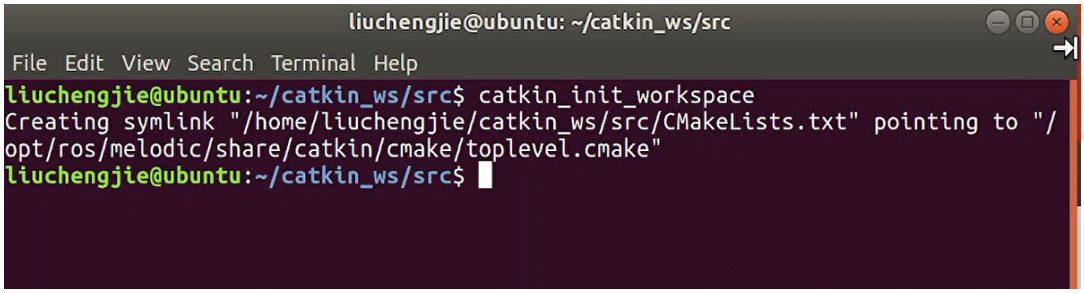
ROS-PID 实验

201250125 刘承杰

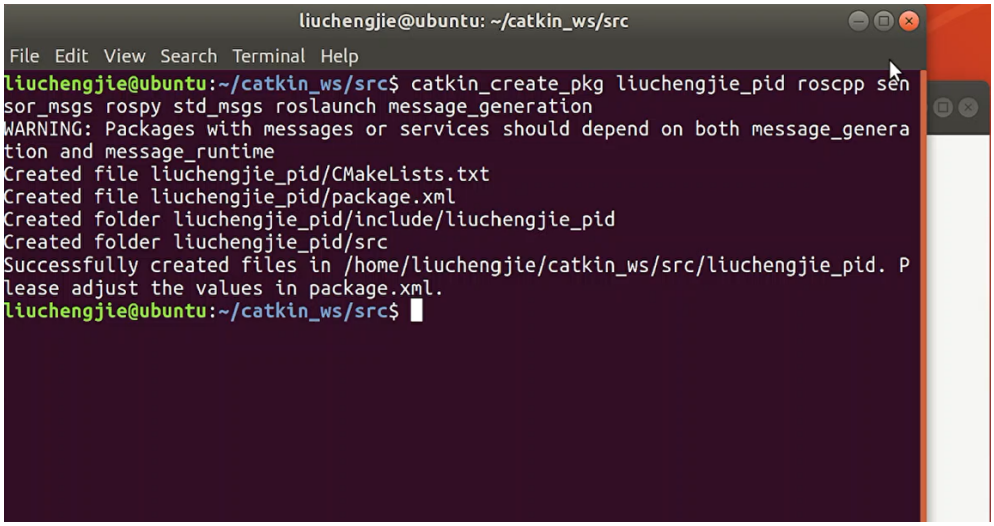
代码参考：https://github.com/Khoo395/F1Tenth-Labs-

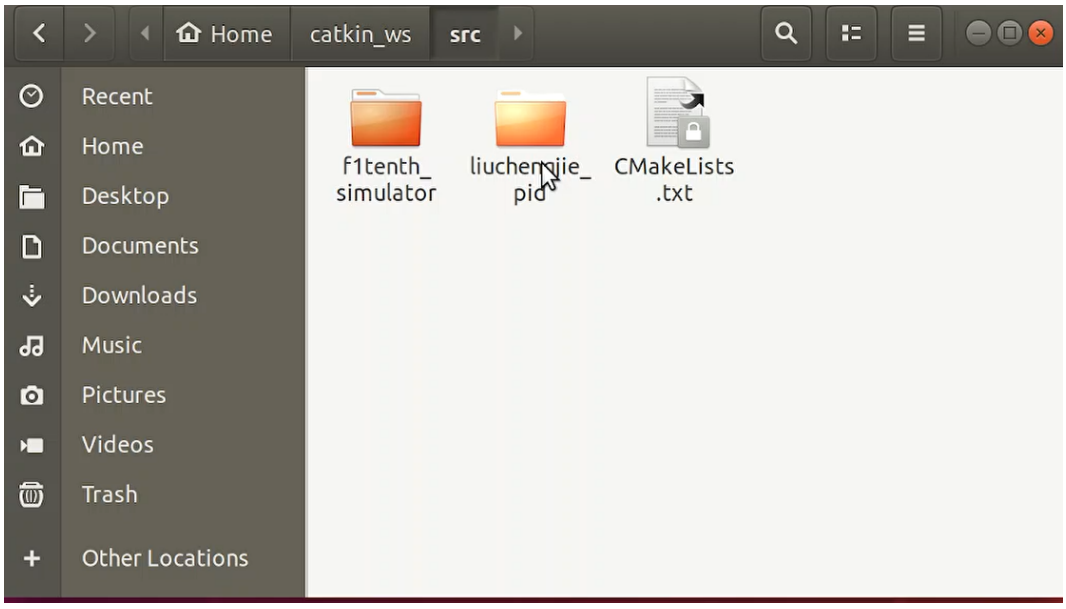
一、基本功能实现

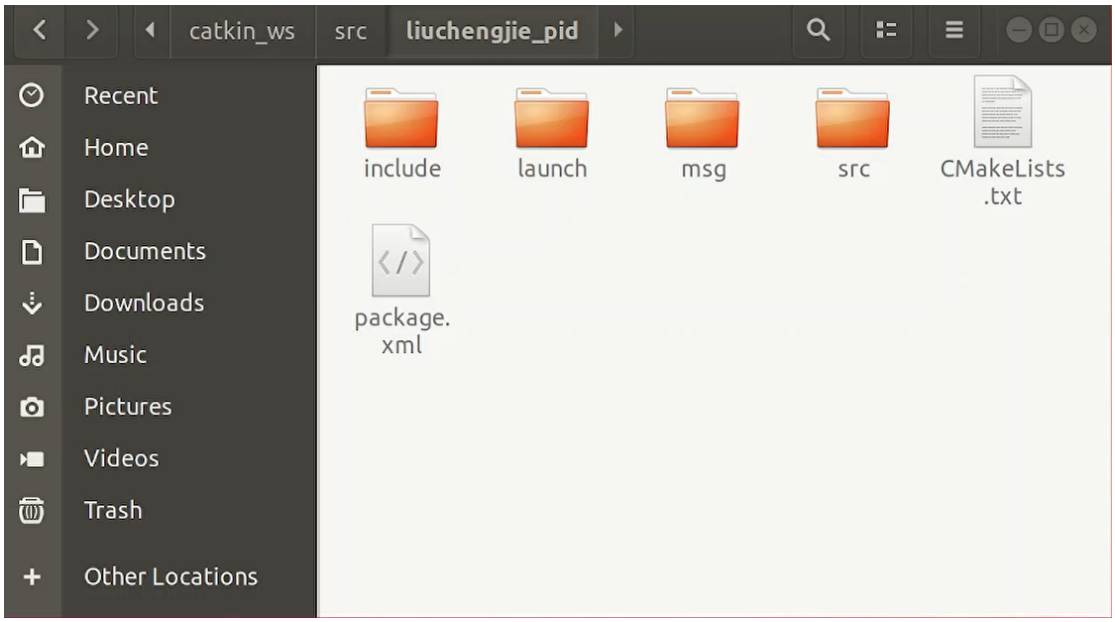
1. init workspace



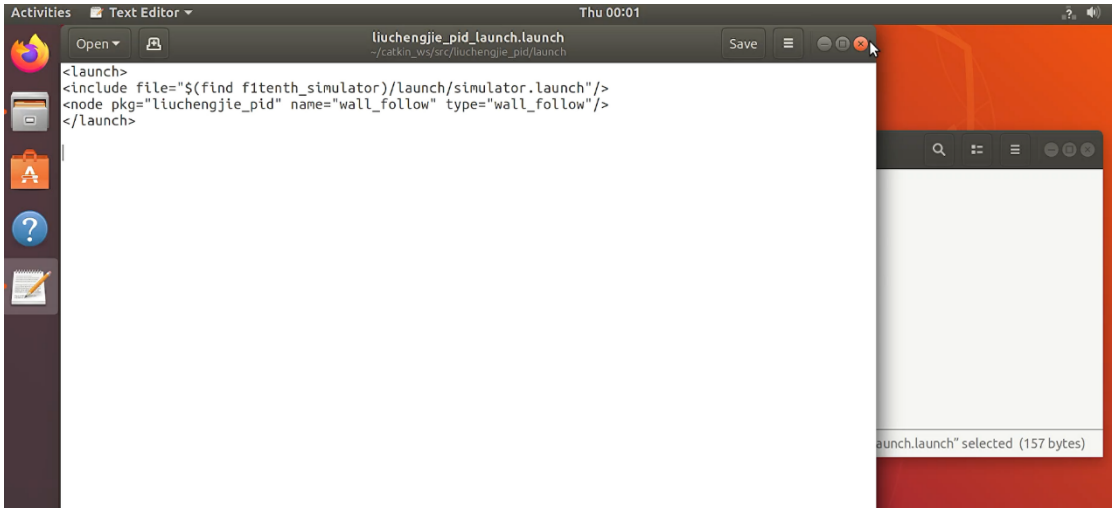
2. 下载 f1tenth 并生成包 liuchengjie_pid



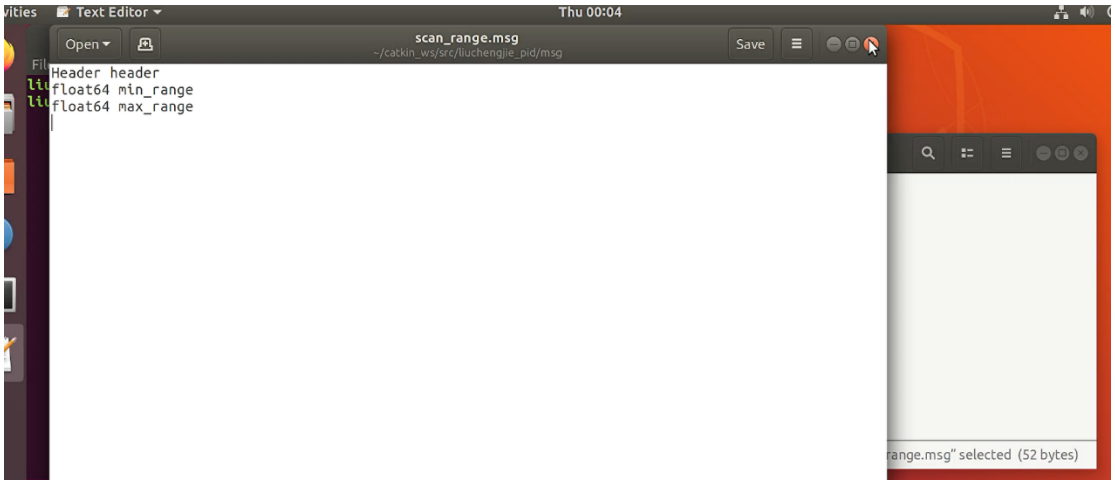




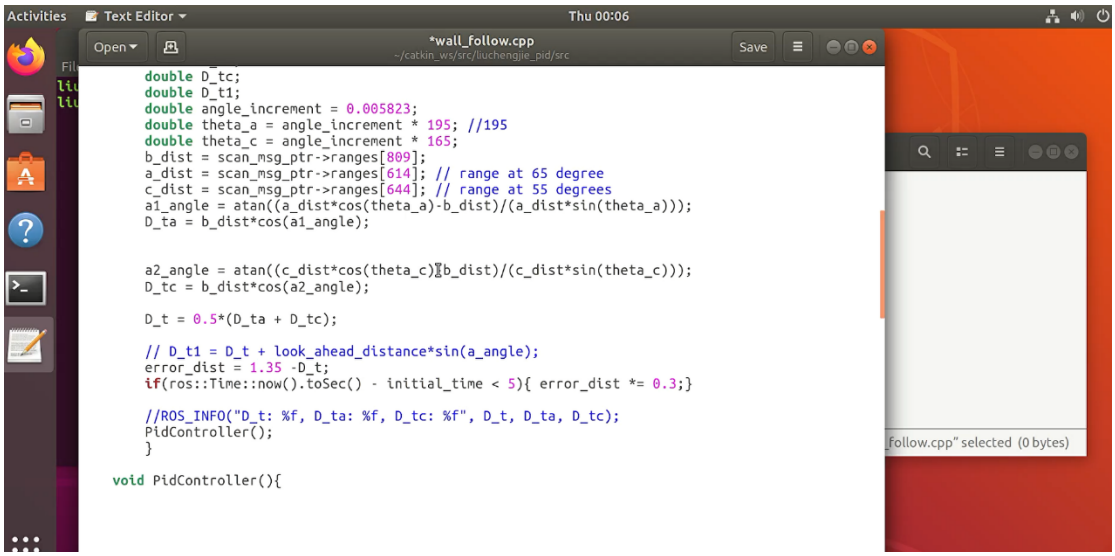
3. 新建 launch/liuchengjie_pid_launch.launch



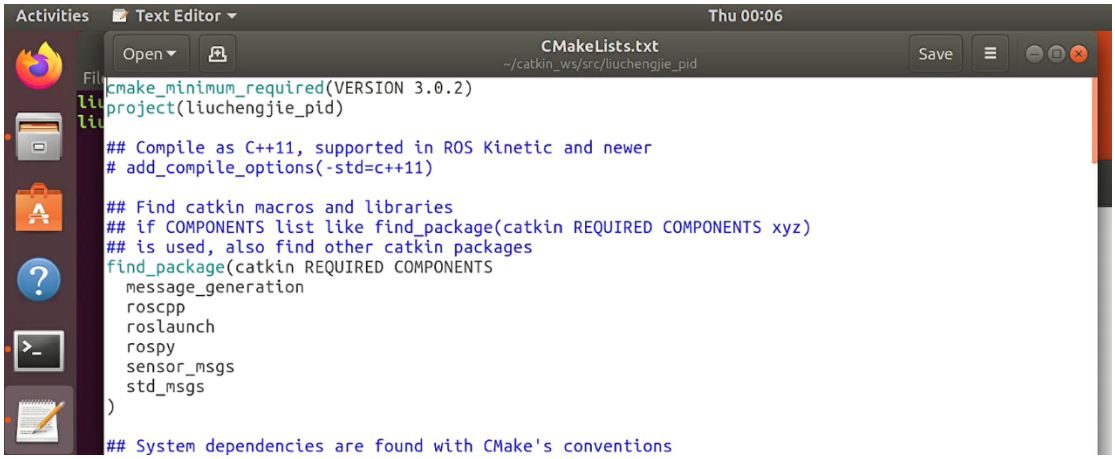
4. 新建 msg/scan_range.msg



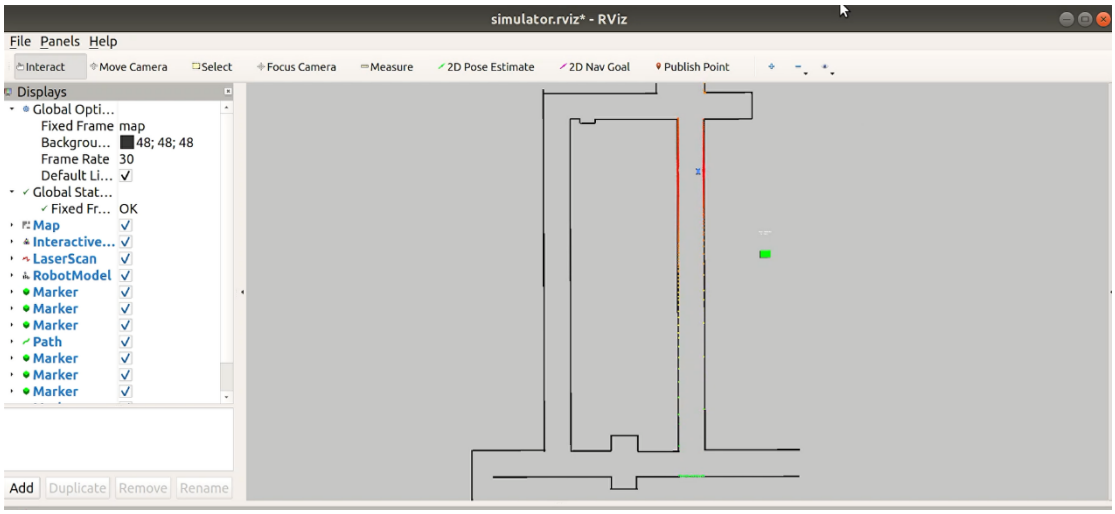
5. 导入 wall_follow.cpp



6. 更改 CMakelists.txt



7. 编译运行（具体效果详解压缩包内的 original.mkv）



二、参数调整

1. 激进的速度调度策略

原来的速度调度策略如图一，这就导致 original.mkv 中小车运行速度较慢，更改后的速度策略如图二，代码见 src/wall_follow_with_intensive_speed_control.cpp。更改后的视频在压缩包内的 advanced1.mkv 中。

```
    if(current_time_reading - initial_time < 5){
        d_term = 0; // To cancel the inital volatile behaviour of d_term
    }
    ROS_INFO("d_error: %f", d_term);
    }
    steer_angle = -K_P * error_dist + K_I * integral + K_D * d_term;
    if(steer_angle> M_PI){steer_angle = M_PI - 0.02; }
    if(steer_angle< -M_PI){steer_angle = -M_PI+ 0.02; }
    //ROS_INFO("dE_dt: %f ", d_term);
    if(abs(steer_angle) < 0.1745){speed = 1.5;}
    else if(abs(steer_angle) > 0.1745 && abs(steer_angle) < 0.349){speed = 1;}
    else if(abs(steer_angle) > 0.349){speed = 0.3;}
    ackermann.drive.speed = speed;
    ackermann.drive.steering_angle = steer_angle;
    ackermann.header.frame_id = "laser";
    ackermann.header.stamp = ros::Time::now();
    AckermannPub.publish(ackermann);
}
```

图一

```
if(current_time_reading - initial_time < 5){
    d_term = 0; // To cancel the inital volatile behaviour of d_term
}
ROS_INFO("d_error: %f", d_term);
}
steer_angle = -K_P * error_dist + K_I * integral + K_D * d_term;
if(steer_angle> M_PI){steer_angle = M_PI - 0.02; }
if(steer_angle< -M_PI){steer_angle = -M_PI+ 0.02; }
//ROS_INFO("dE_dt: %f ", d_term);
if(abs(steer_angle) < 0.15){speed = 4.5;}
else if(abs(steer_angle) > 0.15 && abs(steer_angle) < 0.25){speed = 3;}
else if(abs(steer_angle) > 0.25 && abs(steer_angle) < 0.35){speed = 1.5;}
else if(abs(steer_angle) > 0.35){speed = 0.3;}
ackermann.drive.speed = speed;
ackermann.drive.steering_angle = steer_angle;
ackermann.header.frame_id = "laser";
ackermann.header.stamp = ros::Time::now();
AckermannPub.publish(ackermann);
```

图二

修改后的小车在速度方面有显著提升——运行半圈（至左下角）的时间由原来的 46s 减少为 29s，不足指出就是有时与墙的距离过近，容易撞墙。

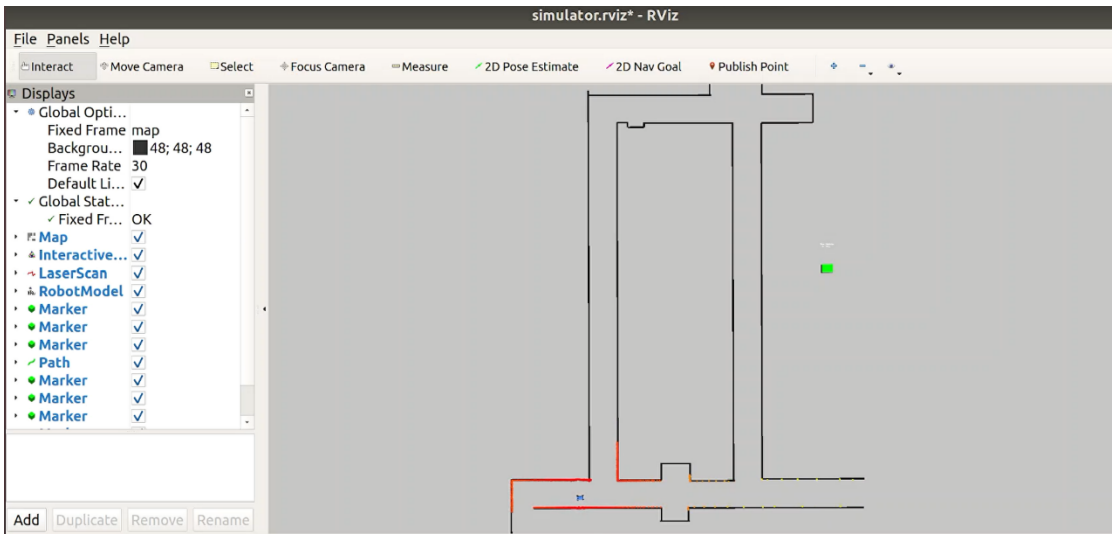
2. 敏捷反应调度策略

为了弥补 1 中的缺陷，我们在保证速度的情况下添加了多个角度——速度判断。同时，我们也调整了角度的调整幅度，让小车及时做出反应，更改后的速度策略如图三，代码见 src/wall_follow_with_quick_reaction.cpp。更改后的视频在压缩包内的 advanced2.mkv 中。

```
if(current_time_reading - initial_time < 5){
    d_term = 0; // To cancel the inital volatile behaviour of d_term
}
ROS_INFO("d_error: %f", d_term);
}
steer_angle = -K_P * error_dist + K_I * integral + K_D * d_term;
if(steer_angle> M_PI){steer_angle = M_PI - 0.03; }
if(steer_angle< -M_PI){steer_angle = -M_PI+ 0.03; }
//ROS_INFO("dE_dt: %f ", d_term);
if(abs(steer_angle) < 0.05){speed = 5;}
else if(abs(steer_angle) > 0.05 && abs(steer_angle) < 0.1){speed = 3;}
else if(abs(steer_angle) > 0.1 && abs(steer_angle) < 0.15){speed = 2.5;}
else if(abs(steer_angle) > 0.15 && abs(steer_angle) < 0.2){speed = 2;}
else if(abs(steer_angle) > 0.2 && abs(steer_angle) < 0.25){speed = 1.5;}
else if(abs(steer_angle) > 0.25 && abs(steer_angle) < 0.3){speed = 1;}
else if(abs(steer_angle) > 0.3 && abs(steer_angle) < 0.35){speed = 0.5;}
else if(abs(steer_angle) > 0.35){speed = 0.25;}
ackermann.drive.speed = speed;
ackermann.drive.steering_angle = steer_angle;
ackermann.header.frame_id = "laser";
ackermann.header.stamp = ros::Time::now();
AckermannPub.publish(ackermann);
```

图三

修改后的小车由 29s 减少到了 27s，并且很明显小车的反应更加敏捷了，它能够在左下角做出正确的转向。



这个方向是正确的，因为之前的贴墙与现在是同一面墙。