Real-Time Loop Closure in 2D LIDAR SLAM

1.ABSTRACT

便携式激光测距仪，进一步称为 LIDAR，同时定位和建图（SLAM）是获取竣工楼层平面图的有效方法。 实时生成和可视化楼层平面图有助于操作员评估捕获数据的质量和覆盖范围。 构建便携式捕获平台需要在有限的计算资源下操作。 我们介绍了我们的背包绘图平台中使用的方法，该平台以 5 厘米的分辨率实现了实时映射和循环闭合。为了实现实时循环闭包，我们使用分支定界方法将扫描到子图匹配计算为约束。我们提供实验结果并与其他众所周知的方法进行比较，这些方法表明，在质量方面，我们的方法与已有技术相比具有竞争力。

2.INTRODUCTION

竣工平面图适用于各种应用。用于收集建筑物管理任务的数据的人工调查通常将计算机辅助设计（CAD）与激光卷尺相结合。这些方法很慢，并且通过将人类对建筑物的偏见视为直线的集合，并不总是准确地描述空间的真实性质。使用 SLAM，可以快速准确地调查大小和复杂程度较高的建筑物，这些建筑物需要花费更长的时间才能手动调查。

在这个领域应用 SLAM 并不是一个新想法，也不是本文的重点。相反，本文的贡献是一种新的方法，用于降低计算激光范围数据的闭环约束的计算要求。这项技术使我们能够绘制数十万平方米的非常大的楼层，同时为操作员提供实时全面优化的结果。

3.RELATED WORK

scan-to-scan 匹配经常用于计算基于激光的 SLAM 方法中的相对姿势变化，例如[1] - [4]。然而，就其本身而言，scan-to-scan 匹配会迅速累积误差。

scan-to-map 匹配有助于限制错误的累积。 一种使用 Gauss-Newton 在线性插值地图上找到局部最优的方法是[5]。 在存在良好的姿势初始估计的情况下，在这种情况下通过使用足够高数据速率的 LIDAR，局部优化的 scan-to-map 匹配是有效且稳健的。在不稳定的平台上，使用惯性测量单元（IMU）将激光扇投影到水平面上以估计重力方向。

像素精确扫描匹配方法，如[1]，进一步减少了局部误差累积。虽然计算上更昂贵，但这种方法对于循环闭包检测也很有用。一些方法侧重于通过匹配激光扫描中提取的特征来降低计算成本[4]。用于闭环检测的其他方法包括基于直方图的匹配[6]，扫描数据中的特征检测以及使用机器学习[7]。

解决剩余局部误差累积的两种常用方法是粒子滤波器和基于图的 SLAM [2], [8]。

粒子滤波器必须保持每个粒子中完整系统状态的表示。对于基于网格的 SLAM，随着地图变大，这很快就会变得非常耗费资源; 例如，我们的一个测试案例是在 3 公里的轨道上收集了 22,000 平方米。较小的维度特征表示，例如[9]，因为其不需要每个粒子的网格图，所以可以用来减少资源需求。当需要最新的网格图时，[10]建议计算子图，仅在必要时更新，以便最终的图是所有子图的栅格化。

基于图优化的方法在表示姿势和特征的节点集合上工作。图中的边是由观察产生的约束。可以使用各种优化方法来最小化由所有约束引入的误差，例如，[11]，[12]。在[13]中描述了这种用于室外 SLAM 的系统。它使用基于图优化的方法，通过局部的 scan-to-scan 匹配以及基于子图特征的直方图重叠局部图进行匹配。

4.SYSTEM OVERVIEW

Google 的 Cartographer 提供了一种实时室内地图解决方案，其形式为配备传感器的背包，可生成 r = 5 cm 分辨率的 2D 网格地图。系统的操作员可以看到在穿过建筑物时创建的地图。 激光扫描被插入到最佳估计位置的子图中，假定在短时间内足够准确。扫描匹配发生在最近的子图上，因此它仅取决于最近的扫描，并且世界坐标系中的姿态估计误差会累积。

为了在适度的硬件要求下获得良好的性能，我们的 SLAM 方法不使用粒子滤波器。为了应对错误的累积，我们定期运行姿态优化。当一个子图完成后，就不会再插入新的扫描，它会参与扫描匹配的闭环。所有已完成的子图和扫描都会被自动考虑进行闭环。如果它们基于当前姿势估计足够接近，则扫描匹配器尝试在子图中找到扫描。如果在当前估计的姿势周围的搜索窗口中找到足够好的匹配，则将其作为循环结束约束添加到优化问题。

Cartographer 通过每隔几秒完成优化，我们在实验中能发现在经过访问过的位置时它能立即闭环。这导致软实时约束，即闭环扫描匹配必须比添加新扫描更快地发生，否则它明显落后。我们通过使用分支定界方法和对每个完成的子图进行预计算网格来实现这一点。

5.LOCAL 2D SLAM

我们的系统将单独的局部和全局方法结合到 2D SLAM 中。两种方法都可以优化姿态。

$\boldsymbol{\xi} = (\xi_x, \xi_y, \xi_\theta)$ 包括雷达观测的一个平移 $(\boldsymbol{x}, \boldsymbol{y})$ 和一个旋转 ξ_θ，这在后面被称为一个 scan。在不稳定的平台上，例如我们的背包，IMU 用于估计重力方向，用于将扫描从水平安装的 LIDAR 投影到 2D 世界。

在我们的本地方法中，使用非线性优化将每个连续扫描与世界的一小块（称为子图 M）进行匹配，该非线性优化使扫描与子图对齐; 该过程进一步称为扫描匹配。扫描匹配随着时间累积误差，后来我们的全局方法将其去除，如 CLOSING LOOPS 中所述。

A.SCANS

子图构造是重复对齐 scan 和 submap 坐标 frame 的迭代过程。当扫描的原点在 $\mathbf{0} \in \mathbf{R}^2$ 时，我们现在将关于扫描点的信息写为 $\boldsymbol{H} = \{\boldsymbol{h_k}\}_{k=1,\ldots,K}, \boldsymbol{h_k} \in \boldsymbol{R}^2$。子图帧中扫描帧的姿态 $\boldsymbol{\xi}$ 表示为变换 $\boldsymbol{T_\xi}$，它将扫描点从扫描帧严格转换为子图帧，定义为

$$\begin{aligned} \boldsymbol{T_\xi} &= \begin{pmatrix} \cos\xi_\theta & -\sin\xi_\theta \\ \sin\xi_\theta & \cos\xi_\theta \end{pmatrix} \boldsymbol{p} + \begin{pmatrix} \xi_x \\ \xi_y \end{pmatrix} \\ \boldsymbol{T_\xi} &= \boldsymbol{R_\xi p} + \boldsymbol{t_\xi} \end{aligned} \tag{1.1}$$

B. SUBMAPS

一些连续扫描用于构建子图。这些子图采用概率网格 M 的形式：$\boldsymbol{rZ} \times \boldsymbol{rZ} \rightarrow [\boldsymbol{p}_{\min}, \boldsymbol{p}_{\max}]$

从给定分辨率 r（例如 5cm）的离散网格点映射到值。这些值可以被认为是网格点被阻挡的概率。对于每个网格点，我们将相应的像素定义为最接近该网格点的所有点。

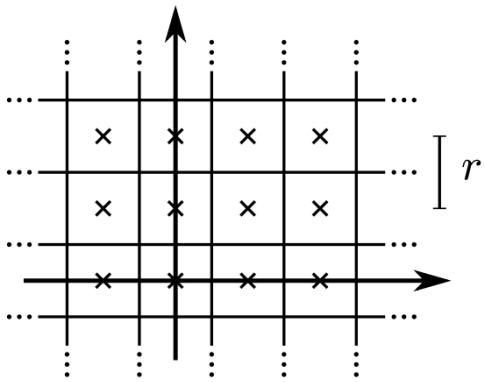

Fig. 1. Grid points and associated pixels.

每当要将一次 scan 插入到概率网格中时，计算用于命中的一组网格点和用于未命中的不相交组。对于每次命中，我们将最近的网格点插入到命中集中。 对于每个未命中，我们插入与每个像素相关联的网格点，该网格点与扫描原点和每个扫描点之间的一条光线相交，不包括已经在命中集中的网格点。如果每个以前未观察到的网格点位于其中一个集合中，则会为其分配概率 $\boldsymbol{p}_{hit}$ 或 $\boldsymbol{p}_{miss}$。如果已经观察到网格点 $\boldsymbol{x}$，我们更新命中和未命中的几率：

$$\boldsymbol{odd}(\boldsymbol{p}) = \frac{\boldsymbol{p}}{1-\boldsymbol{p}} \tag{1.2}$$

$$\boldsymbol{M}_{new}(\boldsymbol{x}) = \boldsymbol{clamp}(\boldsymbol{odds}^{-1}(\boldsymbol{odds}(\boldsymbol{M}_{new}(\boldsymbol{x})) \bullet \boldsymbol{odds}(\boldsymbol{p}_{hits}))) \tag{1.3}$$

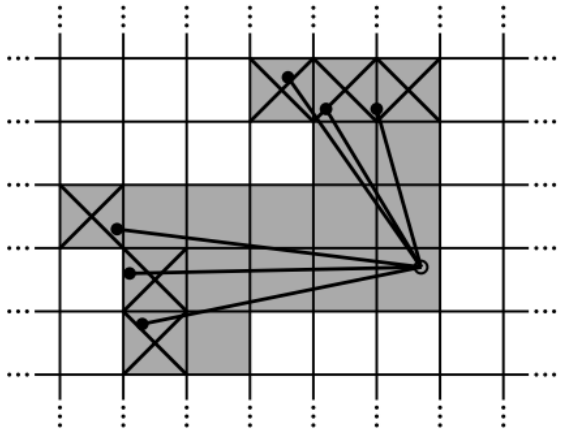

Fig. 2. A scan and pixels associated with *hits* (shaded and crossed out) and *misses* (shaded only).

C. CERES SCCAN MATCHING

在将扫描插入子图之前，使用基于 Ceres 的[14]扫描匹配器优化相对于当前本地子图的扫描姿态 $\boldsymbol{\xi}$。扫描匹配器负责找到最大化子图扫描点处概率的扫描姿势。我们把它作为一个非线性最小二乘问题

$$\arg\min_{\boldsymbol{\xi}} \sum_{\boldsymbol{k}=1}^{\boldsymbol{K}} (1 - \boldsymbol{M}_{smooth}(\boldsymbol{T}_{\boldsymbol{\xi}} \boldsymbol{h}_{\boldsymbol{k}}))^2 \tag{1.4}$$

其中 $\boldsymbol{T}_{\boldsymbol{\xi}}$ 根据扫描姿势将 $\boldsymbol{h}_{\boldsymbol{k}}$ 从扫描帧变换到子图帧。函数 $\boldsymbol{M}_{\boldsymbol{smooth}}$：从 $\boldsymbol{R}^2 \rightarrow \boldsymbol{R}$ 是一个在局部子图中概率值的平滑版本。我们使用双三次插值。结果，在区间[0,1]外的值会出现，但是这些值可以认为是无害的。

这种平滑函数的数学优化通常比网格的分辨率提供更好的精度。由于这是局部优化，因此需要良好的初始估计。能够测量角速度的 IMU 可用于估计扫描匹配之间的姿势的旋转分量 $\boldsymbol{\theta}$。尽管计算量更大，但可以在没有 IMU 的情况下也可以使用更高频率的扫描匹配或像素精确扫描匹配方法。

6.CLOSING LOOPS

由于扫描仅与包含少量近期扫描的子图匹配，因此上述方法会慢慢累积错误。 对于仅几十次连续扫描，累积误差很小。

通过创建许多小子图来处理更大的空间。我们的方法是，先优化所有扫描和子图的位姿，然后进行稀疏姿态调整[2]。插入扫描的相对位姿存储在存储器中以用于闭环优化。 除了这些相对位姿之外，一旦子图不再发生变化，所有其他由扫描和子图组成的对都被认为是闭环。 扫描匹配器在后台运行，如果找到良好匹配，则会将相应的相对位姿添加到优化问题中。

A.OPTIMIZATION PROBLEM

与扫描匹配一样，循环闭包优化也被公式化为非线性最小二乘问题，允许轻松添加残差以考虑其他数据。每隔几秒钟，我们使用 Ceres [14]来计算以下问题的解。

$$\underset{\Xi^m,\Xi^s}{\arg\min}\frac{1}{2}\sum_{ij}\boldsymbol{\rho}(\boldsymbol{E}^2(\boldsymbol{\xi}_i^m,\boldsymbol{\xi}_j^s;\Sigma_{ij},\boldsymbol{\xi}_{ij})) \tag{1.5}$$

其中，$\Xi^m=\{\boldsymbol{\xi}_i^m\}_{i=1,\ldots,m}$，$\Xi^s=\{\boldsymbol{\xi}_j^s\}_{j=1,\ldots,n}$ 在给定一些约束后在世界坐标系中进行优化。这些约束以相对位姿 $\boldsymbol{\xi}_{ij}$ 和协方差矩阵 Σ_{ij} 的形式表现。对一个 submap $\boldsymbol{i}$ 和一个 scan $\boldsymbol{j}$ 而言，位姿 $\boldsymbol{\xi}_{ij}$ 描述了该 scan 在 submap 坐标帧中的哪个位置匹配上了。协方差矩阵 Σ_{ij} 可以被计算，例如，遵循[15]中的方法，或者局部地使用 Ceres [14]与公式(1.4)的协方差估计特征。残差 $\boldsymbol{E}$ 通过下列公式计算：

$$\begin{aligned}\boldsymbol{E}^2(\boldsymbol{\xi}_i^m,\boldsymbol{\xi}_j^s;\Sigma_{ij},\boldsymbol{\xi}_{ij})&=\boldsymbol{e}(\boldsymbol{\xi}_i^m,\boldsymbol{\xi}_j^s;\boldsymbol{\xi}_{ij})\Sigma_{ij}^{-1}\boldsymbol{e}(\boldsymbol{\xi}_i^m,\boldsymbol{\xi}_j^s;\boldsymbol{\xi}_{ij})\\ \boldsymbol{e}(\boldsymbol{\xi}_i^m,\boldsymbol{\xi}_j^s;\boldsymbol{\xi}_{ij})&=\boldsymbol{\xi}_{ij}-\begin{pmatrix}\boldsymbol{R}_{\boldsymbol{\xi}_i^m}^{-1}(\boldsymbol{t}_{\boldsymbol{\xi}_i^m}-\boldsymbol{t}_{\boldsymbol{\xi}_j^s})\\ \boldsymbol{\xi}_{i;\boldsymbol{\theta}}^m-\boldsymbol{\xi}_{j;\boldsymbol{\theta}}^s\end{pmatrix}\end{aligned} \tag{1.6}$$

损失函数 $\boldsymbol{\rho}$，例如 Huber 损失，用于减少当扫描匹配向优化问题添加不正确约束时可能出现在公式(1.5)中的异常值的影响。例如，这可能发生在局部对称环境中，例如办公室隔间。异常值的替代方法包括[16]。

B.BRANCH-AND-BOUND SCAN MATCHING

我们对最佳的像素级精确匹配感兴趣：

$$\xi^* = \arg\max_{\xi \in W} \sum_{k=1}^{K} M_{nearest}(T_\xi h_k) \tag{1.7}$$

上式中，W 是搜索窗口，$M_{nearest}$ 是通过首先将其参数舍入到最近的网格点来扩展到所有 R^2，即将网格点的值扩展到相应的像素。使用(1.4)可以进一步提高匹配的质量。通过仔细选择步长来提高效率。我们选择角度步长 δ_θ，使得最大范围 $d_{\max}$ 处的扫描点不会移动超过一个像素的宽度 r。我们使用余弦定律推导出

$$d_{\max} = \max_{k=1,\ldots,K} \|h_k\| \tag{1.8}$$

$$\delta_\theta = \arccos\left(1 - \frac{r^2}{2d_{\max}^2}\right) \tag{1.9}$$

我们计算了包含给定线性和角度搜索窗口大小的整数步长，例如 $W_x = W_y = 7m, W_\theta = 30^\circ$

$$w_x = \left\lceil \frac{W_x}{r} \right\rceil, w_y = \left\lceil \frac{W_y}{r} \right\rceil, w_\theta = \left\lceil \frac{W_\theta}{r} \right\rceil \tag{1.10}$$

这个公式导出了一个组成搜索窗口的有限集合 W。该集合的中心是 ξ_0。

$$\bar{W} = \{-w_x, \ldots, w_x\} \times \{-w_y, \ldots, w_y\} \times \{-w_\theta, \ldots, w_\theta\} \tag{1.11}$$

$$W = \{\xi_0 + (rj_x, rj_y, \delta_\theta j_\theta) : (j_x, j_y, j_\theta)\} \tag{1.12}$$

一个用于寻找 ξ^* 的算法如下，参见算法 1，但是考虑到搜索窗口的大小，这样的算法太过缓慢。

Algorithm 1 Naive algorithm for (BBS)

```
best_score ← −∞
for j_x = −w_x to w_x do
  for j_y = −w_y to w_y do
    for j_θ = −w_θ to w_θ do
      score ← Σ_{k=1}^{K} M_nearest(T_{ξ0+(rj_x, rj_y, δ_θ j_θ)} h_k)
      if score > best_score then
        match ← ξ0 + (rj_x, rj_y, δ_θ j_θ)
        best_score ← score
      end if
    end for
  end for
end for
return best_score and match when set.
```

相反，我们使用分支定界方法在较大的搜索窗口上有效地计算 ξ^*。 有关通用方法，请参见算法 2。 这种方法首先在混合整数线性程序的背景下提出[17]。 关于这个主题的文献很

广泛，见[18]的简短概述。

Algorithm 2 Generic branch and bound

```
best_score ← −∞
C ← C_0
while C ≠ ∅ do
  Select a node c ∈ C and remove it from the set.
  if c is a leaf node then
    if score(c) > best_score then
      solution ← n
      best_score ← score(c)
    end if
  else
    if score(c) > best_score then
      Branch: Split c into nodes C_c.
      C ← C ∪ C_c
    else
      Bound.
    end if
  end if
end while
return best_score and solution when set.
```

主要思想是将可能性子集表示为树中的节点，其中根节点表示所有可能的解，在我们的示例中为 W. 每个节点的子节点形成其父节点的分区，因此它们一起表示同一组可能性。叶节点是单体，每个代表一个可行的解决方案。 请注意，算法是准确的。它提供与未优化的方法相同的解决方案，只要内部节点 c 的得分 score(c)是其元素得分的上限即可。在这种情况下，每当节点有界时，在该子树中不存在比目前最熟知的解决方案更好的解决方案。
为了得到具体的算法，我们必须决定节点选择，分支和上界计算的方法。
1）节点选择：在没有更好的替代方案的情况下，我们的算法使用深度优先搜索（DFS）作为默认选择：算法的效率取决于被修剪的树的大部分。这取决于两件事：良好的上限和良好的当前解决方案。后一部分由 DFS 帮助，它可以快速评估许多叶节点。由于我们不希望将不良匹配作为循环结束约束添加，我们还会引入一个分数阈值，低于该分数阈值我们对最优解决方案不感兴趣。由于实际上不会经常超过阈值，这降低了节点选择或找到初始启发式解决方案的重要性。关于在 DFS 期间访问孩子的顺序，我们计算每个孩子的分数的上限，访问具有最大边界的最有希望的子节点。算法 3 是这种方法。

Algorithm 3 DFS branch and bound scan matcher for (BBS)

```
best_score ← score_threshold
Compute and memorize a score for each element in C_0.
Initialize a stack C with C_0 sorted by score, the maximum
score at the top.
while C is not empty do
  Pop c from the stack C.
  if score(c) > best_score then
    if c is a leaf node then
      match ← ξ_c
      best_score ← score(c)
    else
      Branch: Split c into nodes C_c.
      Compute and memorize a score for each element
      in C_c.
      Push C_c onto the stack C, sorted by score, the
      maximum score last.
    end if
  end if
end while
return best_score and match when set.
```

2）减枝规则：树中的每个节点由整数元组描述$\boldsymbol{c}=(\boldsymbol{c}_x,\boldsymbol{c}_y,\boldsymbol{c}_\theta,\boldsymbol{c}_h)\in\boldsymbol{Z}^4$。高度为$\boldsymbol{c}_h$的节点可以组合成$2^{c_h}\times 2^{c_h}$种平移但只代表一种具体的旋转：

$$\bar{\bar{\boldsymbol{W}}}_c=\left(\left\{(\boldsymbol{j}_x,\boldsymbol{j}_y)\in\boldsymbol{Z}^2:\begin{matrix}\boldsymbol{c}_x\le\boldsymbol{j}_x<\boldsymbol{c}_x+2^{c_h}\\ \boldsymbol{c}_y\le\boldsymbol{j}_y<\boldsymbol{c}_y+2^{c_h}\end{matrix}\right\}\times\{\boldsymbol{c}_\theta\}\right) \tag{1.13}$$

$$\bar{\boldsymbol{W}}_c=\bar{\bar{\boldsymbol{W}}}_c\cap\bar{\boldsymbol{W}} \tag{1.14}$$

叶子节点的高度为$\boldsymbol{c}_h=0$，对应的可行解为$\boldsymbol{W}\in\boldsymbol{\xi}_c=\boldsymbol{\xi}_0+(\boldsymbol{rc}_x,\boldsymbol{rc}_y,\boldsymbol{\delta}_\theta\boldsymbol{c}_\theta)$

在我们的算法 3 的公式中，包含所有可行解的根节点没有明确地出现并且在覆盖搜索窗口的固定高度$\boldsymbol{h}_0$处分支成一组初始节点$\boldsymbol{C}_0$。搜索窗口为:

$$\begin{aligned}
\bar{\boldsymbol{W}}_{0,x}&=\left\{-\boldsymbol{w}_x+2^{h_0}\boldsymbol{j}_x:\boldsymbol{j}_x\in\boldsymbol{Z},0\le 2^{h_0}\boldsymbol{j}_x\le 2\boldsymbol{w}_x\right\}\\
\bar{\boldsymbol{W}}_{0,y}&=\left\{-\boldsymbol{w}_y+2^{h_0}\boldsymbol{j}_y:\boldsymbol{j}_y\in\boldsymbol{Z},0\le 2^{h_0}\boldsymbol{j}_y\le 2\boldsymbol{w}_y\right\}\\
\bar{\boldsymbol{W}}_{0,\theta}&=\left\{\boldsymbol{j}_\theta\in\boldsymbol{Z},-\boldsymbol{w}_\theta\le\boldsymbol{j}_\theta\le\boldsymbol{w}_\theta\right\}\\
\boldsymbol{C}_0&=\bar{\boldsymbol{W}}_{0,x}\times\bar{\boldsymbol{W}}_{0,y}\times\bar{\boldsymbol{W}}_{0,\theta}\times\{\boldsymbol{h}_0\}
\end{aligned} \tag{1.15}$$

在$\boldsymbol{c}_h>1$的给定节点 c 处，我们分支到最多四个高度为$\boldsymbol{c}_h-1$的子节点

$$\boldsymbol{C}_c=\left(\left(\{\boldsymbol{c}_x,\boldsymbol{c}_x+2^{c_h-1}\}\times\{\boldsymbol{c}_y,\boldsymbol{c}_y+2^{c_h-1}\}\right)\cap\bar{\boldsymbol{W}}\right)\times\{\boldsymbol{c}_h-1\} \tag{1.16}$$

3）计算上限：分支定界方法的剩余部分是计算内部节点上限的有效方法，无论是在计算工作量还是在边界质量方面。我们用

$$\begin{aligned}
&\boldsymbol{score}(\boldsymbol{c})=\sum_{\boldsymbol{k}=1}^{\boldsymbol{K}}\max_{\boldsymbol{j}\in\bar{\bar{\boldsymbol{W}}}_c}\boldsymbol{M}_{\boldsymbol{nearest}}(\boldsymbol{T}_{\boldsymbol{\xi}_j}\boldsymbol{h}_k)\\
&\ge\sum_{\boldsymbol{k}=1}^{\boldsymbol{K}}\max_{\boldsymbol{j}\in\bar{\boldsymbol{W}}_c}\boldsymbol{M}_{\boldsymbol{nearest}}(\boldsymbol{T}_{\boldsymbol{\xi}_j}\boldsymbol{h}_k)\\
&\ge\max_{\boldsymbol{j}\in\bar{\boldsymbol{W}}_c}\sum_{\boldsymbol{k}=1}^{\boldsymbol{K}}\boldsymbol{M}_{\boldsymbol{nearest}}(\boldsymbol{T}_{\boldsymbol{\xi}_j}\boldsymbol{h}_k)
\end{aligned} \tag{1.17}$$

为了能够有效地计算最大值，我们使用预先计算的网格$\boldsymbol{M}^{c_h}_{\boldsymbol{precomp}}$。每个可能的高度$\boldsymbol{c}_h$预先计算一个网格允许我们以正比于扫描点数量的时间来计算得分。请注意，为了能够做到这一点，我们还计算了$\bar{\bar{\boldsymbol{W}}}_c$上的最大值，它可以大于搜索空间边界附近的$\bar{\boldsymbol{W}}_c$。

$$\boldsymbol{score}(\boldsymbol{c})=\sum_{\boldsymbol{k}=1}^{\boldsymbol{K}}\boldsymbol{M}^{c_h}_{\boldsymbol{precomp}}(\boldsymbol{T}_{\boldsymbol{\xi}_j}\boldsymbol{h}_k) \tag{1.18}$$

$$M^{h}_{precomp}(x,y) = \max_{\substack{x'\in[x,x+r(2^h-1)]\\ y'\in[y,y+r(2^h-1)]}} M_{nearest}(x',y') \tag{1.19}$$

对于叶子节点来说，ξ_c 与先前一样。注意 $M^{h}_{precomp}$ 有与 $M_{nearest}$ 一样的像素结构，但 $M^{h}_{precomp}$ 在每个像素中存储了从该像素开始的最大值，这样的像素 box 有 $2^h \times 2^h$ 个。图 3 给出了这种预先计算的网格的一个例子。

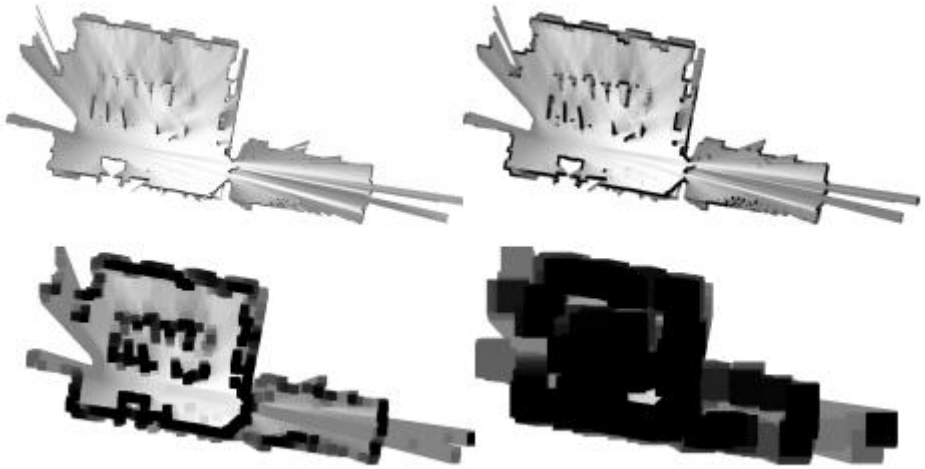

Fig. 3. Precomputed grids of size 1, 4, 16 and 64.

为了使构建预先计算的网格的计算工作量保持在较低水平，我们要等到概率网格不再接收更新。 然后我们计算一组预先计算的网格，并开始匹配它。

对于每个预先计算的网格，我们计算从每个像素开始的行方向上 2^h 像素宽范围内的最大值。使用该中间结果，然后构造下一个预先计算的网格。

如果按照添加顺序删除值，则可以按摊销的 O(1)保持更新的值集合的最大值。连续最大值保存在一个双端队列中，可以递归地定义为包含当前集合中所有值的最大值，然后是第一次出现最大值后所有值的连续最大值列表。 对于空值集合，此列表为空。使用这种方法，可以在 O(n)中计算预先计算的网格，其中 n 是每个预先计算的网格中的像素数。

计算上限的另一种方法是计算较低分辨率的概率网格，连续减半分辨率，见[1]。 由于我们的方法的额外内存消耗是可接受的，我们更喜欢使用较低分辨率的概率网格，这导致比(1.17)更差的界限，从而对性能产生负面影响。

7.EXPERIMENTAL RESULTS

略

8.CONCLUSIONS

略

9.REFERENCES

[1] E. Olson, “M3RSM: Many-to-many multi-resolution scan matching,” in Proceedings of the IEEE International Conference on Robotics and Automation (ICRA), June 2015.
[2] K. Konolige, G. Grisetti, R. Kummerle, W. Burgard, B. Limketkai, ¨and R. Vincent, “Sparse pose adjustment for 2D mapping,” in IROS, Taipei, Taiwan, 10/2010 2010.
[3] F. Lu and E. Milios, “Globally consistent range scan alignment for environment mapping,” Autonomous robots, vol. 4, no. 4, pp. 333–349, 1997.
[4] F. Mart´ın, R. Triebel, L. Moreno, and R. Siegwart, “Two different tools for three-dimensional mapping: DE-based scan matching and feature-based loop detection,” Robotica, vol. 32, no. 01, pp. 19–41,2014.
[5] S. Kohlbrecher, J. Meyer, O. von Stryk, and U. Klingauf, “A flexible and scalable SLAM system with full 3D motion estimation,” in Proc. IEEE International Symposium on Safety, Security and Rescue Robotics (SSRR). IEEE, November 2011.
[6] M. Himstedt, J. Frost, S. Hellbach, H.-J. Bohme, and E. Maehle, “Large scale place recognition in 2D LIDAR scans using geometrical landmark relations,” in Intelligent Robots and Systems (IROS 2014), 2014 IEEE/RSJ International Conference on. IEEE, 2014, pp. 5030–5035.
[7] K. Granstrom, T. B. Sch ¨on, J. I. Nieto, and F. T. Ramos, “Learning to ¨ close loops from range data,” The International Journal of Robotics Research, vol. 30, no. 14, pp. 1728–1754, 2011.
[8] G. Grisetti, C. Stachniss, and W. Burgard, “Improving grid-based SLAM with Rao-Blackwellized particle filters by adaptive proposals and selective resampling,” in Robotics and Automation, 2005. ICRA 2005. Proceedings of the 2005 IEEE International Conference on. IEEE, 2005, pp. 2432–2437.
[9] G. D. Tipaldi, M. Braun, and K. O. Arras, “FLIRT: Interest regions for 2D range data with applications to robot navigation,” in Experimental Robotics. Springer, 2014, pp. 695–710.
[10] J. Strom and E. Olson, “Occupancy grid rasterization in large environments for teams of robots,” in Intelligent Robots and Systems (IROS), 2011 IEEE/RSJ International Conference on. IEEE, 2011, pp. 4271– 4276.
[11] R. Kummerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g2o: A general framework for graph optimization,” in Robotics and Automation (ICRA), 2011 IEEE International Conference on. IEEE, 2011, pp. 3607–3613.
[12] L. Carlone, R. Aragues, J. A. Castellanos, and B. Bona, “A fast and accurate approximation for planar pose graph optimization,” The International Journal of Robotics Research, pp. 965–987, 2014.
[13] M. Bosse and R. Zlot, “Map matching and data association for largescale two-dimensional laser scan-based SLAM,” The International Journal of Robotics Research, vol. 27, no. 6, pp. 667–691, 2008.
[14] S. Agarwal, K. Mierle, and Others, “Ceres solver,” http://ceres-solver. org.
[15] E. B. Olson, “Real-time correlative scan matching,” in Robotics and Automation, 2009. ICRA’09. IEEE International Conference on. IEEE, 2009, pp. 4387–4393.
[16] P. Agarwal, G. D. Tipaldi, L. Spinello, C. Stachniss, and W. Burgard, “Robust map optimization using dynamic covariance scaling,” in Robotics and Automation (ICRA), 2013 IEEE International Conference on. IEEE, 2013, pp. 62–69.
[17] A. H. Land and A. G. Doig, “An automatic method of solving discrete programming problems,” Econometrica, vol. 28, no. 3, pp. 497–520, 1960.

[18] J. Clausen, “Branch and bound algorithms-principles and examples,” Department of Computer Science, University of Copenhagen, pp. 1–30, 1999.
[19] A. Howard and N. Roy, “The robotics data set repository (Radish),” 2003. [Online]. Available: http://radish.sourceforge.net/
[20] K. Konolige, J. Augenbraun, N. Donaldson, C. Fiebig, and P. Shah, “A low-cost laser distance sensor,” in Robotics and Automation, 2008. ICRA 2008. IEEE International Conference on. IEEE, 2008, pp.3002–3008.
[21] R. Kummerle, B. Steder, C. Dornhege, M. Ruhnke, G. Grisetti, C. Stachniss, and A. Kleiner, “On measuring the accuracy of SLAM algorithms,” Autonomous Robots, vol. 27, no. 4, pp. 387–407, 2009.