# LOAM: Lidar Odometry and Mapping in Real-time

Ji Zhang and Sanjiv Singh

Abstract— We propose a real-time method for odometry and mapping using range measurements from a 2-axis lidar moving in 6-DOF. The problem is hard because the range measurements are received at different times, and errors in motion estimation can cause mis-registration of the resulting point cloud. To date, coherent 3D maps can be built by off-line batch methods, often using loop closure to correct for drift over time. Our method achieves both low-drift and low-computational complexity without the need for high accuracy ranging or inertial measurements. The key idea in obtaining this level of performance is the division of the complex problem of simultaneous localization and mapping, which seeks to optimize a large number of variables simultaneously, by two algorithms. One algorithm performs odometry at a high frequency but low fidelity to estimate velocity of the lidar. Another algorithm runs at a frequency of an order of magnitude lower for fine matching and registration of the point cloud. Combination of the two algorithms allows the method to map in real-time. The method has been evaluated by a large set of experiments as well as on the KITTI odometry benchmark. The results indicate that the method can achieve accuracy at the level of state of the art offline batch methods.

摘要：我们提出了一种实时的里程计和测绘方法，该方法使用来自 6 自由度移动的 2 轴激光雷达的距离测量。这个问题很困难，因为距离测量是在不同的时间接收的，并且运动估计中的错误会导致结果点云的错误配准。迄今为止，连贯的 3D 地图可以通过离线构建批处理方法，通常使用闭环来纠正随时间的漂移。我们的方法无需高精度测距或惯性测量即可实现低漂移和低计算复杂度。获得这种性能水平的关键思想是同时定位和映射的复杂问题的划分，该问题旨在通过两种算法同时优化大量变量。

一种算法以高频率但低保真度执行里程计来估计激光雷达的速度。

另一种算法以低一个数量级的频率运行，用于点云的精细匹配和配准。两种算法的组合允许该方法实时映射。该方法已通过大量实验以及 KITTI 里程计基准进行了评估。

结果表明，该方法可以达到最先进的离线批处理方法水平的准确性。

I. INTRODUCTION

3D mapping remains a popular technology [1]–[3]. Mapping with lidars is common as lidars can provide high frequency range measurements where errors are relatively constant irrespective of the distances measured. In the case that the only motion of the lidar is to rotate a laser beam, registration of the point cloud is simple. However, if the lidar itself is moving as in many applications of interest, accurate mapping requires knowledge of the lidar pose during continuous laser ranging. One common way to solve the problem is using independent position estimation (e.g. by a GPS/INS) to register the laser points into a fixed coordinate system. Another set of methods use odometry measurements such as from wheel encoders or visual odometry systems [4], [5] to register the laser points. Since odometry integrates small incremental motions over time, it is bound to drift and much attention is devoted to reduction of the drift (e.g. using loop closure)

3D 映射仍然是一种流行的技术 [1]-[3]。使用激光雷达进行映射很常见，因为激光雷达可以提供高频范围测量，其中无论测量距离如何，误差都相对恒定。在激光雷达的唯一运动是旋转激光束的情况下，点云的配准很简单。然而，

如果激光雷达本身在许多感兴趣的应用中移动，则准确的映射需要在连续激光测距期间了解激光雷达的位姿。解决该问题的一种常见方法是使用独立的位置估计（例如通过 GPS/INS）将激光点注册到固定坐标系中。

另一组方法使用里程计测量，例如来自车轮编码器或视觉里程计系统 [4]、[5] 来记录激光点。由于里程计随着时间的推移集成了小的增量运动，它必然会漂移，并且非常注重减少漂移（例如使用闭环）

Here we consider the case of creating maps with lowdrift odometry using a 2-axis lidar moving in 6-DOF. A key advantage of using a lidar is its insensitivity to ambient lighting and optical texture in the scene. Recent developments in lidars have reduced their size and weight. The lidars can be held by a person who traverses an environment [6], or even attached to a micro aerial vehicle [7]. Since our method is intended to push issues related to minimizing drift in odometry estimation, it currently does not involve loop closure.

在这里，我们考虑使用在 6 自由度中移动的 2 轴激光雷达创建具有低漂移里程计的地图的情况。使用激光雷达的一个关键优势是它对场景中的环境照明和光学纹理不敏感。激光雷达的最新发展减少了它们的尺寸和重量。激光雷达可由穿越环境的人持有

The method achieves both low-drift and low-computational complexity without the need for high accuracy ranging or inertial measurements. The key idea in obtaining this level of performance is the division of the typically complex problem of simultaneous localization and mapping (SLAM) [8], which seeks to optimize a large number of variables simultaneously, by two algorithms. One algorithm performs odometry at a high frequency but low fidelity to estimate velocity of the lidar. Another algorithm runs at a frequency of an order of magnitude lower for fine matching and registration of the point cloud. Although unnecessary, if an IMU is available, a motion prior can be provided to help account for high frequency motion. Specifically, both algorithms extract feature points located on sharp edges and planar surfaces, and match the feature points to edge line segments and planar surface patches, respectively. In the odometry algorithm, correspondences of the feature points are found by ensuring fast computation. In the mapping algorithm, the correspondences are determined by examining geometric distributions of local point clusters, through the associated eigenvalues and eigenvectors.

该方法无需高精度测距或惯性测量即可实现低漂移和低计算复杂度。获得这种性能水平的关键思想是对同时定位和映射 (SLAM) [8] 的典型复杂问题进行划分，

它试图通过两种算法同时优化大量变量。一种算法以高频率但低保真度执行里程计来估计激光雷达的速度。另一种算法以低一个数量级的频率运行，用于点云的精细匹配和配准。虽然没有必要，如果 IMU 可用，则可以提供运动先验以帮助解释高频运动。具体来说，两种算法都提取位于锐利边缘和平面表面上的特征点，并将特征点分别与边缘线段和平面表面块进行匹配。在里程计算法中，通过确保快速计算找到特征点的对应关系。在映射算法中，通过检查局部点簇的几何分布，通过相关的特征值和特征向量来确定对应关系。

By decomposing the original problem, an easier problem is solved first as online motion estimation. After which, mapping is conducted as batch optimization (similar to iterative closest point (ICP) methods [9]) to produce high-precision motion estimates and maps. The parallel algorithm structure ensures feasibility of the problem to be solved in real-time. Further, since the motion estimation is conducted at a higher frequency, the mapping is given plenty of time to enforce accuracy. When running at a lower frequency, the mapping algorithm is able to incorporate a large number of feature points and use sufficiently many iterations for convergence.

通过分解原始问题，首先解决了一个更简单的问题，即在线运动估计。之后，映射作为批量优化（类似于迭代最近点 (ICP) 方法 [9]）进行，以生成高精度运动估计和映射。

并行算法结构保证了实时解决问题的可行性。此外，由于运动估计是以更高的频率进行的，因此映射有足够的时间来提高准确性。以较低频率运行时，

映射算法能够合并大量特征点并使用足够多的迭代来收敛。

图片包含 图形用户界面

描述已自动生成

Fig. 1. The method aims at motion estimation and mapping using a moving 2-axis lidar. Since the laser points are received at different times, distortion is present in the point cloud due to motion of the lidar (shown in the left lidar cloud). Our proposed method decomposes the problem by two algorithms running in parallel. An odometry algorithm estimates velocity of the lidar and corrects distortion in the point cloud, then, a mapping algorithm matches and registers the point cloud to create a map. Combination of the two algorithms ensures feasibility of the problem to be solved in real-time.

图 1. 该方法旨在使用移动 2 轴激光雷达进行运动估计和映射。由于激光点是在不同时间接收的，因此由于激光雷达的运动，点云中会出现失真（如左侧激光雷达云所示）。我们提出的方法通过并行运行的两种算法来分解问题。、里程计算法估计激光雷达的速度并纠正点云中的失真，然后，映射算法匹配并注册点云以创建地图。两种算法的结合保证了实时解决问题的可行性。

II. RELATED WORK

Lidar has become a useful range sensor in robot navigation [10]. For localization and mapping, most applications use 2D lidars [11]. When the lidar scan rate is high compared to its extrinsic motion, motion distortion within the scans can often be neglected. In this case, standard ICP methods [12] can be used to match laser returns between different scans. Additionally, a two-step method is proposed to remove the distortion [13]: an ICP based velocity estimation step is followed by a distortion compensation step, using the computed velocity. A similar technique is also used to compensate for the distortion introduced by a single-axis 3D lidar [14]. However, if the scanning motion is relatively slow, motion distortion can be severe. This is especially the case when a 2-axis lidar is used since one axis is typically much slower than the other. Often, other sensors are used to provide velocity measurements, with which, the distortion can be removed. For example, the lidar cloud can be registered by state estimation from visual odometry integrated with an IMU [15]. When multiple sensors such as a GPS/INS and wheel encoders are available concurrently, the problem is usually solved through an extended Kalman filer [16] or a particle filter [1]. These methods can create maps in real-time to assist path planning and collision avoidance in robot navigation.

激光雷达已成为机器人导航中有用的距离传感器 [10]。对于定位和映射，大多数应用程序使用 2D 激光雷达 [11]。当激光雷达扫描速率与其外部运动相比较高时，扫描内的运动失真通常可以忽略不计。在这种情况下，标准 ICP 方法 [12] 可用于匹配不同扫描之间的激光回波。此外，提出了一种两步法来消除失真 [13]：基于 ICP 的速度估计步骤之后是失真补偿步骤，使用计算的速度。类似的技术也用于补偿单轴 3D 激光雷达 [14] 引入的失真。但是，如果扫描运动相对较慢，则运动失真可能会很严重。当使用 2 轴激光雷达时尤其如此，因为一个轴通常比另一个轴慢得多。经常，其他传感器用于提供速度测量，从而可以消除失真。例如，激光雷达云可以通过与 IMU 集成的视觉里程计的状态估计来注册 [15]。当 GPS/INS 和车轮编码器等多个传感器同时可用时，该问题通常通过扩展卡尔曼滤波器 [16] 或粒子滤波器 [1] 来解决。这些方法可以实时创建地图，以辅助机器人导航中的路径规划和避碰。

If a 2-axis lidar is used without aiding from other sensors, motion estimation and distortion correction become one problem. A method used by Barfoot et al. is to create visual images from laser intensity returns, and match visually distinct features [17] between images to recover motion of a ground vehicle [18]–[21]. The vehicle motion is modeled as constant velocity in [18], [19] and with Gaussian processes in [20], [21]. Our method uses a similar linear motion model as [18], [19] in the odometry algorithm, but with different types of features. The methods [18]–[21] involve visual features from intensity images and require dense point cloud. Our method extracts and matches geometric features in Cartesian space and has a lower requirement on the cloud density.

如果在没有其他传感器帮助的情况下使用 2 轴激光雷达，运动估计和失真校正将成为一个问题。 Barfoot 等人使用的一种方法。是从激光强度返回创建视觉图像，并在图像之间匹配视觉上不同的特征[17]，以恢复地面车辆的运动[18]-[21]。车辆运动在 [18]、[19] 中被建模为恒定速度，在 [20]、[21] 中使用高斯过程建模。我们的方法在里程计算法中使用与 [18]、[19] 类似的线性运动模型，但具有不同类型的特征。方法 [18]-[21] 涉及来自强度图像的视觉特征，并且需要密集的点云。我们的方法在笛卡尔空间中提取和匹配几何特征，对云密度的要求较低。

The approach closest to ours is that of Bosse and Zlot [3], [6], [22]. They use a 2-axis lidar to obtain point cloud which is registered by matching geometric structures of local point clusters [22]. Further, they use multiple 2-axis lidars to map an underground mine [3]. This method incorporates an IMU and uses loop closure to create large maps. Proposed by the same authors, Zebedee is a mapping device composed of a 2D lidar and an IMU attached to a hand-bar through a spring [6]. Mapping is conducted by hand nodding the device. The trajectory is recovered by a batch optimization method that processes segmented datasets with boundary constraints added between the segments. In this method, the measurements of the IMU are used to register the laser points and the optimization is used to correct the IMU biases. In essence, Bosse and Zlot’s methods require batch processing to develop accurate maps and therefore are unsuitable for applications where maps are needed in real-time. In comparison, the proposed method in real-time produces maps that are qualitatively similar to those by Bosse and Zlot. The distinction is that our method can provide motion estimates for guidance of an autonomous vehicle. Further, the method takes advantage of the lidar scan pattern and point cloud distribution. Feature matching is realized ensuring computation speed and accuracy in the odometry and mapping algorithms, respectively

最接近我们的方法是 Bosse 和 Zlot [3]、[6]、[22] 的方法。他们使用 2 轴激光雷达来获取点云，该点云通过匹配局部点簇的几何结构来注册 [22]。此外，他们使用多个 2 轴激光雷达来绘制地下矿井地图 [3]。这种方法结合了 IMU 并使用闭环来创建大地图。 Zebedee 由同一作者提出，是一种测绘设备，由 2D 激光雷达和通过弹簧连接到手杆的 IMU 组成 [6]。映射是通过用手点头设备进行的。

轨迹通过批处理优化方法恢复，该方法处理分段数据集，在分段之间添加边界约束。在该方法中，IMU 的测量值用于配准激光点，优化用于校正 IMU 偏差。在本质上，

Bosse 和 Zlot 的方法需要批处理来开发准确的地图，因此不适合需要实时地图的应用。相比之下，所提出的实时方法生成的地图在质量上与 Bosse 和 Zlot 的相似。区别在于我们的方法可以为自动驾驶汽车的引导提供运动估计。此外，该方法利用了激光雷达扫描模式和点云分布。实现特征匹配，分别确保里程计和映射算法的计算速度和准确性

III. NOTATIONS AND TASK DESCRIPTION

The problem addressed in this paper is to perform egomotion estimation with point cloud perceived by a 3D lidar, and build a map for the traversed environment. We assume that the lidar is pre-calibrated. We also assume that the angular and linear velocities of the lidar are smooth and continuous over time, without abrupt changes. The second assumption will be released by usage of an IMU, in Section VII-B.

本文解决的问题是使用 3D 激光雷达感知的点云进行自我运动估计，并为遍历的环境构建地图。我们假设激光雷达是预先校准的。我们还假设激光雷达的角速度和线速度随着时间的推移是平滑和连续的，没有突然变化。

第二个假设将在第 VII-B 节中通过使用 IMU 来释放。

As a convention in this paper, we use right uppercase superscription to indicate the coordinate systems. We define a sweep as the lidar completes one time of scan coverage. We use right subscription k, k ∈ Z + to indicate the sweeps, and Pk to indicate the point cloud perceived during sweep k. Let us define two coordinate systems as follows。

作为本文的惯例，我们使用右大写上标来表示坐标系。我们将扫描定义为激光雷达完成一次扫描覆盖。我们使用右订阅 k, 表示扫描， 表示在扫描 k 期间感知的点云。让我们定义两个坐标系如下：

* Lidar coordinate system {L} is a 3D coordinate system with its origin at the geometric center of the lidar. The xaxis is pointing to the left, the y-axis is pointing upward, and the z-axis is pointing forward. The coordinates of a point , in are denoted as .
* 激光雷达坐标系 {L} 是一个 3D 坐标系，其原点位于激光雷达的几何中心。 x 轴指向左侧，y 轴指向上方，z 轴指向前方。 中点 ,的坐标表示为 .。
* World coordinate system is a 3D coordinate system coinciding with at the initial position. The coordinates of a point , in are .
* 世界坐标系是在初始位置与 重合的 3D 坐标系。 {Wk} 中点, 的坐标是 。

With assumptions and notations made, our lidar odometry and mapping problem can be defined as

**Problem:** Given a sequence of lidar cloud , compute the ego-motion of the lidar during each sweep k, and build a map with Pk for the traversed environment.

有了假设和符号，我们的激光雷达里程计和映射问题可以定义为

问题：给定一系列激光雷达云 ，计算激光雷达在每次扫描 k 期间的自我运动，并为遍历的环境构建带有 的地图。

IV. SYSTEM OVERVIEW

A. Lidar Hardware

The study of this paper is validated on, but not limited to a custom built 3D lidar based on a Hokuyo UTM-30LX laser scanner. Through the paper, we will use data collected from the lidar to illustrate the method. The laser scanner has a field of view of 180◦ with 0.25◦ resolution and 40 lines/sec scan rate. The laser scanner is connected to a motor, which is controlled to rotate at an angular speed of 180◦/s between −90◦ and 90◦ with the horizontal orientation of the laser scanner as zero. With this particular unit, a sweep is a rotation from −90◦ to 90◦ or in the inverse direction (lasting for 1s). Here, note that for a continuous spinning lidar, a sweep is simply a semispherical rotation. An onboard encoder measures the motor rotation angle with a resolution of 0.25◦ , with which, the laser points are projected into the lidar coordinates, {L}.

本文的研究在但不限于基于 Hokuyo UTM-30LX 激光扫描仪的定制 3D 激光雷达上进行了验证。通过本文，我们将使用从激光雷达收集的数据来说明该方法。激光扫描仪的视野为 180°，分辨率为 0.25°，扫描速率为 40 线/秒。

激光扫描仪连接到电机，电机被控制以 180°/s 的角速度在 -90° 和 90° 之间旋转，激光扫描仪的水平方向为零。使用这个特定单位，扫描是从 -90° 到 90° 或反向旋转（持续 1 秒）。这里，

请注意，对于连续旋转的激光雷达，扫描只是半球形旋转。板载编码器以 0.25° 的分辨率测量电机旋转角度，将激光点投影到激光雷达坐标 {L} 中。

图片包含 游戏机

描述已自动生成

Fig. 2. The 3D lidar used in this study consists of a Hokuyo laser scanner driven by a motor for rotational motion, and an encoder that measures the rotation angle. The laser scanner has a field of view of 180◦ with a resolution of 0.25◦. The scan rate is 40 lines/sec. The motor is controlled to rotate from −90◦ to 90◦ with the horizontal orientation of the laser scanner as zero.

图 2. 本研究中使用的 3D 激光雷达由一个由电机驱动旋转运动的 Hokuyo 激光扫描仪和一个测量旋转角度的编码器组成。激光扫描仪的视野为 180°，分辨率为 0.25°。扫描速率为 40 行/秒。控制电机从 -90° 旋转到 90°，激光扫描仪的水平方向为零。

图示

描述已自动生成

Fig. 3. Block diagram of the lidar odometry and mapping software system

图 3 激光雷达测距与测绘软件系统框图

B. Software System Overview

Fig. 3 shows a diagram of the software system. Let be the points received in a laser scan. During each sweep, is registered in {L}. The combined point cloud during sweep k forms . Then, is processed in two algorithms. The lidar odometry takes the point cloud and computes the motion of the lidar between two consecutive sweeps. The estimated motion is used to correct distortion in . The algorithm runs at a frequency around 10Hz. The outputs are further processed by lidar mapping, which matches and registers the undistorted cloud onto a map at a frequency of 1Hz. Finally, the pose transforms published by the two algorithms are integrated to generate a transform output around 10Hz, regarding the lidar pose with respect to the map. Section V and VI present the blocks in the software diagram in detail.

图 3 显示了软件系统的示意图。设 是激光扫描中接收到的点。在每次扫描期间， 被注册在 {L} 中。扫描 k 期间的组合点云形成 。然后， 用两种算法处理。激光雷达里程计采用点云并计算激光雷达在两次连续扫描之间的运动。估计的运动用于校正 中的失真。该算法以大约 10Hz 的频率运行。激光雷达映射进一步处理输出，该映射以 1Hz 的频率将未失真的云匹配并注册到地图上。最后，两种算法发布的位姿变换被集成以生成大约 10Hz 的变换输出，关于激光雷达相对于地图的位姿。第五节和第六节详细介绍了软件图中的模块。

V. LIDAR ODOMETRY

**A. Feature Point Extraction**

We start with extraction of feature points from the lidar cloud, Pk. The lidar presented in Fig. 2 naturally generates unevenly distributed points in Pk. The returns from the laser scanner has a resolution of 0.25◦ within a scan. These points are located on a scan plane. However, as the laser scanner rotates at an angular speed of 180◦/s and generates scans at 40Hz, the resolution in the perpendicular direction to the scan planes is 180◦/40 = 4.5 ◦ . Considering this fact, the feature points are extracted from Pk using only information from individual scans, with co-planar geometric relationship.

我们从激光雷达云 Pk 中提取特征点开始。图 2 中的激光雷达自然会在 中生成不均匀分布的点。来自激光扫描仪的回波在一次扫描内具有 0.25° 的分辨率。这些点位于扫描平面上。然而，由于激光扫描仪以 180°/s 的角速度旋转并产生 40Hz 的扫描，因此与扫描平面垂直方向的分辨率为 180°/40 = 4.5°。考虑到这一事实，特征点是从 Pk 中提取的，仅使用来自单个扫描的信息，具有共面几何关系。

We select feature points that are on sharp edges and planar surface patches. Let i be a point in , , and let S be the set of consecutive points of i returned by the laser scanner in the same scan. Since the laser scanner generates point returns in CW or CCW order, S contains half of its points on each side of i and 0.25◦ intervals between two points. Define a term to evaluate the smoothness of the local surface

我们选择位于锐利边缘和平面表面补丁上的特征点。令 i 为 中的一个点，，令 S 为激光扫描仪在同一扫描中返回的 i 的连续点的集合。由于激光扫描仪以 CW 或 CCW 顺序生成点返回，因此 S 在 i 和 0 的每一侧都包含其一半的点。两点之间的 25° 间隔。定义一个术语来评估局部表面的平滑度，

**图片包含 文本

描述已自动生成**

The points in a scan are sorted based on the c values, then feature points are selected with the maximum c’s, namely, edge points, and the minimum c’s, namely planar points. To evenly distribute the feature points within the environment, we separate a scan into four identical subregions. Each subregion can provide maximally 2 edge points and 4 planar points. A point i can be selected as an edge or a planar point only if its c value is larger or smaller than a threshold, and the number of selected points does not exceed the maximum.

根据c值对扫描中的点进行排序，然后选择c最大的特征点，即边缘点，以及最小c的特征点，即平面点。为了在环境中均匀分布特征点，我们将扫描分成四个相同的子区域。每个子区域最多可以提供 2 个边缘点和 4 个平面点。一个点 i 只有在其 c 值大于或小于某个阈值，并且选择的点数不超过最大值时，才能被选为边或平面点。

While selecting feature points, we want to avoid points whose surrounded points are selected, or points on local planar surfaces that are roughly parallel to the laser beams (point B in Fig. 4(a)). These points are usually considered as unreliable. Also, we want to avoid points that are on boundary of occluded regions [23]. An example is shown in Fig. 4(b). Point A is an edge point in the lidar cloud because its connected surface (the dotted line segment) is blocked by another object. However, if the lidar moves to another point of view, the occluded region can change and become observable. To avoid the aforementioned points to be selected, we find again the set of points S. A point i can be selected only if S does not form a surface patch that is roughly parallel to the laser beam, and there is no point in S that is disconnected from i by a gap in the direction of the laser beam and is at the same time closer to the lidar then point i (e.g. point B in Fig. 4(b)).

在选择特征点时，我们要避免选择包围点的点，或者与激光束大致平行的局部平面上的点（图 4（a）中的点 B）。这些点通常被认为是不可靠的。此外，我们希望避免位于遮挡区域边界上的点 [23]。一个例子如图 4(b) 所示。点 A 是激光雷达云中的一个边缘点，因为它的连接表面（虚线段）被另一个物体挡住了。但是，如果激光雷达移动到另一个视点，被遮挡的区域可能会发生变化并变得可观察到。为避免上述点被选中，我们再次找到点 S 的集合。只有当 S 没有形成大致平行于激光束的表面片时，才能选择点 i，并且 S 中没有点与 i 不存在间隙。激光束的方向，同时更靠近激光雷达，然后是点 i（例如图 4（b）中的点 B）。

In summary, the feature points are selected as edge points starting from the maximum c value, and planar points starting from the minimum c value, and if a point is selected,

综上所述，特征点从c值最大开始选择为边缘点，c值从最小开始选择平面点，如果选择了一个点，

* The number of selected edge points or planar points cannot exceed the maximum of the subregion, and
* 选择的边缘点或平面点的数量不能超过子区域的最大值
* None of its surrounding point is already selected, and
* 它的周围点都没有被选中
* It cannot be on a surface patch that is roughly parallel to the laser beam, or on boundary of an occluded region.
* 它不能位于与激光束大致平行的表面补丁上，也不能位于遮挡区域的边界上。

An example of extracted feature points from a corridor scene is shown in Fig. 5. The edge points and planar points are labeled in yellow and red colors, respectively

从走廊场景中提取的特征点示例如图 5 所示。边缘点和平面点分别用黄色和红色标记

手机屏幕截图

低可信度描述已自动生成

Fig. 4. (a) The solid line segments represent local surface patches. Point A is on a surface patch that has an angle to the laser beam (the dotted orange line segments). Point B is on a surface patch that is roughly parallel to the laser beam. We treat B as a unreliable laser return and do not select it as a feature point. (b) The solid line segments are observable objects to the laser. Point A is on the boundary of an occluded region (the dotted orange line segment), and can be detected as an edge point. However, if viewed from a different angle, the occluded region can change and become observable. We do not treat A as a salient edge point or select it as a feature point.

图 4. (a) 实线段代表局部表面斑块。点 A 位于与激光束（橙色虚线段）成一定角度的曲面片上。 B 点位于与激光束大致平行的表面片上。我们将 B 视为不可靠的激光返回，不选择它作为特征点。

(b) 实线段是激光可观察到的物体。 A点位于遮挡区域（橙色虚线段）的边界上，可以检测为边缘点。但是，如果从不同的角度观察，遮挡区域可能会发生变化并变得可观察到。我们不将 A 视为显着边缘点或选择它作为特征点。

图片包含 游戏机, 灯光, 钟表

描述已自动生成 Fig. 5. An example of extracted edge points (yellow) and planar points (red) from lidar cloud taken in a corridor. Meanwhile, the lidar moves toward the wall on the left side of the figure at a speed of 0.5m/s, this results in motion distortion on the wall

图 5. 从走廊中提取的激光雷达云中提取的边缘点（黄色）和平面点（红色）的示例。同时，激光雷达以 0.5m/s 的速度向图中左侧的墙壁移动，导致墙壁运动失真

**B. Finding Feature Point Correspondence 寻找特征点对应**

The odometry algorithm estimates motion of the lidar within a sweep. Let be the starting time of a sweep k. At the end of each sweep, the point cloud perceived during the sweep, ,is reprojected to time stamp , illustrated in Fig. 6. We denote the reprojected point cloud as . During the next sweep k+1, is used together with the newly received point cloud, , to estimate the motion of the lidar

里程计算法估计激光雷达在一次扫描中的运动。令 为扫描 k 的开始时间。在每次扫描结束时，扫描期间感知的点云 被重新投影到时间戳 ，如图 6 所示。我们将重新投影的点云表示为 。在下一次扫描 k+1 期间，

与新接收的点云 一起用于估计激光雷达的运动

Let us assume that both k and are available for now, and start with finding correspondences between the two lidar clouds. With , we find edge points and planar points from the lidar cloud using the methodology discussed in the last section. Let and be the sets of edge points and planar points, respectively. We will find edge lines from . k as the correspondences for the points in , and planar patches as the correspondences for those in

让我们假设 和 现在都可用，并从寻找两个激光雷达云之间的对应关系开始。通过 ，我们使用上一节中讨论的方法从激光雷达云中找到边缘点和平面点。令 和 分别是边缘点和平面点的集合。我们将从 中找到边缘线作为 中的点的对应关系，平面块作为 中的点的对应关系

Note that at the beginning of sweep k+1, is an empty set, which grows during the course of the sweep as more points are received. The lidar odometry recursively estimates the 6- DOF motion during the sweep, and gradually includes more points as increases. At each iteration, and are reprojected to the beginning of the sweep using the currently estimated transform. Let and be the reprojected point sets. For each point in and , we are going to find the closest neighbor point in . Here, is stored in a 3D KD-tree [24] for fast index.

请注意，在扫描 k+1 开始时， 是一个空集，在扫描过程中随着接收到更多点而增长。激光雷达里程计在扫描过程中递归地估计 6 自由度运动，并随着 的增加逐渐包括更多的点。在每次迭代中，使用当前估计的变换将 和 重新投影到扫描的开始。令 和 为重投影点集。对于 和 中的每个点，我们将在 中找到最近的邻居点。在这里， 存储在 3D KD-tree [24] 中用于快速索引。

Fig. 7(a) represents the procedure of finding an edge line as the correspondence of an edge point. Let i be a point in , The edge line is represented by two points. Let j be the closest neighbor of i in , , and let l be the closest neighbor of i in the two consecutive scans to the scan of j. (j, l) forms the correspondence of i. Then, to verify both j and l are edge points, we check the smoothness of the local surface based on (1). Here, we particularly require that j and l are from different scans considering that a single scan cannot contain more than one points from the same edge line. There is only one exception where the edge line is on the scan plane. If so, however, the edge line is degenerated and appears as a straight line on the scan plane, and feature points on the edge line should not be extracted in the first place.

图7(a)表示寻找边缘线作为边缘点对应的过程。令 i 为 中的一个点，。边缘线由两个点表示。令 j 为 i 在 中的最近邻 ，，令 l 为 i 在对 j 的扫描的两次连续扫描中的最近邻。 (j,l) 形成 i 的对应关系。然后，为了验证 j 和 l 都是边缘点，我们根据（1）检查局部表面的平滑度。在这里，考虑到单个扫描不能包含来自同一边缘线的多个点，我们特别要求 j 和 l 来自不同的扫描。只有一个例外是边缘线位于扫描平面上。但是，如果是这样，则边缘线退化并在扫描平面上显示为直线，并且首先不应提取边缘线上的特征点。

图片包含 图形用户界面

描述已自动生成

Fig. 6. Reprojecting point cloud to the end of a sweep. The blue colored line segment represents the point cloud perceived during sweep k, . At the end of sweep k, is reprojected to time stamp to obtain (the green colored line segment). Then, during sweep k + 1, and the newly perceived point cloud (the orange colored line segment) are used together to estimate the lidar motion.

图 6. 将点云重新投影到扫描结束。蓝色线段表示在扫描 k， 期间感知的点云。在扫描 k 结束时， 被重新投影到时间戳 以获得 （绿色线段）。然后，在扫描 k + 1 期间，

和新感知的点云 （橙色线段）一起用于估计激光雷达运动。

手机屏幕的截图

低可信度描述已自动生成

Fig. 7. Finding an edge line as the correspondence for an edge point in (a), and a planar patch as the correspondence for a planar point in (b). In both (a) and (b), j is the closest point to the feature point, found in P¯ k. The orange colored lines represent the same scan of j, and the blue colored lines are the two consecutive scans. To find the edge line correspondence in (a), we find another point, l, on the blue colored lines, and the correspondence is represented as (j, l). To find the planar patch correspondence in (b), we find another two points, l and m, on the orange and blue colored lines, respectively. The correspondence is (j, l, m).

图 7. 找到一条边缘线作为 (a) 中的边缘点的对应关系，以及平面贴片作为 (b) 中的平面点的对应关系。在 (a) 和 (b) 中，j 是距离特征点最近的点，在 P¯ k 中找到。橙色线代表 j 的相同扫描，

蓝色线条是两次连续扫描。为了找到 (a) 中的边缘线对应关系，我们在蓝色线上找到另一个点 l，对应关系表示为 (j, l)。为了找到 (b) 中的平面补丁对应关系，我们分别在橙色和蓝色线上找到另外两个点 l 和 m。对应关系为 (j, l, m)。

Fig. 7(b) shows the procedure of finding a planar patch as the correspondence of a planar point. Let i be a point in , k+1. The planar patch is represented by three points. Similar to the last paragraph, we find the closest neighbor of i in P¯ k, denoted as j. Then, we find another two points, l and m, as the closest neighbors of i, one in the same scan of j, and the other in the two consecutive scans to the scan of j. This guarantees that the three points are non-collinear. To verify that j, l, and m are all planar points, we check again the smoothness of the local surface based on (1).

图 7(b) 显示了寻找平面贴片作为平面点的对应关系的过程。设 i 为 中的一个点，。平面贴片由三个点表示。与上一段类似，我们在 中找到 i 的最近邻，记为 j。然后，我们找到另外两个点 l 和 m，作为 i 的最近邻，一个在j的同一次扫描中，另一个在j的连续两次扫描中。这保证了三个点是非共线的。为了验证 j、l 和 m 都是平面点，我们根据 (1) 再次检查局部表面的平滑度。

With the correspondences of the feature points found, now we derive expressions to compute the distance from a feature point to its correspondence. We will recover the lidar motion by minimizing the overall distances of the feature points in the next section. We start with edge points. For a point , if (j, l) is the corresponding edge line, , the point to line distance can be computed as

通过找到特征点的对应关系，现在我们推导出计算从特征点到其对应关系的距离的表达式。我们将在下一节中通过最小化特征点的总距离来恢复激光雷达运动。我们从边缘点开始。对于一个点 ，如果 (j,l) 是对应的边线，，点到线的距离可以计算为

文本, 信件

描述已自动生成

where , , and are the coordinates of points i, j, and l in {L}, respectively. Then, for a point , if (j, l, m) is the corresponding planar patch, , the point to plane distance is

其中 , , 和 分别是 {L} 中点 i、j 和 l 的坐标。那么，对于一个点 ，如果 (j, l, m) 是对应的平面块，，则点到平面的距离为

文本, 信件

描述已自动生成

Whe is the coordinates of point m in {L}

**C. Motion Estimation 运动估计**

The lidar motion is modeled with constant angular and linear velocities during a sweep. This allows us to linear interpolate the pose transform within a sweep for the points that are received at different times. Let t be the current time stamp, and recall that is the starting time of sweep . Let be the lidar pose transform between . contains rigid motion of the lidar in 6-DOF, , where are translations along the axes of {L}, respectively, and are rotation angles, following the right-hand rule. Given a point, let be its time stamp, and let be the pose transform between . can be computed by linear interpolation of

激光雷达运动在扫描过程中以恒定的角速度和线速度建模。这使我们能够在扫描内对在不同时间接收到的点进行位姿变换的线性插值。令 t 为当前时间戳，并回想 是扫描 k+1 的开始时间。

令 为 之间的激光雷达姿态变换。 包含激光雷达在 6 自由度下的刚性运动， ，其中 分别是沿 {L} 的 x 轴、y 轴和 z 轴的平移， 是旋转角度，遵循右手法则。给定一个点 ，令 为其时间戳，令 为 之间的位姿变换。 可以通过 的线性插值计算

Recall that and are the sets of edge points and planar points extracted from , and and are the sets of points reprojected to the beginning of the sweep, . To solved the lidar motion, we need to establish a geometric relationship between and , or and . Using the transform in (4), we can derive,

回想一下， 和 是从 中提取的边缘点和平面点的集合，而 和 是重投影到扫描开始的点集 。为了求解激光雷达运动，我们需要建立 和，或者 和 之间的几何关系。使用（4）中的变换，我们可以推导出:



where is the coordinates of a point i in or , is the corresponding point in or , is the a-th to b-th entries of , and R is a rotation matrix defined by the Rodrigues formula [25],

其中 为 或 中点 i 的坐标， 为 或 中的对应点 是 ) 的第a到第b个条目，R是由罗德里格斯公式[25]定义的旋转矩阵:

In the above equation, θ is the magnitude of the rotation

在上面的等式中，θ是旋转的幅度，

ω is a unit vector representing the rotation direction

ω 是表示旋转方向的单位向量

and is the skew symmetric matrix of [25].

是 ω [25] 的斜对称矩阵。

Recall that (2) and (3) compute the distances between points in and and their correspondences. Combining (2) and (4)-(8), we can derive a geometric relationship between an edge point in and the corresponding edge line,

Similarly, combining (3) and (4)-(8), we can establish another geometric relationship between a planar point in and the corresponding planar patch.

类似地，结合（3）和（4）-（8），我们可以在 中的一个平面点和对应的平面补丁之间建立另一种几何关系

Finally, we solve the lidar motion with the LevenbergMarquardt method [26]. Stacking (9) and (10) for each feature point in and , we obtain a nonlinear function,

最后，我们使用 Levenberg Marquardt 方法 [26] 求解激光雷达运动。对 Ek+1 和 Hk+1 中的每个特征点堆叠 (9) 和 (10)，我们得到一个非线性函数，

where each row of f corresponds to a feature point, and d contains the corresponding distances. We compute the Jacobian matrix of f with respect to , denoted as J, where . Then, (11) can be solved through nonlinear iterations by minimizing d toward zero,

其中 f 的每一行对应一个特征点，d 包含相应的距离。我们计算 f 关于 的 Jacobian 矩阵，记为 J，其中 。然后，(11) 可以通过非线性迭代通过将 d 最小化到零来求解，

λ is a factor determined by the Levenberg-Marquardt method.

λ 是由 Levenberg-Marquardt 方法确定的因子。

文本

低可信度描述已自动生成

**D. Lidar Odometry Algorithm**

The lidar odometry algorithm is shown in Algorithm 1. The algorithm takes as inputs the point cloud from the last sweep, , the growing point cloud of the current sweep, , and the pose transform from the last recursion, . If a new sweep is started, is set to zero (line 4-6). Then, the algorithm extracts feature points from to construct and in line 7. For each feature point, we find its correspondence in (line 9-19). The motion estimation is adapted to a robust fitting [27]. In line 15, the algorithm assigns a bisquare weight for each feature point. The feature points that have larger distances to their correspondences are assigned with smaller weights, and the feature points with distances larger than a threshold are considered as outliers and assigned with zero weights. Then, in line 16, the pose transform is updated for one iteration. The nonlinear optimization terminates if convergence is found, or the maximum iteration number is met. If the algorithm reaches the end of a sweep, is reprojected to time stamp using the estimated motion during the sweep. Otherwise, only the transform is returned for the next round of recursion

激光雷达测距算法如算法 1 所示。该算法将上次扫描的点云 、当前扫描的增长点云 和上次递归的位姿变换 作为输入。如果开始新的扫描，则 设置为零（第 4-6 行）。然后，

该算法从 中提取特征点以在第 7 行构造 和 。对于每个特征点，我们在 中找到其对应关系（第 9-19 行）。运动估计适用于鲁棒拟合[27]。在第 15 行，该算法为每个特征点分配一个二方权重。

与其对应的距离较大的特征点被分配较小的权重，距离大于阈值的特征点被认为是异常值并分配零权重。然后，在第 16 行，位姿变换形式被更新一次迭代。

如果发现收敛或达到最大迭代次数，非线性优化将终止。如果算法到达扫描结束，则使用扫描期间估计的运动将 重新投影到时间戳 。否则，只返回变换 用于下一轮递归

图示

描述已自动生成

Fig. 8. Illustration of mapping process. The blue colored curve represents the lidar pose on the map, , generated by the mapping algorithm at sweep k. The orange color curve indicates the lidar motion during sweep k + 1, , computed by the odometry algorithm. With and , the undistorted point cloud published by the odometry algorithm is projected onto the map, denoted as (the green colored line segments), and matched with the existing cloud on the map, (the black colored line segments).

图 8. 映射过程示意图。蓝色曲线表示地图上的激光雷达姿态 ，由映射算法在扫描 k 处生成。橙色曲线表示扫描 k + 1、 期间的激光雷达运动，由里程计算法计算得出。对于 和 ，

将里程计算法发布的未失真点云投影到地图上，表示为 （绿色线段），并与地图上现有的点云 （黑色线段）匹配。

VI. LIDAR MAPPING

The mapping algorithm runs at a lower frequency then the odometry algorithm, and is called only once per sweep. At the end of sweep k + 1, the lidar odometry generates a undistorted point cloud, , and simultaneously a pose transform, , containing the lidar motion during the sweep, between . The mapping algorithm matches and registers in the world coordinates, {W}, illustrated in Fig. 8. To explain the procedure, let us define as the point cloud on the map, accumulated until sweep k, and let be the pose of the lidar on the map at the end of sweep k, . With the outputs from the lidar odometry, the mapping algorithm extents for one sweep from to , to obtain , and projects into the world coordinates, {W}, denoted as . Next, the algorithm matches with by optimizing the lidar pose .

映射算法以低于里程计算法的频率运行，并且每次扫描仅调用一次。在扫描 k + 1 结束时，激光雷达里程计生成一个未失真的点云 ，同时生成一个姿态变换 ，其中包含扫描期间的激光雷达运动，介于 。

映射算法在世界坐标 {W} 中匹配并注册 ，如图 8 所示。为了解释该过程，让我们将 定义为地图上的点云，累积直到扫描 k，并让 是扫描k结束时激光雷达在地图上的位姿，。使用激光雷达里程计的输出，

映射算法将 从 扩展到 一次扫描，以获得 ，并将 投影到世界坐标 {W} 中，记为 。接下来，该算法通过优化激光雷达姿态 将 与 匹配。

The feature points are extracted in the same way as in Section V-A, but 10 times of feature points are used. To find correspondences for the feature points, we store the point cloud on the map, , in 10m cubic areas. The points in the cubes that intersect with are extracted and stored in a 3D KD-tree [24]. We find the points in within a certain region around the feature points. Let be a set of surrounding points. For an edge point, we only keep points on edge lines in , and for a planar point, we only keep points on planar patches. Then, we compute the covariance matrix of , denoted as **M**, and the eigenvalues and eigenvectors of **M**, denoted as **V** and **E**, respectively. If is distributed on an edge line, contains one eigenvalue significantly larger than the other two, and the eigenvector in **E** associated with the largest eigenvalue represents the orientation of the edge line. On the other hand, if is distributed on a planar patch, **V** contains two large eigenvalues with the third one significantly smaller, and the eigenvector in **E** associated with the smallest eigenvalue denotes the orientation of the planar patch. The position of the edge line or the planar patch is determined by passing through the geometric center of .

特征点的提取方法与第 V-A 节相同，但使用了 10 倍的特征点。为了找到特征点的对应关系，我们将点云存储在地图 上，在 10m 立方区域中。立方体中与 相交的点被提取并存储在 3D KD-tree [24] 中。

我们在特征点周围的某个区域内找到 中的点。令 为一组周围点。对于边缘点，我们只保留 中边缘线上的点，而对于平面点，我们只保留平面块上的点。然后，我们计算 的协方差矩阵，记为 **M**，以及 **M** 的特征值和特征向量，分别表示为 **V** 和 **E**。如果 分布在一条边线上，则**V**包含一个明显大于其他两个的特征值，**E**中与最大特征值相关的特征向量代表了边线的方向。另一方面，如果 分布在平面贴片上，则 **V** 包含两个大特征值，第三个明显更小，并且 **E** 中与最小特征值相关的特征向量表示平面贴片的方向。通过 的几何中心确定边缘线或平面贴片的位置。

图片包含 图形用户界面

描述已自动生成

Fig. 9. Integration of pose transforms. The blue colored region illustrates the lidar pose from the mapping algorithm, , generated once per sweep. The orange colored region is the lidar motion within the current sweep, , computed by the odometry algorithm. The motion estimation of the lidar is the combination of the two transforms, at the same frequency as .

图 9. 姿势变换的集成。蓝色区域说明了映射算法 的激光雷达姿态，每次扫描生成一次。橙色区域是当前扫描 内的激光雷达运动，由里程计算法计算得出。激光雷达的运动估计是两种变换的组合，频率与 相同。

To compute the distance from a feature point to its correspondence, we select two points on an edge line, and three points on a planar patch. This allows the distances to be computed using the same formulations as (2) and (3). Then, an equation is derived for each feature point as (9) or (10), but different in that all points in share the same time stamp, . The nonlinear optimization is solved again by a robust fitting [27] through the Levenberg-Marquardt method [26], and is registered on the map. To evenly distribute the points, the map cloud is downsized by a voxel grid filter [28] with the voxel size of 5cm cubes.

为了计算一个特征点到它的对应点的距离，我们在一条边线上选择两个点，在一个平面补丁上选择三个点。这允许使用与 (2) 和 (3) 相同的公式计算距离。然后，为每个特征点推导出一个方程为（9）或（10），

但不同之处在于 中的所有点共享相同的时间戳 。非线性优化通过 Levenberg-Marquardt 方法 [26] 的鲁棒拟合 [27] 再次解决，并且 被注册在地图上。为了均匀分布这些点，地图云通过体素网格过滤器 [28] 缩小尺寸，体素大小为 5 厘米立方体。

Integration of the pose transforms is illustrated in Fig. 9. The blue colored region represents the pose output from the lidar mapping, , generated once per sweep. The orange colored region represents the transform output from the lidar odometry, , at a frequency round 10Hz. The lidar pose with respect to the map is the combination of the two transforms, at the same frequency as the lidar odometry.

位姿变换的集成如图 9 所示。蓝色区域表示来自激光雷达映射的位姿输出， ，每次扫描生成一次。橙色区域表示来自激光雷达里程计的变换输出，，频率约为 10Hz。相对于地图的激光雷达姿态是两种变换的组合，频率与激光雷达里程计相同。

VII. EXPERIMENTS

During experiments, the algorithms processing the lidar data run on a laptop computer with 2.5GHz quad cores and 6Gib memory, on robot operating system (ROS) [29] in Linux. The method consumes a total of two cores, the odometry and mapping programs run on two separate cores. Our software code and datasets are publicly available1,2 .

在实验期间，处理激光雷达数据的算法在 Linux 中的机器人操作系统 (ROS) [29] 上运行在具有 2.5GHz 四核和 6Gib 内存的笔记本电脑上。该方法总共消耗两个核心，里程计和映射程序在两个独立的核心上运行。我们的软件代码和数据集是公开的1,2。

图表, 图示

描述已自动生成

Fig. 11. Matching errors for corridor (red), lobby (green), vegetated road (blue) and orchard (black), corresponding to the four scenes in Fig. 10.

图 11 走廊（红色）、大厅（绿色）、植被道路（蓝色）和果园（黑色）的匹配错误，对应于图 10 中的四个场景。

电脑游戏画面

中度可信度描述已自动生成

Fig. 10. Maps generated in (a)-(b) a narrow and long corridor, (c)-(d) a large lobby, (e)-(f) a vegetated road, and (g)-(h) an orchard between two rows of trees. The lidar is placed on a cart in indoor tests, and mounted on a ground vehicle in outdoor tests. All tests use a speed of 0.5m/s.

图 10. (a)-(b) 狭长走廊、(c)-(d) 大大厅、(e)-(f) 植被道路和 (g)-(h) 中生成的地图两排树之间的果园。激光雷达在室内测试中放置在推车上，在室外测试中安装在地面车辆上。所有测试均使用 0.5m/s 的速度。

表格

描述已自动生成

**A. Indoor & Outdoor Tests**

The method has been tested in indoor and outdoor environments. During indoor tests, the lidar is placed on a cart together with a battery and a laptop computer. One person pushes the cart and walks. Fig. 10(a) and Fig. 10(c) show maps built in two representative indoor environments, a narrow and long corridor and a large lobby. Fig. 10(b) and Fig. 10(d) show two photos taken from the same scenes. In outdoor tests, the lidar is mounted to the front of a ground vehicle. Fig. 10(e) and Fig. 10(g) show maps generated from a vegetated road and an orchard between two rows of trees, and photos are presented in Fig. 10(f) and Fig. 10(h), respectively. During all tests, the lidar moves at a speed of 0.5m/s.

该方法已在室内和室外环境中进行了测试。在室内测试期间，激光雷达与电池和笔记本电脑一起放置在推车上。一个人推着车走。图 10(a) 和图 10(c) 显示了在两个有代表性的室内环境中构建的地图，一个狭窄而长的走廊和一个大大厅。如图。图 10(b) 和图 10(d) 显示了从相同场景拍摄的两张照片。在户外测试中，激光雷达安装在地面车辆的前部。图 10(e) 和图 10(g) 显示了从两行树木之间的植被道路和果园生成的地图，照片分别显示在图 10(f) 和图 10(h) 中。在所有测试中，激光雷达以 0.5m/s 的速度移动

To evaluate local accuracy of the maps, we collect a second set of lidar clouds from the same environments. The lidar is kept stationary and placed at a few different places in each environment during data selection. The two point clouds are matched and compared using the point to plane ICP method [9]. After matching is complete, the distances between one point cloud and the corresponding planer patches in the second point cloud are considered as matching errors. Fig. 11 shows the density of error distributions. It indicates smaller matching errors in indoor environments than in outdoor. The result is reasonable because feature matching in natural environments is less exact than in manufactured environments.

为了评估地图的局部准确性，我们从相同的环境中收集了第二组激光雷达云。在数据选择期间，激光雷达保持静止并放置在每个环境中的几个不同位置。使用点对面ICP方法[9]对两个点云进行匹配和比较。匹配完成后，

一个点云与第二个点云中对应的平面斑块之间的距离被认为是匹配错误。图 11 显示了误差分布的密度。它表明室内环境中的匹配误差比室外环境小。

结果是合理的，因为自然环境中的特征匹配不如人造环境中精确。

Additionally, we conduct tests to measure accumulated drift of the motion estimate. We choose corridor for indoor experiments that contains a closed loop. This allows us to start and finish at the same place. The motion estimation generates a gap between the starting and finishing positions, which indicates the amount of drift. For outdoor experiments, we choose orchard environment. The ground vehicle that carries the lidar is equipped with a high accuracy GPS/INS for ground truth acquisition. The measured drifts are compared to the distance traveled as the relative accuracy, and listed in Table I. Specifically, Test 1 uses the same datasets with Fig. 10(a) and Fig. 10(g). In general, the indoor tests have a relative accuracy around 1% and the outdoor tests are around 2.5%.

此外，我们进行测试以测量运动估计的累积漂移。我们选择包含闭环的室内实验走廊。这使我们可以在同一个地方开始和结束。运动估计会在起始位置和结束位置之间产生一个间隙，该间隙指示漂移量。

对于户外实验，我们选择果园环境。携带激光雷达的地面车辆配备了高精度 GPS/INS，用于获取地面实况。将测量的漂移与作为相对精度的行进距离进行比较，并列在表 I 中。具体而言，测试 1 使用与图 10(a) 和图 10 相同的数据集。一般来说，室内测试的相对准确度在 1% 左右，室外测试的相对准确度在 2.5% 左右。

**B. Assistance from an IMU**

We attach an Xsens MTi-10 IMU to the lidar to deal with fast velocity changes. The point cloud is preprocessed in two ways before sending to the proposed method, 1) with orientation from the IMU, the point cloud received in one sweep is rotated to align with the initial orientation of the lidar in that sweep, 2) with acceleration measurement, the motion distortion is partially removed as if the lidar moves at a const velocity during the sweep. The point cloud is then processed by the lidar odometry and mapping programs.

我们将 Xsens MTi-10 IMU 连接到激光雷达以应对快速的速度变化。点云在发送到所提出的方法之前以两种方式进行预处理，1) 使用来自 IMU 的方向，在一次扫描中接收到的点云被旋转以与该扫描中激光雷达的初始方向对齐，2) 使用加速度测量,运动失真被部分消除，就好像激光雷达在扫描过程中以恒定速度移动一样。然后由激光雷达里程计和映射程序处理点云。

The IMU orientation is obtained by integrating angular rates from a gyro and readings from an accelerometer in a Kalman filter [1]. Fig. 12(a) shows a sample result. A person holds the lidar and walks on a staircase. When computing the red curve, we use orientation provided by the IMU, and our method only estimates translation. The orientation drifts over 25◦ during 5 mins of data collection. The green curve relies only on the optimization in our method, assuming no IMU is available. The blue curve uses the IMU data for preprocessing followed by the proposed method. We observe small difference between the green and blue curves. Fig. 12(b) presents the map corresponding to the blue curve. In Fig. 12(c), we compare two closed views of the maps in the yellow rectangular in Fig. 12(b). The upper and lower figures correspond to the blue and green curves, respectively. Careful comparison finds that the edges in the upper figure are sharper

IMU 方向是通过在卡尔曼滤波器 [1] 中集成来自陀螺仪的角速率和来自加速度计的读数来获得的。图 12(a) 显示了一个示例结果。一个人拿着激光雷达走在楼梯上。在计算红色曲线时，我们使用 IMU 提供的方向，我们的方法只估计平移。在 5 分钟的数据收集过程中，方向漂移超过 25°。假设没有可用的 IMU，绿色曲线仅依赖于我们方法中的优化。蓝色曲线使用 IMU 数据进行预处理，然后是所提出的方法。我们观察到绿色和蓝色曲线之间的微小差异。如图。图 12(b) 给出了对应于蓝色曲线的图。在图 12（c）中，我们比较了图 12（b）中黄色矩形中地图的两个闭合视图。上图和下图分别对应蓝色和绿色曲线。仔细对比发现上图的边缘更锐利

Table II compares relative errors in motion estimation with and without using the IMU. The lidar is held by a person walking at a speed of 0.5m/s and moving the lidar up and down at a magnitude around 0.5m. The ground truth is manually measured by a tape ruler. In all four tests, using the proposed method with assistance from the IMU gives the highest accuracy, while using orientation from the IMU only leads to the lowest accuracy. The results indicate that the IMU is effective in canceling the nonlinear motion, with which, the proposed method handles the linear motion.

表 II 比较了使用和不使用 IMU 的运动估计的相对误差。激光雷达由一个人以 0.5m/s 的速度行走并以 0.5m 左右的幅度上下移动激光雷达。基本事实是由卷尺手动测量的。在所有四个测试中，在 IMU 的帮助下使用所提出的方法可以获得最高的准确度，而使用 IMU 的方向只会导致最低的准确度。结果表明，IMU有效地消除了非线性运动，所提出的方法利用它来处理线性运动。

图片包含 图示

描述已自动生成

Fig. 12. Comparison of results with/without aiding from an IMU. A person holds the lidar and walks on a staircase. The black dot is the starting point. In (a), the red curve is computed using orientation from the IMU and translation estimated by our method, the green curve relies on the optimization in our method only, and the blue curve uses the IMU data for preprocessing followed by the method. (b) is the map corresponding to the blue curve. In (c), the upper and lower figures correspond to the blue and green curves, respectively, using the region labeled by the yellow rectangular in (b). The edges in the upper figure are sharper, indicating more accuracy on the map.

图 12. 有/无 IMU 辅助的结果比较。一个人拿着激光雷达走在楼梯上。黑点是起点。在（a）中，红色曲线是使用来自 IMU 的方向和我们的方法估计的平移来计算的，绿色曲线仅依赖于我们方法中的优化，

蓝色曲线使用 IMU 数据进行预处理，然后是该方法。 (b) 是对应于蓝色曲线的地图。在（c）中，上图和下图分别对应于蓝色和绿色曲线，使用（b）中黄色矩形标记的区域。上图中的边缘更锐利，

表示地图上的准确性更高。

表格

描述已自动生成