实验报告

211250175 王艺羲

1. 实验步骤及结果

在simulator.launch文件夹中添加以下内容

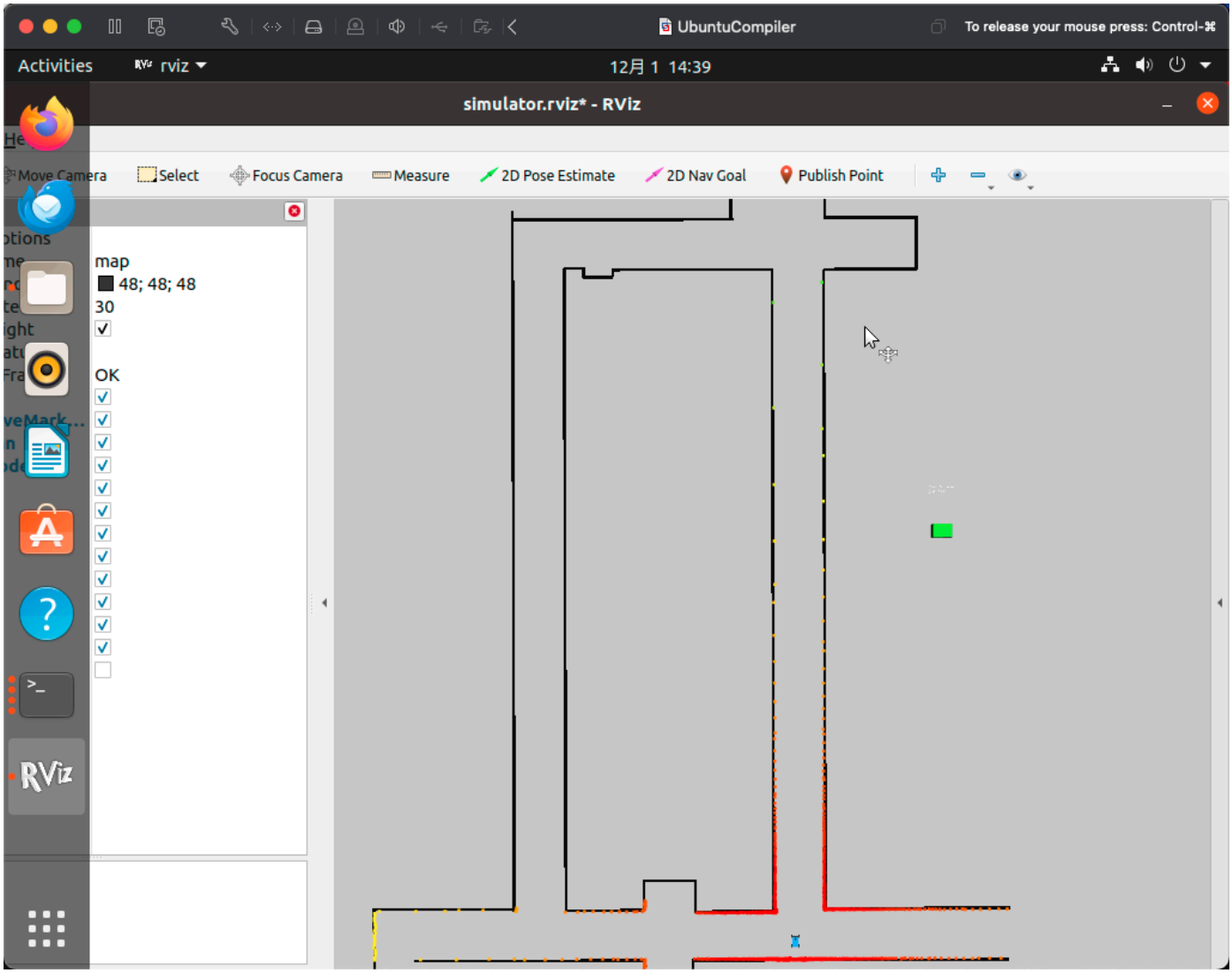
```
<!-- Launch the Safety Node -->
<node pkg="f1tenth_simulator" name="safety_node" type="safety_node" output="screen">
  <rosparam command="load" file="$(find f1tenth_simulator)/params.yaml"/>
</node>
```

在node文件夹中导入safety_node.cpp文件并编译运行

```
catkin_make
source devel/setup.bash
roslaunch f1tenth simulator simulator.launch
```

实验结果

小车成功在即将到来的碰撞时进行制动



2. TTC模型描述

本实验实现了一个TTC模型，用于判断是否需要紧急刹车

TTC模型原理

TTC是基于两个运动物体之间的距离和相对速度的概念，通过将其关联起来，提供了一种动态评估碰撞风险的方法。在我们的实现中，我们使用了一个基于相对速度和距离的简化常量速度模型。

TTC模型C++代码

```
void scan_callback(const sensor_msgs::LaserScan::ConstPtr& scan_msg) {
  // Calculate Time-to-Collision (TTC) - a simple example using the closest point
  //double closest_distance = *min_element(scan_msg->ranges.begin(), scan_msg-
>ranges.end());
  //ROS_INFO("Closest Distance: %f", closest_distance);

  //double ttc = (closest_distance-0.2) / speed;
  auto min_range_it = std::min_element(scan_msg->ranges.begin(), scan_msg->ranges.end());
  size_t min_range_index = std::distance(scan_msg->ranges.begin(), min_range_it);

  // Calculate the angle corresponding to the minimum range
```

```
  double angle = scan_msg->angle_min + min_range_index * scan_msg->angle_increment;

  // Calculate Time-to-Collision (TTC) considering relative motion direction and angle
  double closest_distance = *min_range_it;
  ROS_INFO("Closest Distance: %f, Angle: %f, abs cos value: %f",
closest_distance,angle,std::abs(cos(angle)));

  // Calculate relative speed based on the sign of the velocity
  double relative_speed = (speed >= 0) ? speed : -speed;

  // Calculate TTC with adjusted formula
  // 0.25:the distance between jiguang and the body of the car
  double ttc = (closest_distance) / (relative_speed * std::abs(cos(angle)));
  ROS_INFO("relative_speed: %f, ttc : %f",relative_speed,ttc);

  // Publish drive/brake message based on TTC
  if (ttc < 0.6) {
    // If TTC is less than 1 second, apply emergency brake
    publishEmergencyBrake();
  } else {
    // If TTC is greater than or equal to 1 second, no emergency action needed
    publishNoEmergency();
  }
}
```

TTC计算公式

```
double TTC = obs_dist / std::max(0.0, (speed * cos(i * scan_msg->angle_increment -
3.14159)));
```

TTC的计算公式为障碍物距离（`obs_dist`）除以车辆与障碍物之间相对速度的水平分量。相对速度的水平分量由车辆的速度（`speed`）乘以障碍物与车辆之间的夹角的余弦值（`cos(i * scan_msg- \>angle_increment - 3.14159)`）得到。

TTC实现细节

在TTC计算中，我们假设了车辆的反应时间以及驾驶人的反应时间等和为0.6秒,当ttc<0.6秒时我们就让小车进行紧急制动.

我们将TTC模型集成到ROS中. 具体的实现细节涉及订阅来自激光雷达和车辆里程计的数据，计算TTC，包括以下流程:

1. 根据激光雷达获得min_distance
2. 获得与最近点之间的夹角
3. 计算小车速度在 到最近点方向的投影上的 速度
4. 根据min_distance和投影速度计算ttc

并根据计算结果发布相应的制动指令。
若ttc<react_time: 紧急制动(发布速度为0,并将紧急制动的标志改为true)

3. 亮点

引入了横向速度分量的计算，通过计算速度在垂直于运动方向的分量，考虑横向速度对TTC的影响。

计算投影速度,防止小车贴墙行驶这种情景下小车启动制动的发生.