Robot URDF model

Note: The virtual machine needs to be in the same LAN as the car, and the ROS_DOMAIN_ID needs to be the same. You can check [Must Read Before Use] to set the IP and ROS_DOMAIN_ID on the board.

1. Program Function Description

The car connects to the proxy, runs the program, and the URDF model will be displayed in rviz.

2. Start the program

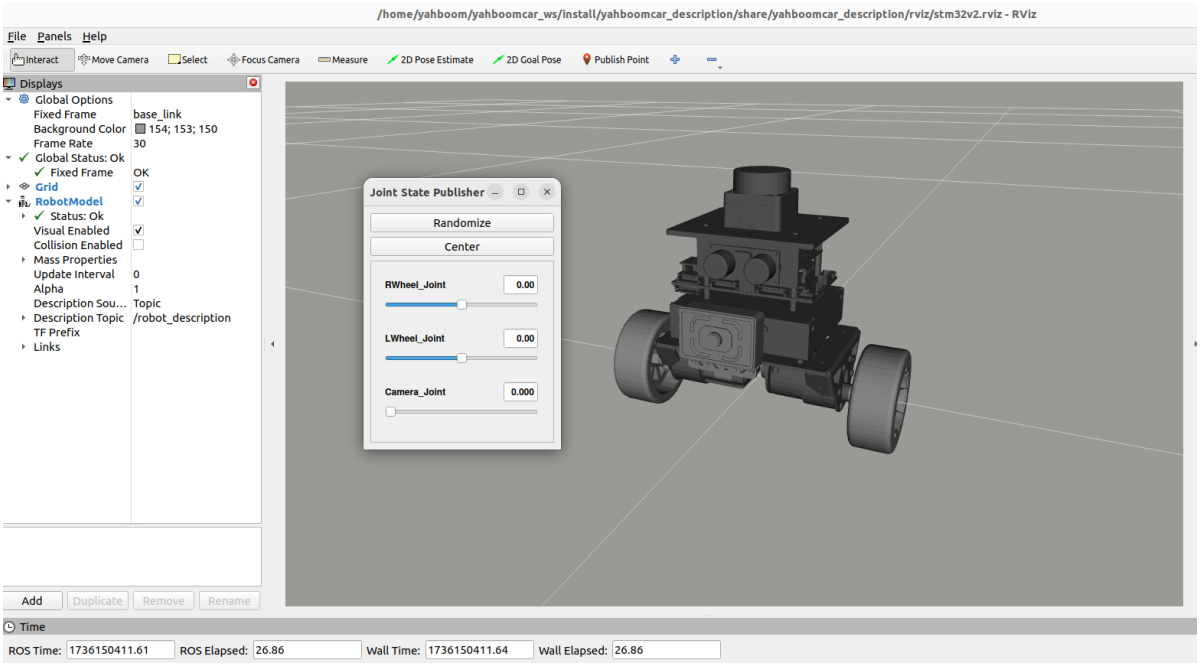
2.1 Run the command

Take the matching virtual machine as an example, load URDF and generate a simulation controller, terminal input,

```
ros2 launch yahboomcar_description display_launch.py
```

Wait for rviz to open,

Then, use the mouse to adjust the viewing angle, slide the simulation controller just generated, and you can see the tires/camera of the car changing,



RWheel_Joint: Control the left wheel

LWheel_Joint: Control the right wheel

Camera_Joint: Control the camera bracket

Randomize: Randomly publish values to each joint

Center: All Joints return to the center

3. Code analysis

Code location (taking the virtual machine as an example),

```
/home/yahboom/yahboomcar_ws/src/yahboomcar_description/launch
```

display_launch.py

```
from ament_index_python.packages import get_package_share_path

from launch import LaunchDescription
from launch.actions import DeclareLaunchArgument
from launch.substitutions import Command, LaunchConfiguration

from launch_ros.actions import Node
from launch_ros.parameter_descriptions import ParameterValue

def generate_launch_description():
    urdf_tutorial_path = get_package_share_path('yahboomcar_description')
    default_model_path = urdf_tutorial_path / 'urdf/STM32-V2-V1.SLDASM.urdf'

    model_arg = DeclareLaunchArgument(name='model',
default_value=str(default_model_path),
                                      description='Absolute path to robot urdf
file')
    robot_description = ParameterValue(Command(['xacro ',
LaunchConfiguration('model')]),
                                       value_type=str)

    robot_state_publisher_node = Node(
        package='robot_state_publisher',
        executable='robot_state_publisher',
        parameters=[{'robot_description': robot_description}]
    )

    joint_state_publisher_node = Node(
        package='joint_state_publisher',
        executable='joint_state_publisher'
    )


    tf_base_footprint_to_base_link = Node(
        package='tf2_ros',
        executable='static_transform_publisher',
        arguments=['0', '0', '0.0335', '0.0', '0.0', '0.0', 'base_footprint',
'base_link'],
    )

    return LaunchDescription([
        model_arg,
        joint_state_publisher_node,
        robot_state_publisher_node,
        tf_base_footprint_to_base_link
    ])
```

- model_arg: load model parameters, the loaded model is STM32-V2-V1.SLDASM.urdf, located in `/home/yahboom/yahboomcar_ws/src/yahboomcar_description/urdf`
- joint_state_publisher_gui_node: publish sensor_msgs/JointState message
- robot_state_publisher_node: robot state publishing
- tf_base_footprint_to_base_link: publish static transformation from base_footprint to base_link

4. URDF model

URDF, the full name is Unified Robot Description Format, translated into Chinese as Unified Robot Description Format, is a robot model file described in XML format, similar to D-H parameters.

```
<?xml version="1.0" encoding="utf-8"?>
```

The first line is a required field for XML and describes the version information of XML.

```
<robot
  name="STM32-V2-V1">
</robot>
```

The second line describes the current robot name; all information about the current robot is contained in the [robot] tag.

4.1. Components

- Link, connecting rod, can be imagined as a human arm
- Joint, joint, can be imagined as a human elbow joint

Relationship between link and joint: two links are connected by joints, imagine that the arm has a forearm (link) and an upper arm (link) connected by an elbow joint (joint).

4.1.1. Link

1), Introduction

In the URDF descriptive language, link is used to describe physical properties,

- Describe visual display, label.
- Describe collision properties, label.
- Describe physical inertia, label is not commonly used.

Links can also describe the size of the connecting rod (size)\color (color)\shape (shape)\inertial matrix (inertial matrix)\collision parameters (collision properties), etc. Each Link will become a coordinate system.

2), Sample code

```
<link
  name="base_link">
<inertial>
  <origin
    xyz="0.00192307327839142 0.000501949810846121 0.0484593354002855"
    rpy="0 0 0" />
  <mass
```

```
    value="0.310378730787665" />
  <inertia
    ixx="0.000163083342089287"
    ixy="-1.69941998317735E-07"
    ixz="-1.34826644634636E-07"
    iyy="9.03265342034757E-05"
    iyz="-1.10451361772956E-07"
    izz="0.000222531354148277" />
</inertial>
<visual>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://yahboomcar_description/meshes/base_link.STL" />
  </geometry>
  <material
    name="">
    <color
      rgba="0.35 0.35 0.35 1" />
  </material>
</visual>
<collision>
  <origin
    xyz="0 0 0"
    rpy="0 0 0" />
  <geometry>
    <mesh
      filename="package://yahboomcar_description/meshes/base_link.STL" />
  </geometry>
</collision>
</link>
```

3. Tag introduction

- origin: describes the pose information; the xyz attribute describes the coordinate position in the environment, and the rpy attribute describes its own posture.
- mess: describes the quality of the link.
- inertia: inertial reference system. Due to the symmetry of the rotational inertia matrix, only 6 upper triangular elements ixx, ixy, ixz, iyy, iyz, izz are required as attributes.
- geometry: the tag describes the shape; the main function of the mesh attribute is to load the texture file, and the filename attribute is the file address of the texture path
- material: the tag describes the material; the name attribute is a **required item**, which can be empty and repeated. The rgba attribute in the [color] tag is used to describe red, green, blue, and transparency, separated by spaces.

4.1.2, joints

1. Introduction

Describes the relationship between two joints, motion position and speed limits, kinematic and dynamic properties. There are several types of joints:

- fixed: fixed joints. No movement is allowed, it serves as a connection.
- continuous: revolute joint. It can rotate continuously, without rotation angle limit.
- revolute: revolute joint. Similar to continuous, with rotation angle limit.
- prismatic: sliding joint. It moves along a certain axis, with position limit.
- floating: floating joint. It has six degrees of freedom, 3T3R.
- planar: planar joint. It allows translation or rotation above the plane orthogonal to the plane.

2), sample code

```
<joint
  name="Camera_Joint"
  type="revolute">
<origin
  xyz="0.055721 0 0.024677"
  rpy="1.5708 -1.5708 0" />
<parent
  link="base_link" />
<child
  link="Camera_Link" />
<axis
  xyz="0 0 -1" />
<limit
  lower="0"
  upper="1.63"
  effort="10"
  velocity="1.0" />
</joint>
```

In the [joint] tag, the name attribute is a **required item**, describing the name of the joint, and it is unique. In the [joint] tag, the type attribute is filled in with the six major joint types.

3), Tag introduction

- origin: sub-tag, refers to the relative position of the rotation joint in the parent coordinate system.
- parent, child: parent, child sub-tags represent two links to be connected; parent is a reference object, and child rotates around the parent.
- axis: sub-tag indicates which axis (xyz) the link corresponding to the child rotates around and the amount of rotation around the fixed axis.
- limit: sub-tag is mainly used to limit the child. The lower attribute and upper attribute limit the range of rotation, the effort attribute limits the force range during the rotation process. (positive and negative value, in Newton or N), the velocity attribute limits the speed of rotation, in meters/second or m/s.
- mimic: describes the relationship between this joint and the existing joints.
- safety_controller: describes the safety controller parameters. Protect the movement of the robot joints.