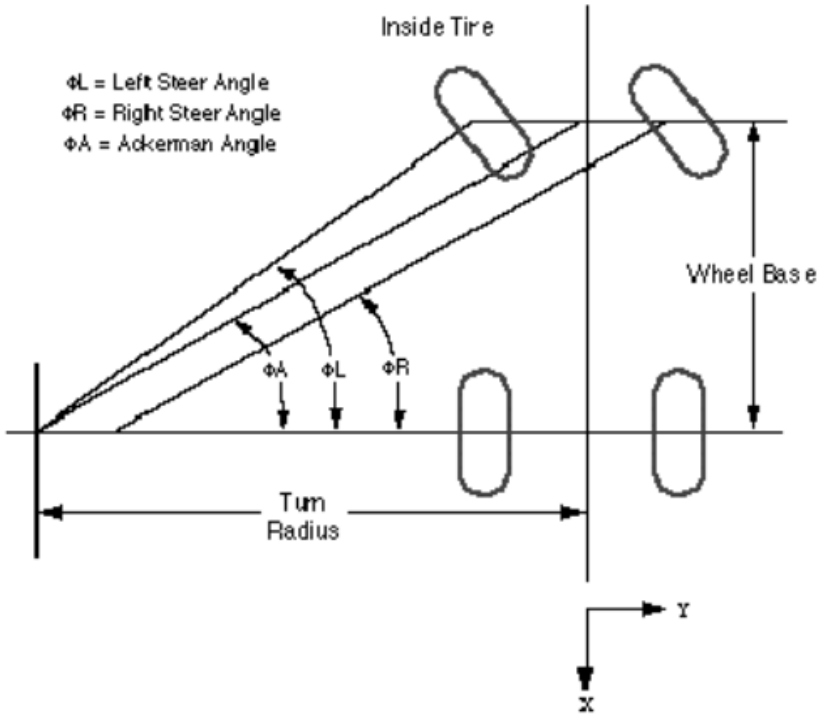
cooneo_mini仿真——01

一、原理简介

1、阿克曼转向结构

阿克曼转向是一种现代汽车的转向方式，在汽车转弯的时候，内外轮转过的角度不一样，内侧轮胎转弯半径小于外侧轮胎，下图就是理想的阿克曼转向。



这种转向方式最初是由的德国马车工程师的Georg Lankensperger 1817年提出，他的代理商Rudolph Ackerman于1818年在英国申请专利，所以从今往后这个转向原理就叫阿克曼转向几何了。

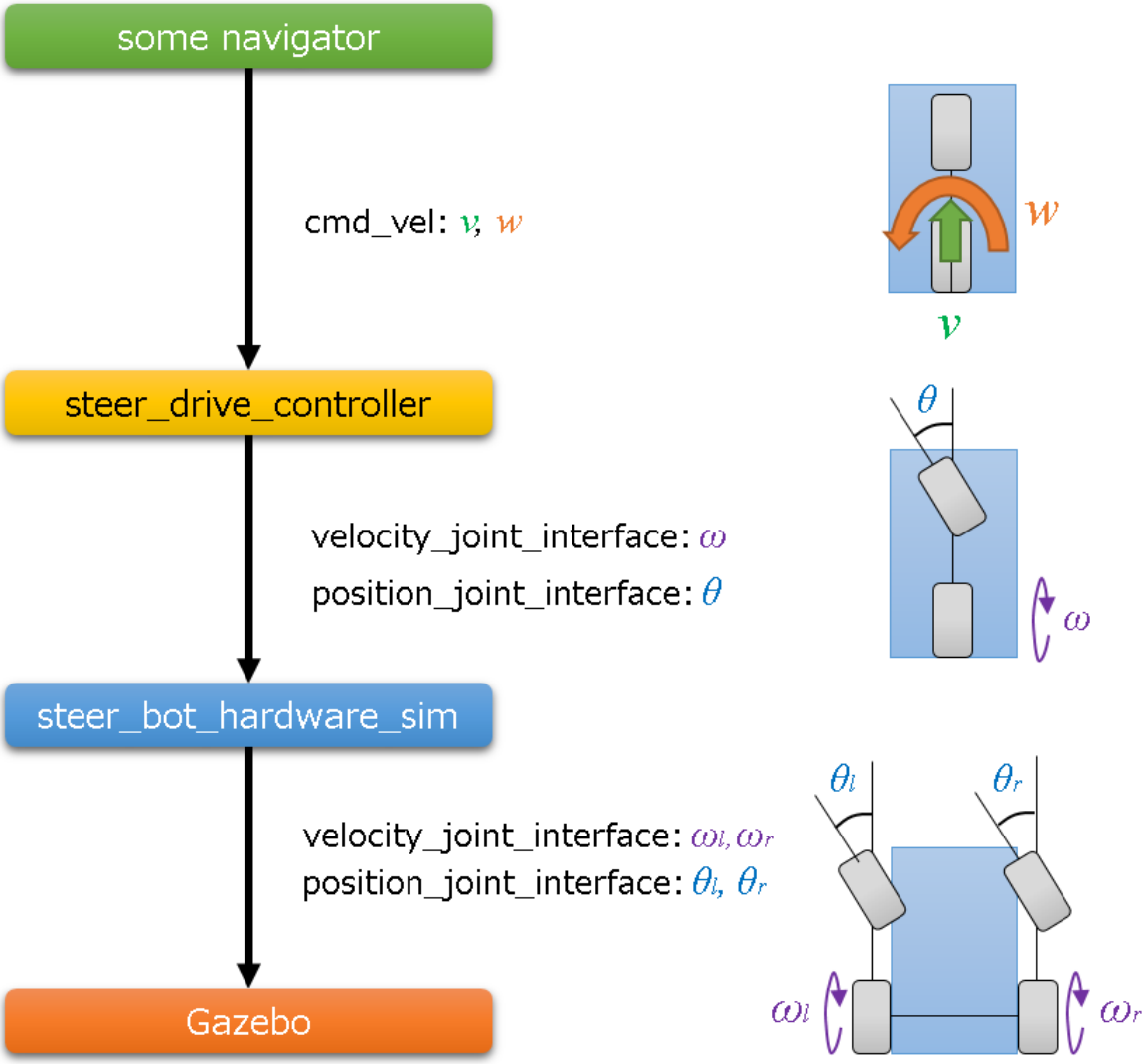
由上图所示的示意图可知，在保证车辆尽量稳定运行，不漂移、不侧移的前提下。在车辆转弯的时候，后边内侧的轮子的转速必然会小于外侧的轮子。传统汽车是在后轮中间使用了一种机械装置——差速桥，该桥是一个机械式差速装置，在汽车行进的过程中合理分配左右轮转速。

除此之外，前轮的转向角度和后轮的转速也有关系。比如：车辆以最大角度Max_angle转向时，此时后轮内侧的转速应该是**大于0**的。所以结论是：**想要满足车辆行进过程中不漂移，车速和转向角度是有关联关系的**。

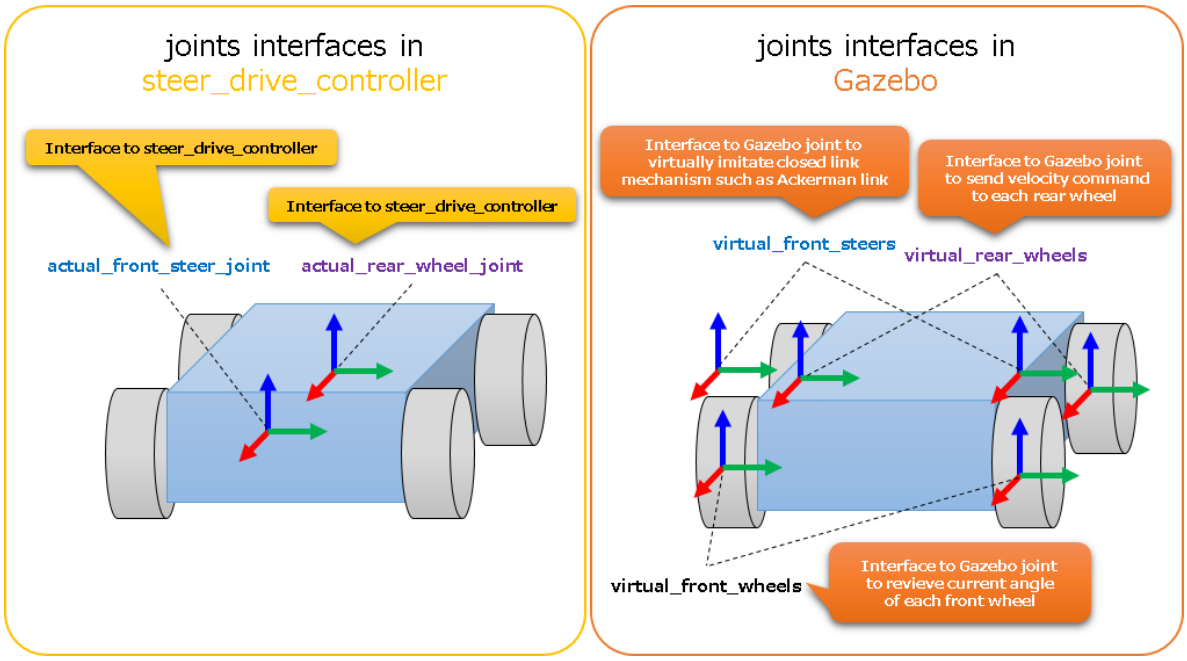
2、Gazebo仿真插件

为了便于学习和拓展，需要将该种结构的小车模型置入在Gazebo中仿真。想要在Gazebo中仿真出车体的运动效果，需要如下部件：模型文件（URDF、XACRO）、对应的仿真插件、控制命令。模型文件将在下面一节讲述，这里主要讲使用的仿真插件——**steer_bot_handware_gazebo**。

该仿真插件的功能就是接收外部发布的基于base_link(基座点)的线速度和角速度，然后根据车的尺寸，输出为前轮的转向角度和后轮的转速，示意图如下：



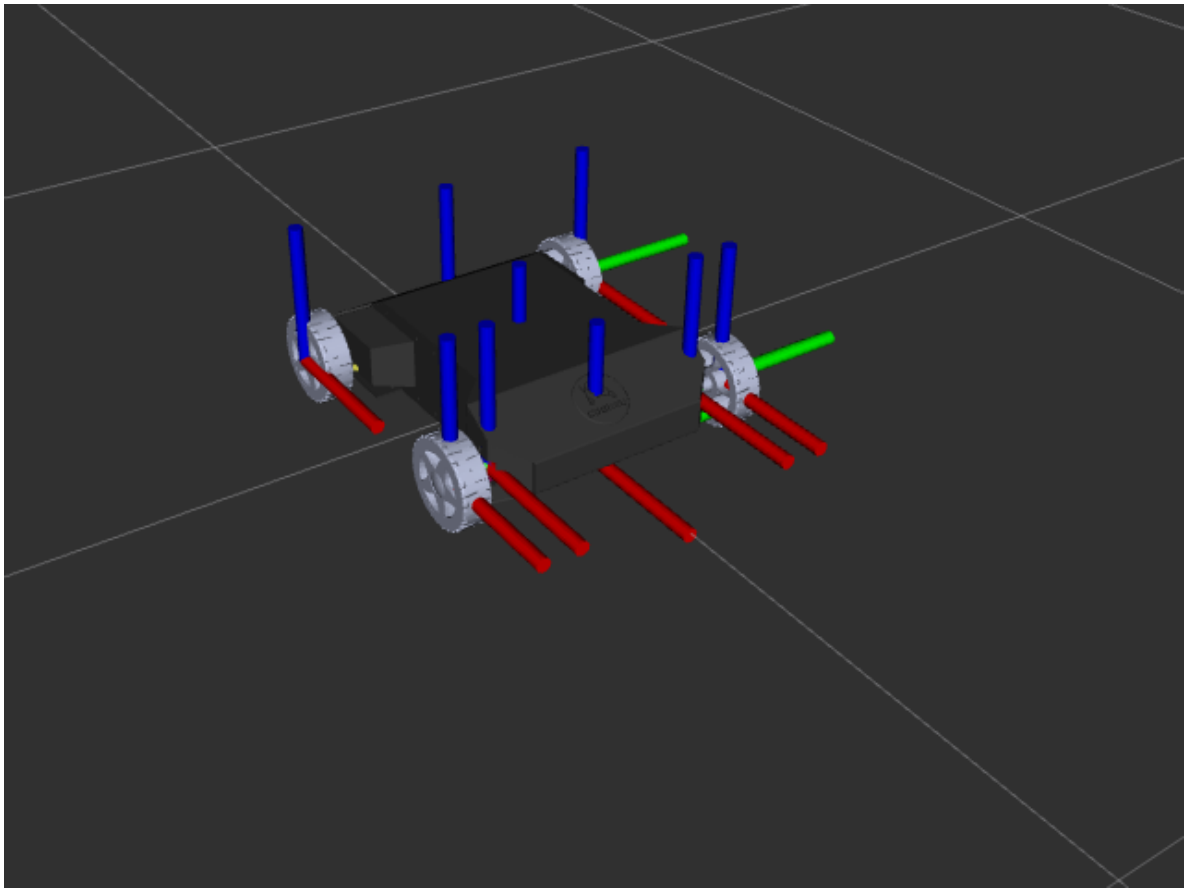
为了能够实现上述的转换，需要在模型上增加两个虚拟的点，分别是后轮中间的：rear_wheel_link/joint和前轮中间的：front_steer_link/joint。(上述的两个虚拟点的名称少了actual，不影响)官方给的示意图如下：

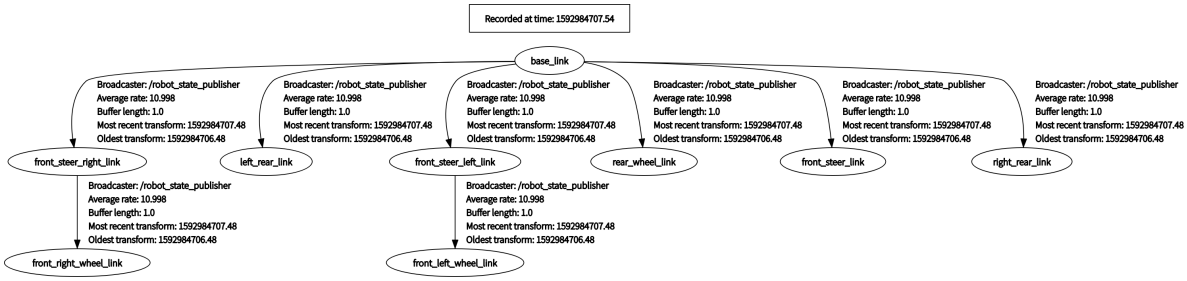


如上图右边所示，两个后轮速度关节接口用作命令接口，用于控制模型在Gazebo中线性运动。其他两个前轮关节只是用作转向。

3、cooneo_mini URDF模型文件

模型示意图与其TF转换树如下：





值得注意的是，两个前轮的”parent_link“并不是base_link，而是另外加的” front_steer_left_link“和"front_ steer_right_link"。添加这两个的目的是控制前轮转向，因为在使用urdf expoter插件时，一个关节只能绕一个轴转动，前轮除了转向以外，还需要滚动，所以添加这两个。

二、模型和仿真插件之间的适配

1、本仿真使用到的ROS package

cooneo_mini：由Solidworks 2018 软件和 URDF expoter 插件，在原始三维设计图上导出的。导出时需要注意不同Link之间所包含的实体，在solidworks中的命名应该不同，其次，导出模型的过程中应该让计算机有充足的动态内存容量。否则，导出的模型容易错位。

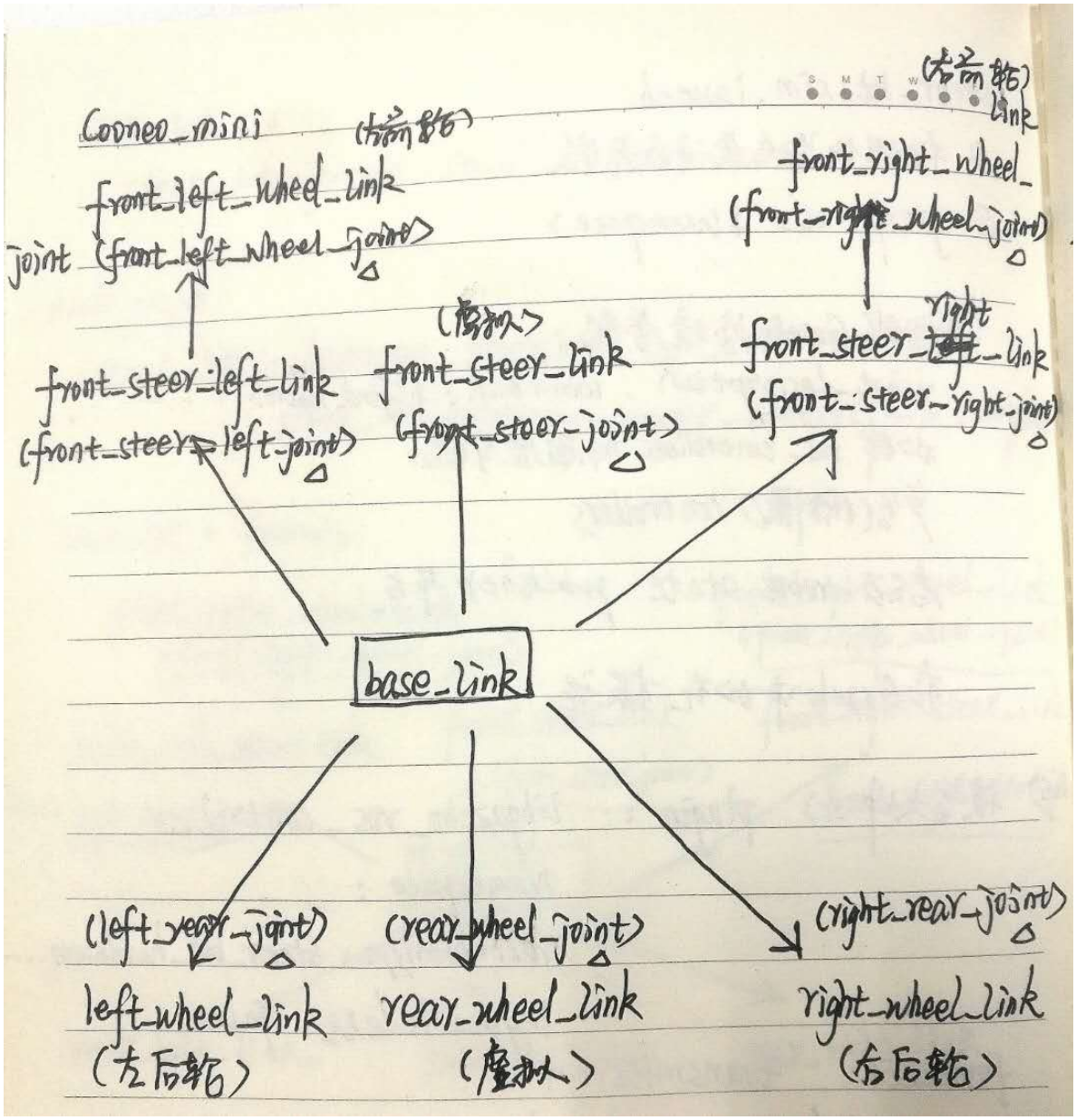
steer_mini_gazebo: 自己创建的用于存放仿真过程中所需的配置文件和启动文件，分别在” config“和”launch“文件夹下。

steer_drive_ros：使用的阿克曼仿真插件

2、给关节添加动力标签

给cooneo_mini中的原始urdf文件添加关节驱动标签，这样，关节才能在gazebo中动起来。cooneo_mini的各个Link之间的连接关系示意图如下：

带三角符号标识的表示Link之间连接的关节名称



由上图可知，需要给原始最初的模型文件cooneo_mini_rviz.urdf中的部分关节添加动力标签，内容如下：

```
<robot>
    ......

  <!--  add trans plugin to front steer_joint      -->
  <transmission name="front_steer_joint_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <actuator name="front_steer_joint_motor">
      <mechanicalReduction>1</mechanicalReduction>
```

```
 <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface
>
    </actuator>
    <joint name="front_steer_joint">

 <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface
>
    </joint>
  </transmission>
  <!--**********************************************************************-->

  <!-- add trans plugins to steer_joint and steer wheel joint-->
  <transmission name="front_steer_right_joint_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <actuator name="front_steer_right_joint_motor">
      <mechanicalReduction>1</mechanicalReduction>

 <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface
>
    </actuator>
    <joint name="front_steer_right_joint">

 <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface
>
    </joint>
  </transmission>

  <transmission name="front_right_wheel_joint_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <actuator name="front_right_wheel_joint_motor">
      <mechanicalReduction>1</mechanicalReduction>

 <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface
>
    </actuator>
    <joint name="front_right_wheel_joint">

 <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface
>
    </joint>
  </transmission>

  <transmission name="front_steer_left_joint_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <actuator name="front_steer_left_joint_motor">
      <mechanicalReduction>1</mechanicalReduction>

 <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface
>
    </actuator>
    <joint name="front_steer_left_joint">

 <hardwareInterface>hardware_interface/PositionJointInterface</hardwareInterface
>
    </joint>
  </transmission>
```

```
  <transmission name="front_left_wheel_joint_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <actuator name="front_left_wheel_joint_motor">
      <mechanicalReduction>1</mechanicalReduction>

 <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface
>
    </actuator>
    <joint name="front_left_wheel_joint">

 <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface
>
    </joint>
  </transmission>
  <!--*******************************************************************-->

  <!-- add trans plugin to rear wheel joint-->
  <transmission name="left_rear_joint_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <actuator name="left_rear_joint_motor">
      <mechanicalReduction>1</mechanicalReduction>

 <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface
>
    </actuator>
    <joint name="left_rear_joint">

 <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface
>
    </joint>
  </transmission>

  <transmission name="right_rear_joint_trans">
    <type>transmission_interface/SimpleTransmission</type>
    <actuator name="right_rear_joint_motor">
      <mechanicalReduction>1</mechanicalReduction>

 <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface
>
    </actuator>
    <joint name="right_rear_joint">

 <hardwareInterface>hardware_interface/VelocityJointInterface</hardwareInterface
>
    </joint>
  </transmission>
  <!--*******************************************************************-->

  <!--  add gazebo_ros_control plugin -->
  <gazebo>
    <plugin name="gazebo_ros_control" filename="libgazebo_ros_control.so">
      <robotNamespace>/</robotNamespace>

 <robotSimType>steer_bot_hardware_gazebo/SteerBotHardwareGazebo</robotSimType>
    </plugin>
  </gazebo>
  <!--*******************************************************************-->
```

```
    ......

</robot>
```

解释：

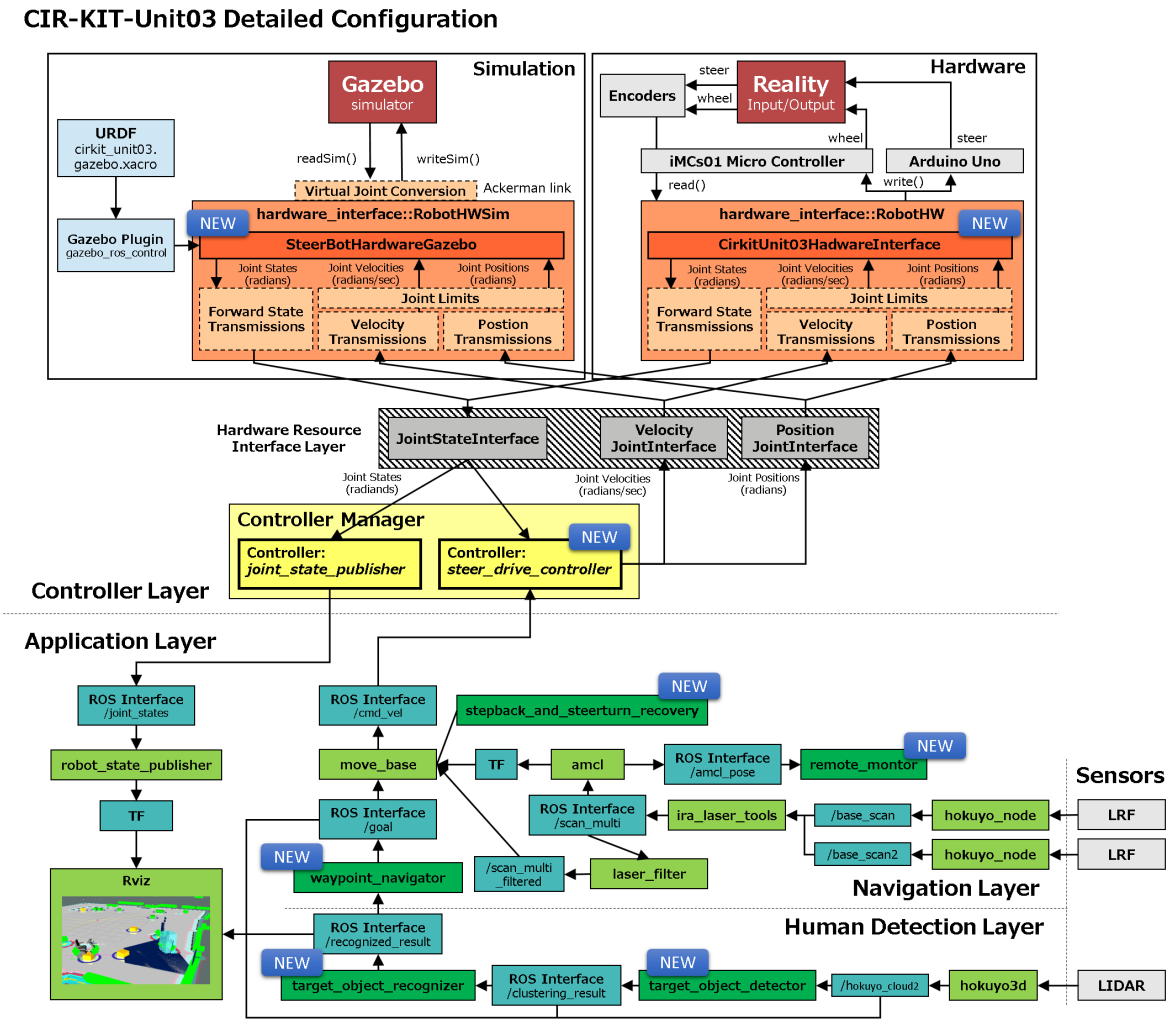
上面的动力标签有两种类型分别如下：

hardware_interface / VelocityJointInterface: 转速的物理仿真插件，四个轮子的类型。

hardware_interface /PositionJointInterface: 转向的物理仿真插件，前轮转向节的类型。

如上所述，给相应的关节添加相应的转动标签，然后相应的关节就能在Gazebo中动起来了，这样才能在仿真插件的控制下，实现Gazebo中仿真运动。

细心的你会发现，最后还添加了一个标签，该标签是一个插件，主要的作用就是连通Gazebo和ROS，是它俩的中间件，缺少它，就不能接收ROS下的控制指令。他们之间的关系可如下图：



3、修改仿真插件的配置文件

经过前面的步骤，我们的模型已经具备在Gazebo中运动的前提条件，现在，我们将利用仿真插件驱动模型在Gazebo中运动。先查看启动的launch文件(/steer_mini_gazebo/launch/steer_cooneo_mini_sim.launch)，知道需要加载那些配置文件：

```
<?xml version="1.0"?>
<launch>
    <!-- setting the start position of the model -->
```

```
    <arg name="x" default="0.0"/>
    <arg name="y" default="0.0"/>
    <arg name="z" default="0.0" />
    <arg name="roll" default="0.0"/>
    <arg name="pitch" default="0.0"/>
    <arg name="yaw" default="0.0"/>

    <!-- load environment into Gazebo  -->
    <include file="$(find gazebo_ros)/launch/empty_world.launch" >
        <arg name="world_name" value="$(find
cooneo_mini)/worlds/cooneo_office.world"/>
    </include>

    <!-- Load the robot description -->
    <param name="robot_description" command=" cat $(find
cooneo_mini)/urdf/cooneo_mini.urdf"/>

    <!-- Load ros_controllers configuration parameters -->
    <rosparam file="$(find
steer_mini_gazebo)/config/ctrl_ackermann_steering_controller_mini.yaml"
command="load"  />
    <rosparam file="$(find steer_mini_gazebo)/config/ctrl_gains_mini.yaml"
command="load" />
    <rosparam file="$(find
steer_mini_gazebo)/config/ctrl_joint_state_publisher_mini.yaml" command="load"
 />
    <rosparam file="$(find
steer_mini_gazebo)/config/ctrl_steer_bot_hardware_gazebo_mini.yaml"
command="load" />

    <!-- Spawn the controllers -->
    <node pkg="controller_manager" type="spawner" name="controller_spawner"
        args="joint_state_publisher ackermann_steering_controller"
        output="screen" respawn="false" >
    </node>

    <!-- Launch  the robot state publisher -->
    <node name="robot_state_publisher" pkg="robot_state_publisher"
type="robot_state_publisher">
        <param name="publish_frequency" value="50.0"/>
    </node>

    <!-- Launch a rqt steering GUI for publishing to
steer_drive_controller/cmd_vel -->
    <node pkg="rqt_robot_steering" type="rqt_robot_steering"
name="rqt_robot_steering" >
        <param name="default_topic"
value="ackermann_steering_controller/cmd_vel"/>
    </node>



    <!-- Spawn robot in Gazebo -->
    <node name="spawn_vehicle" pkg="gazebo_ros" type="spawn_model" args="-urdf -
param robot_description -model cooneo_mini -gazebo_namespace /gazebo
              -x $(arg x) -y $(arg y) -z $(arg z)
              -R $(arg roll) -P $(arg pitch) -Y $(arg yaw)"
          respawn="false" output="screen" />
```

```
</launch>
```

由上可见，涉及到的配置文件如下几个：

```
ctrl_ackermann_steering_controller_mini.yaml   # 001
ctrl_gains_mini.yaml                           # 002
ctrl_joint_state_publisher_mini.yaml           # 003
ctrl_steer_bot_hardware_gazebo_mini.yaml       # 004
```

下面将分别讲述：

(1)ctrl_ackermann_steering_controller_mini.yaml

```
# Configuration for the ackermann_steering_controller.
ackermann_steering_controller:
    type: 'ackermann_steering_controller/AckermannSteeringController'

    # Odometry related
    publish_rate: 50                    # 里程计发布频率 default: 1.0
    open_loop: false

    # Joints
    rear_wheel: 'rear_wheel_joint'
    front_steer: 'front_steer_joint'

    # Geometry
    wheel_separation_h: 0.3492       #后轮与前轮的距离
    wheel_radius: 0.0625            #车轮半径

    # Odometry calibration and tuning
    wheel_separation_h_multiplier: 1.0  # 乘数应用于车轮分离参数。这用于解决机器人模型与实际机器人之间的差异（例如，里程计调整） default: 1.0
    wheel_radius_multiplier: 1.0        # 乘数应用于车轮半径参数。这用于解决机器人模型与实际机器人之间的差异（例如，里程计调整）。 default: 1.0
    steer_pos_multiplier: 1.0           # 转向位置角度倍增器可进行微调。 default: 1.0

    # Odometry covariances for the encoder output of the robot.
    pose_covariance_diagonal: [0.001, 0.001, 0.001, 0.001, 0.001, 0.03]
    twist_covariance_diagonal: [0.001, 0.001, 0.001, 0.001, 0.001, 0.03]

    # Top level frame (link) of the robot description
    base_frame_id: 'base_link'

    # Transform from odom -> base_link
    enable_odom_tf: true              #是否直接发布odom --> base_link的tf转换
    odom_frame_id: '/odom'

    # Set to false if the motor driver provides velocity data.
    estimate_velocity_from_position: true

    # Commands
    publish_cmd: true
    allow_multiple_cmd_vel_publishers: false

    # Velocity and acceleration limits for the robot
```

```
    linear:
        x:
            has_velocity_limits    : true
            max_velocity           : 10.0   # m/s
            has_acceleration_limits: true
            max_acceleration       : 2.0    # m/s^2
    angular:
        z:
            has_velocity_limits    : true
            max_velocity           : 3.0   # rad/s
            has_acceleration_limits: true
            max_acceleration       : 3.0    # rad/s^2

    # Other (undocumented but in source code)
    # velocity_rolling_window_size: 10
    # cmd_vel_timeout: 0.5               #连续两条速度命令之间允许的时间间隔，超过间隔后会立即停车

    # Deprecated...
    # publish_wheel_joint_controller_state: false
```

(2)ctrl_gains_mini.yaml

```
# PID gains for the steer_bot_hardware_gazebo plugin.
gains:
    right_rear_joint:
        p: 100000.0
        d: 10.0
        i: 0.50
        i_clamp: 3.0
    left_rear_joint:
        p: 100000.0
        d: 10.0
        i: 0.50
        i_clamp: 3.0
```

(3)ctrl_joint_state_publisher_mini.yaml

```
# Configuration for the ros_controllers joint_state_controller.
joint_state_publisher:
    type: "joint_state_controller/JointStateController"
    publish_rate: 50
```

(4)ctrl_steer_bot_hardware_gazebo_mini.yaml

```
# Configuration for the steer_bot_hardware_gazebo plugin.
steer_bot_hardware_gazebo:
  rear_wheel  : 'rear_wheel_joint'        #后轮关节名称，用于接收来自
steer_drive_controller的线性驱动命令。
  front_steer : 'front_steer_joint'       #前转向接头名称，用于从
steer_drive_controller接收角驱动命令。

  virtual_rear_wheels: ['right_rear_joint', 'left_rear_joint']                #虚
拟后轮接头名称，用于向Gazebo的对应的轮接头发送速度指令。
  virtual_front_wheels: ['front_right_wheel_joint','front_left_wheel_joint']  #虚
拟前轮接头名称以掌握Gazebo相应轮接头的当前角度。
  virtual_front_steers: ['front_steer_right_joint', 'front_steer_left_joint'] #虚
拟前转向关节名称，用于向Gazebo中的相应转向关节发送位置命令。

  enable_ackermann_link : true            #如果为true，则启用虚拟ackerman链接机制。否
则，将启用paralell链接机制。
  wheel_separation_h : 0.4035             #左右两侧的车轮间隔长度，用于计算阿克曼连杆机构
的转向角。
  wheel_separation_w : 0.3492             #前后轮距，用于计算阿克曼连杆机构的转向角。
```

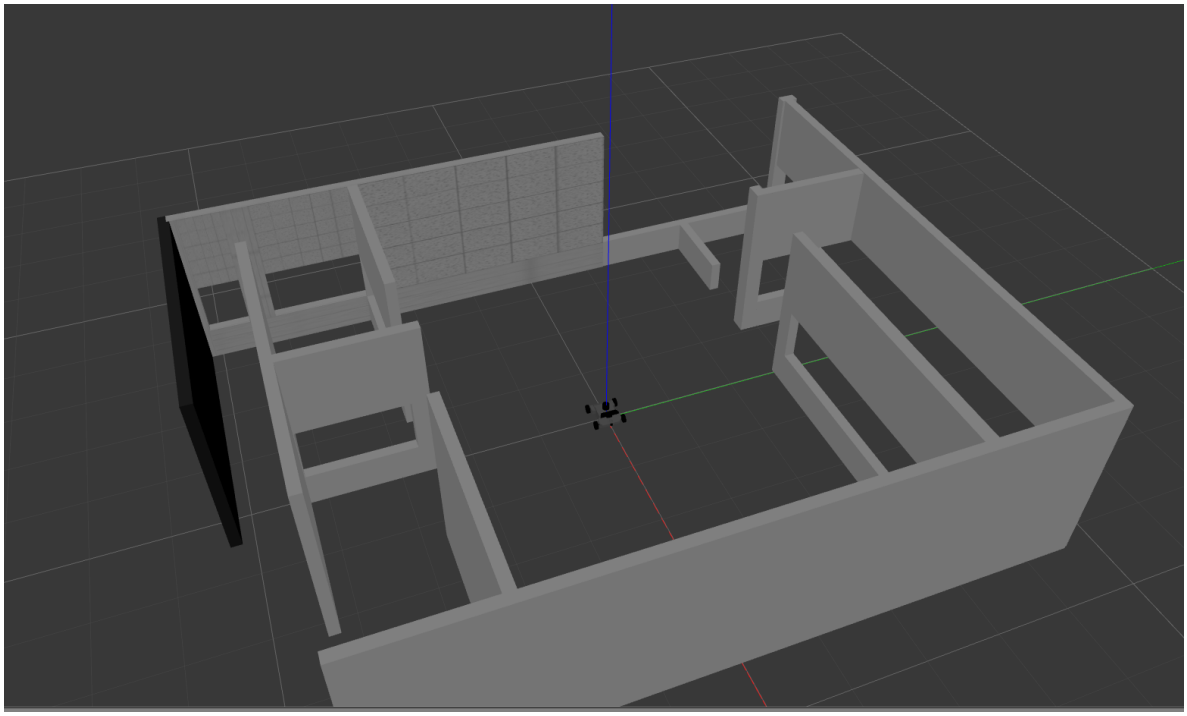
三、实际运行效果展示

1、将pkg中的功能放入自己的工作空间中编译，source。

```
# 复制 cooneo_mini  steer_drive_ros  steer_mini_gazebo 进入自己的工作空间/src下，再退
出到根目录下
catkin_make
source devel/setup.bash
```

2、运行

```
roslaunch steer_mini_gazebo steer_cooneo_mini_sim.launch
```



3、rostopic list 运行结果是：

```
/ackermann_steering_controller/cmd_vel
/ackermann_steering_controller/odom
/clock
/gains/left_rear_joint/parameter_descriptions
/gains/left_rear_joint/parameter_updates
/gains/right_rear_joint/parameter_descriptions
/gains/right_rear_joint/parameter_updates
/gazebo/link_states
/gazebo/model_states
/gazebo/parameter_descriptions
/gazebo/parameter_updates
/gazebo/set_link_state
/gazebo/set_model_state
/gazebo_gui/parameter_descriptions
/gazebo_gui/parameter_updates
/joint_states
/rosout
/rosout_agg
/tf
/tf_static
```

由此可知，即可接收：”**/ackermann_steering_controller/cmd_vel**” 话题数据，也可以发布里程计信息：

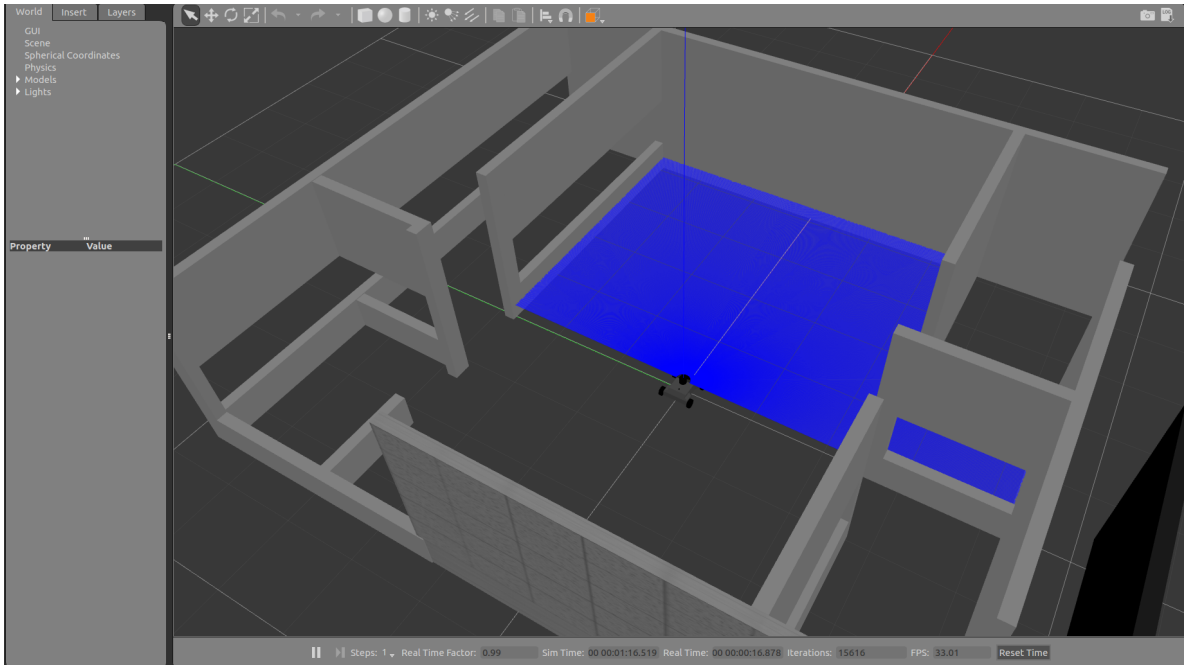
“**/ackermann_steering_controller/odom**”。

四、拓展

详情见cooneo_mini仿真——02章节

1、添加传感器标签，可以仿真传感器：

```
cd <catkin_workspace>
source devel/setup.bash
roslaunch steer_mini_gazebo steer_cooneo_mini_sensors_sim.launch
```



2、在1的基础上使用算法，建立栅格地图：

```
cd <catkin_workspace>
source devel/setup.bash
roslaunch steer_mini_gazebo gmapping_steer_cooneo_mini_sensors_sim.launch
```