(n)}\right]^{-1} \quad (1)$$

该更新公式取决于当前测量 zt、先验概率 P(n)和先验估计 P(n j z1：t_1)。术语 P(n j zt)表示给定测量 zt 的体素 n 被占据的概率。这个值对于产生 ZT 的传感器是特定的。我们提供详细的传感器模型在整个实验中使用的 SCT。5.1。

一致先验概率的普遍假设导致 P(n)=0:5，并且通过使用 log-odds 符号，等式(1)可以改写为

$$\mathrm{L}(n \mid z_{1:t}) = \mathrm{L}(n \mid z_{1:t-1}) + \mathrm{L}(n \mid z_t), \quad (2)$$

with

$$\mathrm{L}(n) = \log\left[\frac{P(n)}{1-P(n)}\right]. \quad (3)$$

该更新公式取决于当前测量 zt、先验概率 P(n)和先验估计 P(n j z1：t_1)。术语 P(n j zt)表示给定测量 zt 的体素 n 被占据的概率。这个值对于产生 ZT 的传感器是特定的。我们提供详细的传感器模型在整个实验中使用的 SCT。5.1。

统一先验概率的普遍假设导致 P(n)=0:5，并且通过使用 log-odds 符号，方程(1)可以被重写，因为更新规则的这个公式允许更快的更新，因为乘法被加法所代替。在预先计算的传感器模型的情况下，不需要在更新步骤期间计算对数。注意，log-odds 值可以转换为概率，反之亦然，因此我们为每个体素存储这个值而不是占用概率。值得注意的是，对于对称的传感器模型的某些配置，即，作为命中更新的节点具有与作为未命中更新的节点相同的权重，这种概率更新与类似地计算命中和未命中具有相同的效果(Kelly 等人，2006)。

当使用 3D 地图进行导航时，常常应用占用概率 P(n j z1:t)的阈值。当达到阈值时，体素被认为被占用，否则假定它是自由的，从而定义两个离散状态。从等式（2）可以明显看出，要改变体素的状态，我们需要集成与已集成一样多的观察来定义其当前状态。换言之，如果一个体素被观测到自由 k 次，那么它必须被观测到至少被占用 k 次，然后根据阈值（假设在传感器模型中自由和被占用的测量同样可能）认为被占用。虽然这种特性在静态环境中是需要的，但是移动机器人经常面临环境中的临时或永久变化，并且地图必须快速适应这些变化。为了确保这种适应性，YGEL 等人。(2007a)提出了一种钳位更新策略，该钳位更新策略定义了占用估计的上限和下限。代替直接使用等式（2），根据估计更新占用估计值。

$$\mathrm{L}(n \mid z_{1:t}) = \max(\min(\mathrm{L}(n \mid z_{1:t-1}) + \mathrm{L}(n \mid z_t), l_{\max}), l_{\min}), \quad (4)$$

其中 Lmin 和 LMAX 表示对数几率值的下界和上界。直观地，这个修改的更新公式限制了更改体素的状态所需的更新次数。在我们的方法中应用箝位更新策略将带来两个优点：我们确保对映射的信任保持有界，因此模型可以快速适应环境的变化。此外，我们还可以通过修剪来压缩相邻体素（见 SeCT）。3.4）。正如我们将要讨论的。5.4，这导致必须保持的体素的数量显著减少。从完全概率的角度来看，通过箝位实现的压缩不再是完全无损的，因为接近零的信息和一个信息丢失。然而，在夹持阈值之间，完全概率被保留。

3.3 种多分辨率查询

当测量集成到映射结构中时，只对八叉树中的叶节点执行概率更新。但是由于八叉树是一种分层的数据结构，我们可以利用树中的内部节点来实现多分辨率查询。请注意，当树只被遍历到一个给定的深度（不是叶节点的深度）时，我们对 3D 空间进行了更粗略的分割。每个内部节点跨越其八个子节点占用的体积，因此为了确定内部节点的占用概率，我们必须聚合其子节点的概率。

考虑到节点 n 的八个子卷 ni，可以采用几种策略来确定节点 n 的占用概率(Kraetzschmar 等人，2004)。根据手头的应用，平均占有率

$$\bar{l}(n) = \frac{1}{8}\sum_{i=1}^{8} \mathrm{L}(n_i) \quad (5)$$

or the maximum occupancy

$$\hat{l}(n) = \max_i \mathrm{L}(n_i) \quad (6)$$

可以使用，其中 L(n)返回节点 n 的当前 log-odds 占用值。使用最大子占用来更新内部节点可以被认为是一种保守策略，非常适合于机器人导航。通过假设一个卷被占用，如果它的任何部分已经被占用，则可以规划无冲突的路径，并且由于这个原因，在我们的系统中使用最大占用更新。注意，在更保守的设置中，L(n)也可以被定义为返回未知小区的正占用概率。图 3 显示了一个八叉树在多个分辨率下查询占用体素的例子。

3.4 八叉树图压缩

在教派中。3.1，我们解释了树剪枝如何减少具有离散占用状态的八叉树中的冗余信息量，其中体素既可以被占用，也可以被释放。同样的技术也可以应用于使用概率占用估计来对占用空间和自由空间进行建模的地图。然而，一般来说，即使两个体素被相同的物理障碍物占用，也不能期望相邻节点的占用概率是相同的。传感器噪声和离散化误差可能导致不同的概率，从而干扰依赖于相同节点信息的压缩方案。该问题的一个可能的解决方案是对体素概率应用阈值，例如 0.5，并且以这种方式生成 Fairfield 等人建议的离散状态估计。（2007）。然而，使用这种方法，在修剪树之后，无法恢复单个的概率估计。在我们的方法中，我们通过应用等式（4）中给出的箝位更新策略来实现地图压缩。每当体素的 log-odds 值达到下界 lmin 或上界 lmax 时，我们认为节点在我们的方法中是稳定的。直观地，稳定的节点已经被自由地测量或者以高置信度占用。在静态环境中，所有体素在集成了足够数量的测量值之后将收敛到稳定状态。

例如，在我们的实验中选择的参数下，五个一致的测量值足以将未知体素渲染成稳定的体素。如果内部树节点的所有子节点都是具有相同占用状态的稳定叶节点，则可以修剪这些子节点。如果将来的测量与相应的内部节点的状态相矛盾，那么它的子节点将相应地重新生成和更新。应用此压缩仅导致接近 P(n)=0 和 P(n)=1 的信息丢失，同时保持两者之间的概率。在我们的实验中，结合八叉树修剪和钳位导致高达 44%的压缩改善。在许多机器人导航任务中，例如避障或定位，只有包含空闲或占用节

点的最大似然图就足够了。在这些情况下，基于占用阈值的有损压缩，如 Fairfield 等。（2007）可以执行。对于这种压缩，所有节点都被转换为其最大似然概率（钳位），然后进行树剪枝。这就产生了更大的压缩和更少的内存需求。

Fig. 5 Detail of a volumetric indoor OctoMap containing color information. The complete map covers an area of 7.3 m × 7.9 m × 4.6 m at 2 cm resolution.

3.5 扩展

3.5.1 具有丰富信息的地图

可以扩展八叉树节点来存储额外的数据以丰富地图表示。例如，体素可以存储地形信息、环境数据（如温度）或颜色信息。每个额外的体素属性都需要一个允许融合多个测量的方法。作为示例，我们扩展了映射框架来存储每个体素的平均颜色。这为用户创建了可视化，并且能够根据虚拟视图对环境进行基于颜色的分类或基于外观的机器人定位（类似于（Einhorn 等人，2011；Mason 等人，2011））。它也可以用作创建彩色、高分辨率表面网格的起点(Hoppe 等人，1992)。图 5 显示了一个八叉树图，它是通过集成用手持式 Microsoft Kinect 传感器记录的彩色点云创建的。这些数据在 RGBD 数据集的 freiburg1 360 序列中可用（Sturm 等人，2012），并使用 RGB-D SLAM（Endres 等人，2012）对齐。

3.5.2 八叉树层次结构

我们开发了对映射方法的扩展，它利用了环境中的层次依赖性（Wurm 等人，2011）。此扩展维护树结构中的子映射集合，其中每个节点表示环境的子空间。在我们的系统中应用的细分是基于用户定义的输入分段和表达分段之间关系的给定空间关系。

图 6 给出了基于对象位于支持平面顶部的假设的层次结构的图示。在这个应用中，我们首先估计输入中的支持平面。然后，在这些支持平面之上的对象在输入数据中被分割，并被建模为单个的体积子图。因此，该表是位于地板上的子映射，而若干家用对象又被表示为位于桌面上的子映射。

与单个的、整体的环境地图相比，我们的分层方法显示出许多优点：首先，每个子地图被独立维护，并且映射参数（例如分辨率）可以适应于每个子地图。其次，子图可以独立操作。例如，可以移动表示单个对象的一个子映射，而其余子映射保持静态。第三，子映射的层次依赖性可以在层次结构中编码。例如，表上的所有对象都可以与该表相关联，如果表被移动，则对象也随之移动。该方法已在桌

面操作的上下文中进行了评估。以非常精细的分辨率映射表上的对象，而以较低分辨率映射表和背景结构。这种方法导致模型比单个表示完整场景的地图要紧凑一个数量级。

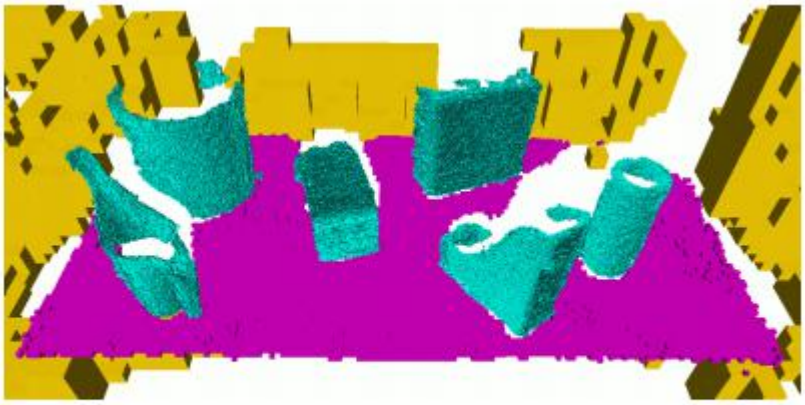

Fig. 6 Hierarchical octree model of a tabletop scene. Background (yellow), table (magenta), and objects (cyan) are represented by individual octree maps of different resolutions.

4 实施细则

4.1 内存高效节点实现

在直接的八叉树实现中，树中的每个节点除了数据有效负载之外还存储其中心位置的坐标、体素大小和指向其子节点的指针。然而，这会导致大量的内存开销。由于节点位置及其体素大小可以在遍历八叉树时重建，因此我们不显式地将此信息存储在节点中以减少内存开销。

一般来说，八叉树节点需要维护他们的孩子的有序列表。这可以通过使用每个节点的八个指针来直接实现。如果对稀疏数据进行建模，那么这些指针（在 32 位体系结构上为 8×4 字节=32 字节）的内存需求将导致显著的内存开销（Wilhelms 和 Van Gelder，1992）。通过每个节点使用一个指向八个指针的数组的子指针（图 7，左），我们克服了这一点。只有当节点确实有子节点并且不为叶节点分配时，才分配这个数组。

因此，八叉树中的任何叶节点仅存储映射数据本身（例如，占用概率）和一个（空）指针。内部节点另外存储八个指针到它们的子节点。在我们的评估中使用的机器人相关的数据集中，80%-85%的八叉树节点是叶子。在我们的实验中，与为每个节点分配 8 个指针相比，上述实现节省 60%-65%的内存。

为了存储每个体素的占用概率，一个浮点值（通常是 4 个字节）就足以表示 log-odds 值。这导致对于内部节点来说，节点大小为 40 字节，对于 32 位架构上的 leaf 来说，节点大小为 8 字节。注意，为了运行时效率，大多数编译器将内存中的成员数据对齐，也就是说，节点的数据被填充为一个字大的倍数（32 位架构上的 4 字节）。64 位架构可以以两倍大小的指针和单词为代价来处理大量内存。在这种架构中，内部节点的内存大小增加到 80 字节，叶子节点的大小增加到 16 字节。注意，大多数编译器将数据结构的实际大小（内部节点 76 字节，叶节点 12 字节）再次填充为字大小的倍数（64 位架构上的 8 字节）。

在我们的方法中，八叉树在设计上是同构的，也就是说，所有节点具有相同的结构和存储占用。虽然内部节点可以通过省略占用信息来潜在地节省 8 字节，但是根据等式(5)或(6)维护它允许多分辨率查询，其中树遍历在固定深度处停止。f 每个对象实例指向虚拟函数表（vtable）的一个额外指针。为了最小化内存占用，我们在八叉树节点实现中避免了这种开销。我们对节点应用直接继承和转换，并且只在八叉树类中使用虚拟继承。此方法导致每八叉树映射仅一个指针大小的开销。

4.2 八叉树类型

在图 8 中，我们框架中最常见的八叉树和节点类型被概括为 UML 图。基本八叉树功能在 OcTreeBase 中实现，基本节点功能在 OcTreeDataNode 中实现。OcTreeDataNode 是根据存储在节点中的数据模板化的，而 OcTreeBase 是根据节点类型模板化的。

OccupancyOcTreeBase 为树实现添加了占用映射功能，比如扫描插入和光线投射。主要的占用八叉树类 OcTree 从 OccupancyOcTreeBase 派生，使用 OcTreeNode 作为它的节点。这种结构允许我们在不同级别上对框架进行灵活的扩展，例如，使用自定义数据扩展节点或向八叉树添加新功能。一个例子是 ColorOcTree 的实现，它使用 ColorOcTreeNodes（如图 5 所示）。这些节点除了占用估计之外还存储颜色，如 Sect.3.5.1

在当前的实现中，最大树深度被限制在 16 个级别。这通过使用可计算的体素地址实现快速树遍历。然而，深度限制也限制了八叉树的最大空间范围。例如，在 1 厘米的分辨率下，地图在每个维度上可以覆盖最大 216×0:01m=655:36m。

虽然这对于大多数室内应用来说已经足够了，但是该实现可以直接扩展到 32 个深度级别，允许在 1cm 的分辨率下覆盖 232×0:01m=42949672:96m。

4.3 地图文件生成

许多机器人应用需要将地图存储在文件中。这包括在建立阶段生成地图，然后由移动机器人用于路径规划和定位的情况。另一个场景是多机器人系统，其中在机器人之间交换地图。无论哪种情况，都需要紧凑的序列化表示来最小化磁盘空间或通信带宽的消耗。

只要映射的最大似然估计对于手头的任务足够，就可以生成最紧凑的文件。在这种情况下，每个节点的概率被丢弃。如上所述，对于机器人系统，例如，在探索期间，没有记录任何信息的卷可能特别感兴趣。为此，我们明确区分空闲区域和未知区域，并将节点编码为地图文件中的已占用、空闲、未知或内部节点。使用这些标签，八叉树映射可以被递归地编码为紧凑的比特流。每个节点仅由其子代的八个标签来表示。从根节点开始，将不是叶子的每个子节点递归地添加到位流中。叶节点不必添加，因为它们可以在解码过程中从它们的标签中重构。图 7（右）说明了比特流编码。每行表示一个具有对应于根节点的上行的节点。最后一行仅包含叶子，因此不添加其他节点。

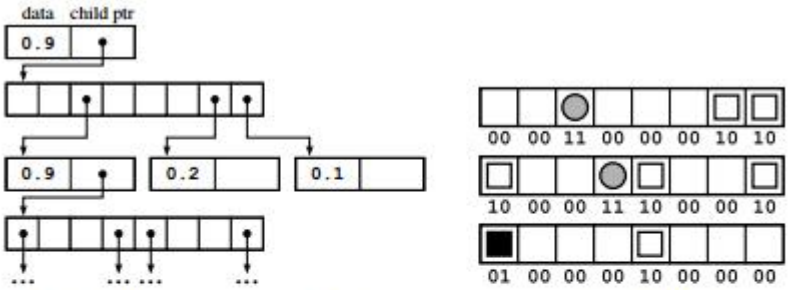

Fig. 7 *Left*: The first nodes of the octree example from Fig. 2 in memory connected by pointers. Data is stored as one float denoting occupancy. *Right*: The complete tree from Fig. 2 as compact serialized bitstream. All maximum-likelihood occupancy information can be stored serially in only six bytes, using two bits for each of a node's eight child labels (00: unknown; 01: occupied; 10: free; 11: inner node with child next in the stream).

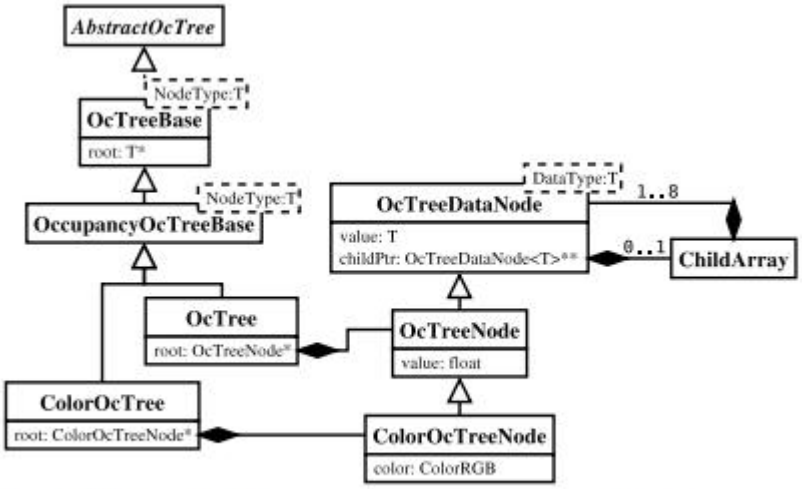

Fig. 8 UML diagram of the most common octree and node classes.

在这个最大似然表示中，每个节点占用 16 位内存，每个子节点占用 2 位，从而得到一个紧凑的映射文件。在我们的实验中，即使对于高分辨率的大型户外环境，文件大小也不会超过 15MB（参见 Sect.5.4 用于详细说明)存在需要存储和维护地图中所有信息的应用程序。这包括将硬盘空间用作辅助存储器，并将映射临时保存到磁盘，直到需要再次访问它们为止。另一个用例是存储附加节点数据，例如颜色或地形信息，这些数据在最大似然编码中将丢失。在这些情况下，我们通过存储节点数据（占用和附加数据）以及指定是否存在子节点的每个节点 8 位来对节点进行编码。然而，正如我们将在实验中展示的那样，这会导致相当大的文件。注意，类似于内存中的八叉树表示，序列化流不包含任何实际的 3D 坐标。为了重构一个映射，只需要知道根节点的位置。节点之间的所有其他空间关系隐式存储在编码中。

4.4 我们的 OctoMAP 实现

OctoMAP 是一个独立的 C++库。它是在 BSD 许可下发布的，可以从 http://octomap.github.com.获得。源代码被完整地记录下来，库使用 CMake 来支持几个平台（Linux 和 Mac OS X 与 GCC、Windows 与 MinGW 或 Visual Studio）。在机器人操作系统（ROS）中，OctoMap 可以作为预编译的 Debian 包提供，例如，用于 Ubuntu 发行版 1。进一步的 ROS 集成可以在 octomap ros 和 octomap msgs 包中获得。借助于 pkg-config 或 CMake 构建系统中的查找包机制，通过对其进行编译和链接，可以容易地将 OctoMap 集成到任何其他框架中。

基于 OpenGL 的 3D 可视化应用程序可以与库一起查看存储的八叉树文件，并从范围数据增量地构建映射，这简化了故障排除和地图数据检查（参见图 9）。它还提供基本的编辑功能。

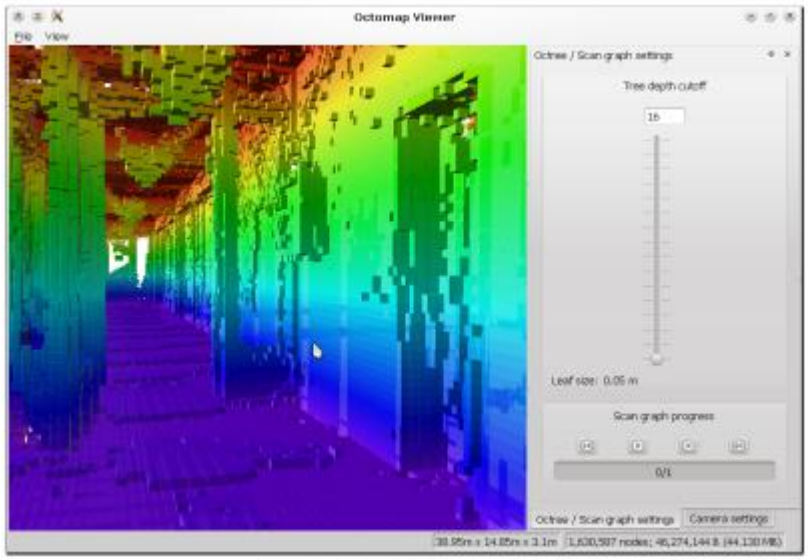

Fig. 9 The OctoMap visualization application octovis

4.4.1 集成传感器测量

通过调用占用八叉树类 OcTree 的 insertRay(·)方法，使用 raycasting 集成各个测距。当沿着射线到传感器原点的所有其他体素被免费更新时，这更新了所占用的测量的终点。

使用 insertScan(·)集成点云，例如来自 3D 激光扫描或立体摄像机的点云。这种批处理操作已经被优化为比从原点跟踪每条射线更有效。最后，通过调用 updateNode(·)，可以用点度量更新八叉树中的单个节点。

4.4.2 访问数据

可以通过搜索它们的坐标来访问个体八叉树节点。对于有效的批查询，我们的实现提供迭代器遍历八叉树，类似于标准 C++容器类。使用这些迭代器，可以查询某个边界框中的所有节点、叶节点或叶节点，或者根据进一步的标准对其进行筛选。光线交叉查询，即将光线从原点投射到给定方向，直到它命中占用的体积，是机器人学中 3D 地图的重要用例。这种查询用于可见性检查或使用范围传感器进行定位。因此，我们在 CASTRAY（·）方法中提供了这种功能。

5 评价

文中提出的方法已经使用若干真实世界数据集和模拟数据集进行了评估。这些实验旨在验证所提出的表示是否满足在导言中提出的要求。更具体地说，我们演示了我们的方法能够充分地建模各种类型的环境，并且它是一个可更新的、灵活的映射结构，可以紧凑地存储。为了评价，我们使用 OctoPAP1.5.12 的当前实现。评估的数据集可在线 3 获得，并且可以利用工具 graph2tree 从 3D 点云转换为八叉树映射，该工具 graph2tree 还打印所有必要的统计数据。

5.2 激光测距数据的传感器模型

OctoMap 可以与任何类型的距离传感器一起使用，只要有反向传感器模型。由于我们的真实世界数据集大部分是用激光测距仪获取的，因此我们采用基于光束的逆传感器模型，该模型假定测量的端点对应于障碍物表面，并且传感器起始端点之间的视线不包含任何障碍。所有卷的占用概率初始化为 P(n)=0:5 的一致先验概率。为了有效地确定需要更新的地图单元，执行射线投射操作，该操作确定沿着从传感器原点到测量端点的光束的体素。为了效率，我们使用 Bresenham 算法的 3D 变体来近似光束（Amanatides 和 Woo，1987）。沿着束的体积如 SECT 中所描述的更新。3.2 使用以下逆传感器模型：

$$\mathrm{L}(n \mid z_t) = \begin{cases} l_{\text{occ}} & \text{if beam is reflected within volume} \\ l_{\text{free}} & \text{if beam traversed volume} \end{cases} \quad (7)$$

在整个实验中，我们使用了 locc=0:85 和 lfree=0:4 的对数比值，分别对应于占空容量和空闲容量的概率分别为 0:7 和 0:4。将钳位阈值设置为 lmin=2 和 lmax=3:5，对应于 0:12 和 0:97 的概率。我们在实验上确定这些值最适合我们使用激光测距仪来映射大多数静态环境的用例，同时仍然保持地图可更新性以适应偶尔的变化。通过适应这些可变的阈值，可以实现更强的压缩。我们将在宗派进行评估。5.6，在映射置信度和压缩之间存在权衡。

当使用扫描激光测距仪时，光线投射操作的离散化效应可能导致不希望的结果。在传感器以浅角扫过平坦表面期间，一次 2D 扫描中所测量的体积可以在后续扫描的光线投射中被标记为自由的。这种效果在图 10 中示出。这种不希望的更新通常会在模型表面产生孔，如图 11 中的示例所示。为了克服这个问题，我们在映射方法中处理来自与单个 3D 点云相同位置的传感器扫描中的扫描线集合。

由于激光扫描仪的测量通常是由障碍物表面的反射引起的，因此我们确保对应于端点的体素在被占用时更新。更准确地说，每当根据等式(7)将体素更新为占用体素时，在地图的同一测量更新中，体素不会免费更新。通过这种方式更新地图，可以防止所描述的影响，并且准确地表示环境，如图 11（右）所示。

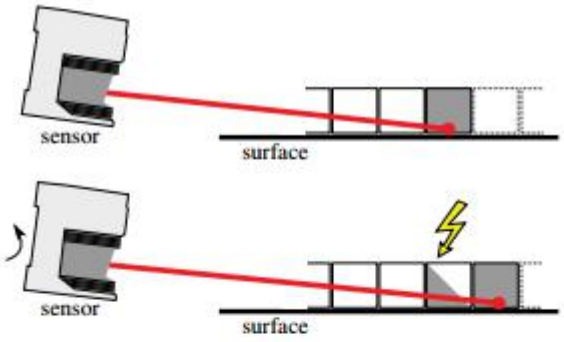

Fig. 10 A laser scanner sweeps over a flat surface at a shallow angle by rotating. A cell measured occupied in the first scan (top) is updated as free in the following scan (bottom) after the sensor rotated. Occupied cells are visualized as gray boxes, free cells are visualized in white.

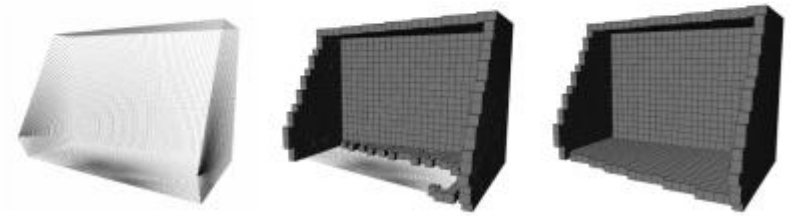

Fig. 11 A simulated noise-free 3D laser scan (left) is integrated into our 3D map structure. Sensor sweeps at shallow angles lead to undesired discretization effects (center). By updating each volume at most once, the map correctly represents the environment (right). For clarity, only occupied cells are shown.

5.2 真实传感器数据的三维模型

在这个实验中，我们演示了我们的方法模拟真实环境的能力。已经使用了各种不同的数据集。注意，

自由空间在实验中明确建模，但是为了清楚起见，在图中没有显示。室内数据集 FR-079 走廊是使用先锋 2 AT 平台在平底倾斜装置上装备 SICK LMS 激光测距仪记录的。通过应用 3D 扫描匹配方法，减少了测径误差。该机器人三次穿越弗莱堡校区 079 大楼的走廊，共完成 66 次 3D 扫描，总计 600 万个终点。在处理这个数据集时，我们把激光束的最大射程限制为 10 米，这样就消除了通过窗户观测到的建筑物外部的杂散测量。图 12 显示了最终的映射。一个相当大的室外数据集被记录在弗莱堡的计算机科学校园 4。它由 81 个密集的 3D 扫描组成，覆盖面积 292m×167m，沿着 723m 的轨迹。在进一步的实验中，我们使用新学院数据集的激光距离数据(Smith 等人，2009)(纪元 C，总共 1400 万个终点)。这些数据是在一个大型的室外环境中记录的，两个固定的激光扫描仪在机器人前进时扫描到机器人的左侧和右侧。对于这个数据集，使用由视觉里程计产生的机器人轨迹的优化估计(Sibley 等人，2009)。所得的户外地图如图 13 所示。最后，我们将 freiburg1 360RGBD 数据集的数据集成到我们的地图表示中，来自 Microsoft Kinect 传感器的总计 2.1 亿个端点（参见 Sect.3.5.1）。最终的地图，如图 5 所示，代表了一个分辨率为 2 厘米的办公室环境。在这个地图中，我们另外存储每个体素颜色信息。

Map dataset	Accuracy	Cross-validation
FR-079 corridor (5 cm)	97.27%	96.00%
Freiburg campus (10 cm)	97.89%	95.80%
New College (Ep. C) (10 cm)	98.79%	98.46%

Table 1 Map accuracy and cross-validation as percentage of correctly mapped cells between evaluated 3D scans and the built map. For the accuracy, we used all scans for map construction and evaluation. For cross-validation, we used 80% of all scans to build the map, and the remaining 20% for evaluation.

5.3 地图精度

这个实验演示了 3D 地图如何精确地表示用于构建该地图的数据。注意，这个特定的评估独立于底层的八叉树结构，因为我们的映射方法能够对与 3D 网格相同的数据进行建模。我们测量精确度作为所有 3D 扫描中正确映射的单元的百分比。如果 3D 地图单元在地图和评估的 3D 扫描中具有相同的最大似然状态（空闲或占用），则其计数为正确映射。由此，扫描被当作插入到已经构建的映射中，即，端点必须被占用，并且沿着传感器和端点之间的射线的所有单元必须是自由的。作为第二个度量，我们在构建映射时通过跳过每次第 5 次扫描来交叉验证映射，并使用这些跳过扫描来评估正确映射的单元格的百分比。

表 1 中的结果显示我们的映射方法准确地表示了环境。剩余的误差很可能是由于传感器噪声、离散化效应或者不完全完美的扫描对准。交叉验证结果只损失很少的精度，这说明概率传感器模型可以得到逼真的预测结果。

5.4 内存消耗

在这个实验中，我们评估我们的方法的内存消耗。在不同的树分辨率处理多个数据集。我们分析了有八叉树压缩和不执行八叉树压缩的表示的内存使用情况，以及将每个节点转换为完全空闲或占用的最大似然压缩。

为了进行比较，我们还确定了在内存中线性初始化的最小尺寸的最佳对齐的 3D 网格所需的内存量。根据教派。4.1，在 32 位架构上存储在八叉树中的占用的内存消耗由以下给出

$$\mathrm{mem}_{\mathrm{tree}} = n_{\mathrm{inner}} \times 40\,\mathrm{B} + n_{\mathrm{leafs}} \times 8\,\mathrm{B}\,, \qquad (8)$$

其中 nNIN 是内部节点的数量，而 NLIP 是叶节点的数量。存储相同信息（占用概率为一个浮点数）的最小 3D 网格的大小由

$$\mathrm{mem}_{\mathrm{grid}} = \frac{x \times y \times z}{r^3} 4\mathrm{B}\,, \qquad (9)$$

其中 x;y;z 是地图在每个维度上的最小边界框的大小，r 是地图分辨率。此外，我们使用完整的概率模型和 Sect.4.3，并对生成的文件大小进行评估。
表 2 给出了示例性分辨率的内存使用情况。可以看出，高压缩比可以达到，特别是在大型户外环境。在这里，

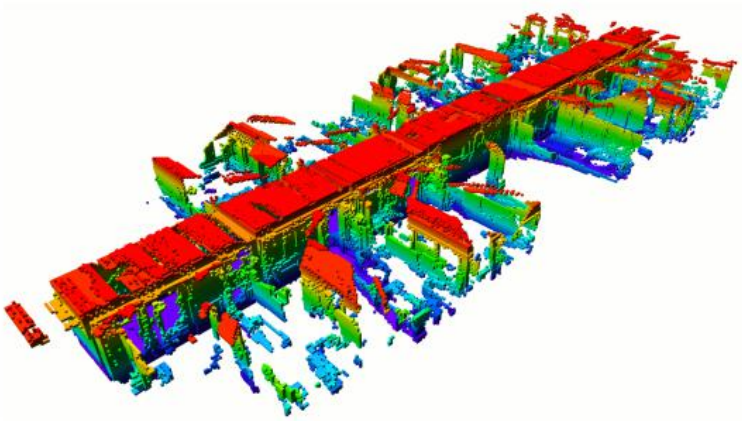

Fig. 12 3D map of the FR-079 corridor dataset, as seen from the top. The structure of the adjacent rooms has been partially observed through the glass doors (size of the scene: 43.7 m × 18.2 m × 3.3 m).

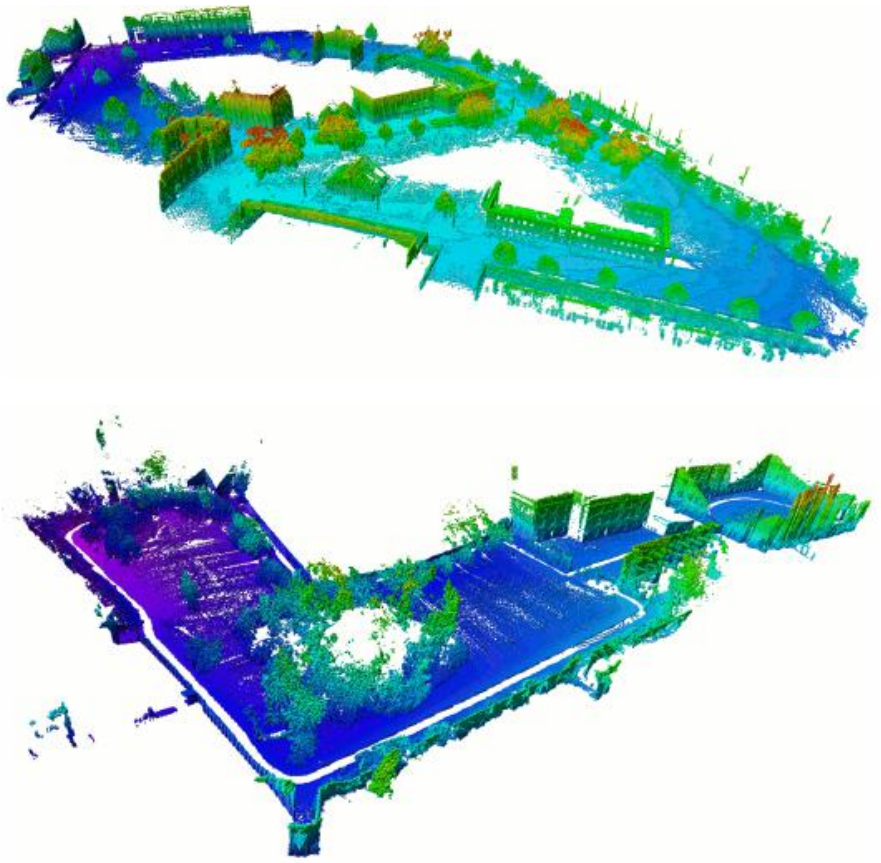

Fig. 13 Resulting octree maps of two outdoor environments at 0.2 m resolution. For clarity, only occupied volumes are shown with height visualized by a color (gray scale) coding. Top: Freiburg campus dataset (size of the scene: 292 m × 167 m × 28 m), bottom: New College dataset (size of the scene: 250 m × 161 m × 33 m).

				Memory w. octree compression [MB]			File size [MB]	
Map dataset	Mapped area [m^3]	Res. [cm]	Mem. 3D grid [MB]	None	Pruned	Max. likelih.	Full	Lossy
FR-079 corridor	43.7 × 18.2 × 3.3	5	78.88	73.55	41.62	24.72	15.76	0.67
		10	10.01	10.87	7.22	5.02	2.70	0.14
Freiburg campus	292 × 167 × 28	10	5162.90	1257.57	990.66	504.76	379.70	13.82
		20	648.52	187.93	130.24	74.12	49.68	2.00
		80	10.58	4.55	4.12	3.09	1.53	0.08
New College (Ep. C)	250 × 161 × 33	10	5058.76	607.92	395.42	230.33	148.75	6.40
		20	633.64	91.33	50.57	35.95	18.65	0.99
		80	10.13	2.34	1.79	1.69	0.63	0.05
freiburg1_360 (RGBD)	7.9 × 7.3 × 4.6	2	252.99*	159.97*	45.52*	20.05	21.59*	0.52
		5	16.19*	11.24*	4.55*	2.52	2.11*	0.07

Table 2 Memory consumption of different octree compression types compared to full 3D occupancy maps (called 3D grid) on a 32-bit architecture. Octree compression in memory is achieved by merging identical children into the parent node (called Pruned). A more efficient but more lossy compression in memory is achieved by converting each node to its maximum-likelihood value (completely free or occupied) followed by pruning the complete tree. A maximum-likelihood tree containing only free and occupied nodes can then be serialized to a compact binary file format (called Lossy file). (*): Voxels contain the full color information from the RGBD dataset.

其中 x;y;z 是地图在每个维度上的最小边界框的大小，r 是地图分辨率。此外，我们使用完整的概率模型和 Sect.4.3，并对生成的文件大小进行评估。

表 2 给出了示例性分辨率的内存使用情况。可以看出，高压缩比可以达到，特别是在大型户外环境。在这里，修剪将合并大量空闲空间卷，未知空间的区域不使用任何内存。

注意，分辨率为 10cm 的室外数据集的 3D 网格甚至不能适合 32 位机器的可寻址主存储器。另一方

面，我们的地图结构也能够模拟具有中等存储要求的精细分级的室内环境。在非常有限的空间中，最佳对齐的 3D 网格可能比未压缩的映射八叉树占用更少的内存。然而，一旦压缩技术被使用，这种效应就会减弱。

随着时间的推移，存储器消耗的演变如图 14 所示。当机器人探索新领域时，内存使用量增加（在 FR-079 走廊扫描 1-22 和 39-44，在弗莱堡校园扫描 1-50 和 65-81）。在剩余的时间里，重新访问之前映射的区域，其中由于修剪，内存使用保持几乎恒定甚至减少。
正如预期的那样，内存使用量随着树分辨率的增加而呈指数增长。这种效应在图 15 中可以看到，我们在图中使用了对数标度。表 2 给出了序列化二进制最大似然图（表示为“Lossy”）和全概率模型（“Full”）的文件大小。注意，通过使用标准文件压缩方法，可以进一步压缩映射文件。即使是弗莱堡大学和新学院相当大的户外数据集的地图，文件大小也不到 14MB。

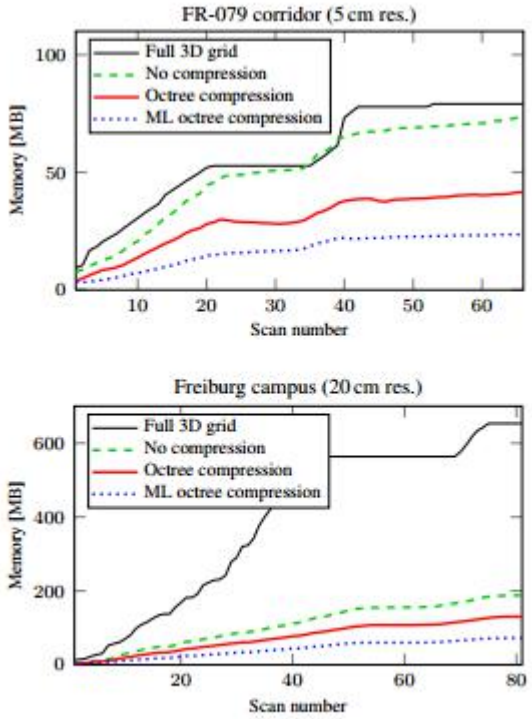

Fig. 14 Memory usage while mapping the two data sets FR-079 corridor and Freiburg campus.

5.5 运行时间

在以下实验中，我们分析了在框架中集成和访问数据所需的时间。对于各种地图数据集，所有运行时都在标准桌面 CPU（Intel Core i7-2600，3.4GHz）的单个核心上进行评估。

5.5.1 地图生成

首先，我们分析了通过集成范围数据来生成地图所需的时间。这个时间取决于地图分辨率和集成的光束长度。我们处理了 FR-079 走廊和弗莱堡校园数据集，它们都具有全激光范围（高达 50 米）和具有有限最大范围 10 米的多个分辨率。在图 16 中给出了一个梁的平均插入时间。
在我们的实验中，3D 扫描通常由大约 9 万到 25 万个有效测量组成。通常，这样的扫描可以在不到一秒钟的时间内整合到地图中。这表明，我们目前的实现甚至可以应付 RGBD-相机的要求数据，在快速帧速率，尽管在较短的范围，输出高达 30 万点。
利用长测量光束和大型户外区域，如弗莱堡大学校园数据集，可以通过限制地图更新范围来获得加速。然而，在只有很少的传感器光束到达的室内，通过限制传感器范围没有明显的加速。

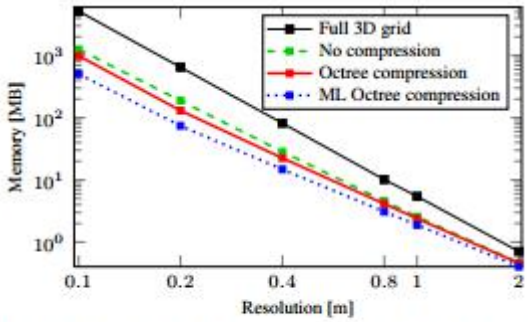

Fig. 15 Effect of resolution on memory usage of the Freiburg campus dataset. Note that a logarithmic scaling is used.

5.5.2 地图查询

我们使用迭代器（参见 Sect.4.4.2）。查询的深度可以在运行时受到限制，在我们的数据结构中，这相当于粗略映射中的映射查询。当粗略的分辨率足够时，这允许更高效的树遍历。图 17 显示了遍历多个映射到对应于完整映射分辨率（深度截止=0）的最大树深度 16 的时间。该图还给出了当查询深度有限时查询所有叶节点的时间。深度截断中的每个增量使最小体素的边缘长度加倍，并使遍历速度提高大约两倍。可以看出，地图遍历是有效的。即使在全地图分辨率下，包含 1087 014 个被占用和 3 377 882 个自由叶节点的弗赖堡校园的大地图也可以在 51ms 内通过。

5.6 夹紧参数

最后，分析了钳位阈值对地图精度和压缩的影响。由于这些阈值提供了占用概率的下界和上界，所以与没有钳位的完整映射相比，接近 P=0 和 P=1 的信息丢失。钳位映射表示完整映射的近似，因此我们使用在完整映射上求和的 Kullback-Leibler 散度（KLD）作为度量。由于占用是具有自由和占用的离散状态的二进制随机变量，因此可以通过对所有地图节点 n：

$$\mathrm{KLD}(M_f, M_c) = \sum_n \left(\ln\left(\frac{P(n)}{Q(n)}\right) P(n) + \ln\left(\frac{1-P(n)}{1-Q(n)}\right)(1-P(n)) \right), \tag{10}$$

其中 p（n）是 Mn 中节点 n 的占用概率，Mc 中的 q（n）。一系列占用结果的范围从[0:1]（没有夹紧，无损耗）到[0:4:0:6]（强夹紧，大部分损耗），不同的映射可以在图 18 中看到。我们选择的默认阈值[0:12:0:97]的值显示为细水平线，虚线蓝色表示内存消耗，红色表示 KLD。

选择这个夹紧范围主要是为了在基于激光的绘图和环境的偶尔变化（例如人们通过扫描或关门）的情况下工作得最好。可以看到，通过更高的钳位可以获得更强的压缩，而代价是失去映射置信度。在最退化的情况下，一个传感器更新就足以将体素标记为完全空闲或占用，从而失去用概率更新过滤噪声的任何能力。注意，虽然钳位对地图压缩是有益的，但是即使没有钳位，无损压缩地图也比 3D 网格小（参见表 2）。

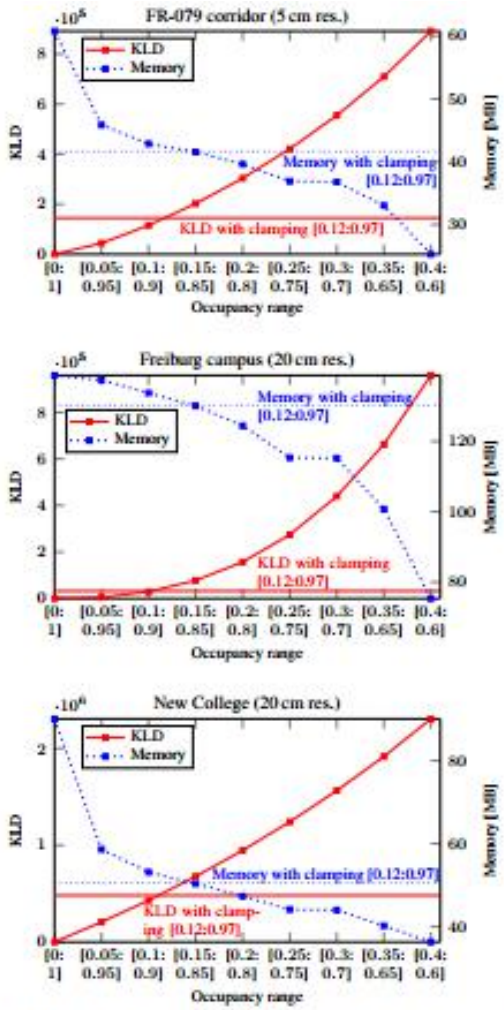

Fig. 18 Effect of different clamping ranges on map compression and accuracy in our three datasets. A higher clamping, resulting in a smaller occupancy range, increases the efficiency of the octree compression (memory consumption, dashed blue). The Kullback-Leibler divergence (KLD, red) measures the information loss between the unclamped map with full probabilities in [0:1] and a clamped representation. Our default clamping range [0.12:0.97] is shown for comparison by horizontal lines in blue (dashed) for memory consumption and red for the KLD.

6 案例研究

自从在 2010 年首次引入（Wurm 等人，2010）以来，OctoMap 框架受到了极大的关注，并且已经在多个应用程序中使用。这些包括 6D 定位(Hornung 等人，2010)、带有飞行器的自主导航(Heng 等人，2011；Muller 等人。，2011），使用类人机器人的自主导航（Owald 等人，2012；Maier 等人，2012），3D 探索（Shade 和 Newman，2011；Dornhege 和 Kleiner，2011），3DSLAM（Hertzberg 等人，2011），3D 臂导航（Ciocarlie 等人，2010），语义映射（Blodow 等人，2011），以及导航杂乱环境中的 N（HONNUG 等，2012）。在下文中，我们将更详细地描述这些用例中的一些，以便演示 OctoMap 库的多功能性和集成的简易性。

6.1 三维定位技术

在以前的工作中（Hornung 等人，2010），我们开发了一种基于 OctoMap 作为 3D 环境模型的定位方法。该方法基于二维激光测距、IMU 和联合编码器数据，利用蒙特卡罗定位技术跟踪复杂室内环境下仿人机器人的 6D 躯干姿态。对于粒子滤波观测模型，我们首先使用端点模型，然后结合用于局部细化的视觉观测，使用优化的光线投射方法(Owald 等人，2012)。由此产生的定位是高度精确的，甚至能够使人形爬上螺旋楼梯。我们的实现是可用的开放源码 5，并且在 OctoMap 中使用光线投射功

能（参见 Sect.4.4.2）。这使得其他机器人定位系统能够重复使用。

6.2 桌面操作

ROS 对撞机 PACKAG6 构建基于 3D 点云的碰撞地图。来自多个源的传感器数据，例如倾斜激光和立体相机，使用 OctoMap 进行融合。八叉树节点被扩展为存储时间戳属性，该属性允许在动态变化的环境中逐渐清除节点。这种新的碰撞地图使 ROS 手臂导航和抓取管道（Ciocarlie 等人，2010）能够动态地响应变化并应对传感器噪声。与以前的固定大小的体素网格相比，新的实现允许初始无限的工作空间、集成来自多个传感器的数据，并且它更加节省内存。

6.3 杂乱环境中的导航

OctoMap 还用于创建用于移动操作的导航模块。在这个项目中，一个 PR2 机器人从一张有两只胳膊的桌子上拿起大物体，然后通过窄通道把它带到另一张桌子上（Hornung 等人，2012）。该系统集成了 3D 传感器数据在 OctoMap。然后，根据机器人的全部运动学配置和所附物体，使用得到的 3D 占用图来执行碰撞检查。多层投影 2D 地图和使用运动原语的任意时间规划器允许几乎实时地规划具有有界次最优性。在 ROS7 中，导航系统和基于 OctoMap 的增量式映射框架都是开源的。

7 结论

在本文中，我们提出了一个用于三维映射的开源框架 OctoMap。我们的方法使用基于八叉树的有效数据结构，它支持紧凑的内存表示和多分辨率映射查询。使用概率占用估计，我们的方法可以表示包括自由和未知区域的体积三维模型。所提出的方法使用允许无损压缩方案的有界每卷置信度，并导致内存使用显著减少。我们评估了我们的方法与各种现实世界的数据集。结果表明，我们的方法能够以精确的方式对环境建模，同时最小化内存需求。

OctoMap 可以很容易地集成到机器人系统中，并且已经成功地应用于各种机器人项目中。该实现可作为 BSDCONVICE C++源代码使用。数据集可以在线验证我们的实验结果并与之进行比较。作者要感谢 J.Muller，S.Owald，R.B.Rusu，R.Schmitt 和 C.Sprunk 对十月地图图书馆的卓有成效的讨论和贡献。