

# Mapping 3.0 阶段性测试报告

<https://idriverplus.com>

智行者定位组

## 摘要

本文档总结了智行者定位组 2019 年 8 月份在 Mapping 3.0 项目的工作。我们完善了 Mapping 3.0 的流水线式建图，提出了针对 GPS 跳变、自动化准出、多 Bag 包地图拼接、点云去重影等重要问题的解决方案。同时，我们在以往建图数据上建立了共 748.1GB, 59 万平米的规模化测试数据集。在该数据集上的综合测试表明，Mapping 3.0 目前已达到 96.25% 建图成功率，一次性成功率为 93.75%，建图速度达到 4.2-4.9 分钟/千平米，较过去算法有显著提升。

关键词：建图算法，激光点云，高精地图

## 1 Mapping 3.0 计划

### 1.1 3.0 目标与整体计划

Mapping 3.0 计划（以下简称 3.0）是为了针对既有 2.0 建图框架提出的新算法框架。在该算法框架中，我们希望实现以下目标：

- 消除 2.0 存在的建图困难问题。2.0 在许多建图能力边界附近会产生一些需要人工反复调试的场景，例如 GPS 缺失、跳变、多 Bag 包拼接、闭环检查、地图重影，等等。同时，其整体操作流程较为复杂，对地图专员要求较高，操作过程以人工调用算法为主，无法实现 24 小时在线。在 3.0 中，我们希望对上述问题进行消解或简化，以达到自动化建图的目的。
- 增加 3.0 的新特性。3.0 将使用基于建图服务器的全自动建图流水线。在默认情况下，绝大多数建图请求，从数据上传到建图结果的准出都无需人

工参与。服务器将实时接管上传完毕的数据并启动流水线，而维护人员可以在建图结果完成后收到建图质量报告，决定通过与否。

- 实现车端和现场制图的支持。在 3.0 中，我们将允许现场人员采集数据之后，直接在现场或车端进行制图。从数据采集到完成制图应该在一个小时以内。

综上，3.0 是定位室在 2019 年下半年对建图算法的整体优化方案。按照定位室整体规划，3.0 在近期主要有以下几个里程碑：

- 在 2019 年 7 月底，实现 2.0 的算法移植与重构，完成 2.0 的基础功能。
- 在 2019 年 8 月底，实现流水线式自动建图过程，完成本地服务器部署。部分或完全地解决 2.0 遗留问题。
- 在 2019 年 9 月底，完成 3.0 的调试工具，完全解决 2.0 存在的遗留问题。在云端部署 3.0 系统并取代 2.0，承接新的建图任务。

### 1.2 7 份完成情况简述

在七月份，我们主要完成了：

- 对 2.0 代码进行重构和规范化；
- 建立了地图数据集，对格式规范的数据集可自动化运行；
- 服务器制图功能。开启远程服务器后，可与服务器通信进行任务的建立。

### 1.3 8 月任务列表

在 8 月份，我们预计完成的任务主要有：

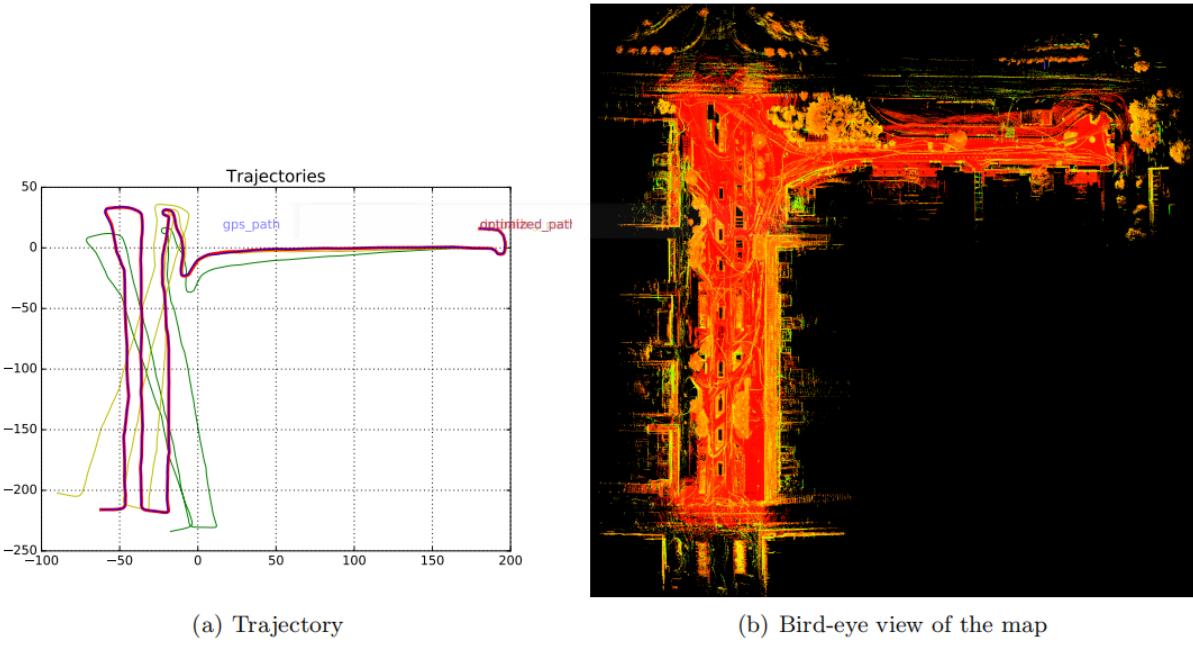


图 1: Mapping 3.0 建图的例子 (深圳华强北)。(a): 3.0 计算的运动轨迹; (b): 3.0 建立的点云地图

1. 完善流水线功能: 自动准入准出、报告生成与发送、内存管理。
2. 部分或完全解决 2.0 的遗留问题: GPS 跳变处理、多 Bag 包数据拼接、地图去重影。

图 1 为 3.0 建立地图的例子。该例为 2.0 来自深圳华强北, 为 2.0 建图困难场景之一, 在 3.0 中可以一次性建成。我们在后文会详细描述 3.0 针对 2.0 的算法改进和详细测试结果。

## 1.4 本报告测试内容

本报告主要描述 3.0 在 2019 年 8 月底的整体改进和详细测试过程与结果。针对定位室对外输出, 我们量化测试以下几个指标。

### Mapping3.0 系统平台基本性能指标

- 1 可以连续 24 小时正常运行, 无宕机和严重内存泄露问题;
- 2 在服务器配置上 (内存  $\geq 32\text{GB}$ , 核心数量  $\geq 8$  个), 能够同时响应多于 4 条建图请求;
- 3 能够自动给出建图最终报告, 描述建图情况;

**Mapping3.0 建图能力指标** 针对 minimal、standard 和 large 三类地图, 总测试数量不少于 80 个, 测试以下指标:

- 1 在 ODD 内<sup>1</sup>, 建图成功率  $\geq 95\%$ , 无需人工干预;
- 2 建图平均速度  $\geq 6000\text{m}^2/\text{h}$ , 或每 GB 数据耗时  $\leq 30$  分钟;
- 3 对于失败的地图, 可以通过阅读报告分析具体失败原因。

对于单个主题的算法改进 (例如点云去重影), 难以量化描述的, 我们将描述改进原理, 并给出一些改进前后的对比结果。这部分内容将在实验章节描述。

本报告接下来阐述 3.0 各项改进的原理, 然后给出测试结果, 最后给出总结和展望。对原理不感兴趣的读者可跳过第 2 章。

## 2 8 月改进内容与方法

本节主要描述各主题的改进内容和原理, 可能含有一部分专业性描述。

<sup>1</sup>指数据符合采集规范, 场景在能力边界以内的建图任务。

## 2.1 8月代码改动概述

Mapping 3.0 的 master 分支在 8 月共有 112 次<sup>2</sup>提交，代码新增 23311 行，删除 16607 行，净增 6704 行。目前源代码共 9000 行。

## 2.2 服务器功能拓展

8 月份建图服务主要有以下功能拓展。

1. **完整的流水线。**现在流水线包括“数据下载”、“数据准备”、“Lidar Odom”、“后端优化”、“数据验证”五个环节。8 月份我们完善了整条流水线，修复了一些 bug，使之能在绝大多数数据集上完整运行。
2. **自动化报告生成。**在建图任务完成之后，Mapping 3.0 会自动生成一个建图报告，主要有建图长度、数据包大小、建图耗时、建图是否成功、仿真是否通过、点云和 DB 下载链接等信息。图 2 是一份报告的首页，其简报部分含有建图轨迹、地图点云俯视图等信息，第二页的详细报告则给出运行时的详细数据。报告还会提示出回环点的位置（浅蓝色框），以供检查人员检查局部的结果。
3. **建图完成后自动邮件发送。**现在 3.0 支持完成建图后自动发送邮件的功能，以便今后上线查看结果。3.0 会把报告作为邮件附件发送给定位组成员，地图质检人员可以点击报告中的下载链接，下载点云和 DB 数据。
4. **服务器并发数量的控制。**现在服务器有最大并发数量限制。最大数量可以由用户控制，也可以由程序自动分配（需要指定最大可用内存数量）。图 3 显示了 3.0 的命令行界面。在最大内存为 32G 的情况下，3.0 自动分配了 9 条并发流水线<sup>3</sup>。由图可见它们各自处于不同的运行阶段，互不干扰。

### Report of Mapping in fujian fuzhouwanfuzhongxin

idriverplus.com

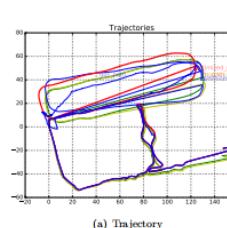
Mapping 3.0 version 0.1, Builder: Peng GuoQi

August 22, 2019

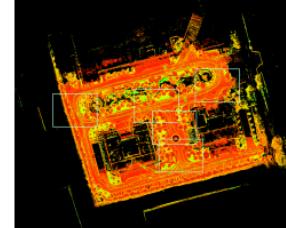
#### 1 Brief Report

Table 1: General Status

Map Name	fujian_fuzhouwanfuzhongxin					
Time Usage	11m19.2s	Valid Bags	5	Total Size	2.31 GB	
Length	657m	Area	6.57e+03m <sup>2</sup>	Build Success	Yes	Total Kfs   1517
Checkout	YES	Reason				
Map DB Link	<a href="http://127.0.0.1:8000/fujian_fuzhouwanfuzhongxin/map.db">http://127.0.0.1:8000/fujian_fuzhouwanfuzhongxin/map.db</a>					
Point Cloud Link	<a href="http://127.0.0.1:8000/fujian_fuzhouwanfuzhongxin/result.pcd">http://127.0.0.1:8000/fujian_fuzhouwanfuzhongxin/result.pcd</a>					



(a) Trajectory



(b) Bird-eye view of the map

#### 2 Detail Report

data fetching: file already exist, skip downloading.  
data fetching: ok.  
Data Fetching time usage: 30.793 seconds.

图 2：自动生成报告的首页，该例为福建福州万福中心地图。

## 2.3 GPS 跳变处理

### 2.3.1 GPS 跳变的分类

GPS 信号跳变是 2.0 难以处理大规模地图的一个主要原因。由于 GPS 多径效应的存在，我们拿到的 GPS 信号值可能与真实情况不符，但 GPS 接收机认为信号有高置信度。详细来说，GPS 跳变可以分为以下四类：

- **GPS 跳变。**指 GPS 信号短时间内跳至别处，一段时间内跳回正确值的情况。
- **GPS 阶跃。**指 GPS 跳至错误值并持续较长时间，表现为阶跃特征。
- **GPS 渐变。**指 GPS 缓慢变化至一个错误的值。
- **GPS 失效。**指 GPS 信号变化至零或极远处值。

以上四种情况在实际当中均可能出现。由于实际行驶轨迹的不可预测性，GPS 信号跳变也可能是各种情况

<sup>2</sup>从 2019.08.01 至 2019.08.26。

<sup>3</sup>我们建议在 32G 机器上并发数为 4，图中运行小型数据集，可以适当多开。

```

Success: 0/Failed: 0/Running: 9 of 9/TODO: 32
id Status Step Name
0 working Pose Optimization beijing_yizhuang
1 working Lider Odom beijing_jinghuen
2 working Lider Odom beijing_shiyuanhuayushu
3 working Data Preparing beijing_haidianshuangyushu
4 working Lider Odom beijing_qinghuazhenshi
5 working Data Preparing beijing_dianxin
6 working Data Preparing beijing_mudanyuankejilou
7 working Data Fetching beijing_tengxunzhongchuandon
8 working Data Fetching beijing_shougangbingqiu
9 TODO None chongqing_keirui
10 TODO None chongqing_datang
11 TODO None fujian_donghuxiaozhen2
12 TODO None guangdong_zhongshannantou
13 TODO None guangdong_zhongshannanlishen
14 TODO None guangdong_foshanbiguiyuanzongbu
15 TODO None guangdong_foshanbiguiyuanywuye
16 TODO None guangdong_eokeq12
17 TODO None guangdong_guangjiaohui
18 TODO None guangdong_guangzhoukexuezhongxin1
19 TODO None guangdong_guangzhoukexuezhongxin2
20 TODO None guangdong_guangzhoukexuezhongxin3
21 TODO None guangdong_guangzhouxiangxue
22 TODO None guangdong_haizhushidi
23 TODO None guangdong_huanguzhuque
24 TODO None guangdong_kemeiqingjie
25 TODO None guizhou_guanshanhu
26 TODO None guizhou_yidongdasha
27 TODO None hebei_beodingbaixia
28 TODO None hebei_qinhuangdeotienxiediyiguan
29 TODO None henan_xuchangbeiyou
30 TODO None xiangan_guojiichangbashi
31 TODO None hunan_changshameixihu
32 TODO None jiangsu_xuzhuanggaoxin
33 TODO None jiangsu_suzhouchengguanju2
34 TODO None neimenggu_menzhoulitaowei1
35 TODO None neimenggu_menzhoulitaowei2
36 TODO None shanxi_xiandetangbuyecheng1
37 TODO None shanxi_xianjingxueyuan
38 TODO None shanghai_zhongguoyidong1
39 TODO None shanghai_lvdi
40 TODO None zhejiang_hangzhouxiti

```

图 3: Mapping 3.0 在小型数据集上自动分配了 9 条流水线。

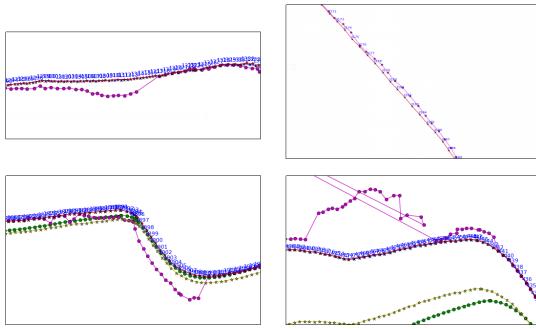


图 4: GPS 跳变的实际例子：紫色线：GPS 轨迹；红色线 + 蓝色标记：优化轨迹

的混合，而非属于单一类别。以图 4 为例<sup>4</sup>，左上角 GPS 信号出现了渐变，然后跳回正确位置；右上角 GPS 出现随机跳动；左下角 GPS 跳变至错误值，然后返回；右下角 GPS 跳至极远处无效值，然后返回。由此可见，对 GPS 跳变进行归类，是一种人为的，仅

<sup>4</sup>本例以优化轨迹，即红色轨迹/蓝色数字作为参考。

为了方便讨论的做法，实际对每次跳变进行分类是比较困难的。

### 2.3.2 后端轨迹优化

GPS 跳变主要影响后端轨迹优化的结果。后端轨迹优化是一个状态估计问题，它从若干种数据来源中计算最优的轨迹。数据来源主要有以下几类：

1. GPS 定位；
2. IMU+ 轮速积分，在 Mapping 3.0 中称为 Dead Reckoning (DR)；
3. 激光前后帧匹配 (Lidar Odom)。

其中，仅有 GPS 能够提供绝对定位信息（相对于真实世界），其他两种定位来源均为相对定位（相对于上个时刻或之前某个时刻）。所以原则上说，若全部区域没有 GPS，那我们无法给出带绝对位置信息的地图；若长时间缺少 GPS 信号，那么在缺失 GPS 的区域无法保证绝对位置的准确性。

我们以图优化形式描述上述问题，那么整个问题可抽象为一个 Pose Graph [1, 2]。以  $\mathbf{x}_k \in \text{SE}(3)$  表示第  $k$  时刻位姿，那么各种观测表达为：

1. GPS 位姿： $\mathbf{x}_{k,\text{GPS}}$ ；
2. DR 相对位姿（从  $k-1$  至  $k$ ）： $\mathbf{x}_{k-1,k,\text{DR}}$ ；
3. Lidar 相对位姿（从  $k-1$  至  $k$ ）： $\mathbf{x}_{k-1,k,\text{LIDAR}}$ 。

每个数据对应真实位姿的一次观测：

$$\begin{aligned} \mathbf{x}_{k,\text{GPS}} &= \mathbf{x}_k \oplus \mathbf{n}_{\text{GPS}}, \\ \mathbf{x}_{k-1,k,\text{DR}} &= \mathbf{x}_{k-1}^{-1} \mathbf{x}_k \oplus \mathbf{n}_{\text{DR}} \\ \mathbf{x}_{k-1,k,\text{LIDAR}} &= \mathbf{x}_{k-1}^{-1} \mathbf{x}_k \oplus \mathbf{n}_{\text{LIDAR}} \end{aligned} \quad (1)$$

其中  $\oplus$  指  $\text{SE}(3)$  上广义加法， $\mathbf{n}$  为各种噪声，假设服从高斯分布：

$$\begin{aligned} \mathbf{n}_{\text{GPS}} &\sim \mathcal{N}(0, \Sigma_{\text{GPS}}), \\ \mathbf{n}_{\text{DR}} &\sim \mathcal{N}(0, \Sigma_{\text{DR}}), \\ \mathbf{n}_{\text{LIDAR}} &\sim \mathcal{N}(0, \Sigma_{\text{LIDAR}}) \end{aligned} \quad (2)$$

那么整个 Pose Graph 问题由最小二乘问题

$$\{\mathbf{x}\}^* = \arg \min_{\mathbf{x}_k, k=1, \dots, n} \sum \{\rho(\mathbf{e}_{\text{GPS}}^T \Sigma_{\text{GPS}}^{-1} \mathbf{e}_{\text{GPS}}) + \rho(\mathbf{e}_{\text{DR}}^T \Sigma_{\text{DR}}^{-1} \mathbf{e}_{\text{DR}}) + \rho(\mathbf{e}_{\text{LIDAR}}^T \Sigma_{\text{LIDAR}}^{-1} \mathbf{e}_{\text{LIDAR}})\} \quad (3)$$

描述，其中  $\rho$  表示鲁棒核函数（实用中使用 Cauchy 核，理论证明 [3] 加权 Cauchy 核等价于自适应估计，因此 Cauchy 核在噪声大小不确定时比较好用）[4]，不同下标的  $e$  表示不同种类的误差，整个优化问题为各类误差在高斯协方差意义下的  $\Sigma$  范数。

Pose Graph 在理论上是简单易用的，但实际处理中存在几个重要的问题：

- 如何构建合适的 Pose Graph？虽然各类观测的数学定义是明确的，但实际上对于 Lidar 和 DR 的测量，仅添加  $k-1$  到  $k$  时刻的测量信息并不足以保证局部轨迹形状不发生畸变；而局部轨迹形状一旦和测量不符，点云拼接时即可能造成重影。因此，除了 GPS 信息仍然为对单次定位的测量之外，我们会在 Pose Graph 中添加  $k-l, k-l+1, \dots, k-1$  至  $k$  时刻的相对测量，其中  $l$  为参与局部测量的关键帧个数。在实用中，取  $l_{\text{LIDAR}} = 4, l_{\text{DR}} = 2$ 。
- 如何制定优化的流程？为了让建图结果更加完善，我们并不是简单地构建一个 Pose Graph 并求解，而需要进行一定的处理。算法 1 描述了后端优化的整体流程。

---

#### Algorithm 1 后端优化算法流程

---

- 1: 第一轮
  - 2: 第 1 遍优化
  - 3: 添加所有传感器数据，添加 Robust Kernel；
  - 4: 得到优化结果；
  - 5: 判断 outlier，调整 outlier 阈值，去除 outlier，去除 Robust Kernel
  - 6: 第 2 遍优化
  - 7: 得到仅使用 inlier，无 Robust Kernel 的结果；
  - 8: 第二轮
  - 9: 根据第一轮优化结果计算回环检测
  - 10: 带入回环检测信息，执行 2-7 步骤
  - 11: 输出最终结果
- 

它的主要改进在于：第 1 轮优化后我们已得到一个大致准确的轨迹，它们可能来自多个包，所以直接拼接可能会有明显的重影。但是，第 1 轮轨

迹可以为回环检测提供初始位姿，使我们可以在限定区域内搜索有效的回环信息。这些回环信息加入之后，第 2 轮优化会使得全局轨迹更加准确，拼接处重影更少。

- 如何确定各种测量的协方差矩阵？协方差矩阵描述了单次测量的精度信息。在融合各种传感器输入时，必须十分注意它们的原始数据噪声大小，否则可能使融合失效。按照现实中的传感器精度，例如单点 GPS，在信号有效时，我们取：

$$\Sigma_{\text{GPS}} = \begin{bmatrix} \text{diag}(\underbrace{0.15, 0.15, 0.15}_{\text{平移部分}}, \underbrace{0.08, 0.08, 0.08}_{\text{角度部分}}) \end{bmatrix}^2 \quad (4)$$

表明单点 GPS 标准差的平移为 15cm，旋转为  $5^\circ$ ，这是符合现实传感器的。若输入时判断 GPS 信号不好，则会以倍率扩大该协方差矩阵。

对于激光匹配和 DR，它们在局部比较准确，目前取：

$$\begin{aligned} \Sigma_{\text{DR}} &= [\text{diag}(0.05, 0.05, 0.05, 0.008, 0.008, 0.008)]^2 \\ \Sigma_{\text{LIDAR}} &= [\text{diag}(0.05, 0.05, 0.05, 0.008, 0.008, 0.008)]^2 \end{aligned} \quad (5)$$

即它们在两个关键帧之间的标准差为 5cm 和 0.5 度。这是一个比较小的噪声，它保证我们的优化结果局部与 Lidar 和 DR 更加相似。

- 如何判断测量信息是否有效？根据前面所述，GPS 即使在输入时判定为有效，实际也可能出现各种形式的跳变。因此，在优化时，我们检查各类传感器数据的代价函数是否超出阈值  $\delta$ ，若超过，则会将该测量信息在第 2 遍优化时移除。实用中的取值为<sup>5</sup>：

$$\delta_{\text{GPS}} = 0.535, \delta_{\text{DR}} = 1.566, \delta_{\text{LIDAR}} = 1.437. \quad (6)$$

针对 GPS 部分，如果超过 90% 的数据无效，我们会增大  $\delta_{\text{GPS}}$  至两倍，以适应 GPS 不好的场景，

---

<sup>5</sup>这些取值是根据实验中的误差值分布确定的，例如某张 GPS 信号良好的区域，误差的 90% 分位数约在 0.29，而最大值为 0.657。这种分布可以辅助我们确定各类测量的外点阈值。

这种增大最多运行三次，即 GPS 外点的阈值最多为初始值的 8 倍。

总体而言，上述的算法保证我们在信息源存在部分异常值时，算法仍能给出良好的轨迹估计。同时，我们也可以方便地检测地图中的回环，使拼接处更加合理。我们将在实验章节给出一些定量的结果。

## 2.4 仿真与自动准出

仿真和准出作为整个建图中最后一个环节，又称数据验证环节。该环节主要完成点云地图的激光匹配验证、准出项检测以及相关数据保存等内容，任意过程出现问题或者不满足准出检测标准，均会给出相关原因，根据原因优先级输出数据验证最终检测结果，仿真错误优先级将高于准出错误优先级。

---

### Algorithm 2 仿真与自动准出

---

- 1: 仿真
  - 2: 前期数据准备；
  - 3: 获取单个激光匹配数据包；
  - 4: 定位初始化；
  - 5: 若初始化定位成功，进行后续激光匹配；若定位初始化不成功，记录并执行 3 - 5 步骤，进行后续 bag 数据验证；
  - 6: 完成单个数据包定位后，进行 MSF 融合，若还存在未仿真数据包，执行 3 - 6 步骤；
  - 7: 计算自适应激光匹配阈值；
  - 8: 若存在故障信息，则需要判断类型并进行相应操作；
  - 9: 准出检测
  - 10: 仿真结果数据输入；
  - 11: 数据完整性检测；
  - 12: 平滑性、断层、位置和角度偏差检测与统计；
  - 13: 相关数据保存；
  - 14: 输出数据验证最终检测结果
- 

### 2.4.1 仿真过程

为了更好的定义仿真过程中出现的问题类型，以及是否需要重建点云地图等，将仿真过程输出的故障（错误）等级分为 3 级：0. 正常；1. 部分不正常；2. 致命错误。详细的故障信息会在 report (pdf 文档) 中给出，结合故障等级和详细的故障信息快速定位问题类型和原因，为有效解决故障或问题提供了重要依据。

仿真过程主要包含：前期数据准备、定位初始化、激光匹配定位、MSF 融合<sup>6</sup>以及自适应激光匹配阈值生成。

1. **前期数据准备。** 获取需要仿真数据包相关信息和地图加载模式，若为非瓦片加载模式，此阶段需要加载整张地图；若为瓦片模式，此阶段只需要获取所有瓦片地图相关信息，无需加载。
2. **定位初始化。** 目前初始化方式有三种：GPS、功能点以及广域匹配；首先进行 10 次 GPS 初始化，未成功，在进行 20 次功能点初始化，依然未成功，最后进行 80 次广域匹配初始化，若依然未成功，则需记录并给出具体原因，便于后期问题分析。
3. **激光匹配定位。** 目前激光匹配算法为 NDT，若定位过程中出现定位丢失，需重定位，若成功，记录并继续激光匹配；若失败，记录失败原因并结束此数据包的激光匹配过程。
4. **MSF 融合。** 完后单个数据包激光匹配后，需要将激光定位数据与 GPS、IMU 以及轮速计进行融合定位，由于最终仿真的位置输出，同时，可以用于检测地图是否存在较大畸变等问题。
5. **自适应激光阈值生成** 完成所有数据包定位仿真后，通过基于时间序列的分割聚类方法优化激光匹配阈值，获取本场景各处最佳的定位阈值。

至此，整个仿真过程结束，其中图 8 为瓦片地图加载例子，现阶段瓦片地图形状是边长为 50m 的正方形，每次加载车辆周围 25 块，当车辆运行至新瓦片区域时，加载的地图需要删除远距离瓦片地图和新载入近距离瓦片地图。如仿真过程中，，车辆出现定位丢失

<sup>6</sup>激光、GPS、轮速及以及 IMU 传感器数据进行的多传感器定位融合。

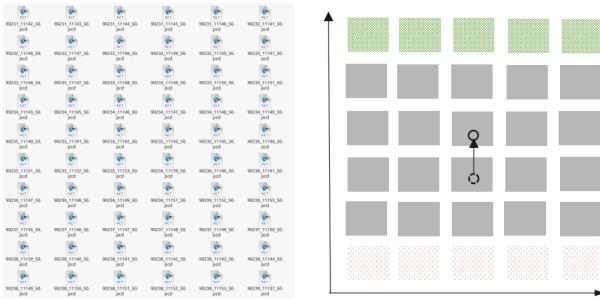


图 5：瓦片地图加载例子：车辆移动过程中，绿色为新增瓦片地图，白色为剔除以前瓦片地图，灰色为保持不变地图

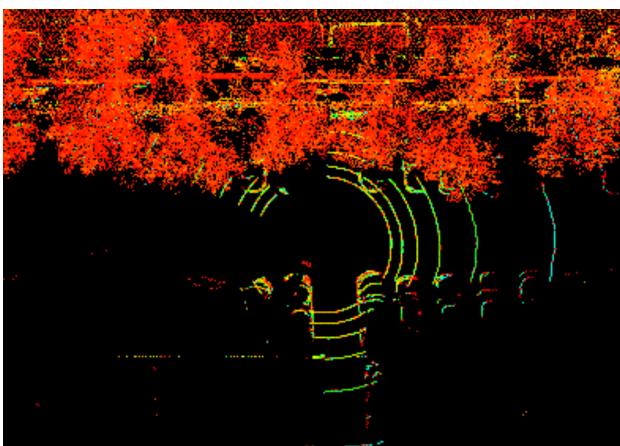


图 6：某场地初始化定位失败事例：黄色为车辆实际点云数据，红色为场地图点云数据

或者初始化定位失败，均会在报告中生成问题指定区域范围内的点云数据，如图 6 所示，通过最后生成的报告可以看出采集时，贴边数据在点云地图外。

#### 2.4.2 准出过程

为了对点云地图各项准出指标的自动检测，以及是否需要重建点云地图等，将准出过程输出的故障（错误）等级分为 3 级：0. 正常；1. 部分不正常；2. 致命错误，其效果和仿真过程相同。

准出过程主要包含：激光匹配（NDT）轨迹自身的跳变检测、GPS 与融合轨迹偏差以及存储数据检测。

1. **激光匹配（NDT）轨迹自身的跳变检测。** 激光匹配轨迹跳变分为两大类：a. 匹配曲线的不平滑；b. 轨迹点出现断裂；针对 a 类：使用上一帧定位

结果和 DR 递推当前帧位置，计算当前帧和递推位置之间的欧式距离，不大于 30cm 则认为定位轨迹平滑，超出则认为定位轨迹不平滑，此时输出检测不合格，并输出出现不平滑轨迹编号、坐标以及跳点偏差，并记录这些点的位置坐标，将轨迹状态改成 1；针对 b 类：直接计算前后两帧之间的欧式距离，若超过 70cm（暂定），则认为该处轨迹存在断裂问题，将轨迹状态改成 2；并统计不平滑的点个数以及断裂次数。

由于上述现象连续出现的长度很短（以轨迹而言），因此在显示时，可以将该点的位置进行相应的放大（取前后 20 个点），来直观显示此类问题。

2. **GPS 与融合轨迹偏差。** 首先根据 GPS 数据状态，将 GPS 数据进行分类（0-5），由于 GPS 本身存在误差，为了降低其噪声干扰，在比较时，只使用类型 4 的点，统计其角度和位置偏差的平均值和指定范围内的分布情况，根据分布情况和平均值判断该地图是否满足准出标准。

#### 3. 存储数据检测。

- 数据是否为空检测，除了轨迹之外，其他相关数据缺失时只提示一个警告，而轨迹为空时，为致命错误，需要强制退出；
- 功能点数据存储过程检测。基于需求和版本问题，功能点数据为可选的，但后期该数据为必须项；
- 图像数据存储过程检测，图像数据目前是可选项，主要方便 HAMO<sup>7</sup>软件制作贴边数据，但是在后期的点云地图构建中，除 5G 车辆外，图像数据是必需项。
- 广域初始化原始数据存储过程检测，该数据用于车辆广域初始化，为了提高车辆初始化定位成功率，此数据为必须项。
- 轨迹数据检测。若建图数据包轨迹为空时，检出失败并退出，若贴边数据包轨迹为空时，记录并提示一个警告，无需退出。

在整个仿真和自动准出结束后，会在报告中生成去全

<sup>7</sup>HAMO：制作高精度语义地图的自研软件工具。

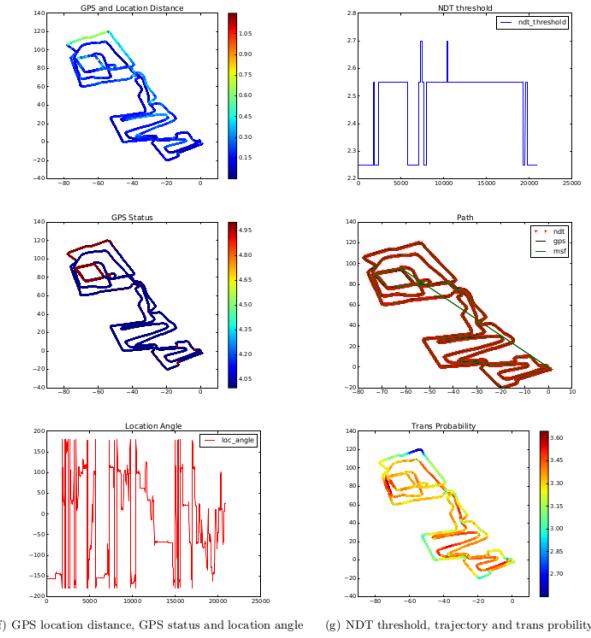


图 7: 数据验证输出的相关曲线图

场景 GPS 状态、轨迹偏差、航向、匹配阈值以及自适应激光匹配阈值，如图 7，便于直观查看整个场景定位状态。

## 2.5 广场情况匹配处理

### 2.5.1 广场情况匹配问题

在广场这种无明显结构特征的场景下，激光点云匹配基本无效，匹配出的轨迹存在严重的跳变情况，而在“局部更相信 Lidar Odom”的后端优化策略中，这种匹配跳变将会使得最终优化后的位姿也是严重跳变的，从而使得点云地图重影较大。

图 8 是一个 Lidar 匹配失效的例子。在这个图中，黄色线代表的 Lidar 轨迹尽管给出了结果，但其局部形状与现实并不匹配。该图中，车辆实际走的是直线，但 Lidar 匹配结果有明显的侧向抖动。这种匹配失效的原因是由于大广场场景激光点云特征稀少，导致匹配效果差。如果我们不识别这种情况，后端优化将继续相信激光的局部形状，导致拼图不正确。一般称这种情况为“**匹配退化**”。

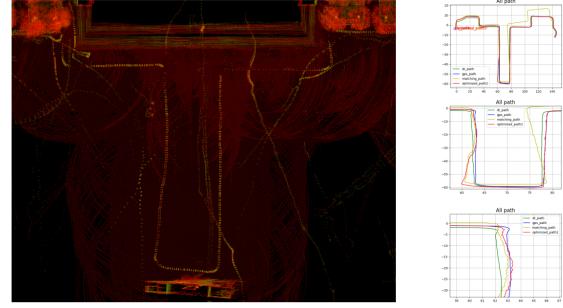


图 8: 广场情况匹配跳变的实际例子：优化后的轨迹受匹配的轨迹影响，严重跳边，地图重影

### 2.5.2 系统退化识别

为了解决广场区域激光匹配无效带来的优化后轨迹仍然跳变的问题，我们的方法是当广场区域出现时，Lidar Odom 能够准确识别出广场并将识别的结果传递给后端，后端位姿优化在面对广场区域时采取“局部更相信 DR”的策略。

一般情况下，基于激光点云匹配的状态估计问题在无明显结构特征的场景下（平面场景等）在某个方向上是退化的 [5]，退化使得求解不够稳定，从而导致求解的位姿是跳变的。我们可以借助匹配时的状态估计问题是否退化来识别场景中的广场区域。

我们知道，基于特征匹配（LOAM）的状态估计问题同样也是一个最小二乘问题，采用高斯牛顿法去求解这个问题的本质就是求解增量方程：

$$\mathbf{H}\Delta\mathbf{x} = \mathbf{g} \quad (7)$$

其中  $\mathbf{H} = \mathbf{J}^T \mathbf{J}$ ， $\mathbf{J}$  为雅可比矩阵。为了求解增量方程，通常需要  $\mathbf{H}$  是可逆的，也就是满秩的，假设  $\mathbf{H}$  是  $n$  维的，则  $\text{rank}(\mathbf{H}) = n$ ，此时：

$$\Delta\mathbf{x} = \mathbf{H}^{-1}\mathbf{g} \quad (8)$$

然而在一些情况下， $\mathbf{J}^T \mathbf{J}$  是半正定的， $\mathbf{H}$  不满秩，那么增量方程便会存在一系列通解：

$$\Delta\mathbf{x} = \underbrace{\mathbf{H}^{-1}\mathbf{g}}_{\text{特解}} + \underbrace{k_1\Delta\mathbf{x}_1 + k_2\Delta\mathbf{x}_2 + \dots + k_r\Delta\mathbf{x}_r}_{\text{通解}}. \quad (9)$$

公式(9)中的 $\Delta\mathbf{x}_1, \Delta\mathbf{x}_2 \dots \Delta\mathbf{x}_r$ 属于增量方程的零空间, 此时 $\text{rank}(\mathbf{H}) = n - r$ , 在这种情况下,  $\mathbf{H}$ 为奇异矩阵, 增量的稳定性较差, 导致算法不收敛, 我们认为此时系统是退化的。

通过判断 $\mathbf{H}$ 是否满秩即可判断系统是否退化, 在建图的后端优化里便可针对性的处理退化场景。然而实际应用中, 在一些容易使得系统退化的场景中(如广场区域), 由于各种精度误差的存在,  $\mathbf{H}$ 往往都是满秩的, 需要通过对 $\mathbf{H}$ 进行特征值分解或者对 $\mathbf{J}$ 进行奇异值分解的手段来判断系统是否退化。对 $\mathbf{H}$ 进行特征值分解如下所示:

$$\mathbf{H} = \mathbf{V}\Lambda\mathbf{V}^T \quad (10)$$

$$\Lambda = [\text{diag}(\lambda_1, \lambda_2 \dots \lambda_n)], \quad \lambda_1 \geq \lambda_2 \geq \dots \lambda_n$$

当 $\mathbf{H}$ 最小特征值 $\lambda_n$ 与最大特征值 $\lambda_1$ 的比值非常小(实际取 $10^{-4}$ 量级)的时候, 我们认为 $\mathbf{H}$ 接近奇异矩阵, 此时即使数值上是满秩的 $\mathbf{H}$ 同样会使得整个系统是病态的, 增量的稳定性较差。因此对 $\mathbf{H}$ 特征值分解, 通过判断最小特征值是否非常小判断系统是否退化的有效手段, 对 $\mathbf{J}$ 进行奇异值分解同理。

判断为退化的激光匹配位姿, 将会被赋予非常大的噪声, 后端优化便更相信DR递推结果, 使得广场区域的轨迹整体较为平滑, 有效地减少地图重影。测试结果见实验章节。

## 2.6 多 bag 包拼接和连通性检测

不能大范围的闭环是2.0难以构建大规模地图的一个很重要原因。**多 bag 包拼接和连通性检测**主要实现bag内和bag之间的大范围闭环检测, 给建图提供全局的闭环约束, 极大提高建图能力与建图精度。如图9, 左边为2.0建图效果, 由于蓝色框内累计偏差将近2m, 未闭环成功; 右边通过多分辨率匹配检测到闭环, 添加了闭环约束, 建图累计误差得到纠正。

多 bag 包拼接和连通性检测的算法流程如3所示:

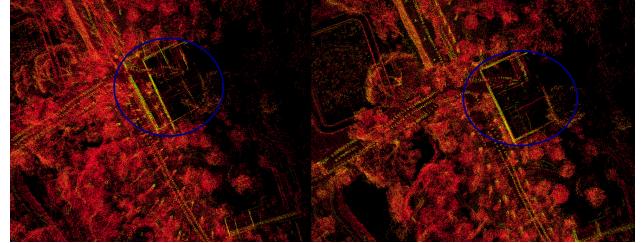


图9: 亭林建图效果对比, 左侧: 2.0 建图, 右侧: 3.0 建图

---

### Algorithm 3 连通性检测流程

---

- 1: 计算轨迹间与轨迹内的备选闭环;
  - 2: 为了节约计算资源, 从备选闭环中选择固定数量的闭环约束, 这部分备选闭环才会真正参与匹配计算;
  - 3: 针对一个备选闭环, 生成匹配需要的多分辨率地图;
  - 4: 基于多分辨率地图, 按照广度优先的分支限界搜索策略进行匹配, 并校验计算结果;
  - 5: 将上一步筛选出的匹配闭环通过NDT匹配进行细粒度匹配, 如果匹配成功则加入闭环约束集合;
  - 6: 回到第三步, 直到所有备选闭环计算完毕。
-

### 2.6.1 备选闭环计算

潜在备选闭环计算的实现原理很简单，关键在于算法实现中数据结构的复杂性。备选闭环的计算基于关键帧的初始值，计算每个关键帧与其他关键帧的距离，一定范围内的都作为此关键帧的备选闭环关键帧。

以  $k_{im}$  表示轨迹  $i$  中第  $m$  个关键帧， $C_{ij}$  表示轨迹  $i$  和轨迹  $j$  中所有备选闭环的集合。 $C_{ij}$  中元素为  $c_{mn}$ ，表示轨迹  $i$  中第  $m$  个关键帧和轨迹  $j$  中第  $n$  个关键帧存在备选闭环。如果两个关键帧的  $XY$  平面距离小于阈值距离，即

$$D(k_{im}, k_{jn}) < D_t \quad (11)$$

那么  $c_{mn} \in C_{ij}$ 。如果一共存在  $N$  条轨迹，则两两轨迹的闭环集合  $C_{ij}$  组成了所有备选闭环集合  $C_N$ 。

两条轨迹的备选闭环集合  $C_{ij}$  中，如果  $i = j$ ， $C_{ij}$  表示的是轨迹内的备选闭环集合，否则表示的是轨迹之间的备选闭环集合。备选闭环计算模块最后提供的就是  $C_{ij}$  组成的集合  $C_N$ 。例如，给定 0, 1 两条轨迹，那么  $C_2 = \{C_{00}, C_{01}, C_{11}\}$ 。

### 2.6.2 多分辨率地图

多分辨率地图是离散化的栅格地图，每个栅格内存储两组高斯参数，一组参数表征地面附近点云的高斯分布，一组参数表征地面以上点云的高斯分布。具体的多分辨率地图数据形式可表示如下：

$\text{MAP}_N$  表示具有  $N$  层的多分辨率地图，第  $n$  层地图  $\text{map}_n \in \text{MAP}_N$ ， $\text{map}_n$  地图中的某栅格  $c_{ij}^n$  由四个参数描述：

$$c_{ij}^n = \{\mu_0, \sigma_0, \mu_1, \sigma_1\} \quad (12)$$

其中  $\mu_0, \sigma_0$  为地面附近点云高斯分布的参数， $\mu_1, \sigma_1$  为地面以上点云高斯分布的参数。以  $r_n$  表示第  $n$  层地图的分辨率，这里取  $r_{n+1}/r_n = 2$ 。

假设有点云  $P$ ，那么点云在  $n$  层地图对应的概率

参数为：

$$\begin{aligned} p_n &= \prod_{p_i \in P} N_j^n(p_i) \\ &= \prod_{p_i \in P} \frac{1}{\sqrt{2\pi}\sigma_j^n(p_i)} \exp\left(-\frac{(z_i - \mu_j^n(p_i))^2}{2(\sigma_j^n(p_i))^2}\right) \end{aligned} \quad (13)$$

其中  $\sigma_j^n(p_i)$ ， $\mu_j^n(p_i)$  为点  $p_i$  在第  $n$  层地图对应栅格的方差与均值， $j$  取 0 或者 1。为了加速计算对上式两边取对数可得：

$$\begin{aligned} \ln(p_n) &= \sum_{p_i \in P} \ln(N_j^n(p_i)) \\ &\sim \sum_{p_i \in P} \left( -\ln \sigma_j^n(p_i) - \frac{(z_i - \mu_j^n(p_i))^2}{(\sigma_j^n(p_i))^2} \right) \end{aligned} \quad (14)$$

进行此处理的好处是：

- 极大减少了计算量；
- 将小数的乘法变为加法，可防止数据超出 float64 的表达能力。

### 2.6.3 匹配策略

基于多分辨率地图的匹配策略与基于广度优先的分支限界算法相似。首先在最低分辨率地图上按照搜索范围遍历搜索最优解，然后将最优解作为下一层地图的初始值进行搜索，以此循环直到最高分辨率地图；在最高分辨率地图上得到最优值以后，以此作为最后 NDT 匹配的初始值，匹配得出最后的结果。算法实现过程中有几个关键点需要注意：

- 基于多分辨率地图的搜索只关注  $(x, y, yaw)$  三个维度，其他三个维度可使用待匹配点云的初始值或者匹配地图的位姿。
- 地图的最低分辨率对应的栅格不能太大，太大的栅格可能导致点云失真容易误匹配，一般栅格边长不超过 2m。

### 2.7 点云去重影拼接

地图构建中，地图重影是指点云建图中出现的墙面变厚甚至双墙现象。地图重影的几项原因：

1. 激光里程计累计偏差导致轨迹交叉处出现重影；

2. 后端优化导致的关键帧相对位置偏差；
3. 非交叉的重复轨迹导致墙面远近共视。

其中，通过优化闭环检测可减少交叉轨迹处的重影偏差，通过优化后端策略可获得更加准确的关键帧位置，上述两种方案均在上述章节中有所改善。为处理重复轨迹导致的远近共视问题导致的重影问题，通过优化后点云的最终拼接策略进行改善，正是本节所述优化。

mapping2.0 周期内，优化完成后，地图输出采用点云累加拼接方案，直接根据各关键帧优化位姿累加所有对应点云。当采图轨迹不可避免重复时，重复的轨迹可能会导致墙面变厚，地图重影增大，导致地图质量不佳，定位偏差加大。本轮优化中，针对重复轨迹进行检测，进行重复轨迹对应点云的去重叠处理及配准微调，实现重复轨迹导致的重影剔除。

### 2.7.1 拼接优化算法流程

点云拼接优化算法主要包含三部分：重复轨迹检测，重叠点的剔除，重复点云位姿微调。算法 4 描述了完整的拼接优化算法设计流程。

---

#### Algorithm 4 拼接优化算法流程

---

- 1: 输入：优化后关键帧及对应点云
  - 2: 提取关键帧，构建建图轨迹；
  - 3: 建图轨迹分类，提取重复轨迹及参考轨迹；
  - 4: 根据建图轨迹预先构建全地图空间栅格，参考轨迹对应点云进行投影；
  - 5: 重复估计对应点云进行栅格投影，判断栅格状态，提取非重叠点；
  - 6: 提取参考轨迹点云作为配准参考；
  - 7: 重叠轨迹关键帧进行配准，校验输入状态及配准结果；
  - 8: 更新各帧点云变换矩阵及对应点云；
  - 9: 输出：更新后的关键帧及点云向量
- 

### 2.7.2 重复轨迹检测

重复轨迹检测时采用空间距离作为检测标准，具体实现采用栅格进行轨迹筛选，规则如下：

1. 将整个建图轨迹进行栅格划分，栅格默认大小为 5；
2. 若两帧非相邻关键帧位于同一个栅格中，则认为两帧相交并重复；
3. 当 5\*5 相邻栅格中存在两条或以上非相邻轨迹段时，认为时序靠后的轨迹段为重复路段；
4. 考虑到转弯轨迹对于整体建图影响，提取出曲率较大的轨迹段认为重复轨迹；
5. 建图轨迹的前段 10 米不允许设置为重复轨迹，其余轨迹按上述分类并对距离较近的重复轨迹进行合并

### 2.7.3 重叠点剔除及重复点云微调

基于检测出的重复轨迹，通过重叠点剔除及重复点云微调，实现点云拼接去重影。重叠点的剔除使用三维栅格，栅格大小设置为 1 米。首先将参考轨迹的点全部投影到对应索引的栅格中，再将重叠轨迹对应关键帧各点进行投影。某重叠轨迹对应点投影时，若对应索引中已存在点云点，则认为该点已重复，剔除该点；若对应索引中不存在点，则认为该处需由重叠轨迹填充，保留该点。由于采用三维栅格进行索引，整体计算效率较高。同时 1 米的栅格间距不会对定位造成较大影响。

重叠点的剔除优化目标在于减少重叠轨迹导致的墙面变厚问题，对于部分场景效果并不明显，故增加了点云微调步骤。点云微调实际参考了定位思路，以参考轨迹对应点云作为地图，重复轨迹作为配准目标，采用 NDT 算法进行重复点云的姿态微调。

## 3 实验

在实验章节，我们一共描述两大部分：整体实验和专题实验。第一部分为现有建图数据集上的建图实验，我们在整个数据集上运行 Mapping 3.0 建图算法，

并给出统计结果，这些统计结果将对今后的开发布署有一定的理论指导作用。第二部分中，我们给出一些针对每个改进点的专题实验，主要说明算法改进之后的建图效果对比。

### 3.1 整体实验

#### 3.1.1 整体实验统计指标

在整体实验中，我们收集了迄今为止的建图数据，按数据包大小分为 minimal、standard、large 三类，各数据集含有的地图数量见表 1。整个测试在 4 个台式机上进行，主要使用 6 代至 9 代 i7，内存为 16G-32G 不等。在 Minimal 数据集上，我们并行运行 6 条流水线；在 Standard 和 Large 上则运行一个。对于每次运行，我们统计了其总共运行时间、数据包大小、地图面积、仿真结果、建图结果五项主要结果<sup>8</sup>。由于整个数据集较大，完整数据结果请参见附录表 2 和表 3，这里我们给出统计数据分析和一些代表性案例。

表 1：数据集说明

数据集名称	数据包个数	数据包大小范围
Minimal	41	0-2 GB
Standard	25	2-4 GB
Large	16	≥ 4GB
总计	82	

首先，我们定义一些结果指标。在目前的测试中，仿真可能给出 PASS、CHECK、FAIL 三种结论。

1. 对于仿真给 PASS 的，地图将直接通过。
2. 对于仿真给 FAIL 的，地图将直接失败。
3. 对于仿真给 CHECK 的，由建图员确认地图是否可以通过。如果建图员确认通过的，地图仍记为 SUCCESS，此时我们不计人工介入。如果建图员认为不通过，但是可以通过调整建图参数，重建后可以通过的，也记为建图成功，但计一次人工介入。如果调试之后仍认为地图不可通过，则记为建图失败，将给出失败原因分析和改进建

<sup>8</sup>需要注意的是，数据包大小指整个压缩包大小。地图面积由建图包里程推算，和实际场地面积可能有所出入。

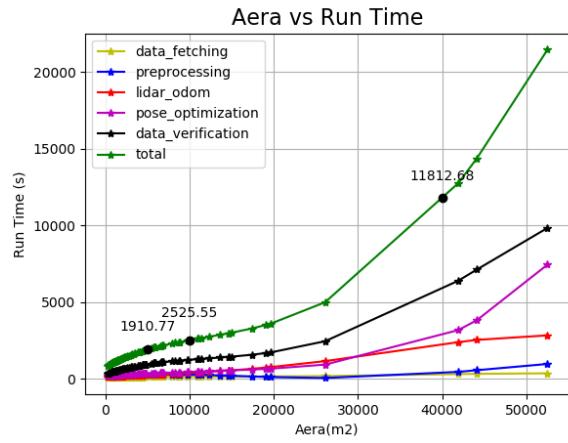


图 10：面积大小和运行时间关系。横轴：面积大小 ( $m^2$ )，纵轴：运行时间。黑色关键点为 5000 平米、10000 平米、40000 平米时的建图总时间。

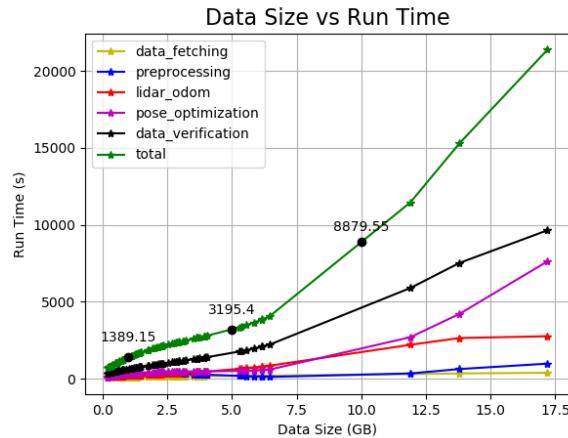


图 11：数据大小和运行时间关系。横轴：地图数据包大小 (GB)，纵轴：建图时间。黑色关键点为：1GB，5GB，10GB 的建图总时间。

议。

根据这个逻辑，我们定义结果指标为：

$$\begin{aligned}
 S_{\text{PASS}} &= \frac{\text{仿真 PASS, 建图成功}}{\text{有效地图数}} \\
 S_{\text{CHECK}} &= \frac{\text{仿真 CHECK, 建图成功}}{\text{有效地图数}} \\
 S_{\text{AUTO}} &= \frac{\text{没有人工介入, 建图成功}}{\text{有效地图数}}
 \end{aligned} \tag{15}$$

并称为通过率，CHECK 率，自动建图成功率，其中

**有效地图数**指排除数据包不完整、格式不正确地图之后的数量。对于本次测试，这三个指标为：

$$\begin{aligned} S_{\text{PASS}} &= 72.5\% \\ S_{\text{CHECK}} &= 23.75\% \quad (16) \\ S_{\text{AUTO}} &= 93.75\% \end{aligned}$$

前两者之和称为**建图成功率**，指仿真给 PASS 或 CHECK 时建图的成功比例：

$$S_{\text{SUCCESS}} = S_{\text{PASS}} + S_{\text{CHECK}} = 96.25\% \quad (17)$$

这些指标综合反映了建图的成功率和自动化建图能力，较好地体现了目前 3.0 的建图能力。

接下来，我们对 3.0 的建图效率进行分析。我们统计了整个测试集建图面积大小与运行时间之关系，见图 10，以及数据包大小与运行时间之关系，见图 11。该图反映出，建图时间基本随面积和数据包大小呈线性增长关系。从这两个图中我们可以统计 3.0 的建图速度：

$$v_{\text{area}} = 4.2 \sim 4.9 \text{分钟}/\text{千平米} \quad (18)$$

或：

$$v_{\text{data}} = 10.65 \sim 14.78 \text{分钟}/\text{GB} \quad (19)$$

以此可以估计每个建图任务的预计时间。例如，对于两万平米的地图，应该需要 42-49 分钟。或者，对于 5GB 的数据包，应需要 50-70 分钟左右的时间<sup>9</sup>。以上两图亦反映了算法各阶段耗时的关系。可以看出，数据验证环节占据了主要的计算时间，约为前端的两倍，而预处理环节和数据下载环节则耗时相对固定。

下面我们举一些较有代表性的建图案例，既有成功的也有失败的。

### 3.1.2 典型案例分析

内蒙古大学 2 是本次测试中所建的最大地图，其建图面积达到 75000 平米，共 24 个有效数据包，占据 15.4GB 空间，建图时间高达 10 个小时。该地图建

<sup>9</sup>注意，由于网速、机器配置等硬件设备的差别，以上速度仅是估计速度。数据包和建图时间的关系变化较大，取决于具体建图轨迹和长度和仿真轨迹的长度。

## Report of Mapping in neimenggu daxue2 idriverplus.com

Mapping 3.0 version 0.1, Builder: Peng GuoQi

August 30, 2019

### 1 Brief Report

Table 1: General Status

Map Name	neimenggu daxue2				
Time Usage	605m44s	Valid Bags	24	Total Size	15.4GB
Length	7.5km	Area	7.5e+04m <sup>2</sup>	Success	No
Checkout	CHECK	Reason	smoothness, large deviation.		
Map DB Link	<a href="http://127.0.0.1:8000/neimenggu_daxue2/sap.db">http://127.0.0.1:8000/neimenggu_daxue2/sap.db</a>				
Point Cloud Link	<a href="http://127.0.0.1:8000/neimenggu_daxue2/result.pcd">http://127.0.0.1:8000/neimenggu_daxue2/result.pcd</a>				

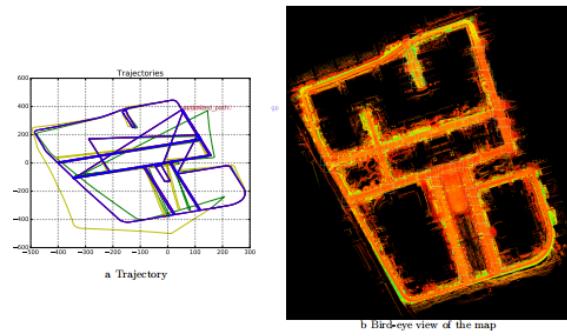


图 12：内蒙古大学建图报告首页

图报告首页见 12，建图结果中仿真给出了 CHECK，通过人工检查后认为建图可以接受，因此属于一次性建图成功的案例。该地图的成功建图，一方面标志着 3.0 建图算法在原有基础上有明显提升，一方面也显示了目前能够接受的最大建图规模，具有一定的代表性。

该报告详细内容反映，GPS 信号在该地图大部分区域有效（GPS 有效值占比 97.8%），因此建图难度主要来源于数据量太大，而建图本身并不困难。有效建图包共 8 个，拼接处较多，回环算法共给出了 856 处回环点，被后端接受的共有 697 处。仿真给 CHECK 的原因主要为平滑性限制，在 24 万个轨迹点中检查出了 22 处不平滑点，人工检查后认为可以通过。

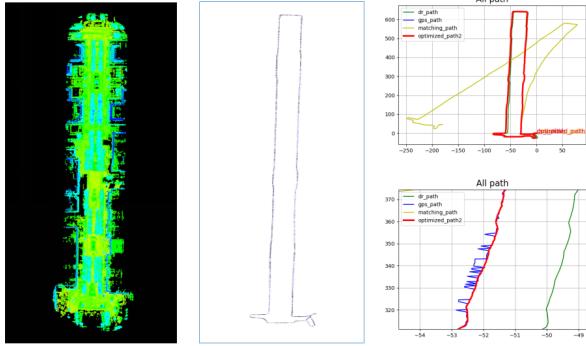


图 13: 西安大唐不夜城全局点云地图、PoseGraph 示意图和轨迹图

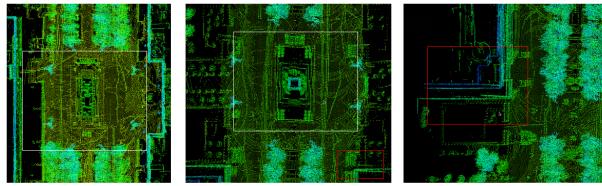


图 14: 西安大唐不夜城地图局部地图

### 3.1.3 失败案例分析

当场景属于比较困难场景时，比如场景的采集路线往复、长时间无合理闭环、共视同一物体且 GPS 较差等情况同时出现，3.0 会存在建图失败的情况，下面我们举两个代表的建图案例，并说明建图失败的原因以及后续改善方向。

**西安大唐不夜城** 场景特点如图 13 所示，右侧上图中优化后轨迹（红色）显示场景路线往复、间隔大于 25 米且单向长度大于 600 米，往复路线上激光点云会共视到物理世界中的相同物体，同时右侧下图显示场景存在 GPS 较差的路段。此种场景由于单方向 lidar-odom 的长时间累计误差导致单方向上地图发生畸变，同时较宽的往复路线无法正常闭环，部分区域 GPS 较差也无法提供较好的全局约束，最终导致地图整体“毛糙”，局部地图存在明显重影。图 13 中间的 Pose Graph 示意图说明往复的路线间没有检测到闭环。图 14 显示最终的点云地图的局部区域重影明显（红色矩形框内）、同一物体点云地图毛糙（白色矩形

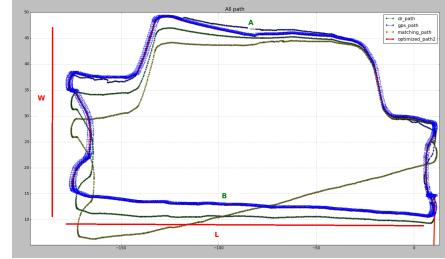


图 15: 西安大雁塔轨迹图

框内）。

**西安大雁塔** 西安大雁塔场景中一条轨迹如图 15 所示，轨迹呈现扁平的矩形，矩形宽度 W 大约 35 米。对激光而言，距离 20 到 40 米是一个比较特殊的位置，两个位置相距此等距离时，激光雷达有大范围共同视野，但匹配算法却不能很好的匹配。

此轨迹正是属于这种情况，这种情况容易导致建图中大距离重影问题，如图 16 所示。左边地图为第一次优化后地图，右边为闭环约束添加后地图，左边地图红色和蓝色框内存在明显重影，右边地图蓝色框内重影消失，而红色框内重影依然存在。这是因为这条轨迹的起始点和终点在图中三角形位置，在此处形成闭环，所以附近的重影被纠正，正如图中蓝色框围绕闭环处，而红色框离闭环处较远。红色框内的重影是由于 A,B 两处的激光雷达同时照射形成，但 AB 之间的轨迹路程又很远，导致很大的累计误差，后端优化时由于 A 处 GPS 不可用，所以无法纠正 A 处的误差。

针对上述的类似场景，可通过以下几种手段去解决。

- 更加合理的采集路线，尽量避免长时间无闭环的情况，比如在图 15 中，在 AB 之间将车辆行驶轨迹闭合。
- 增强算法能力，比如提高 20 ~ 40 米范围闭环能力，比如提高 lidar\_odom 的精度。

### 3.2 专题实验

在专题实验部分，我们描述针对几个专题改进项的相关对比实验。这些实验没有规模化测试，主要是

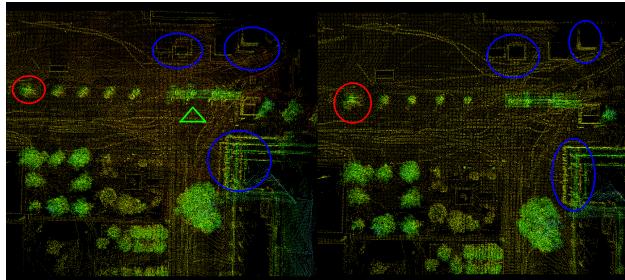


图 16: 西安大雁塔点云图对比

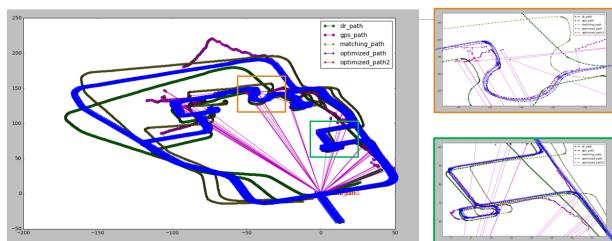


图 17: 万锦领秀小区的轨迹图。左侧: 全局轨迹; 右侧: 局部轨迹

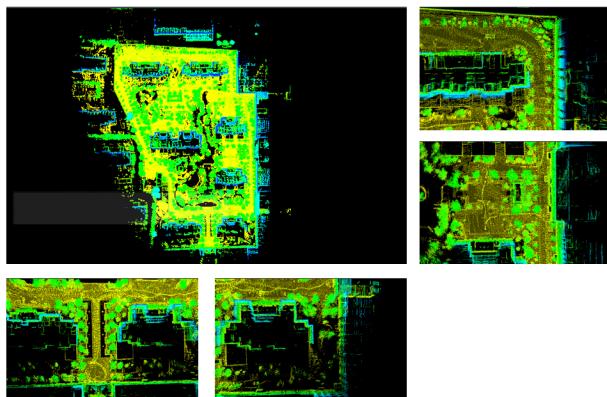


图 18: 万锦领秀小区的点云地图

若干个代表性例子。

### 3.2.1 GPS 跳变检测

GPS 跳变检测已广泛在各种地图中使用，我们举一些有难度和代表性的例子。以内蒙古万锦领秀图为例。该图的轨迹图与点云图见图 17 和图 18。轨迹图中，需要关注的是紫色点的 GPS 轨迹和蓝色的优化轨迹。该图是一个典型的弱 GPS 场景，该小区只有南侧入口处有较稳定的 GPS 信号，其余情况下 GPS 经常不可用，其轨迹图上的紫色线亦反映了该事实。从局部图中可以看出，紫色 GPS 信号经常出现异常值、跳变的情况，即使在正常值中，也由于多径效应存在一定的误差。该图的绿色轨迹和黄色轨迹分别为 DR 和激光里程计的轨迹。由于其里程计的性质，必定会出现累计误差，所以我们应该关注其局部形状，而全局轨迹意义不大。

同时，该地图由两个 Bag 包拼接而成，分别为外侧一圈和内侧一圈组成。如果拼合不当，容易导致地图在拼接处出现重影。在我们后端优化之后，优化结果能够对抗大多数 GPS 跳变的干扰。例如右下的局部轨迹中，GPS 轨迹在拐弯处发生了明显的跳动，但优化结果仍然保持了平滑的形状，根据优化结果拼接而成的地图也未出现明显的重影。

同时，该地图由两个 Bag 包拼接而成，分别为外侧一圈和内侧一圈组成。如果拼合不当，容易导致地图在拼接处出现重影。在我们后端优化之后，优化结果能够对抗大多数 GPS 跳变的干扰。例如右下的局部轨迹中，GPS 轨迹在拐弯处发生了明显的跳动，但优化结果仍然保持了平滑的形状，根据优化结果拼接而成的地图也未出现明显的重影。

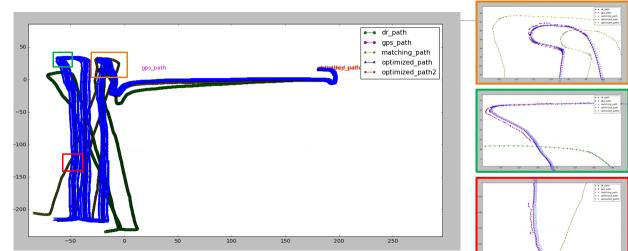


图 19: 华强北步行街轨迹图。左侧: 全局轨迹; 右侧: 局部轨迹

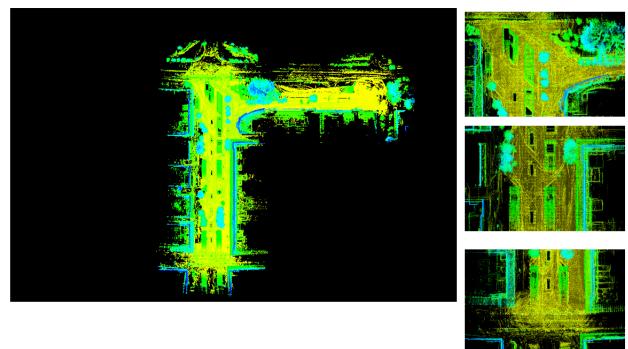


图 20: 华强北步行街地图

另一个例子是深圳华强北地图，见图 19 和图 20。该例中，GPS 信号能够覆盖整个区域，但局部毛刺和跳变较多。如果过于相信 GPS，则拼接的地图会有明显的重影。轨迹图右侧的局部地图显然了几处 GPS

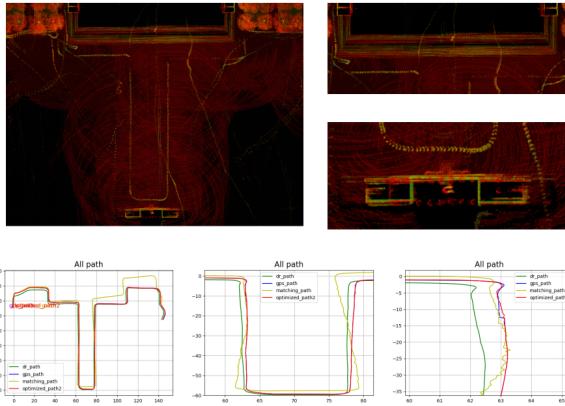


图 21: 市民中心广场区域地图和轨迹曲线。

发生小跳变的场景，这些轨迹都会在优化中被识别为 outlier，从而排除在优化之外。

大体来说，后端优化轨迹的准则是全局与 GPS 对准，局部相信 Lidar 和 DR。这是一条具有普遍意义的优化准则，在大多数 GPS 信号失效时仍可使用。由于 ODD 要求初始点必须有 GPS 信号，这保证了轨迹的绝对位置不会有太大的偏差。另一方面，在局部相信 Lidar 和 DR 的策略保证了局部地图不会有明显的重影，使定位算法在局部匹配时是有效的。

### 3.2.2 广场匹配退化识别

目前广场区域配不稳定的场景数据较少，主要是深圳市民中心广场场景（图 21），其广场面积非常大，导致车辆在广场中心区域时，激光点云基本全部都是广场地面数据，匹配非常不稳定，导致最终的地图重影很大，甚至发生畸变。上图给出了“识别退化后、后端优化更加相信 DR 递推”的点云地图和轨迹曲线，从图的上半部分可以看出广场所台和广场国旗杆台部分“更加规则”，重影变小；图的下半部分可以看出最终优化后的轨迹局部更加相信 DR 递推并且更加光滑，达到了设计的目的。

### 3.2.3 多分辨率地图匹配闭环

一般比较大的场景，因为轨迹较长，闭环处累计了较大误差，这时闭环的作用就会明显体现出来。如

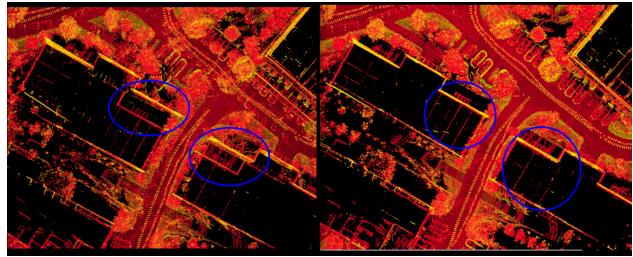


图 22: 恒生建图效果对比

果没有闭环约束，实际物理位置重合或相近处的地图会有明显的距离较远的重影，闭环成功添加约束优化后，重影消失。场景 9, 22 展示了效果的对比，左边为第一次优化后建图效果，蓝色框内位置累计偏差将近 2~3m；右边通过多分辨率匹配检测到闭环，添加了闭环约束，建图累计误差得到纠正。

### 3.2.4 点云去重影拼接

点云去重影拼接主要包含三部分：重复轨迹检测，重叠点剔除及重复点云微调。测试时主要测试重复轨迹检测效果及最终点云拼接效果。

重复轨迹检测效果参考图 23。由图中可以看出，参考轨迹（非重复轨迹）基本覆盖完整地图区域，红色重复轨迹线均是轨迹存在重复区域或曲率较大区域。

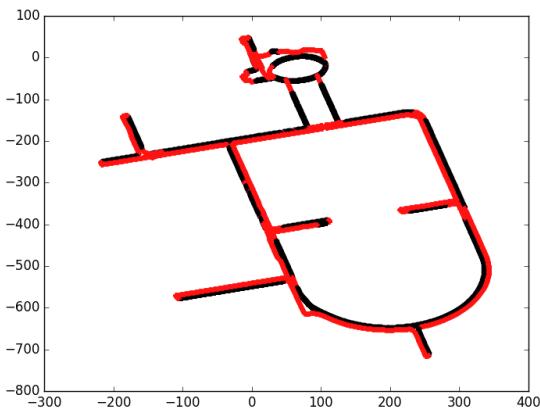


图 23: 内蒙商贸轨迹分类：深蓝色：参考轨迹；红色线：重复轨迹

图 24 为点云拼接完成后效果对比。该场景为海

南诺雅达景区场景，上图为原始部分点云，下图为点云拼接后效果展示，可以看到点云拼接后重影明显改善。

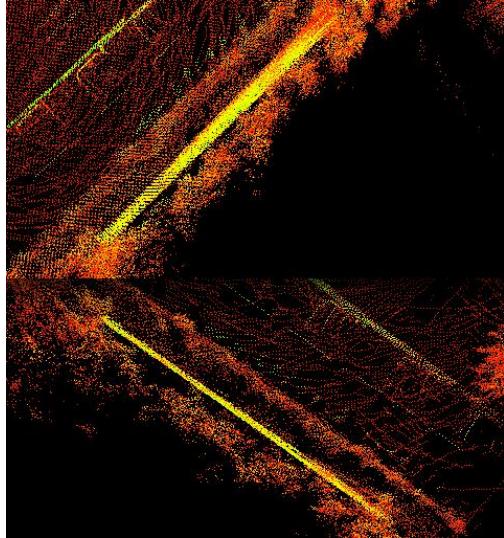


图 24：重叠点云处理效果展示：上图为原始部分点云，下图为点云拼接后效果

## 4 结论

### 4.1 实验结果总结

根据本文描述的实验，我们可以作出以下几点结论：

1. **Mapping 3.0 在算法层面的改进，显著提升了目前建图能力和建图效率。**在建图能力方案，Mapping 3.0 对大部分地图均可一次性建图成功，建图人员只需查看输出报告即可判断地图能否交付。相比于 2.0，3.0 在弱 GPS 场景抗干扰能力更强，无需手动调整 g2o 文件来适配 GPS 跳变点。在闭环和拼接算法上，3.0 也有明显提升，能够准确闭环的地方，3.0 的地图重影很小，几乎无法发现地图拼接痕迹。

在建图速度上，3.0 亦有明显提升。2.0 由于缺少规模化测试，给出的速度指标为 1 小时/6 千平，而 3.0 测试指标为 25-29 分钟/6 千平，相比 2.0 指标提升一倍。

2. 可以认为解决的问题：**GPS 跳变检测、多 Bag 拼接、点云去重影。**GPS 跳变检测已广泛使用于各种建图场景，算法会自动判断 GPS 是否有效，并剔除 GPS 失效的部分。在第一轮优化顺利的情况下，可以认为多 Bag 拼接算法能够给出正确的回环结果，从而拼接各个不同的点云。但是，上述算法有效性成立是有条件的，即 GPS 不能完全无效，多 Bag 拼接和点云去重影依赖于第一轮优化的正确结果。如果客观上前提条件不成立，那么这些算法也会失效。

3. 未完全解决的问题：**自动准人准出。**从测试结果看出，由于建图能力的提升，我们对采集轨迹的要求有所改变。例如：现在我们希望采集轨迹尽量成环，以保证地图一致性；要求 U 型地图尽量少；由于去重影算法的效果，不再回避多次经过同一路段的情况。这些内容应该与采集人员进行沟通，但目前未实施。

准出方面，在仿真给出 CHECK 结果时，目前还没有统一的准则来处理地图是否通过。例如，某些历史数据中，仿真轨迹从地图外出发，导致仿真给出无法初始化的信息；有些车辆的标定参数不对，导致仿真未能正确地给出预测值；有些轨迹不平滑点难以解释，等等。今后应该建立统一的 CHECK 准则，并贯彻实施。

与本文开头部分测试内容中的指标进行对比，可以认为：

1. Mapping 3.0 达到基本性能指标要求：长时间运行无 Core, Minimal 数据集并发数  $\geq 4$ 。
2. 建图成功率  $S_{\text{SUCCESS}} = 96.25\% \geq 95\%$ ，达到要求。
3. 建图速度换算为：12.24-14.28 千平米/小时，达到要求；以数据大小来看，每 GB 耗时 10.65-14.78 分钟，快于目标的 30 分钟。

综上，可以认为 3.0 达到了目前的指标要求。

### 4.2 挑战与展望

我们认为 3.0 在未来还有几个拓展空间。

1. **使用 3.0 建立新地图，收集更多建图问题。**目前

- 我们主要测试了 3.0 在历史数据上的建图情况，但目前现场尚未使用 3.0 建立的地图，其中可能存在一些目前无法预见的问题。我们希望 3.0 能够在试运行期间处理新的建图任务，发布地图，然后解决可能存在的问题。
2. **完善相应的操作手册和文档。** 上述提到的 3.0 使用手册，CHECK 准则、调试方法、采集约束的变更等内容，尚未形成书面的文档以供使用者查询。随着建图工作的推进，我们希望采集人员按照新的规范进行地图采集，而建图人员无需专业训练即可快速掌握 3.0 建图方法和建图失败应对方法。这些文档必须推进落实。
  3. **巨型地图的解决方案。** 目前 3.0 建图能力较多地受制于计算机内存数量，因此，如果地图面积过大，导致采集数据包大于机器内存时，3.0 就无法完成建图任务。尽管服务器的内存可以不受台式机限制，但如何应对特大规模的地图，将是未来的一个挑战。
  4. **现场建图的支持方案。** 考虑到今后建图工作将推向计算性能有限的现场，如何在算力和内存有限的情况下进行建图工作，将是今后重要的发展方向。
- [4] K. De Brabanter, K. Pelckmans, J. De Brabanter, M. Debruyne, J. A. Suykens, M. Hubert, and B. De Moor, “Robustness of kernel based regression: a comparison of iterative weighting schemes,” in *International Conference on Artificial Neural Networks*, pp. 100–110, Springer, 2009.
- [5] J. Zhang, M. Kaess, and S. Singh, “On degeneracy of optimization-based state estimation problems,” in *2016 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 809–816, IEEE, 2016.

## 附录

### A 完整实验结果

完整实验结果见表 2 和 3。

## 参考文献

- [1] Y. Latif, C. Cadena, and J. Neira, “Robust loop closing over time for pose graph slam,” *The International Journal of Robotics Research*, vol. 32, no. 14, pp. 1611–1626, 2013.
- [2] R. Kümmerle, G. Grisetti, H. Strasdat, K. Konolige, and W. Burgard, “g2o: A general framework for graph optimization,” in *2011 IEEE International Conference on Robotics and Automation*, pp. 3607–3613, IEEE, 2011.
- [3] T. D. Barfoot, *State Estimation for Robotics*. Cambridge University Press, 2017.

表2: 完整实验结果

序号	场景	耗时 (s)	大小 (GB)	面积 ( $m^2$ )	仿真结果		建图结果 SUCCESS/FAIL	是否人工介入 YES/NO	备注
					PASS/CHECK/FAIL	SUCCESS			
1	北京亦庄	523.7	0.387	888	CHECK	SUCCESS	NO		
2	北京京环	557.306	0.339	1420	PASS	SUCCESS	NO		
3	北京清华展示	581.134	0.292	818	CHECK	SUCCESS	NO		一个点不平滑
4	北京电信展示	609.694	0.507	832	PASS	SUCCESS	NO		
5	北京海淀双榆树	865.314	0.779	1540	PASS	SUCCESS	NO		
6	北京世园会	1154.419	532	2790	PASS	SUCCESS	NO		
7	重庆大唐移动	308.956	0.249	234	PASS	SUCCESS	NO		
8	北京腾讯众创东	710.285	0.479	806	PASS	SUCCESS	NO		
9	北京牡丹园科技楼	748.235	0.562	1210	CHECK	SUCCESS	NO		贴边轨迹 2 初始点在地图外
10	重庆凯瑞	862.531	0.719	1930	PASS	SUCCESS	NO		
11	北京首钢冰球场	1113.962	0.524	3160	PASS	SUCCESS	NO		
12	广东佛山碧桂园总部	482.517	0.256	621	PASS	SUCCESS	NO		
13	广东佛山碧桂园物业	442.567	0.338	432	PASS	SUCCESS	NO		
14	广东奥科奇 2	358.337	0.233	609	PASS	SUCCESS	NO		
15	广东中山南头	1382.78	1.3	2510	PASS	SUCCESS	NO		
16	广东中山百里山	1754.599	1.27	2290	PASS	SUCCESS	NO		
17	福建东湖小镇 2	1897.586	1.12	3470	PASS	SUCCESS	NO		
18	广东广交会	1842.271	1.35	4600	PASS	SUCCESS	NO		
19	广东黄浦朱雀	486.291	0.336	1170	PASS	SUCCESS	NO		
20	广东广州科学中心 1	2390.862	1.47	4800	PASS	SUCCESS	NO		
21	广东科美清洁	1028.9	0.587	1000	PASS	SUCCESS	NO		
22	贵州移动大厦	188.798	0.173	387	PASS	SUCCESS	NO		
23	广东广州科学中心 2	3415.255	1.79	4780	PASS	SUCCESS	NO		
24	贵州关山湖	1160.875	0.785	2330	PASS	SUCCESS	NO		
25	广东海珠湿地	2835.727	1.21	4060	PASS	SUCCESS	NO		
26	广东广州香雪	2927.482	1.47	3020	PASS	SUCCESS	NO		
27	广东广州科学中心 3	3819.69	2.02	5800	PASS	SUCCESS	NO		
28	香港国际机场巴士	747.77	0.774	1460	CHECK	SUCCESS	NO		轨迹 1 不平滑
29	河南许昌北邮	1186.912	0.766	2640	PASS	SUCCESS	NO		
30	湖南长沙梅溪湖	1028.236	0.655	2070	CHECK	SUCCESS	YES		Lidar 匹配有误, 人工干预优化
31	河北秦皇岛天下第一关	1719.134	0.893	4090	PASS	SUCCESS	NO		
32	河北保定百里峡	2770.785	1.44	5550	PASS	SUCCESS	NO		
33	江苏苏州城关局 2	1751.485	1.11	3480	PASS	SUCCESS	NO		
34	内蒙古满洲里套娃 2	1485.299	0.777	3910	PASS	SUCCESS	NO		
35	内蒙古满洲里套娃 1	2009.945	1.14	2150	PASS	SUCCESS	NO		
36	江苏徐庄高新	2789.384	1.4	4420	PASS	SUCCESS	NO		地图为互不连通的两块
37	山西西安大唐不夜城 1	2133.78	1.3	4770	PASS	SUCCESS	NO		
38	上海绿地	1640.522	1.03	4500	PASS	SUCCESS	NO		
39	山西西安西京学院	2345.442	1.28	4670	PASS	SUCCESS	NO		
40	上海中国移动 1	2371.421	1.92	5770	PASS	SUCCESS	NO		
41	浙江杭州西溪	2386.226	1.74	7920	PASS	SUCCESS	NO		弱 GPS
42	北京腾讯众创新图	2128.832	2.96	8800	PASS	SUCCESS	NO		
43	福建淮安半岛三期 A	2118.854	3.74	9790	PASS	SUCCESS	NO		
44	福建福州万福中心	1849.785	2.31	6570	PASS	SUCCESS	NO		
45	福建东湖小镇 1	1668.903	2.16	5820	PASS	SUCCESS	NO		
46	广东中山剑桥郡	1072.601	2.69	4850	CHECK	SUCCESS	NO		仿真第 1 个包超出地图范围
47	广东南沙蕉门	1974.957	2.54	6000	PASS	SUCCESS	NO		
48	广东深圳华强北	1976.351	3.64	11110	CHECK	SUCCESS	NO		仿真 bag 起始 GPS 差, 初始化失败
49	广东市民中心	1106.115	0.92	12300	CHECK	SUCCESS	NO		部分曲线不光滑
50	广东珠海长隆	2031.963	2.85	6690	PASS	SUCCESS	NO		
51	广西柳州观景滨海	1706.771	3.08	4060	PASS	SUCCESS	NO		
52	海南呀诺	1537.538	2.23	4480	PASS	SUCCESS	NO		
53	江苏南京许昌 2 期	2844.744	3.93	11400	CHECK	SUCCESS	NO		部分曲线不光滑
54	江苏无锡恒生	2698.632	3.66	17400	PASS	SUCCESS	NO		
55	江苏昆山亭林	3435.081	3.91	16200	PASS	SUCCESS	NO		
56	澳门渔人码头	1096.36	2.76	4760	CHECK	SUCCESS	NO		仿真第三个包超出地图范围
57	内蒙古商贸 3	2130.994	3.12	9780	CHECK	SUCCESS	NO		仿真 bag 起始 GPS 差, 初始化失败

表 3: 完整实验结果 (续)

序号	场景	耗时 (s)	大小 (GB)	面积 ( $m^2$ )	仿真结果		建图结果 SUCCESS/FAIL	是否人工介入 YES/NO	备注
					PASS/CHECK/FAIL	SUCCESS/FAIL			
58	内蒙古万锦枫泽园 2	1198.226	2.28	6810	PASS	SUCCESS	NO		
59	山西西安北站	1684.773	2.06	14700	CHECK	FAIL	YES		U型路线严重，重影不易消除
60	山西西安大唐不夜城 2	2342.235	2.87	15000	CHECK	FAIL	YES		U型路线严重，重影不易消除
61	山西西安大雁塔	1632.119	2.94	10300	CHECK	FAIL	YES		U型路线严重，重影不易消除
62	山西西安翻译学院 2	3003.176	4.05	12800	PASS	SUCCESS	NO		
63	上海德邦	784.104	3.22	6900	CHECK	SUCCESS	YES		车型为蜗必达，默认是蜗小白
64	上海提香别墅 2	2847.133	3.5	18300	CHECK	SUCCESS	NO		
65	上海中华艺术宫	1849.442	2.34	11400	CHECK	SUCCESS	NO		
66	北京世界园艺	11615.624	7.7	35450	PASS	SUCCESS	NO		
67	北京柳荫公园	4152.447	6.17	19100	PASS	SUCCESS	NO		
68	福建淮安半岛一期	927.927	9.78	3110	PASS	SUCCESS	NO		存在多个数据包，只建 1 个
69	福建飞凤山	4988.008	6.16	26100	PASS	SUCCESS	NO		
70	福建飞科小镇	2705.715	5.55	8830	PASS	SUCCESS	NO		
71	广东中山燕乐居 1	13648.137	13.8	41900	PASS	SUCCESS	NO		
72	河北雄安市民	2674.011	3.97	10900	PASS	SUCCESS	NO		
73	江苏常州中智云	2258.297	5.85	8330	CHECK	SUCCESS	NO		6 个包仿真失败，建图不完整
74	内蒙古商贸 2	21687.114	17.2	52500	CHECK	SUCCESS	NO		仿真时匹配丢失，推断非地图问题
75	内蒙古大学 1	/	/	/	/	/	NO		拉取数据，解压失败
76	内蒙古大学 2	36343.981	15.4	75000	CHECK	SUCCESS	NO		
77	内蒙古电子	13131.773	11.9	44100	CHECK	SUCCESS	NO		部分曲线不光滑，畸变较大
78	内蒙古万锦枫泽湾	5238.539	5.36	19700	PASS	SUCCESS	NO		
79	内蒙古伊利	12686.379	6.69	29500	CHECK	SUCCESS	NO		1 处轨迹不平滑
80	上海万古科技园	3919.668	5.28	13600	PASS	SUCCESS	NO		
81	内蒙古万锦领秀	3531.174	6.45	14700	PASS	SUCCESS	NO		
82	上海提香别墅 1	/	/	/	/	/	NO		XML 解析失败