上下文敏感的导航分层代价地图

(Layered Costmaps for Context-Sensitive Navigation)

原作： David V. Lu
译作： 罗辉武*

版本：0.1
翻译日期：2019 年 07 月 05 日

摘　要

许多导航系统，包括普遍使用的 ROS 导航功能包，在单个代价地图上规划机器人的移动路径，大部分信息皆存储于单个网格单元中。这种方法对需要生成最小长度、无碰撞路径的应用非常成功，但是在面对动态的、充满人的环境时，需要对 costmap 的网格单元属性扩展到占用空间 (occupied) 或自由空间 (free) 之外，此时会面临非常多的挑战。

我们创建和实现了一种称为分层代价地图 (layered costmap) 的新方法，该方法通过将 costmap 的数据处理分离到语义不相关的图层 (layer) 进行。每个 layer 仅跟踪一种类型的障碍物或者约束，修改用于路径规划的主代价地图 (master costmap)。我们展示了如何将算法与开源的 ROS 导航功能包进行集成，证明了我们的方法比当前使用单源代价地图 (monolithic costmap) 的方法更容易在特定的环境进行调整 (tuning)。这种设计还可以在实际使用中实现加速路径规划的效果，并且可以清晰地表达原始架构所强调的信息。新算法还可以对复杂的代价值进行表达，便于在使用上下文信息导航的环境中创建导航行为。

目录

*Email: huiwu.luo@aliyun.com

1 简介

过去的几十年里，导航算法变得越来越复杂。它们处理大量传感器的数据，高精度跟踪障碍物位置和自由空间位置。结合正确的路径规划算法，它们可以很熟练地在环境中进行巡航。然而，这类导航算法的相当大一部分会遇到同样的问题：算法皆是为了产生无碰撞的自由路径这个目标而进行有效寻优。

这种算法在许多实际用例或抽象环境下表现很好，如果都是要求从点 A 走到点 B。对于其他实际用例来说还不是特别好。像在人口稠密的动态环境中移动的机器人，需要将更复杂的约束集成到优化问题中。从一个点走到另一个点只是一个大区域上下文的一部分。机器人围绕障碍物移动仅是为了避免碰撞是不够的; 机器人必须根据上下文语义的不同区别对待该障碍。例如，大多数情况，在远离桌子几厘米之外行走是完全没有问题的。然而，紧贴着人群行走更是不希望发生的。如果导航算法平等对待所有感知到的障碍物，此时规划器可能无法选择出一条正确的行走路径。

规划路径时，除了尊重他人的个人空间这种场合之外，还有许多其他的场合表明：选择最短的无碰撞路径可能不是最理想的。考虑人群中人经常行走的路线信息，此时应首选避免可能有障碍的长度更长的路径。实用情况下，机器人还必须考虑避免进入有潜在危险的区域，例如厨房。虽然所规划的路径是有效路径，但这些区域段应该带上通行代价。即使对简单情况下的一些因素也应该这么考虑，比如偏向在过道右侧移动。机器人选择走哪条路取决于在大环境提取的额外的上下文信息。

路径规划器使用的环境信息皆存储于一张代价地图上。在传统的代价图中，所有的数据都存储在一个单元网格中，我们称之为单源代价地图 (Monolithic Costmap)。单源代价地图由于只需要在一个地方读写代价的值，非常简单，是一种很流行的技术。它的一个问题是代价地图中相当多的代价值被丢弃，使得对地图的周期维护变得越来越困难。

本文中，我们介绍一种将额外上下文信息集成到代价地图的方案，方案采用了一种称之为分层代价地图 (layered costmaps) 的新方法。通过使用 ROS 导航框架作为试验，我们展示分层代价地图在复现以前导航算法的功能的同时，能增加处理更多上下文信息的灵活性。图1展示了一种分层代价地图的可能的配置方式。我们将讨论该算法和数据结构，以及相对以前方法的改进点。之后，我们将对可以加入到新旧代价地图中的不同层和它们所集成的环境上下文进行检查 (以讨论它们对规划的路径的影响)。

2 相关工作

这项工作的重点是适用于规划路径的地图的网格表示方式。20 世纪 80 年代，由 Moravec 及其合作者在卡内基梅隆大学 (CMU) 开发的占据网格 (occupancy grid) 是现代代价地图鼻祖[1, 2]。占据网格的语义值很直观：每个单元格的值表示该单元格存在障碍物的概率，因此其更新过程是贝叶斯规则的直接应用。Konolige[3] 和 Thrun[4] 改进了概率模型，以便更好地定位障碍物。基于网格的代价地图表示方法 (其中网格值表示的不是概率而是通行代价) 被证明是很实用的方法，尤其是当障碍物的位置是固定不定的情况下 (不使用声纳探测仪)。过去，代价地图主要是二进制的，单元格要么是被占据状态 (occupied)，要么是自由 (free) 状态。现在的情况是，随着将更多的复杂代价的值加入代价地图，必然导致代价地图的语义信息过于混乱。对于那些非致命的 (non-lethal) 代价值，即值介于 occupied 与 free 中间，通常表示软约束 (soft constraints)。自主驾驶车辆采用这样的值进行优化，从而能在街道上正确的一边进行行驶或采取其他优先驾驶行为[5]。Gerkey 和 Agrawal[6] 在代价图中用不同的代价值表示不同类型的地形及其通行性。软约束也可用于基于人机交互的约束。Sisbot 等人的 Costmap 系统[7] 就考虑到个人的活动空间和视野，而 Kirby 等人[8] 等人则对人在路的右边行走行为进行了建模。Svenstrup 等人[9]、Scandolo 和

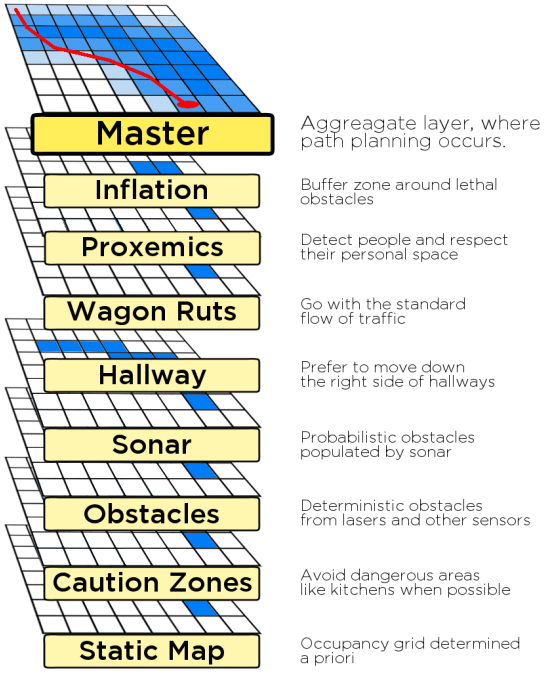


图 1: 一组 costmap 图层，展示了使用分层 costmap 方法可实现的不同上下文行为。

Fraichard[10] 开发了更为复杂为感知人类行为的导航代价计算方法。

3 单源代价地图 (Monolithic Costmap)

单源代价地图 (Monolithic Costmap), 所有数据皆存储于单个网格中，是当前工程实现 costmap 的基础，包括普遍使用的 ROS[11]。本文着重于 ROS 功能包的算法及其实现，这是因为 ROS 已获得大量使用，运行于上十种机器人硬件上[1]。ROS 使用一张单源代价地图进行全局规划，同时在局部规划中使用另一张单源代价地图进行规划。

单源代价地图被证实在规划最短无碰撞路径 (collision-free path) 方面是非常有效的。将初值写入代价地图的做法非常直观，但由于存储空间非常有限，更新过程存在问题，这就限制了它所能实现的功能类型非常有限、代价地图的利用效率不高、可扩展性不强等缺陷。单源代价地图的主要缺点如下：

1) 更新期间的信息很有限：单源代价地图的一个很大的缺陷是代价地图中的大部分信息存储于一个位置。考虑一个简单的传感器数据与全局代价地图中已有值之间冲突的例子。传感器数据表明地图中的某个区域是开阔的，而代价地图上却说明该区域存在障碍。更新代价地图的正确方法应与数据的来源及附加的语义信息有关。一种情况是先前的值是已经移动的人的位置；所以，正确的处理方式是将代价图中标记为“lethal”(致命区域) 的值更新为“free”(自由区域)，使得机器人可以在新腾出的空间通行。但是，一个同等有效的情况是：代价地图中的值来自静态地图的值，静态地图在创建的时候就包含了传感器无法看到的障碍物，例如玻璃墙。此种情况下，代价地图中应保留“lethal”值。

在单源代价地图中是无法区分这两种情况的，因为两种情况都在代价地图中将它标记为“lethal”。只要代价地图中的值为单种类型值，代价地图就会从数据中删除代价地图所表示的任何语义信息。

[1] http://wiki.ros.org/navigation/RobotsUsingNavStack

在单源代价地图正确处理三维障碍物的数据也是有问题的。ROS 的早期开发人员在使用像倾斜的激光传感器这种三维传感器时也遇到该问题。假如障碍物数据仅存储在单源代价地图中，可通过清除观察来移除不同高度的障碍物。因此，他们引入了体素网络 (voxel grids) 以跟踪其他信息[12]。这种解决方案适合在该用例中扩展单源代价地图的功能，但没有推广性。

随着代价地图的数据源和类型的数量增加，这种有限信息的存储结构更成为问题。考虑存在多个"non-lethal"(非致命区域) 的数据源，每个数据源带有自己的语义信息。如果在单源代价地图中将多个"non-lethal" 的值简单相加，其中一个值发生变化将导致每个语义信息需要重新计算。

2）固定更新区域：单源代价地图缺少语义信息，很难判断代价图中任何特定代价的持续时间。因此，如果更新的区域需要后处理 (post-processing) 或要发布到某些外部源，则没有确定的方法来确定最近更新的范围。处理该问题的一种无效方法是保守估计覆盖整个区域的地图区域，这些区域可能已经更新，这是 ROS 的实现所要做的。实际上，这意味着要在机器人周围更新大约 $6m \times 6m$ 的方形区域，无论实际更新了多少空间。

3）临时更新过程：缺乏用于维护和更新代价地图的既定范例，导致只能采用临时的方法进行实现。目前为止，由于实际使用的数据源数量相对较少，该方法已经奏效；但随着数据源数量的增加，这种方法也会变得不可行了。为了确保以正确的方式组合数据，每个数据源都需要清晰了解其他数据源的使用过程。

即使在之前的工作中为计算成本定义了有用的算法，但它们与完整的代价地图进行集成的过程通常是不透明的。如果没有关于如何更新代价地图的信息，则无法准确复制结果。

4）固定的语义解释：除了限制代价地图包含的信息之外，单源代价地图还限制了可以使用的信息类型。单源代价地图也只能对代价地图中的值进行单一解释。代价地图的原始占用网格定义使用了概率解释。即，该值表示在某个位置的代价/惩罚量。对于单源代价地图，并没有明确定义如何将概率数据源与基于代价的数据源结合的方式。

ROS 的代价地图实现还有其他问题，因为它接受的唯一信息类型为是否二进制障碍数据，即要么存在明显的障碍，要么为绝对自由空间。不适合在现有的框架上增加非致命 (non-lethal) 的代价属性。因为 costmap 只有一种数据类型，所以语义信息是固定不变的。

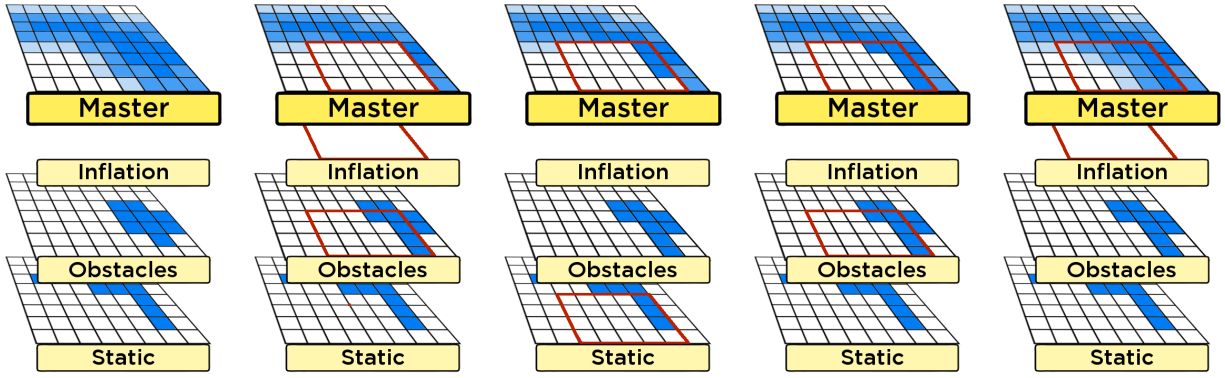
4 分层代价图

4.1 数据结构和更新算法

为了消除上一节中介绍的种种限制，我们设计了分层代价地图 (layered costmap)。数据结构仍包含用于路径规划的二维代价网格。最重要的区别在于如何填充此主代价地图 (master costmap) 的值。分层代价地图不是直接在网格中存储数据，而是维护有序的地层列表 (layered list)，每个层都跟踪与特定功能相关的数据。然后将每层的数据累积到 master costmap 中，这需要在有序的 layered list 进行两次遍历。

在第一次遍历中，即 `updateBounds` 方法中，每个层都被轮询以确定代价地图的更新量。按顺序遍历各层，依次为每层提供前一层需要更新的边框（最初为空框）。每层都可以根据需要扩展边框。第一次遍历会生成一个边框，用于确定 master costmap 需要的更新量。第二次遍历将调用 `updateValues` 方法，在此期间，每个连续层将更新 master costmap 的边框区域内的值。图2展示了使用一组基本代价地图进行的复制行为来说明更新算法。

某些图层会维护自己版本的 costmap 用于缓存结果。这是数据结构维护语义信息的主要方式之一。



(a) Initial Costmap Values (b) UpdateBounds (c) UpdateValues: Static (d) UpdateValues: Obstacles (e) UpdateValues: Inflation

图 2: 更新算法。(a) 中，分层 costmap 由三个功能层和一个 master costmap 组成。障碍物层和静态层保留有自己层的网格副本，而膨胀层没有。为更新 costmap，算法首先在每个图层上调用 `updateBounds` 方法 (b)，从有序列表中的第一个图层开始 (显示在底部)。为了确定新更新区域的边界，障碍层根据传感器的新数据更新自己的 costmap。结果为一个每层都需要更新的区域的边框。接下来，每个层依次调用 `updateValues` 方法更新 master costmap 中的边框区域，从静态层 (c) 开始，然后是障碍层 (d) 和膨胀层 (e)。

例如，障碍层保留与 master costmap 相同大小的私有 costmap，以存储所有光线跟踪 (ray-tracing) 和标记步骤的结果。由于私有 costmap 中的值只能由特定层访问，因此存储在其中的信息不会因为其他数据源的写入操作而丢失。反而最小化了 costmap 由于先前被覆盖写入新值而必须重新计算的频率。

其他层不需要在循环之间保留太多数据，并且将在每个回合中使用其数据更新 master costmap，或者仅对其他层已经写入 master costmap 的数据进行操作。

图2中的例子展示以前用于生成 costmap 的具体操作方法 (ad hoc approach) 如何被细化为一个整洁、良好定义的过程。关于每层如何改变 master costmap 的详细说明，请参照第6节。

4.2 优点

分层代价地图方法显然解决了单源代价地图的局限性。

1) 更清晰的更新步骤：在分层的代价地图方法中，不同类型的代价地图信息添加到单独的层中，使得更新步骤更为清晰。如果希望将静态障碍物、从激光检测到的障碍物、及从声纳检测到的障碍物区别处理，则将这些障碍物存储在各自的层中显然简化了信息的记录方式。每层只需要保持同类型的信息一致即可。

分层的代价地图还消除了同源代价信息之间的争用。每层只需在插入同类型的新信息时更新。如果一个层保有很大的静态区域，则另一个层在更新某个子区域时无需重新计算。静态层只需要更新到主代价地图，然后更新可以移到下一层。

这种更清晰的关注点分离技术使得各个代价地图组件更容易调整。初始用户可以一次只引入一个层，并依次调试每层。

2) 区域动态更新：与单源代价地图中每一轮更新中更新固定区域或更新未知区域相反，由于 updateBounds 通过图层传递，分层代价地图仅更新各个图层认为需要更新的区域。这为 costmap 提供了额外的稳定性，保证只更新边界框内的值。此外，更新量较小的地图可能更有效。

3) 有序更新：与单源代价地图未定义单元格的更新顺序相反，分层代价地图具有显式有序的特点。

在我们的例子中，显然，膨胀层膨胀的是障碍物层和静态层的值，因为膨胀层在有序表中是排在后两者之后的。此外，层之间的交互是显式指定的。每个 costmap 可以配置为将前一个值和层的值合并为最大值、最小值或以两者为输入的的其他数学函数。

4）灵活配置：最后，也最重要的，分层代价地图方法的功能是无限的。实现与先前实现的等效行为所需定义的层仅仅是个开始。可以将机器人操作员所需的多个层添加到分层代价地图中。结果是，各个层可以实现任意复杂的逻辑来更新代价地图，从而扩展了代价地图的语义可能性。每个层还可以独立具有自己的数据表示方式，使得概率占据网格可以与基于代价的层一起存在于它们自己的层中。

5 对比

5.1 实现细节

尽管分层代价地图的算法和数据结构与系统无关，但由于平台的普遍性，我们将重点实现系统与 ROS 导航功能包以演示该方法的功能。分层代价地图的实现使得 costmap 2d 的 API 保持较好的通用性，与导航功能的其他代码一样，采用 C++ 实现；每层皆如此。

工程化实现层非常简单。首先，创建一个新类，它继承自 `costmap 2d::Layer` 类。这意味着需要实现初始化函数（其中层可以独立订阅 ROS 生态系统中的任何数据源）、`updateBounds` 函数 (更新边框)、和 `updateCosts` 函数 (将值写入到 master costmap 中)。可以使用简单的运行时参数更改方法将独立编译的图层插入到分层的 costmap 中。单独编译的层可以在运行时更改参数后以插件形式插入到分层代价地图执行。

在分层代价地图之前，costmap 类在处理是否存在静态地图、或者是否启用三维跟踪障碍物这些不同的情况时，是把它们作为特殊情况处理的。现在的做法变成了对全局代价地图和局部代价地图配置不同类型的图层来进行处理。

我们在 Gazebo 的重复模拟试验中运行两种代价地图。正如 ROS 导航功能包的基准测试采用的评估那样[12]，在 PR2 机器人上测试。经数百次模拟试验，我们发现两种实现方式生成的路径在路径长度、完成时间、及与障碍物的关系方面并没有明显差异。

5.2 时间对比

统计分析中最重要的数据指标之一是 costmap 的更新周期的平均运行时间。由于局部规划需要根据运行速度和新出现的障碍物的情况调整速度，所以更新过程必须非常快。需要达到的标准是至少 5 Hz 的更新频率（和前面的实现采用的一致），将单个周期的运行时间限制在 0.2 秒内。根据环境的细节和运行代价地图的系统，使用单源代价地图的方法能够超过一个量级或两个量级。毕竟分层的 costmap 实现也依赖于这些变量。总之，基于分层思想的实现在某些情况下运行得很快，仅在极少数情况下运行速度变慢。

模拟实验中，在单源实现和分层实现这两种情况下，全局代价地图的平均更新时间分别为 0.00166 秒和 0.00236 秒; 而局部代价地图的更新时间分别为 0.00493 秒和 0.00463 秒。使用单边 t-检验，我们发现局部规划的平均更新时间没有显着差异。在采用分层 costmap 实现时，全局规划的更新时间明显变慢 ($p < 0.001$) 。但是，我们确信这是稀疏模拟环境带来的问题。在这种模拟中，机器人被置于完全开放的环境，而不是一个单独的、机器人与目标点存在一个相对较小障碍物的环境。因此，机器人的激光读

数在多数方向上会延伸到激光所能探测的最大距离，这会导致各地图层实现需要更新一个很大的区域。每一地图层都需要更新该区域，从而降低整体更新速度。

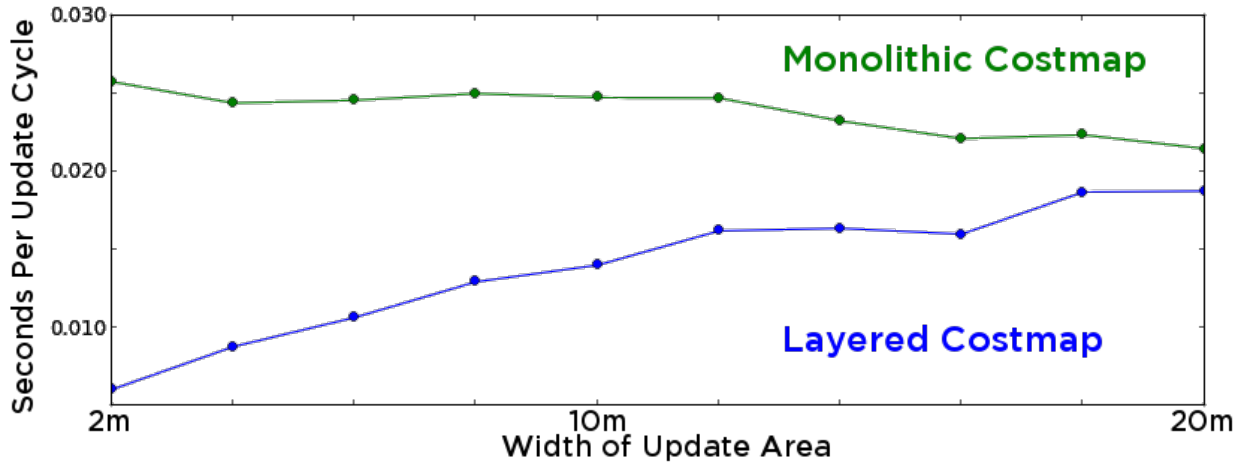


图 3: 更新时间 vs. 更新区域 - 在单源实现的版本中，过程中更新区域大小保持不变，因而计时大致一致。在分层的实现版本中，更新区域会变化，因而计时上也会发生变化。

我们还在一个比较絮乱的模拟环境对机器人作了测试[2]，其中机器人的所有侧面都被一定距离的墙所包围。由于墙壁与机器人很近，因此需要更新的区域要小得多。如图3所示，在这种情况下，分层的代价地图方法更快。随着更新区域的增长，单源版本的更新时间大致保持不变，而分层版本的更新时间会增长以匹配更新的单元格数量发生增加的现实情况。鉴于代价系统的设计多用于絮乱、快速变化的环境，分层代价在这些环境中的速度表现与实际情况较为吻合。

5.3 在真实世界中导航

除了模拟测试外，我们还在真实环境中用 PR2 平台测试了分层代价地图的性能。测试地点主要在 Willow Garage 的办公环境中进行。使用地图层来模拟单源的 costmap 结构，能够让 PR2 成功复现先前实现的所有路径规划行为。但是，当我们修改地图子层以提取使用单源代价地图无法实现的行为时，产生了激动人心的结果。

首先，通过分离静态层 (static layers) 和障碍层 (obstacle layers)，我们测试障碍层是否具有覆盖静态地图的功能。正如第3节所述，使用单源代价地图进行导航的一个问题是可能会错误清除静态地图的某些部分，导致机器人规划出一条穿过实心墙的路径。仅通过允许障碍物层进行光线跟踪并清除检测到的障碍物（不是静态地图上的障碍物），墙从未在主代价地图中被清除，这就消除了令人尴尬的穿墙通行行为。

新层的引入也使以前不可能出现的行为变成可能。激励我们对代价地图进行调查的背景用例是希望创建出类似于第2节相关工作介绍的具有社会行为意识的机器人导航 (socially-aware robot navigation) 方法。我们成功地将这样的地图层集成到 PR2 的路径规划中[3]。我们创建的新地图层和其他地图层的详细信息将在下一节中详细说明。

[2]译者注：絮乱区域是相对于平坦、开阔区域而言

[3]视频:https://www.youtube.com/watch?v=Pzx0yyEcfgI

6 地图层

除了允许分层代价地图复制其他代价地图的功能，地图层的主要优点在于能够轻松集成其他地图层，这些地图层将遵循代价地图处理其他元素的方式。这些附加地图层使代价地图能够表示许多不同的上下文信息，生成需要对上下文信息做出适当反应的动作。

6.1 标准地图层

Static Map Layer(静态地图层)：为执行全局路径规划，机器人需要一幅超出其传感器感知范围的地图，以定位墙壁和其他静态障碍物的位置。可以预先使用 SLAM 算法生成静态地图，也可以从架构图 (architectural diagram) 直接创建静态映射。

当地图层接收到地图数据时，`updateBounds` 方法需要返回一个覆盖整个地图的边框。但是，在后续迭代中，因为该地图层是静态的，所以边框的尺寸不会增大。实际上，静态地图一直是全局代价地图的底层，因此它将其值直接复制到 master costmap 中，因为在此之前其他地图层不会往 master costmap 写入数据。

如果在机器人使用生成的地图进行导航时运行 SLAM 算法，则分层的代价地图方法允许静态地图层获得更新的同时，不会丢失其他地图层中的所有先前信息。在单源代价地图中，整个代价地图的数据会被覆盖写入。

Obstacles Layer(障碍层)：该层从高精度传感器（如激光和 RGB-D 相机）采集数据，并将其置于二维网格中。传感器和传感器读数之间的空间标记为“free”(自由空间)，传感器读数的位置标记为“occupied”(占据空间)。每个周期的 `updateBounds` 期间，新传感器数据会被放入图层的代价地图中，并且边框会自适应扩展。

精确将障碍层的值与代价地图中已有的值组合在一起的方法可能会有所不同，具体取决于传感器数据的信任级别。以前的默认做法是将传感器数据覆盖静态地图的数据。这种做法在静态地图的数据可能不准确并且分层代价地图的方法仍然可以使用的情况下最为有效。但是，如果更信赖静态地图，则可以将该地图层配置为仅向 master costmap(主代价地图) 添加致命障碍 (lethal obstacles) 属性。

Voxels Layer(体素层)：此层与障碍层的功能相同，仅是以三维方式跟踪传感器数据。Marder-Eppstein 等人介绍的三维体素网格方法[12] 可以更智能地检测出障碍物，以反映看到障碍物时的多个高度信息。

Inflation Layer(膨胀层)：正如前述所示，膨胀过程实质上是在每个致命障碍物周围插入一段缓冲区域。机器人绝对会发生碰撞的位置用致命代价 (lethal cost) 进行标记，并且周围的区域具有较小的非致命代价 (non-lethal cost)。这些值的作用是确保机器人不会与致命障碍物发生碰撞，并且不会让机器人过于靠近致命障碍物。`updateBounds` 会增大前一次边框，确保能够将新的致命障碍物进行膨胀，同时将在旧边框之外、而可能膨胀到新边框之内的旧致命障碍物也进行膨胀。`updateValues` 直接在 master costmap 上操作，而不存储局部地存储它的一个副本。

6.2 新功能

Sonar Layer(声纳层)：虽然单源代价地图可以处理声纳数据，但分层代价地图增加了如何处理声纳数据的选项。加入一个处理声纳读数的地图层可以避免激光雷达扫到玻璃墙时没有反射的问题。此外，我们还可以使用该层来处理不同于高精度障碍物层那样算法的声纳数据。我们构建的声纳层实现了概率声纳模型，使用了贝叶斯逻辑更新代价地图。之后，可以设置一个截断概率 (cutoff probability)，只将

相对比较确定的数据写入 master costmap。注意，这种方法允许我们保持概率的语义含义，而不必直接将它们与代价的含义结合起来。

Caution Zones Layer(警告区域层)：此层使我们能够用比 free/occupied 更详细的信息指定机器人的环境区域。虽然没有障碍，但多数机器人都希望不被导航到下楼梯方向。甚至机器人不应该被导航到特定人的办公室。尽管出现可导航的地方，但很多情况下，操作员都希望将机器人限制在可以安全行驶的地方。在这些限制约束下，实践中看到的一种可行技术是在静态地图层上对不偏好通行的区域标记障碍物属性。此技术虽然可以起作用，但会从地图中删除其他应用程序 (例如 AMCL) 可能需要用到的信息。该地图层还可以使我们能够对没有完全封禁但不希望进入的区域进行标记。在厨房中加入非致命代价可确保机器人不会在危险的液体附近行驶，除非没得其他选择。正如 Ferguson 和 Likhachev 所阐述的那样[5]，这些区域亦可用于人类日常社会活动中不太可能接受的区域，例如人与正在交互的物体 (例如一台电视机) 之间的空间。

Claustrophobic Layer(恐惧层)：尽管膨胀层已在致命障碍物外围增加了一个小缓冲区，但恐惧层增加一个更大的缓冲区，用以增加靠近障碍物的相对移动代价。造成的结果就是，机器人宁愿在尽可能远离障碍物的宽敞空间中移动，从而最大化机器人与任何检测到的障碍物的间隙。该层在机器人不确切了解障碍物的位置、具有很多不确定性选择时，机器人选择在障碍物中通行的几率或代价也很高的场景是很有帮助的。

6.3 人机交互层

如第2节所述，为代价地图引入复杂代价的主要动机之一是对人机交互引入建模约束。

Proxemic Layer(空间关系层)：对人与机器人之间的空间关系的研究正逐年上升，研究机器人不违反预期关系的方法亦如此。最常见的方法是将高斯分布或多个高斯分布的混合加入代价地图，正如 Kirby 等人所述那样[8]。这种调整策略可以在检出的人周围添加一个通行代价特别昂贵的区域，使得机器人能够尊重他们的个人空间。

我们的程序创建了一个 Proxemic Layer，实现了 Kirby 等人使用的混合高斯模型。利用检测到的人的位置信息和速度信息（可以从人腿的激光扫描中进行提取），该地图层将每个人的高斯值写入地图层的私有代价地图中，然后将其添加到 master costmap 中。生成值根据两个参数（幅度和方差，the amplitude and the variance）进行缩放。一般来说，增加这些参数的值时，最佳路径会离人更远。然而，正如[13] 所讨论的，最优路径变为最短路径之前，这些参数可以发生多大的改变是有个上限的。这意味着调整参数的意图是让机器人走得更远，但也可能发生相反的情况，这是人类日常社会活动中不希望出现的 (socially suboptimal)。文献[13] 的结果是在 Gazebo 模拟器中完全模拟复现的路径。

我们还基于第2节提到的更为复杂的 proxemic 模型实现了一个地图层。一些模型的假设信息不仅仅是人的位置信息和方向信息，我们的地图层在实现时，认为这些信息已与功能强大的传感器进行了配对，可以额外检测出其他信息，例如人的头部、身体姿态等。

Hallway Layer(过道层)：某些国家的生活文化中，人习惯在右侧行走，正如很多国家的司机在道路右侧行驶一样。我们实现了一种地图层，用于确定机器人是否处于走廊中，动态增加走廊左侧的通行代价，使得机器人倾向于右侧移动。我们在最近的一项用户研究中使用了类似的模型[14]，在地图层改变了通行代价，使机器人倾向于在最接近它的人的走廊的另一侧移动（通常为右侧）。结果显示添加该地图层不仅能有效地将机器人移动到走廊的一侧，而且还使得人的交互变得更加高效。

Wagon Ruts Layer(货运车辙层)：若机器人的目标在于避免造成社会损害 (socially invasive) 并最大

限度减少成为意外障碍的概率，一种有效的策略是让机器人的移动过程模仿人类的交通模式。该层可以降低人们已走过的路径的通行代价，从而使机器人的最优路径尽可能与已走过的路径大体保持一致。你甚至还可以将代价扭转，增加人们经常通行的区域的代价，尽量减少机器人给人类活动带来的中断(social disruption)。

7 讨论

本文中，我们讨论了新的分层 costmap 模型相对于以往的单源模型的好处。由于其效率性和可扩展性，它的实现已被 ROS 发布的所有版本采纳作为默认导航算法 (从 Hydro 开始)，其源代码可在https://github.com/ros-planning/navigation上找到。其他地图层的的实现代码可以在链接http://wiki.ros.org/costmap_2d查找。此外，为利用层结构的插件特性，我们希望创建额外 costmap 的规则，开发以层作为单位的实现方法，并在 ROS 导航框架内进行测试，从而允许算法更加开放以进行交换数据、在不同层之间方便地比较数值。

分层代价地图及其关联的各代价地图层为定义机器人的行为提供了更多的可能性。随着被集成到规划算法中的地图层的增加，机器人对其环境的不同信息掌握得会越多，在导航的时候会考虑这些不同的上下文信息。当前实现的情况是忽略其他上下文信息，或者一次性对它们进行处理。虽然分层的代价地图确实能够集成上下文信息，但我们预测未来的挑战是找到一种动态管理图层集合的方法，以确保在正确的时间对正确的上下文信息进行优先级排序。机器人与人的空间关系行为 (Proxemic behavior) 只是说明了机器人应当尊重个人的私有空间，但是精确地说它对机器人规划路径的效率提升了多少，仍是一个悬而未决的问题。这个问题的一半含义是设计代价地图层，使得数学上最优的路径正是我们所需的路径。而另一半含义为如何平衡机器人的计算需求与人们的期望需求，这是一个社会问题，没有明确的答案。虽然我们不能对这样的问题给出具体的答案，但我们相信拥有一个灵活定制的数据结构来定制机器人的行为将使得回答这样的问题变得更加容易。

参考文献

[1] MATTHIES L, ELFES A. Integration of sonar and stereo range data using a grid-based representation[C]//Proceedings. 1988 IEEE International Conference on Robotics and Automation. [S.l. : s.n.], 1988: 727-733 (引用页: 2).

[2] MORAVEC H P. Sensor fusion in certainty grids for mobile robots[G]//Sensor devices and systems for robotics. [S.l.]: Springer, 1989: 253-276 (引用页: 2).

[3] KONOLIGE K. Improved occupancy grids for map building[J]. Autonomous Robots, 1997, 4(4): 351-367 (引用页: 2).

[4] THRUN S. Learning occupancy grids with forward models[C]//Proceedings 2001 IEEE/RSJ International Conference on Intelligent Robots and Systems. Expanding the Societal Role of Robotics in the the Next Millennium (Cat. No. 01CH37180): vol. 3. [S.l. : s.n.], 2001: 1676-1681 (引用页: 2).

[5] FERGUSON D, LIKHACHEV M. Efficiently using cost maps for planning complex maneuvers[J]. Lab Papers (GRASP), 2008: 20 (引用页: 2, 9).

[6] GERKEY B P, AGRAWAL M. Break on through: Tunnel-based exploration to learn about outdoor terrain[C]//ICRA Workshop on Path Planning on Costmaps. [S.l. : s.n.], 2008 (引用页: 2).

[7] SISBOT E A, MARIN-URIAS L F, ALAMI R, et al. A human aware mobile robot motion planner[J]. IEEE Trans. Robot., 2007, 23(5): 874-883 (引用页: 2).

[8] KIRBY R, SIMMONS R, FORLIZZI J. Companion: A constraint-optimizing method for person-acceptable navigation[C]//RO-MAN 2009-The 18th IEEE International Symposium on Robot and Human Interactive Communication. [S.l. : s.n.], 2009: 607-612 (引用页: 2, 9).

[9] SVENSTRUP M, TRANBERG S, ANDERSEN H J, et al. Pose estimation and adaptive robot behaviour for human-robot interaction[C]//2009 IEEE International Conference on Robotics and Automation. [S.l. : s.n.], 2009: 3571-3576 (引用页: 2).

[10] SCANDOLO L, FRAICHARD T. An anthropomorphic navigation scheme for dynamic scenarios[C]// 2011 IEEE International Conference on Robotics and Automation. [S.l. : s.n.], 2011: 809-814 (引用页: 3).

[11] QUIGLEY M, CONLEY K, GERKEY B, et al. ROS: an open-source Robot Operating System[C]//ICRA workshop on open source software: vol. 3: 3.2. [S.l. : s.n.], 2009: 5 (引用页: 3).

[12] MARDER-EPPSTEIN E, BERGER E, FOOTE T, et al. The office marathon: Robust navigation in an indoor office environment[C]//2010 IEEE international conference on robotics and automation. [S.l. : s.n.], 2010: 300-307 (引用页: 4, 6, 8).

[13] LU D V, ALLAN D B, SMART W D. Tuning cost functions for social navigation[C]//International Conference on Social Robotics. [S.l. : s.n.], 2013: 442-451 (引用页: 9).

[14] LU D V, SMART W D. Towards more efficient navigation for robots and humans[C]//2013 IEEE/RSJ International Conference on Intelligent Robots and Systems. [S.l. : s.n.], 2013: 1707-1713 (引用页: 9).