robot\_gantry digital twin documentation

The gantry\_config ROS package is a fully functional digital twin of the gantry system used for neural recording purpose.It has a physic-based Gazebo model and non-physics-based Rviz model(provided by gantry\_description ROS package). Both of them have been integrated with ros\_control and moveit2 so that they are fully capable of performing simulation in real time or offline.

Config folder- All the configuration files for ros\_control and moveit2

1. Robot\_Gantry\_Gazebo.ros2\_control.xacro

This is an XML file containing a macro definition for a robot gantry that can be controlled using the Robot Operating System 2 (ROS 2) control framework. The macro takes two parameters: the name of the robot and the file path of a YAML file containing initial positions for the robot's joints.

The macro definition includes a ROS 2 control block with a system type, which specifies that this is a complete system with hardware and controllers. The hardware block contains a plugin element that specifies the name of a mock hardware component, which is used for simulation purposes only.

The robot gantry has three joints named "x\_joint", "y\_joint", and "z\_joint". Each joint has two command interfaces for controlling position and velocity, respectively, and two state interfaces for providing information about the joint's position and velocity. The state interfaces are initialized with initial positions loaded from the YAML file specified in the macro call.

Overall, this macro definition provides a template for creating a robot gantry with ROS 2 control capabilities, which can be used in simulations or on real hardware with appropriate modifications.

1. Robot\_Gantry\_Gazebo.srdf

This is an XML file that defines the structure and semantic information about a robot named "Robot\_Gantry\_Gazebo". It is not a replacement for a URDF file, which is required for defining the links and joints of the robot. Instead, this file provides additional information about the robot structure, such as groupings of joints and links, named states for groups, end effector information, virtual joints, and disabled collision pairs.

The file begins with comments that describe the purpose and use of the different elements that are defined in the file. The file defines four groups of joints, named "x\_group", "y\_group", "z\_group", and "end\_effector\_group". The "end\_effector\_group" is a subgroup that includes the other three groups, which represent the joints that control the x, y, and z axes of the end effector.

The file also defines four group states, which specify named joint values for the different groups. The joint values are given for each joint in the group, using the "joint" element within the "group\_state" element.

An end effector named "end\_effector\_def" is defined, which specifies the parent link, group, and parent group for the end effector.

A virtual joint named "base\_world\_vj" is defined, which connects a link of the robot to an external frame of reference. The type of the virtual joint is "fixed", indicating that the joint is rigid and cannot move.

Finally, the file defines several pairs of links for which collision checking is disabled. The reasons for disabling collision checking are given for each pair of links

1. moveit.rviz

This is a configuration file for the Robot Visualization (RViz) tool. RViz is a 3D visualization tool for ROS (Robot Operating System) that allows users to visualize sensor data and robot models in a 3D environment.

The configuration file contains information on the panels, visualization manager, window geometry and various other options for RViz.

The panels section contains a list of panels that are available in the RViz interface. In this case, there are three panels available: Displays, Help and Views.

The visualization manager section contains information on the displays, global options, and tools that are available in RViz. The displays section lists the different types of displays available in RViz, such as Grid and MotionPlanning. The global options section contains the fixed frame option, which determines the frame of reference for the visualization. The tools section lists the tools available in the RViz interface, such as Interact, MoveCamera and Select.

The window geometry section contains the height and width of the RViz window.

Launch folder – All the launch files for starting the simulation

1. demo.launch.py

This code generates a launch file for MoveIt! with the configuration provided by MoveItConfigsBuilder from the moveit\_configs\_utils module. The launch file is generated using the generate\_demo\_launch function also provided by the same module.

MoveItConfigsBuilder is a helper class to generate MoveIt! configuration files for a robot. In this case, it is initialized with the robot name "Robot\_Gantry\_Gazebo" and the package name "robot\_gantry\_config". The to\_moveit\_configs() method is called on the builder instance to generate the configuration files.

generate\_demo\_launch is a function that takes the MoveIt! configuration files as an argument and returns a launch description for a MoveIt! demo. The demo includes RViz for visualization and motion planning, as well as a ROS node to control the robot using MoveIt!.

The resulting launch description can be used to start the demo by running ros2 launch <package\_name> demo.launch.

This launch file launches:

\* static\_virtual\_joint\_tfs

\* robot\_state\_publisher

\* move\_group

\* moveit\_rviz

\* warehouse\_db (optional)

\* ros2\_control\_node + controller spawners

Gantry\_description is a ROS package that contains the Gazebo and Rviz model of gantry system. It provides the model that will be used in gantry\_config ROS package to perform simulation.

Launch folder – Launch files for spawning Gazebo model

1. gazebo.launch.py

This launch file launches Gazebo simulation along with robot\_state\_publisher, joint\_state\_publisher and urdf\_spawner nodes. The urdf\_spawner node will spawn the robot in the Gazebo simulation using the robot\_description topic.

The 'use\_sim\_time' LaunchConfiguration argument is defined with a default value of 'false' and it is used by the robot\_state\_publisher and joint\_state\_publisher nodes as a parameter. If it is set to 'true', the nodes will use the simulation time instead of the system clock time.

The URDF file is specified as a parameter to the robot\_state\_publisher and joint\_state\_publisher nodes. It is loaded using the get\_package\_share\_directory function from the ament\_index\_python.packages module. The path to the URDF file is constructed using the os.path.join function.

The Gazebo simulation is launched using the gazebo executable and the libgazebo\_ros\_factory.so plugin. The --verbose flag is used to output verbose information to the console.

URDF folder- Contains the model/robot description of gantry system

1. Robot\_Gantry\_Gazebo.xacro

The given code is a URDF (Unified Robot Description Format) file for describing a robot model in ROS (Robot Operating System). It defines a robot named "Robot\_Gantry\_Gazebo" which consists of four links - "world", "base\_link", "z\_axis\_1", "y\_axis\_1", and "End\_effector\_1".

Each link is defined with its name, and within each link, there are definitions for its inertial, visual, and collision properties. The inertial properties include the mass and inertia tensor of the link, while the visual properties define how the link should look visually, and the collision properties define the shape of the link used for collision detection.

Each visual and collision property also includes a mesh filename and scale, which are the 3D mesh files for the link's visual and collision representation, respectively. These mesh files are located in the "/home/george/meshes" directory on the robot's file system.

The URDF file also includes three Xacro includes to import additional URDF files: "materials.xacro", "Robot\_Gantry\_Gazebo.trans", and "Robot\_Gantry\_Gazebo.gazebo". These files provide additional information such as material properties and information for simulation in Gazebo.

1. Robot\_Gantry\_Gazebo.gazebo

This is an XML file that defines a robot model for the Gazebo simulator, which is commonly used in robotics for simulation and testing.

The robot model is named "Robot\_Gantry\_Gazebo" and contains several components, each defined using the Gazebo XML syntax.

The first component is a plugin that is used for controlling the robot within Gazebo. The plugin is named "control" and its implementation is specified by the "libgazebo\_ros\_control.so" library.

The next several components define the physical properties of different parts of the robot, such as the base link, z-axis, y-axis, end effector, and stand. These properties include the material used to render the component (specified by the "${body\_color}" variable), as well as physical properties such as friction coefficients and collision detection settings.

Overall, this file is used to define the visual and physical properties of a robot model within the Gazebo simulator, which can be used for simulation and testing of robotic control algorithms.

1. Robot\_Gantry\_Gazebo.trans

This XML file defines three transmissions for a robot named "Robot\_Gantry\_Gazebo". Each transmission consists of a joint and an actuator, both using the "hardware\_interface/EffortJointInterface" interface. The mechanical reduction for each actuator is set to 1, meaning that the actuator output torque is equal to the joint torque. The transmissions are named "Slider 3\_tran", "Slider 4\_tran", and "Slider 7\_tran", and correspond to the joints "Slider 3", "Slider 4", and "Slider 7", respectively.