激光 SLAM 理论与实践 - 作业 6

peng00bo00

January 9, 2021

1. 使用高斯牛顿法对位姿图进行优化的代码可参见./LSSLAMProject/src/ls_slam/src 路径下 gaussian_newton.cpp 文件。为便于调试和调用，这里添加了一个 roslaunch 文件。在终端中输入 “ roslaunch ls_slam LSSLAM.launch arg_data:=test ”即可导入不同的数据集进行位姿图优化,“arg_data:=test”表示导入测试数据,“arg_data:=intel” 表示导入 intel 数据，“arg_data:=killian” 表示导入 killian 数据。优化后的位姿可参见 Fig.1-Fig.3，其中蓝色为优化前位姿，粉色点为优化后位姿。

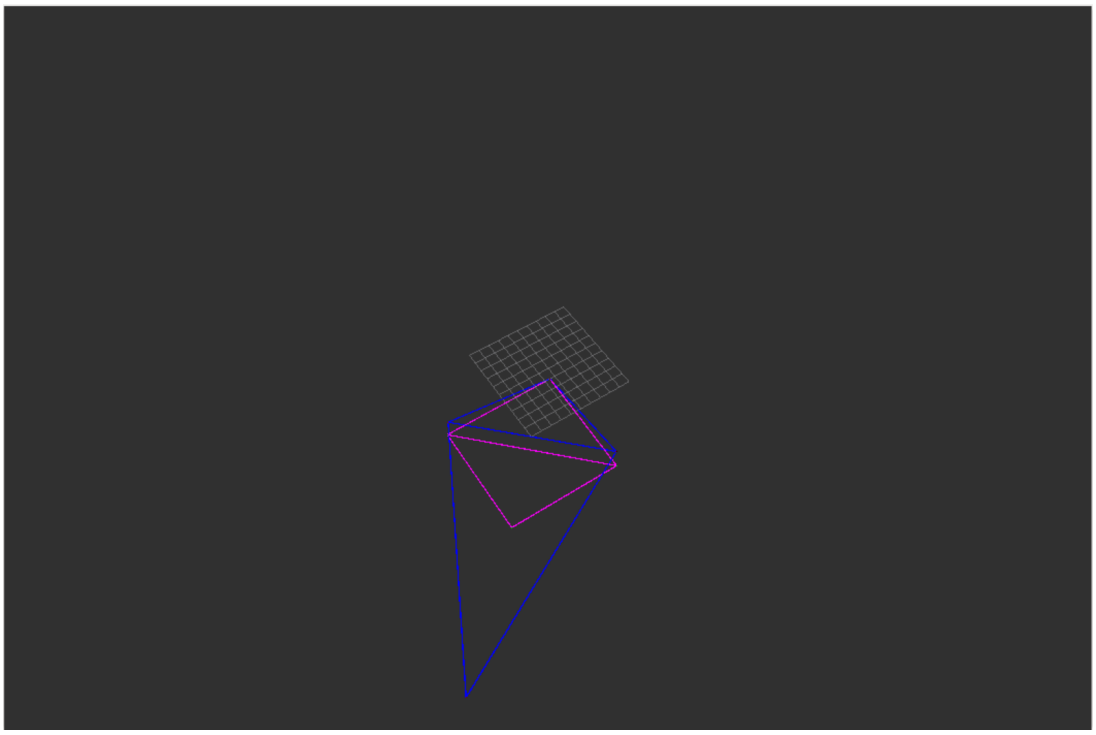

Figure 1: test 数据

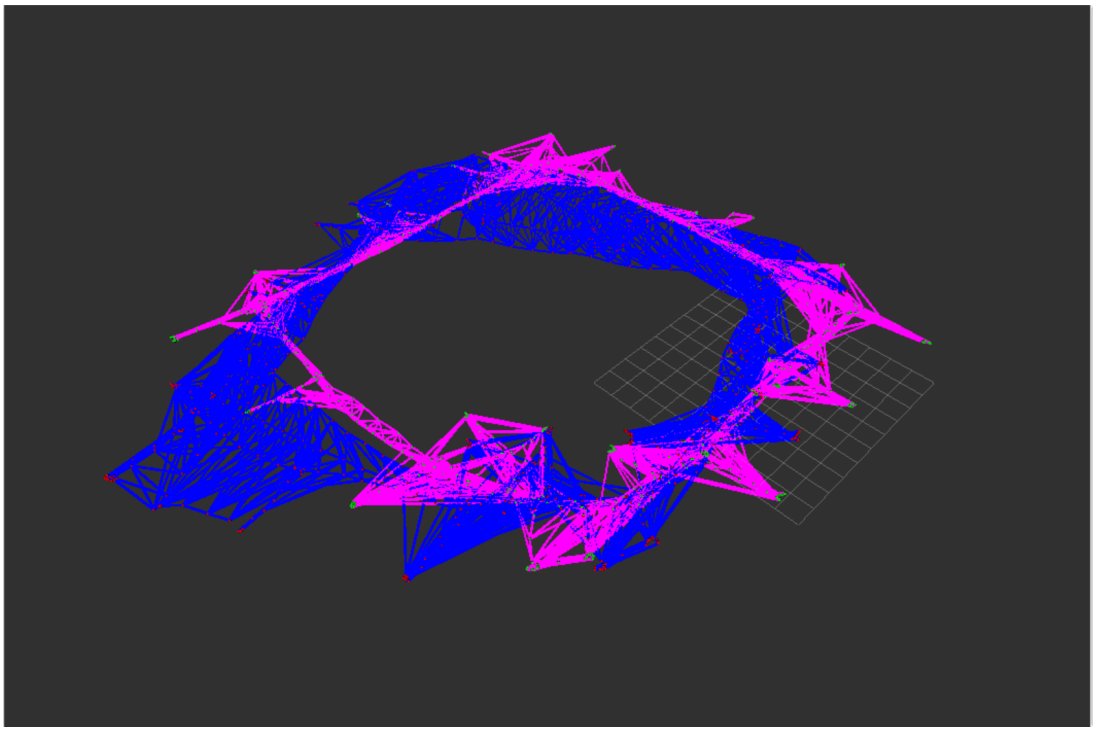

Figure 2: intel 数据

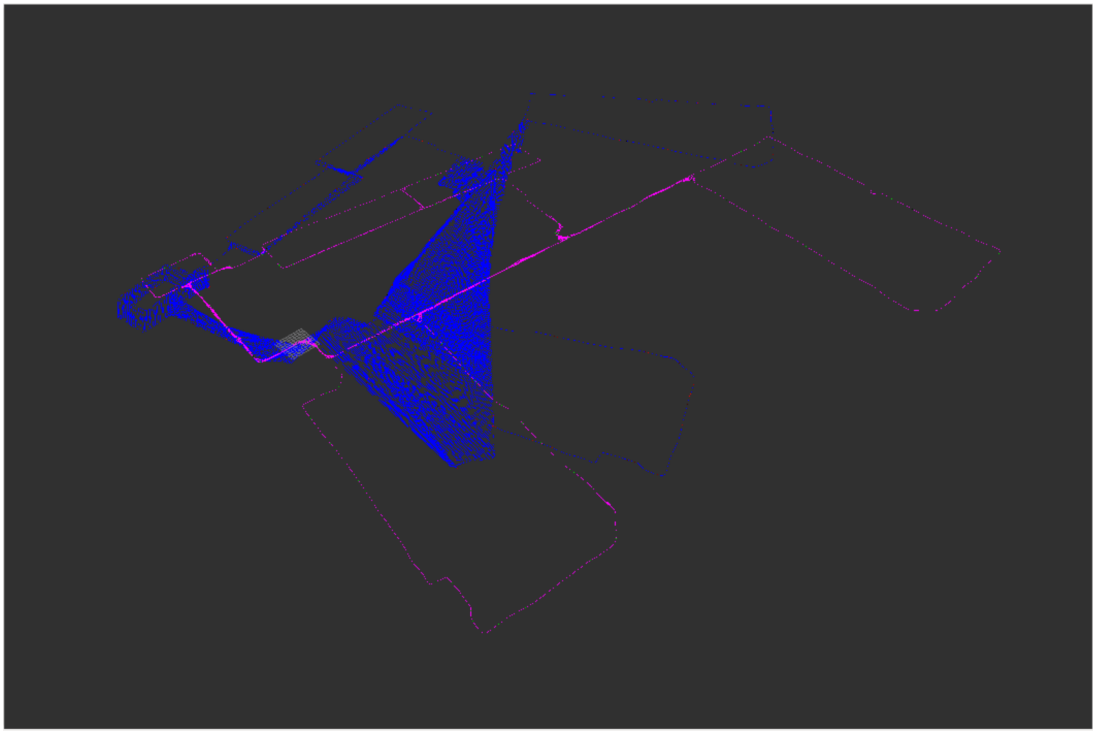

Figure 3: killian 数据

2. 不同数据集下的耗时对比可参见 Fig.4-Fig.6，对比不同数据集下的计算耗时可以发现当位姿较少时计算耗时主要来源于计算 Jacobian 矩阵，而当位姿较多时计算耗时则来自于求解方程 $H\Delta x = b$。

   要降低计算耗时可以考虑以下改进：

   (a) 由于位姿图中每条边互不影响，因此可以考虑在计算分块 Jacobian 矩阵时使用多线程来进行加速；

   (b) 利用 Hessian 矩阵的稀疏性来加速每次方程求解过程；

   (c) 使用更高效的优化算法来减少迭代次数；

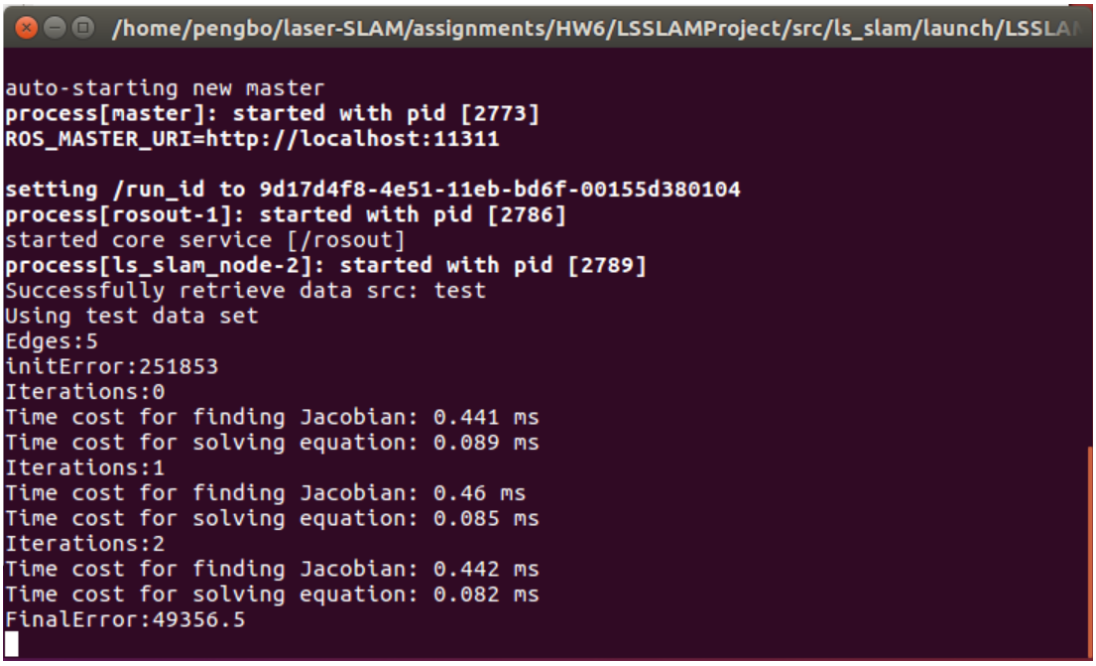

Figure 4: test 耗时

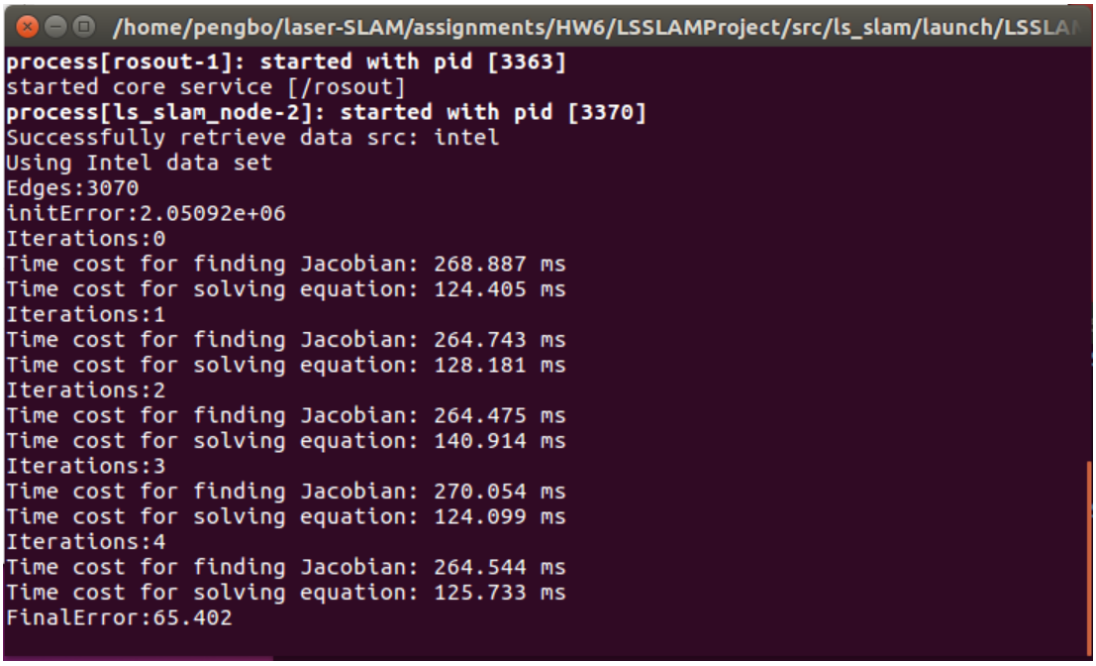

Figure 5: intel 耗时

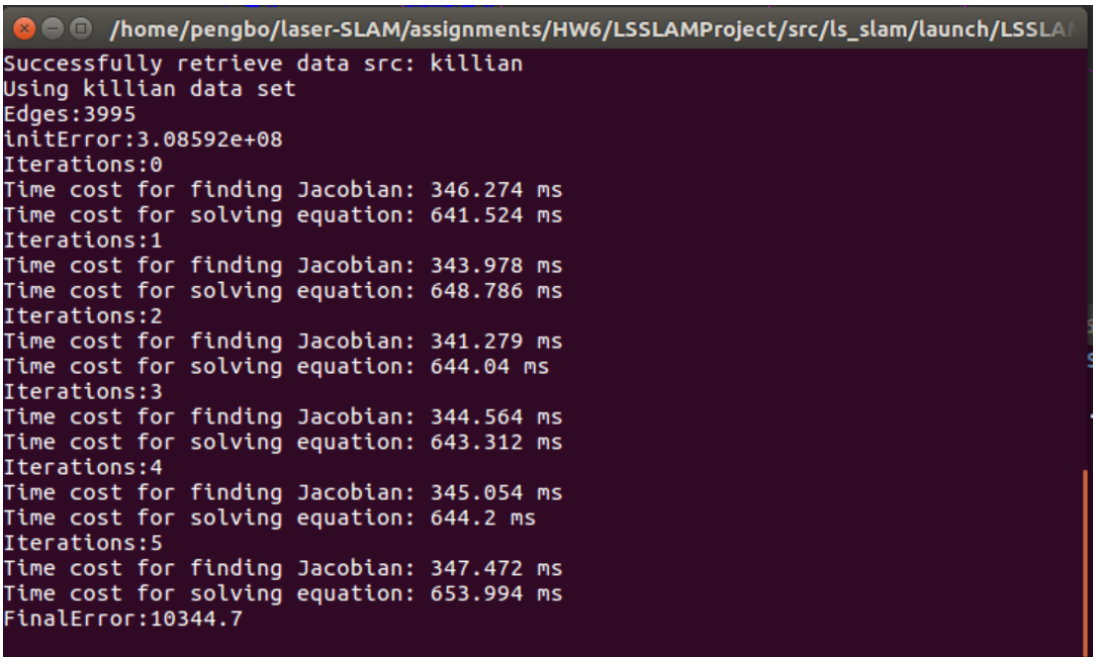

Figure 6: killian 耗时

3. 除了高斯牛顿法之外其他常用的非线性优化算法包括：

   (a) 最速下降法：一阶优化算法，每次计算目标函数在当前点的导数并计算最优步长来优化目标函数；

   (b) 牛顿法：二阶优化算法，每次迭代需要计算目标函数在当前点的 Hessian 矩阵并求解方程 $H\Delta x = b$；

   (c) Levenberg-Marquard 算法（LM 算法）：二阶优化算法，与高斯牛顿法相似都使用了 Jacobian 矩阵来近似 Hessian 矩阵 $H \sim J^T J$。与高斯牛顿法的区别在于 LM 算法在近似 Hessian 矩阵时还考虑了阻尼项 $H \sim J^T J + \mu I$；

   (d) DogLeg 算法：与 LM 算法类似均为 Trust Region 优化方法，区别在于 DogLeg 算法会根据当前置信域的半径以及迭代的步长来选择迭代方法。在每次迭代中 DogLeg 算法首先分别计算当前置信域半径 Δ 以及最速下降法迭代步长 Δx_{SD} 和高斯牛顿迭代步长 Δx_{GN}，若 $||\Delta x_{GN}|| < \Delta$ 则使用高斯牛顿法进行迭代，若 $||\Delta x_{SD}|| > \Delta$ 则使用最速下降法迭代并将 Δx_{SD} 缩减为 Δ，在其他情况下则使用 Δx_{SD} 和 Δx_{GN} 的线性组合来进行迭代。

4. 使用 GTSAM 进行位姿图优化的代码可参见./LSSLAMProject/src/ls_slam_gtsam/src 路径下 main.cpp 文件。在终端中输入 “ roslaunch ls_slam_gtsam LSSLAM.launch arg_data:=test ” 即可导入不同的数据集进行位姿图优化，优化后的位姿可参见 Fig.7-Fig.12。GTSAM 在处理 test 数据集时无法对位姿进行优化，但在其他两个数据集上可以正常工作。检查后发现在 test 数据集上相邻位姿之间的相对旋转均为 0，若将其修改为 $\frac{\pi}{2}$ 则可以正常对位姿进行优化。

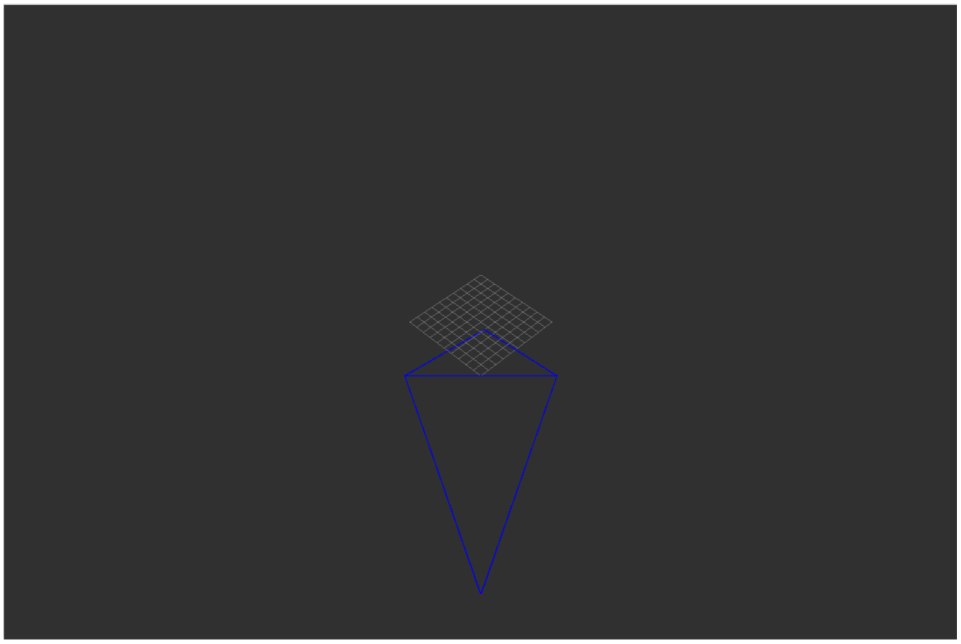

Figure 7: test 数据

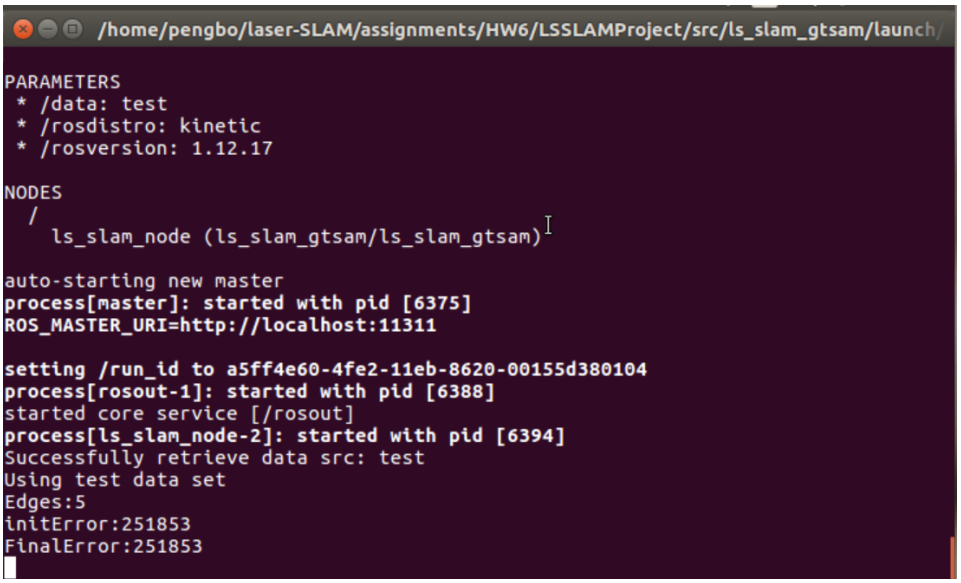

Figure 8: test 数据

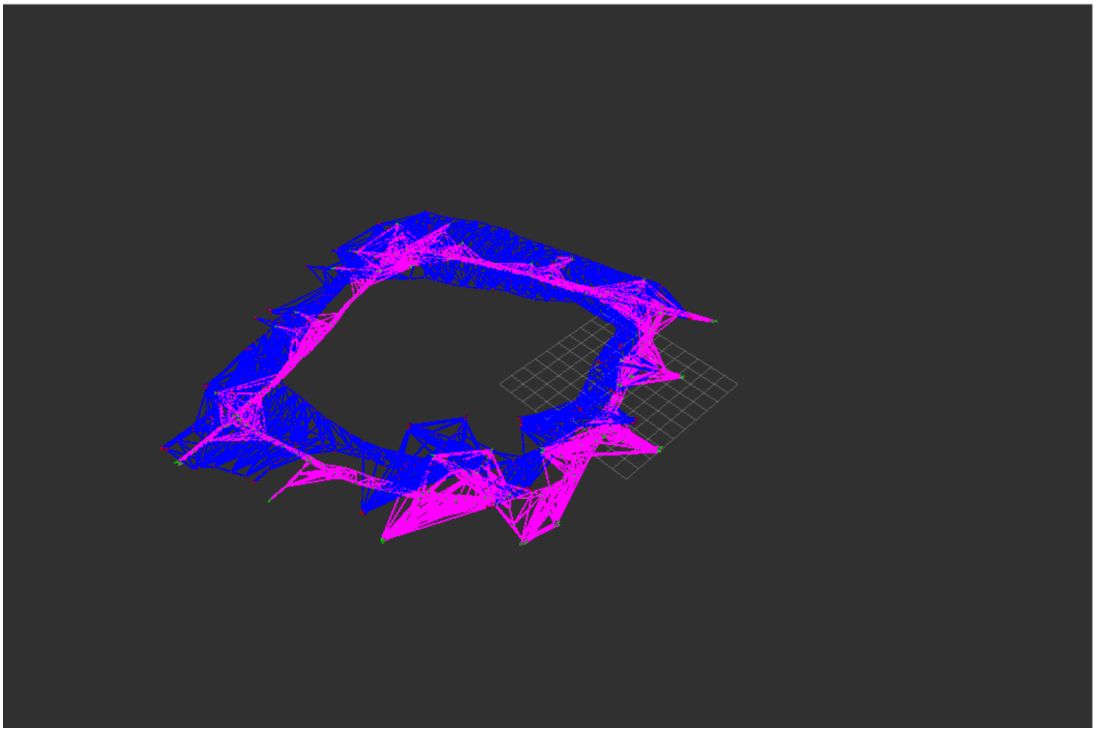

Figure 9: intel 数据

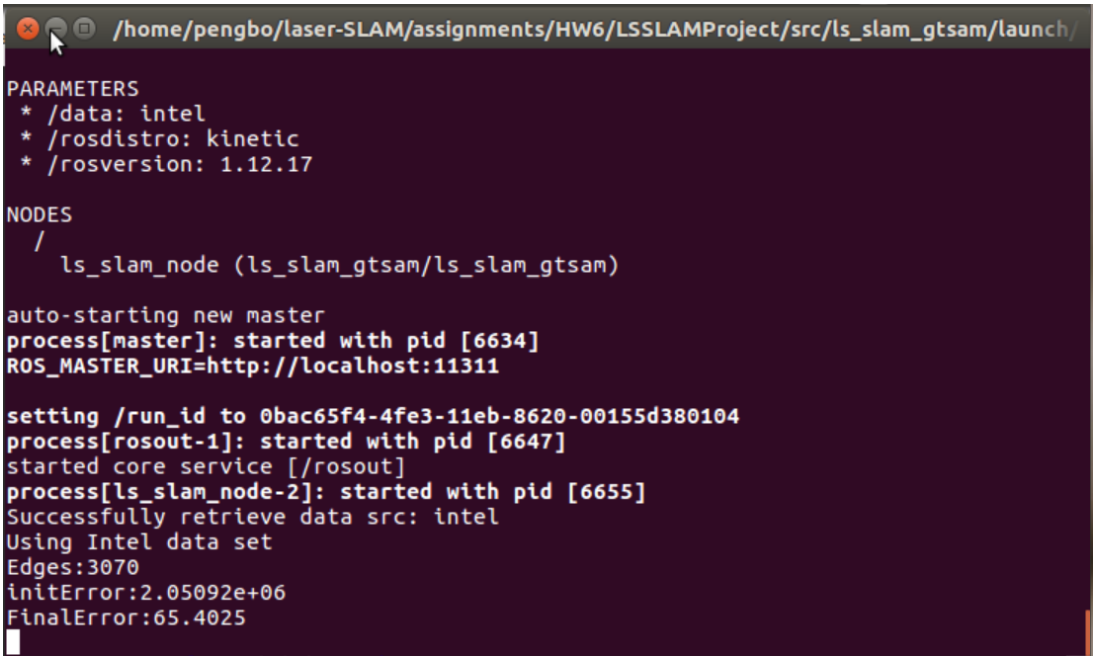

Figure 10: intel 数据

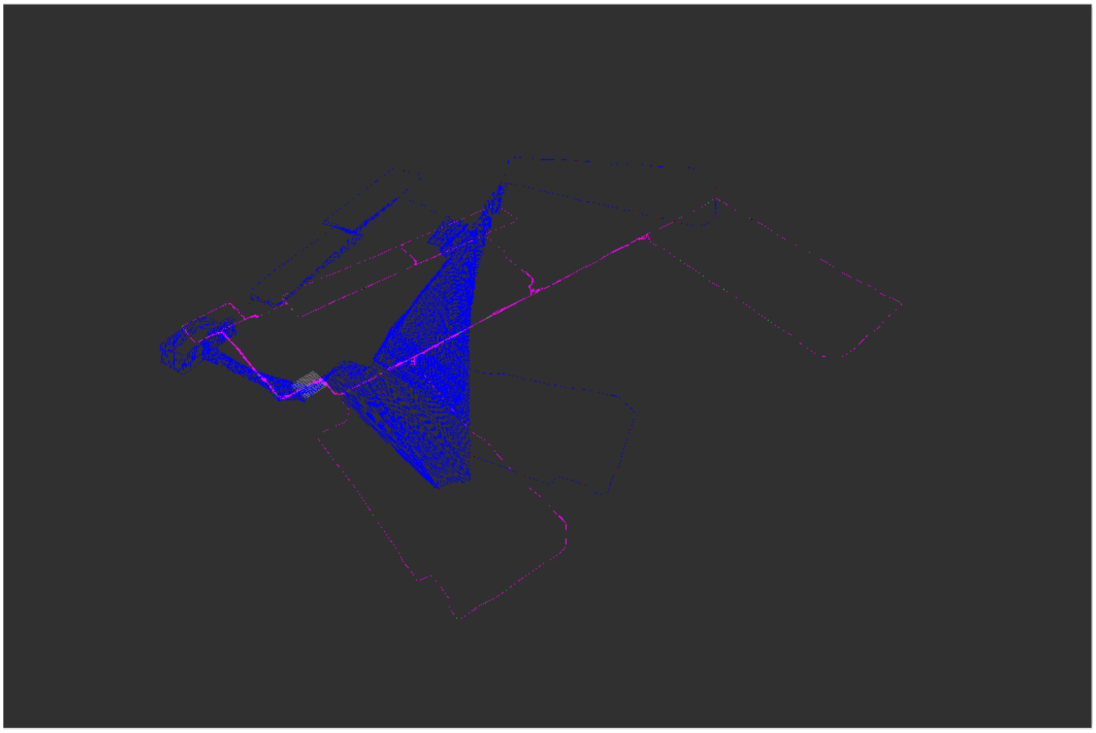

Figure 11: killian 数据

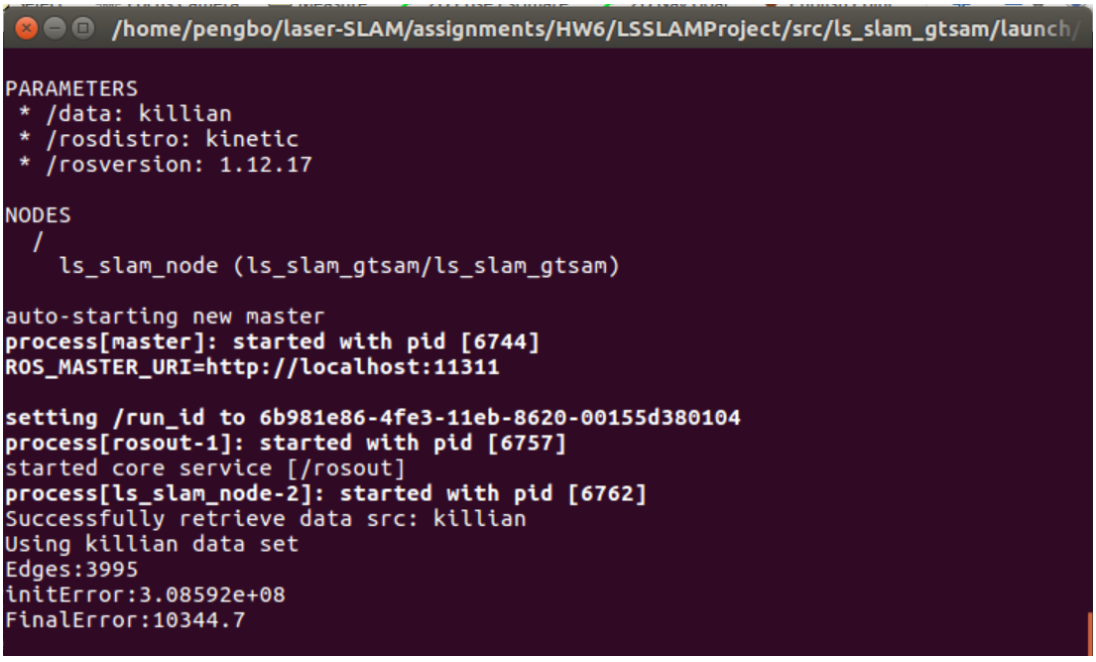

Figure 12: killian 数据