lidar_perception 用户使用说明

主要内容

- 功能概述
- 注意事项
- 准备事项
- 快速开始

1 功能概述

lidar_perception SDK 是一个使用多个 Lidar（2 或 3 个）对自动驾驶车辆提供环境感知算法的软件开发包。其中包括了 Lidar 的外参标定、多个 Lidar 的点云数据融合、将融合后的点云数据映射为 gridmap、对融合后的点云进行地面分割、以及实现对点云的障碍物检测等模块，满足利用多 Lidar 进行环境感知的任务。

2 注意事项

■ 不支持虚拟机和 Windows 系统！您必须安装本地 Linux 系统，例如 Ubuntu。

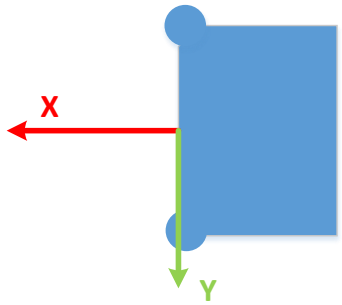
■ 本 SDK 完全基于 C++，只依赖 PCL，Eigen 和 OpenCV 等 C++第三方库，如果您安装了 ROS kinetic desktop-full 版，那么您不需要考虑安装配置这些第三方库的问题。因此，为了方便操作，强烈建议安装 ROS kinetic desktop-full 版。

■ 坐标系统设置

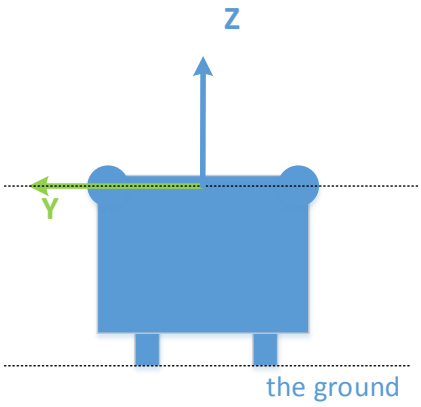
本 SDK 的所有坐标系均为右手坐标系！

(1)车辆坐标系

X 指向车辆的正前方，Y 指向左，Z 指向天空，坐标原点设置为车辆前方两个激光雷达水平连线的几何中心，这意味着原点的 Z 坐标不在地面上，而和激光雷达的安装高度一致。



bird-eye view

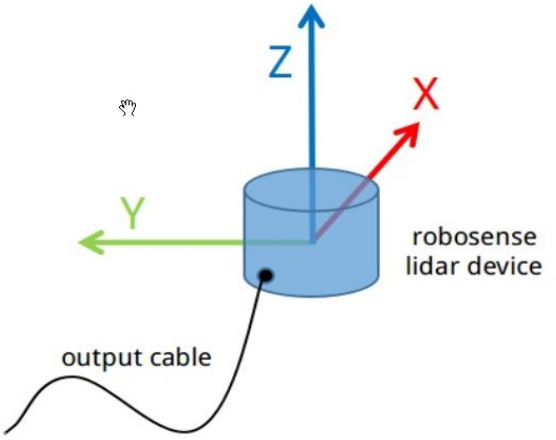


back-eye view

(2)激光雷达坐标系

X 指向输出电缆的相反方向，或者输出电缆指向 X 轴的负方向，如果将激光

雷达指向车辆正前方安装，则 Y 指向左侧，Z 指向天空，如下图所示。



■ yaml 文件参数设置

在运行该 SDK 之前，需对 lidar_perception/cfg/lidar_perception.yaml 进行填写修改，文件参数具体含义见注释。另外，文件中 trans_parms 里旋转角度的正负也遵循右手坐标系定则。

3 准备事项

■ 采集数据对各个 Lidar 进行外参标定：

由于采集的点云数据为激光坐标系下的数据，因此需要将激光坐标系下的数据转换到车辆坐标系下，即获得各个 Lidar 的外参矩阵，或者获得构成外参矩阵的 6 个矢量值：x, y, z, roll, pitch, yaw，其中，前 3 个值代表分别沿 X，Y，Z 方向平移的距离；后 3 个值代表分别沿 X，Y，Z 方向旋转的角度。

将实验车辆开到一个开阔平坦的地方（平整宽阔的路面最佳），在各个激光雷达安装连接完成后，在线接收数据或录制 rosbag 数据利用 lidar_align_tool 进行外参校准。需要注意的是 lidar_align_tool 接收的点云格式为 sensor_msgs::PointCloud2，即只要是符合此格式的点云数据，均可以使用此工具进行外参校准。

■ lidar_align_tool 使用说明：

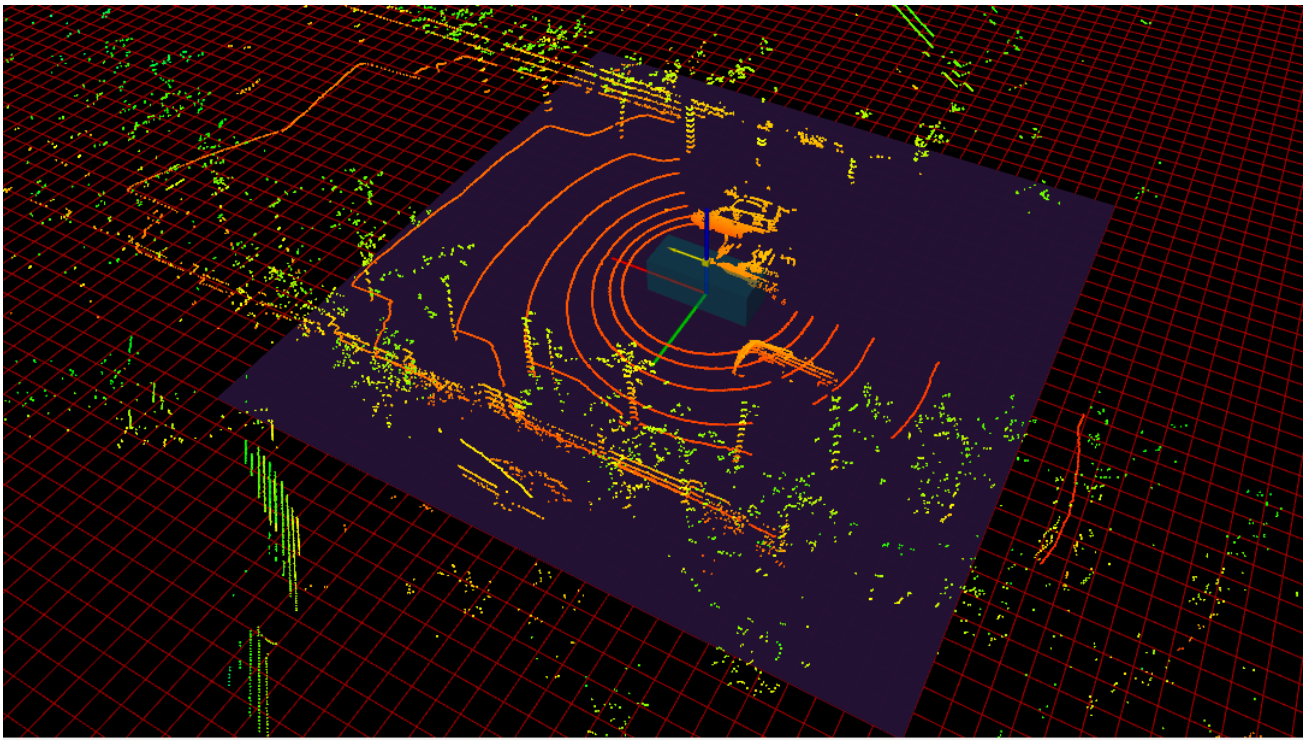
⑴修改 lidar_align_tool/launch/lidar_align.launch 文件，设置 lidar_align_tool 工

具接收待标定的激光点云 topic 名称（lidar_topic）以及帧 id（lidar_frame_id）；

⑵将 lidar_align_tool 放置新建工作空间 catkin_ws/src 下，进行以下操作；

```
cd catkin_ws/
catkin_make
source devel/setup.bash
roslaunch lidar_align_tool lidar_align.launch
```

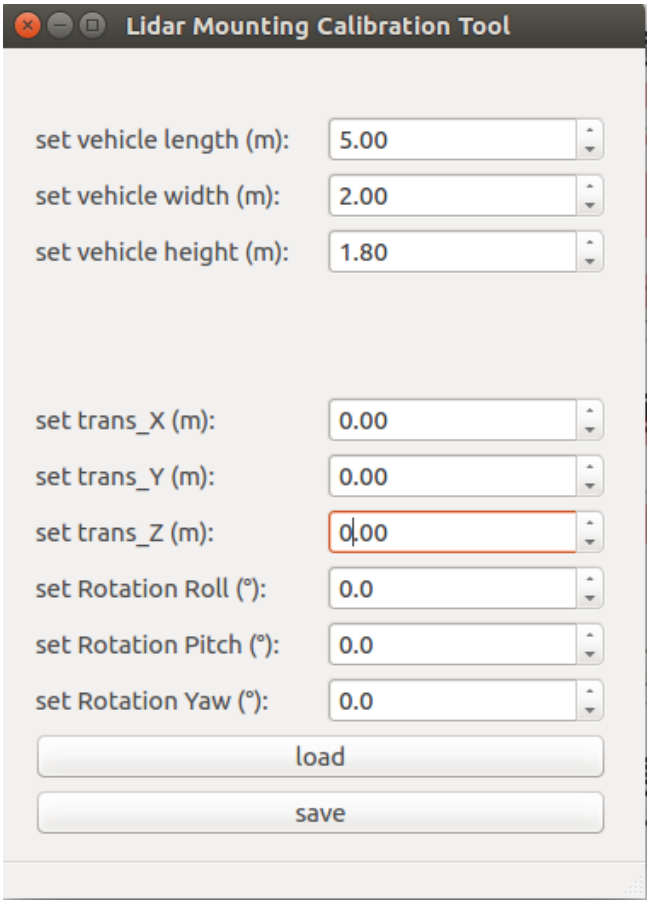
以下是工作时候的界面：



其中，紫色透明平面是虚拟标准地面参考，青色框是虚拟车辆，黄色小盒子以及箭头是虚拟激光雷达及其方向。注意，界面中的车辆坐标系默认为：红色为 x 轴，绿色为 y 轴，蓝色为 z 轴，坐标原点车辆投影到地面的几何中心，而不是前面所说的车辆正前方两个激光雷达水平连线的中心。（注：此标定工具最大的作用是为了获得旋转的 roll, pitch, yaw 值，对于需要平移的 x, y, z 可以不借助此工具获取，实际使用米尺测量确定即可，因此车辆坐标系的原点位置不同不影响旋转角度 roll, pitch, yaw 的确定）

⑶开始校准时，首先设置激光的安装高度，即（set trans_z）；之后根据根据点云与 grid 画线的偏移程度（比如：路面车道线是否平行于 grid 画线等）来确定 yaw 值，最后，调整 roll 值和 pitch 值以保证大部分地面点能很好的隐藏在紫色透

明平面中，调整界面如下图所示；



⑷一个 Lidar 校准完成后，点击 save 保存标定结果，即构成外参矩阵的 6 个值保存在 lidar_align_tool/align_result/align.txt 文件中，之后将此值填写至 lidar_perception/cfg/lidar_perception.yaml 的 trans_parms 中；

⑸重复以上步骤，完成其他 Lidar 的外参标定并进行保存填写。

■ lidar_perception/car_rviz_model 模块为构建的虚拟本车的 robotmodel，可在 rviz 中实时显示。

■ 按照注释完成 lidar_perception/cfg/lidar_perception.yaml 文件的配置，其中包括：接收左右 Lidar（根据实际情况可以进行添加至 3 个或多个）点云数据的 topic、多个 lidar 点云融合后的 center_lidar_topic、去地面的点云 no_ground_topic、障碍物检测的 boundingbox topic、融合后的点云映射生成的 gridmap topic、关注的路面区域 roi（依据车辆坐标系）、生成 gridmap 的分辨率、本车信息、以及各个 Lidar 标定的可构成外参矩阵的 6 个值等，具体配置可参考如下：

```yaml
# subscribe to the lidar pointcloud
left_lidar_topic: /left/rslidar_points

right_lidar_topic: /right/rslidar_points

# fusion to center lidar topic
center_lidar_topic: /center/lidar_points

# no_ground_point_topic
no_ground_topic: /points_no_ground

# detecting bounding box topic
detecting_bbox_topic: /detected_bounding_boxs

# grid map topic
grid_map_topic: /gridmap

#installation height of lidar
lidar_height: 1.025

# roi zone
roi:
  x_min: -20.0
  x_max: 80.0
  y_min: -20.0
  y_max: 20.0
  z_min: -2
  z_max: 5

# map resolution 0.1m/grid
map_resolution: 0.25

# car information
car_info:
  car_width: 1.8
  car_length: 4

```

- 安装 rviz 显示检测 boundingbox 的插件包，安装命令如下：

```
sudo apt-get install ros-kinetic-jsk-rviz-plugins
sudo apt-get install ros-kinetic-jsk-recognition-msgs
```

4 快速开始

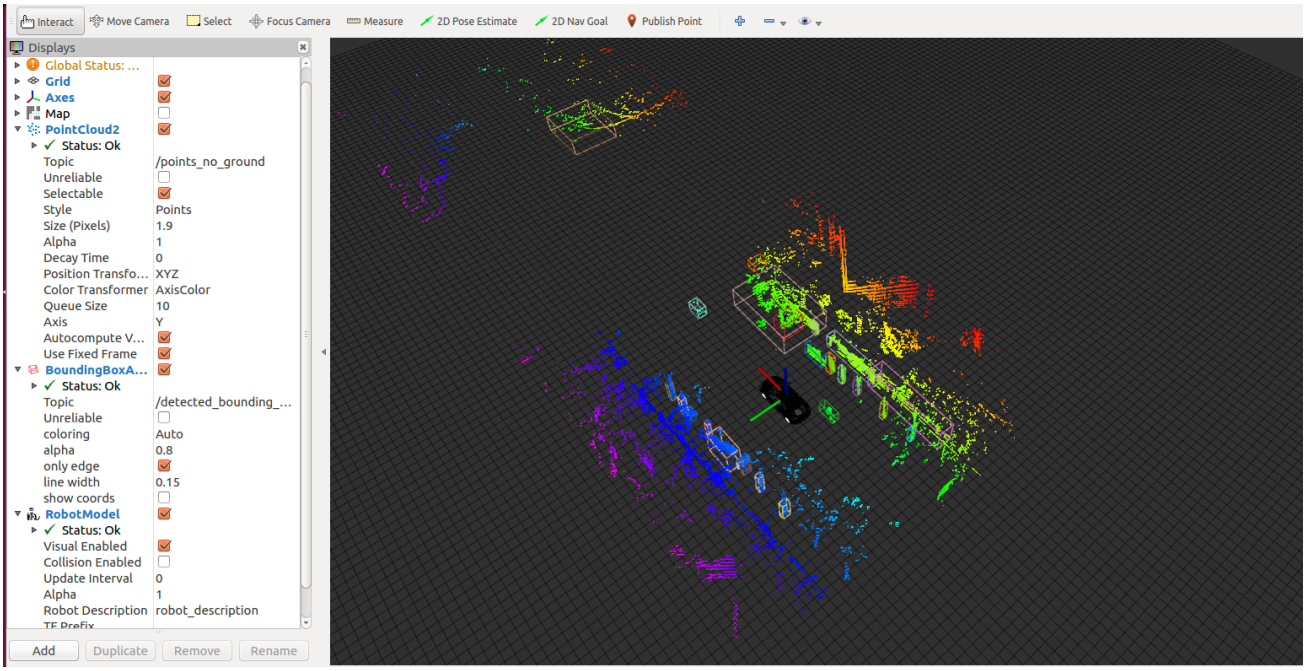
在完成准备事项以后，可以新建工作空间 catkin_ws 以后，将 lidar_perception 包放入 catkin_ws/src 目录下，运行以下命令启动 lidar_perception SDK：

```
catkin_make
source devel/setup.bash
roslaunch lidar_perception lidar_perception.launch
```

关于 launch 文件，具体配置如下，包括启动主节点、显示虚拟车辆模型以及启动 rviz 等，可根据具体需求进行修改。

```xml
<?xml version="1.0"?>

<launch>
  <!-- Console launch prefix -->
  <arg name="launch_prefix" default=""/>

  <!-- Load parameters -->
  <rosparam command="load" ns="lidar_perception" file="$(find lidar_perception)/cfg/lidar_perception.yaml"/>

  <!-- Start node -->
  <!--node pkg="rs_process" type="rs_process_node" name="rs_process" output="screen" launch-prefix="xterm -e gdb -ex run args"-->
  <node pkg="lidar_perception" type="lidar_perception_node" name="lidar_registration" output="screen">
  </node>

  <!--display/-->
  <!--car module for ui display-->
  <group>
    <arg name="gui" default="False" />
    <param name="use_gui" value="$(arg gui)"/>
    <param name="robot_description" textfile="$(find lidar_perception)/car_rviz_model/no_map/default.urdf" />
    <node pkg="joint_state_publisher" type="joint_state_publisher" name="joint_state_publisher"  />
    <node pkg="robot_state_publisher" type="state_publisher" name="robot_state_publisher"  />
  </group>

  <!--rviz show-->
  <node pkg="rviz" type="rviz" name="rviz"  args="-d $(find lidar_perception)/rviz/lidar_perception.rviz"/>


</launch>

```

在程序正常运行后，会自启 rviz，具体界面如下，根据需要在对应的 topic 上打对号：



附：/gridmap 和 /detected_bounding_boxs topic 的效果图