DLIO-PGO For UM_Dataset

Author : Kaho Thanks : XJTU-**Hongqian** **UM-Zezhou**

Direct LiDAR-Inertial Odometry + **Pose Graph Optimized**

设备：LIVOX MID360

代码方案: DLIO(前端) + PGO(基于欧式距离回环 ISAM2回环优化)

数据集地址：

> https://drive.google.com/file/d/1Xb_8QQbWIiBMpbFbQROR1aLZI65EO7u9/view?usp=sharing

1 DLIO具体适配修改

主要参考 DLIO-LIVOX 版本

1.1 DLIO 需要的点云的类型为XYZIT, 需要得到每个点的time，来做点云去畸变

```
void callbackLivox(const livox_ros_driver2::CustomMsgConstPtr& livox);

// Livox-specific structure for livox_ros_driver2/CustomMsg
struct LivoxPoint {
  LivoxPoint(): data{0.f, 0.f, 0.f, 1.f} {}
  PCL_ADD_POINT4D;
  float intensity; // intensity
  std::uint32_t offset_time; // LIVOX: time from beginning of scan in nanoseconds
  EIGEN_MAKE_ALIGNED_OPERATOR_NEW
} EIGEN_ALIGN16;

POINT_CLOUD_REGISTER_POINT_STRUCT(LivoxPoint,
                                  (float, x, x)
                                  (float, y, y)
                                  (float, z, z)
                                  (float, intensity, intensity)
                                  (std::uint32_t, offset_time, offset_time))
```

1.2 LIVOX雷达imu得到的accel 单位为g，DLIO需要m/s^2, 需要修改

```
if (this->sensor == dlio::SensorType::LIVOX) {
    lin_accel[0] = imu->linear_acceleration.x * this->gravity_;
    lin_accel[1] = imu->linear_acceleration.y * this->gravity_;
    lin_accel[2] = imu->linear_acceleration.z * this->gravity_;
  } else {
    lin_accel[0] = imu->linear_acceleration.x;
    lin_accel[1] = imu->linear_acceleration.y;
    lin_accel[2] = imu->linear_acceleration.z;
  }
```

或可参考FASTLIO2中，对重力不同单位进行适配的操作代码,对重力取模

```
        init_state.grav = S2(- mean_acc / mean_acc.norm() * G_m_s2);     //  FASTLIO2
```

2 PGO具体适配修改

主要参考 FASTLIO-PGO 版本, 该版本代码支持ScanContex 回环 与 Radius 欧式距离回环，因为UM-dataset为MID360为主，而ScanContext为基于机械式雷达极坐标系构建描述子，所以个人感觉使用Radius 欧式距离回环效果会更佳。

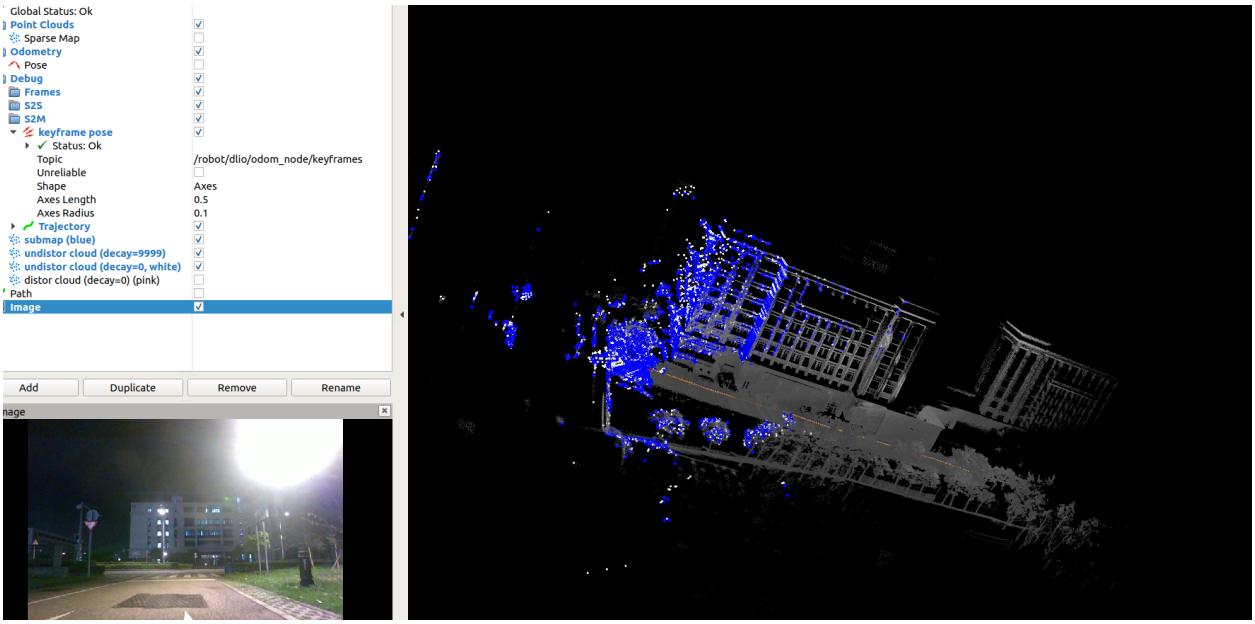
2.1 PGO接受点云为local系，DLIO输出点云为Global系，订阅里程计进行gloabl → local 变换

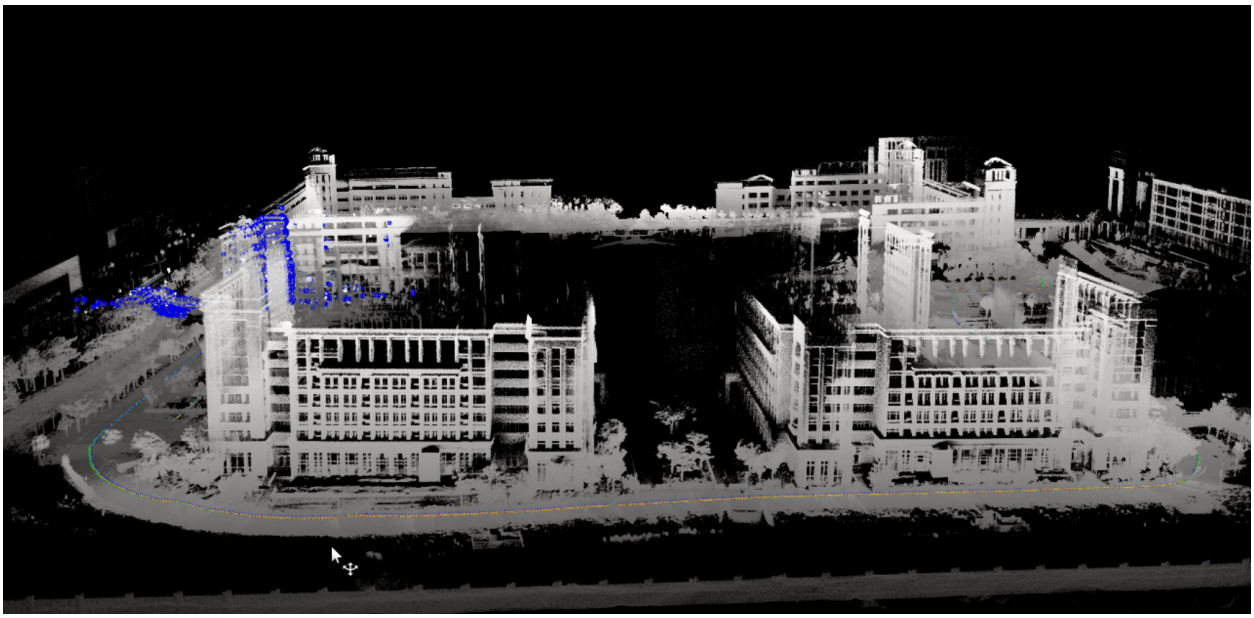
```
// 将keyframe 从global ——> link
            Eigen::Affine3f T_w_link = pcl::getTransformation(pose_curr.x, pose_curr.y, pose_curr.z, pose_curr.roll, pose_curr.pitch, pose_
            Eigen::Matrix4d T_link_w = T_w_link.inverse().matrix().cast<double>() ;
            pcl::transformPointCloud(*thisKeyFrame, *thisKeyFrame, T_link_w);
```

2.2 自动保存TUM 与 KITTI EVO格式

```
saveOptimizedVerticesKITTIformat(isamCurrentEstimate, pgKITTIformat); // pose
            saveOdometryVerticesKITTIformat(odomKITTIformat); // pose
            saveOdometryVerticesTUMformat(odomTUMformat); // pose
            saveOptimizedVerticesTUMformat(isamCurrentEstimate, pgTUMformat);
```

3 DLIO UM 数据集快速测试



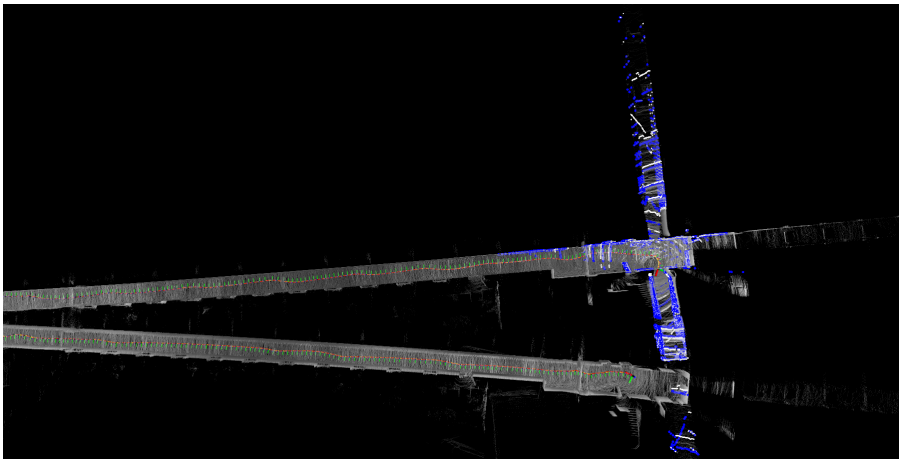


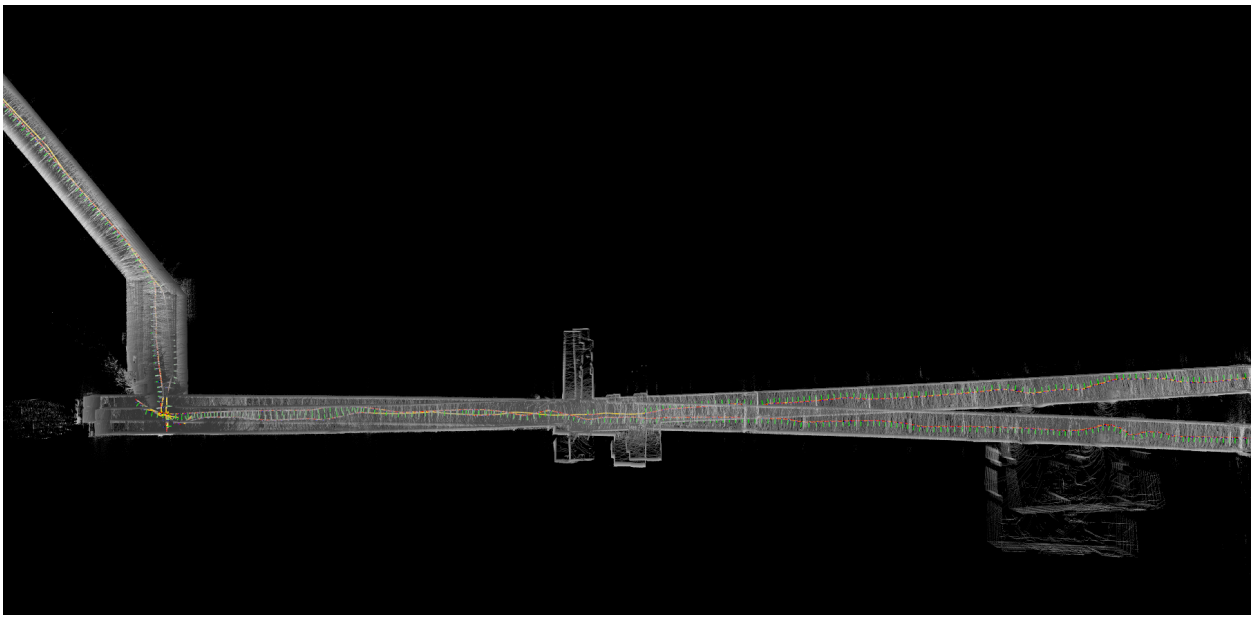
```
#step1 运行数据DLIO， 自动接受MID360点云数据
roslaunch direct_lidar_inertial_odometry um_livox_mid360.launch

#step2 播放数据集
rosbag play  um.bag
```

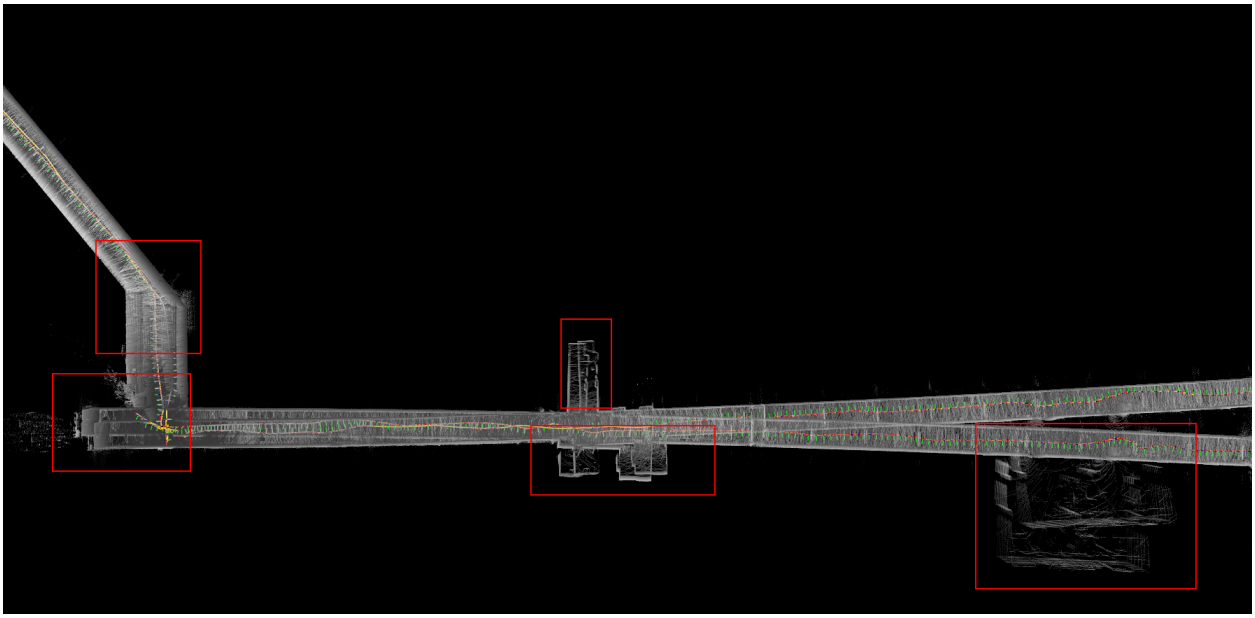
4 DLIO-PGO 数据集快速测试

4.1 ICCV长廊数据集测试

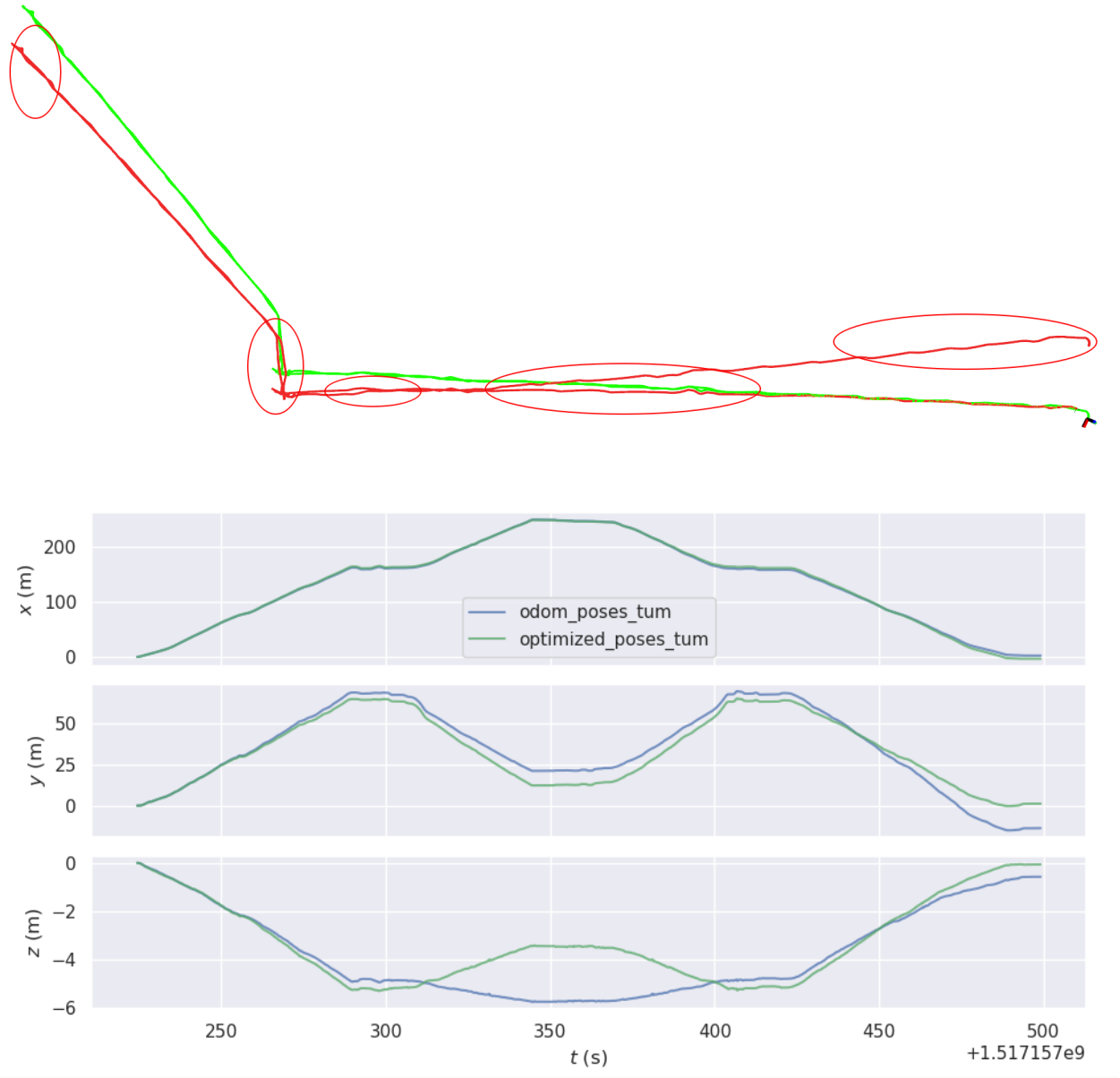
DLIO 开环参数较差，长廊回程累积误差较大



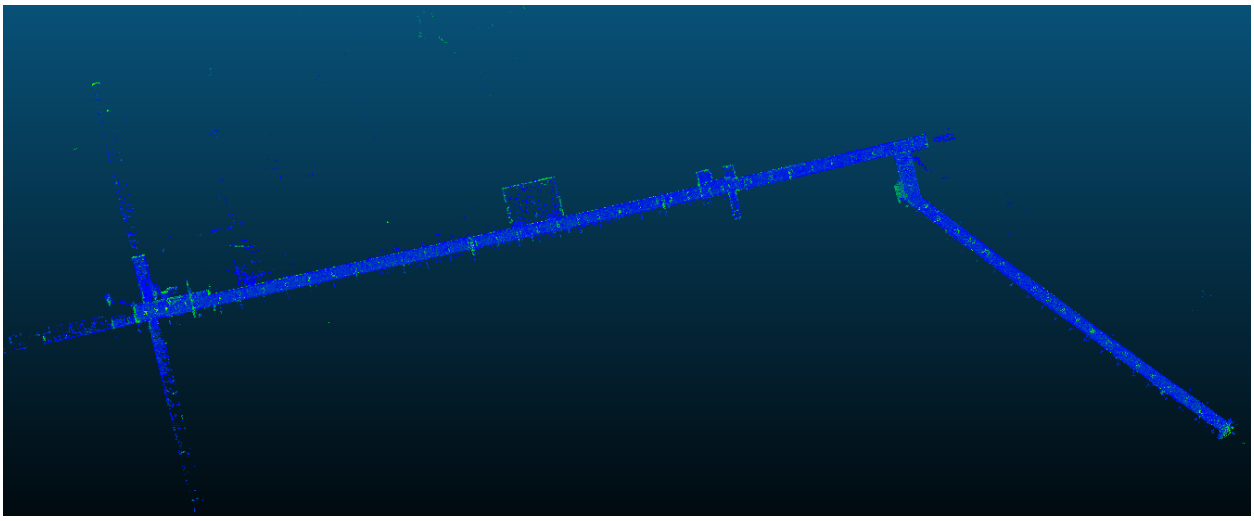


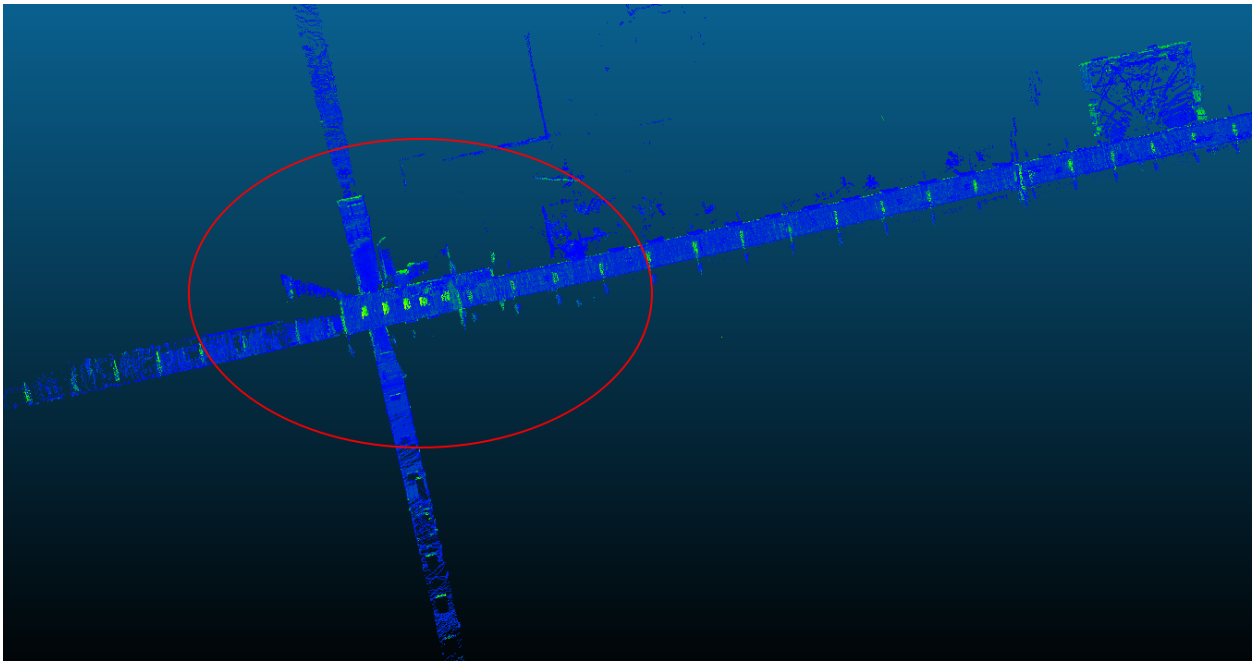


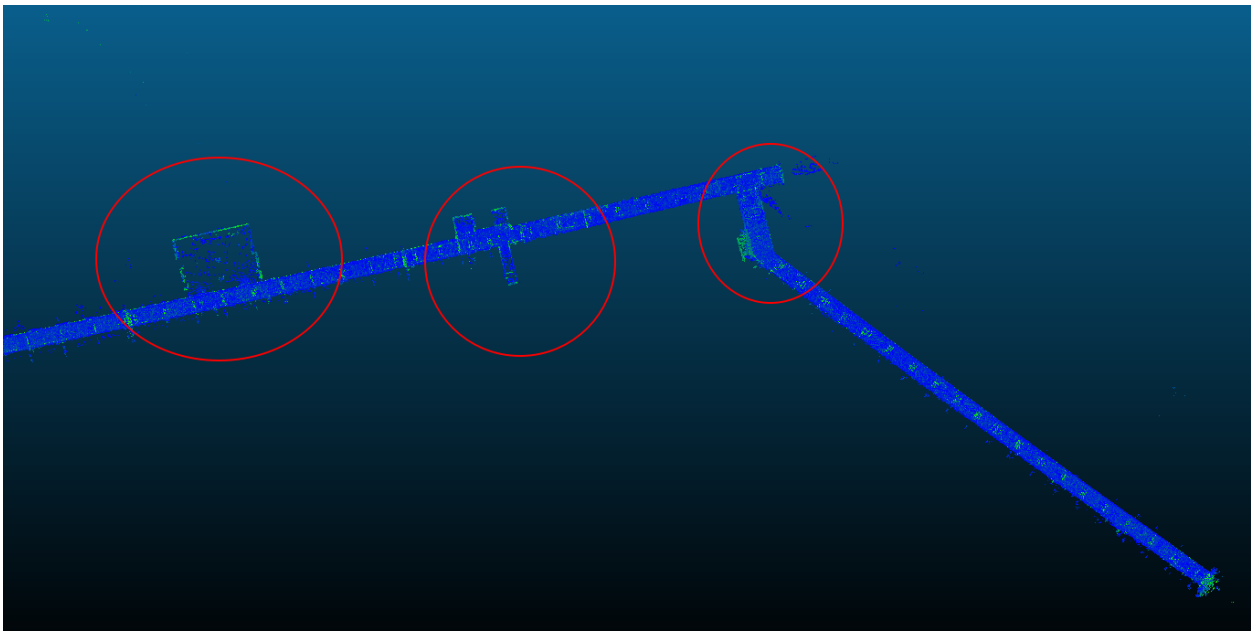
EVO — 红色(DLIO)分叉， 绿色(DLIO-PGO)回环优化，修复累积误差

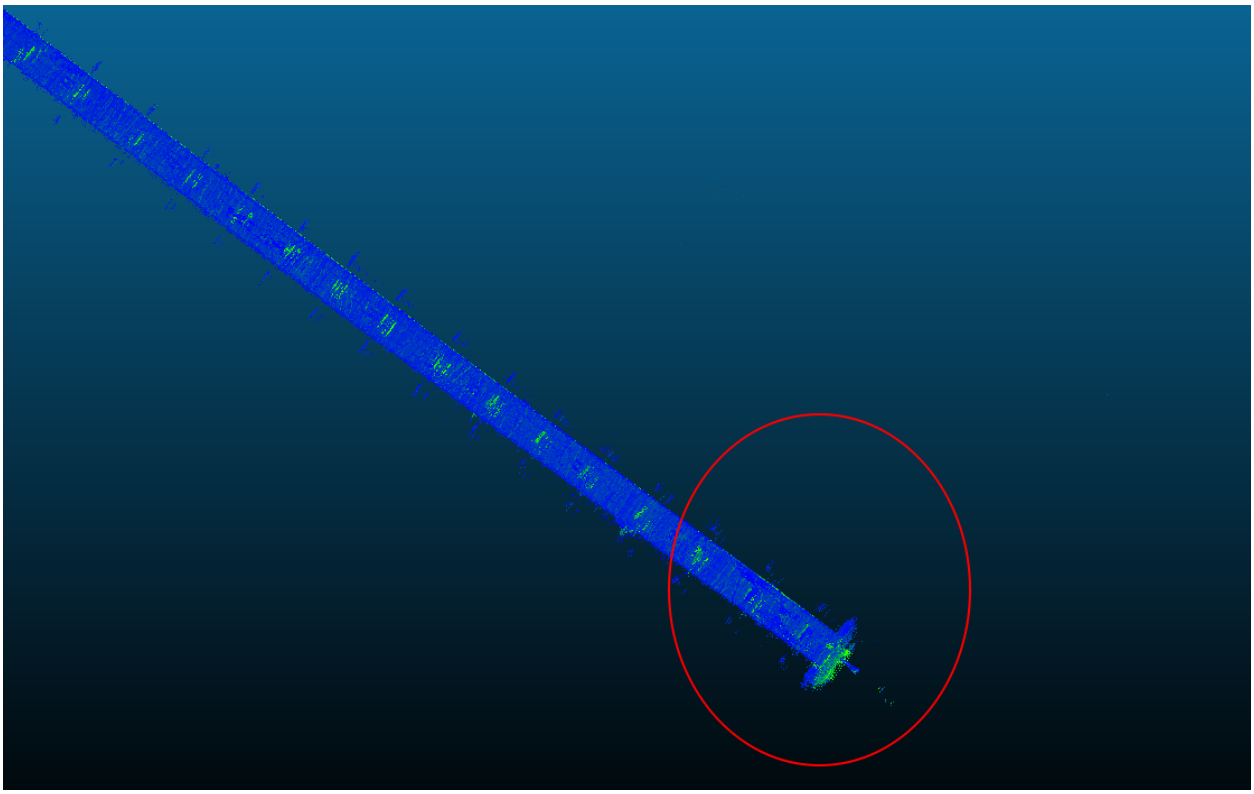


DLIO-PGO回环修正长廊地图（对比DLIO开环，一致性良好，长廊地图没明显分叉）



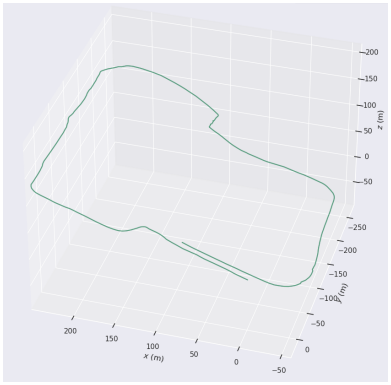


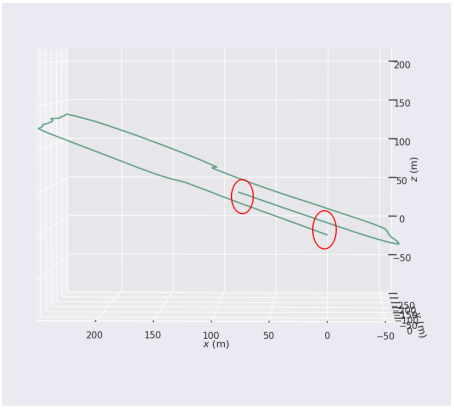


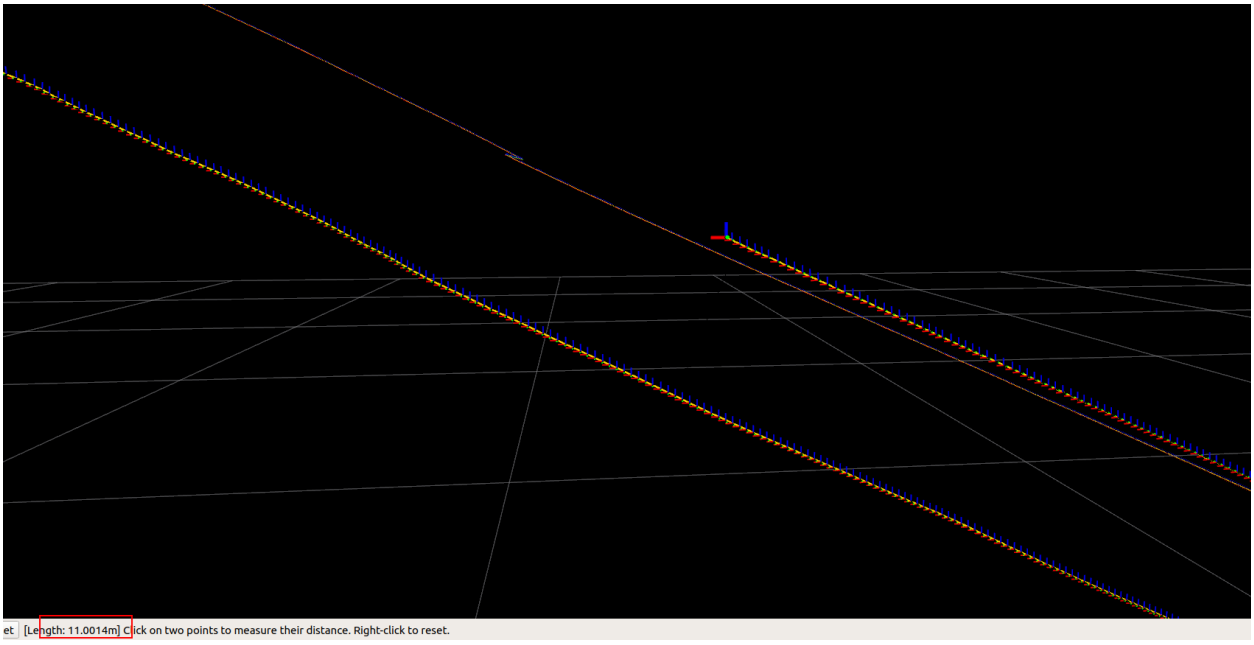


4.2 UM数据集 DLIO-PGO存在问题

地图较大，回环较少，只有最后一部分有回环，导致Z轴累积误差达到10M，触发不到回环检索。







4.3 DLIO-PGO 快速运行,默认欧式距离回环

```
#step1 运行DLIO
roslaunch direct_lidar_inertial_odometry um_livox_mid360.launch
#step2 运行PGO
roslaunch aloam_velodyne dlio_mid360_um.launch
```

切换到ScanContext 回环

```
// FILE： PGO/laserPosegraphOptimization.cpp
while (ros::ok())
    {
        rate.sleep();
        performSCLoopClosure();    //  使用ScanContext回环
       // performRSLoopClosure();     //  使用欧式距离回环
        visualizeLoopClosure();
    }
```

5 其他问题

5.1 将IMU raw_data 中的原始线加速度转换到其他坐标系下，需要考虑平移方向的科氏力，不能只考虑旋转变换

公式暂没找到出处，代码可参考DLIO，将imu raw_data 转换到base_link坐标系下

```
// Transform linear acceleration (need to account for component due to translational difference)
Eigen::Vector3f lin_accel(imu_raw->linear_acceleration.x,          Kenny Chen, 4 months ago • Release v1.0.0 ...
                          imu_raw->linear_acceleration.y,
                          imu_raw->linear_acceleration.z);

Eigen::Vector3f lin_accel_cg = this->extrinsics.baselink2imu.R * lin_accel;

lin_accel_cg = lin_accel_cg
               + ((ang_vel_cg - ang_vel_cg_prev) / dt).cross(-this->extrinsics.baselink2imu.t)
               + ang_vel_cg.cross(ang_vel_cg.cross(-this->extrinsics.baselink2imu.t));

ang_vel_cg_prev = ang_vel_cg;

imu->linear_acceleration.x = lin_accel_cg[0];
imu->linear_acceleration.y = lin_accel_cg[1];
imu->linear_acceleration.z = lin_accel_cg[2];

return imu;
```

旋转变换

平移变换

5.2 DLIO-PGO 外置回环不是最优解，最优解是内置回环

edited by kaho 2023.9.25