

# 路沿检测算法说明

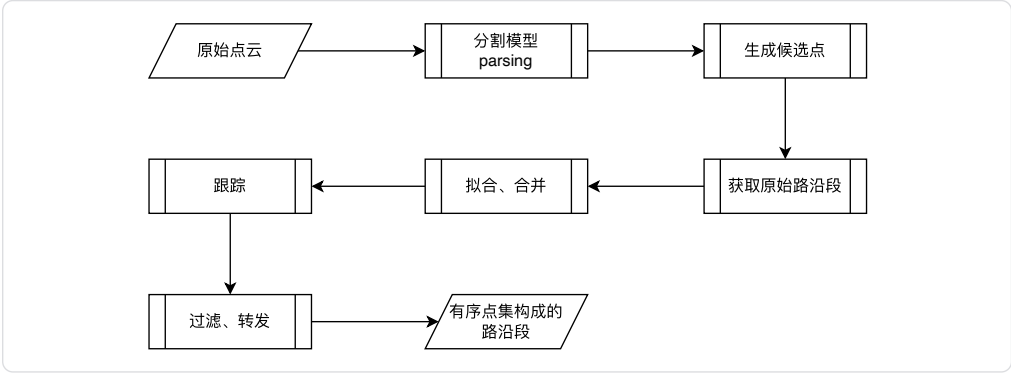
## 思路

输入：点云

算法：模型输出点云语义、后处理选择keypoint、拟合成路沿段

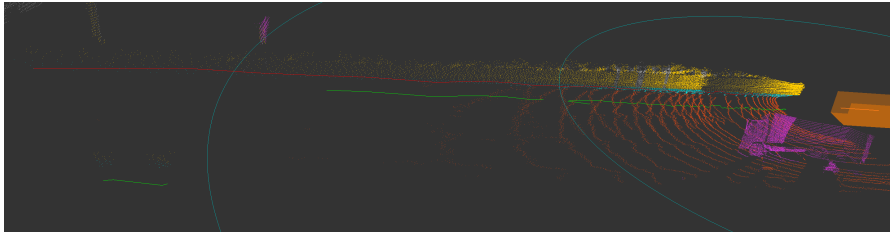
输出：有序点集组成的路沿段

## 算法框图



1	语义分割	候选点	关键点（红色）	关联、过滤结果
2				

### 可视化示意



- 语义分割结果：黄色（栅栏）、橙色（地面）、粉色（障碍物）
- 候选点：浅蓝色（青色）的点，（红色线条附近）
- 原始路沿：绿色线条（为了区分，高度降低1m）
- 拟合合并：红色线条

## 代码

[https://console.cloud.baidu-int.com/devops/icode/repos/baidu/adu-lab/andes/blob/dev-autumn\\_6-2-23/lib/perception/lidar/lib/segmentation/curb\\_segmentation/curb\\_segmentation.cc](https://console.cloud.baidu-int.com/devops/icode/repos/baidu/adu-lab/andes/blob/dev-autumn_6-2-23/lib/perception/lidar/lib/segmentation/curb_segmentation/curb_segmentation.cc)

## 模块

### 分割模型

- 模型输出点云语义，为每一个point增加语义label

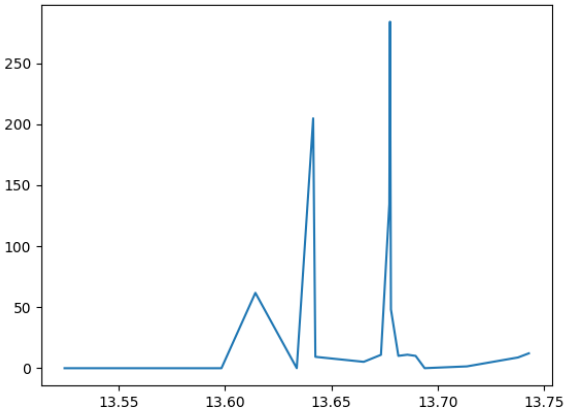
### 生成候选点

- 根据语义选择候选点，并过滤异常点。条件(与&&)：

- point.parsing\_label是路沿或栅栏
  - point属于地面平面内
  - point高度在一定范围内
2. 在柱坐标系中按序组织候选点
- 计算point在柱坐标系中的 r 和 theta
  - 按角度分bin，将point信息保存在每个bin中
  - 每个bin中按 r 从小到大排序点

获取原始路沿段

1. 按theta从0到180的顺序，在每个bin中判断是否存在路沿点
- 在bin中计算相邻2个point之间的梯度，梯度大于一定阈值则判定为路沿。



2. 每次获取到新路沿点判断是否属于上一路沿段
- 由r和theta自适应生成2个point之间的最大距离max\_dis，若两个point的距离大于max\_dis，则创建一个新路沿段
3. 计算路沿基本信息，暂存在curbinfo中
- 将路沿段所有点按x从小到大排序
  - 左右：路沿段所有point的y\_mean，y\_mean > 0则为左路沿
  - 长度：计算该路沿段的长度，一个点长度为0
4. 按所有路沿段长度排序从大到小排序

拟合、合并

1. 依次取最长的路沿段拟合曲线
- 长度大于阈值（3.5m）、点数大于阈值（10）才可以拟合，否则只能被其他路沿段合并
  - 拟合成功才可以输出
  - 拟合曲线阶次随长度自适应调整
2. 依次取比当前路沿段短的路沿，判断该路沿是否在当前路沿的延长线上
- 左右方向一致，才可以合并
  - 路沿段横向误差abs（y-y\*）小于阈值（0.3），才可以合并
  - 按x从小到大排序合并后的路沿，
  - 拟合曲线成功才算合并完成，否则合并失败
  - 添加信息到frame->lane\_boundary\_line中

跟踪

TODO


过滤、转发

1. 按需求条件过滤TODO
2. 将路沿转存到frame->lane\_boundary中，用于可视化

参考

<https://www2.mathworks.cn/help/lidar/ug/curb-detection-in-lidar-point-cloud.html>

[https://github.com/jkk-research/urban\\_road\\_filter/blob/main/src/star\\_shaped\\_search.cpp](https://github.com/jkk-research/urban_road_filter/blob/main/src/star_shaped_search.cpp)

 sensors-22-00194-v2.pdf (35MB)



