Introduction ROS - 5

Jaeseok Kim
The BioRobotics Institute - Service Robotics and Ambient Assisted Living Lab

Contents

- **Manipulator Introduction**
- **OpenManipulator Modeling and Simulation**
- **MoveIt!**

- **Manipulator Introduction**
- **OpenManipulator Modeling and Simulation**
- **MoveIt!**

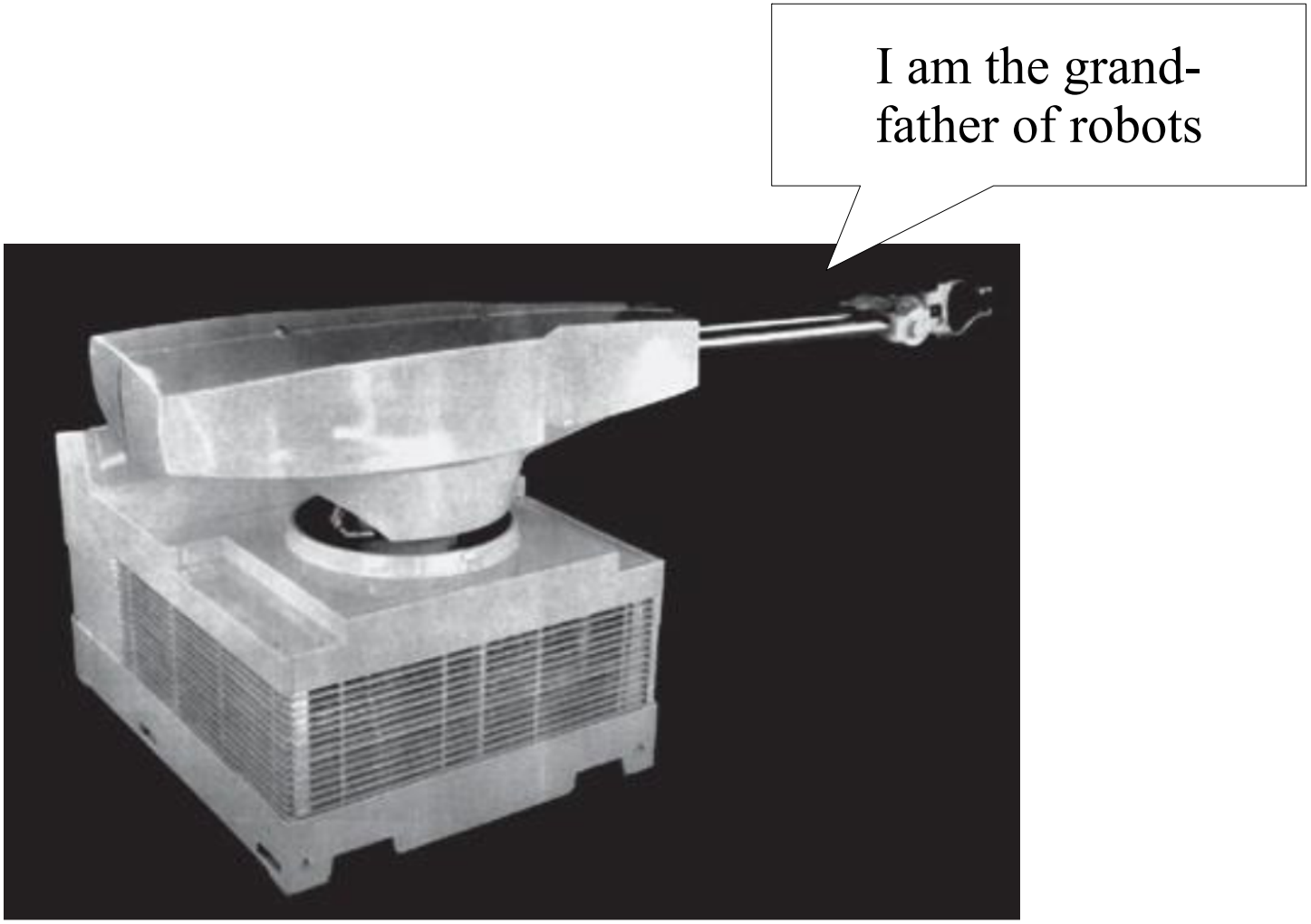
From now on, we will see

Manipulator Modelling

(How to write URDF)

How to Utilize ‘MoveIt!’

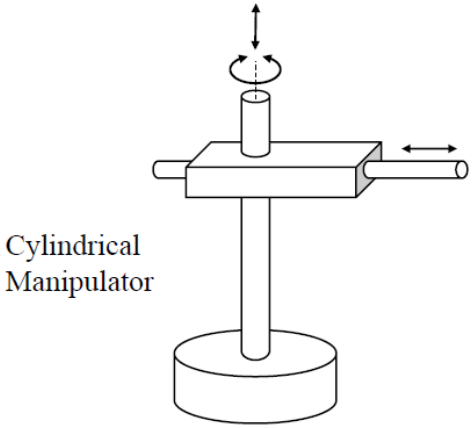
Manipulator?



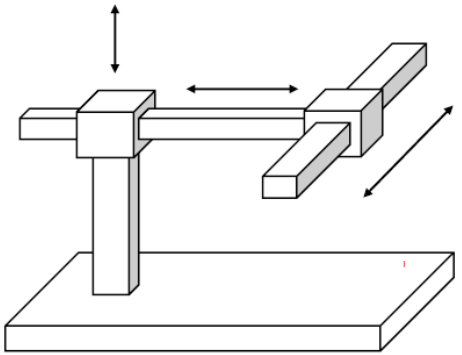
The first industrial robot, unimate, 1961

http://en.wikipedia.org/wiki/Unimate

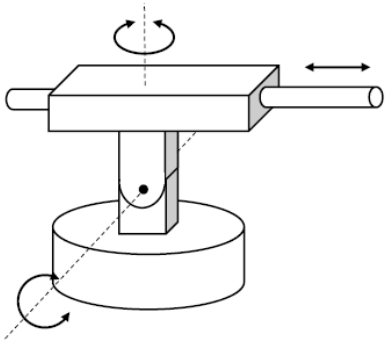
Types of Manipulator



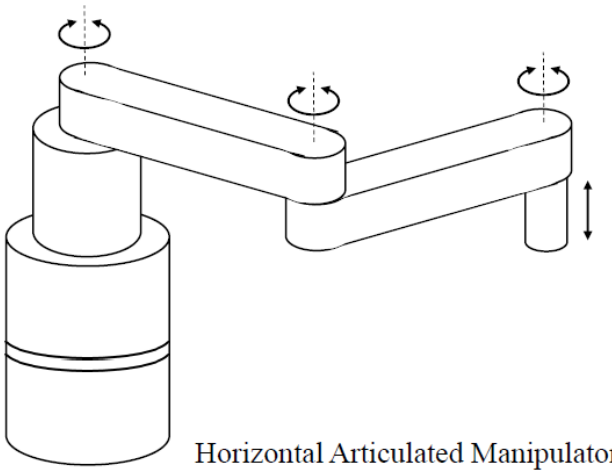
Cylindrical
Manipulator



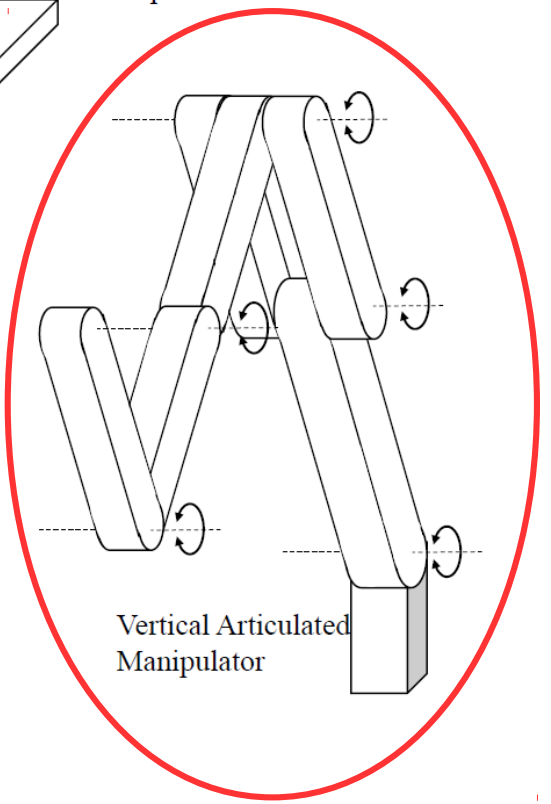
Rectangular-Coordinate
Manipulator



Polar-Coordinate
Manipulator

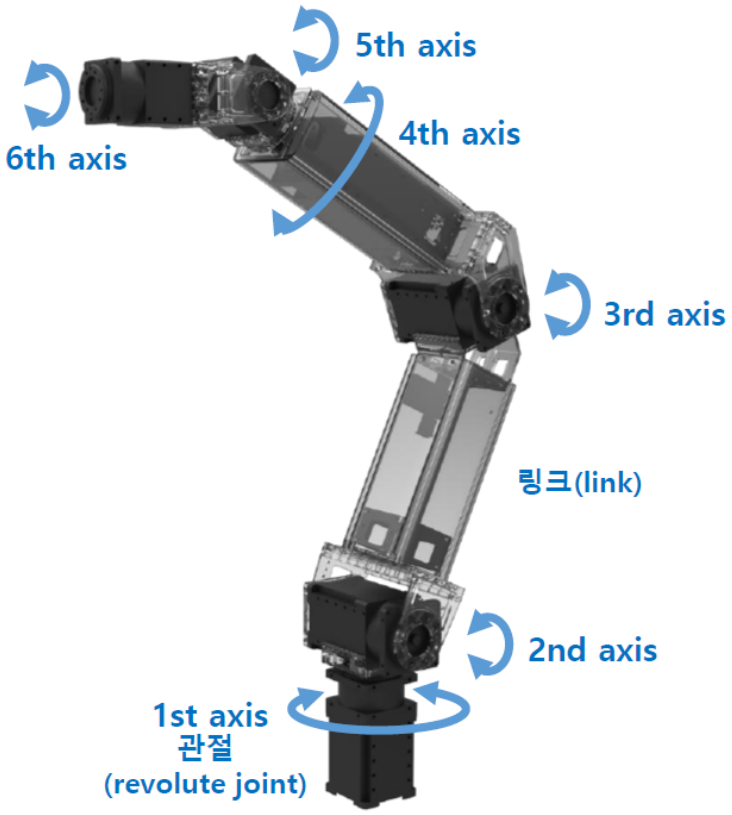


Horizontal Articulated Manipulator
(SCARA type)

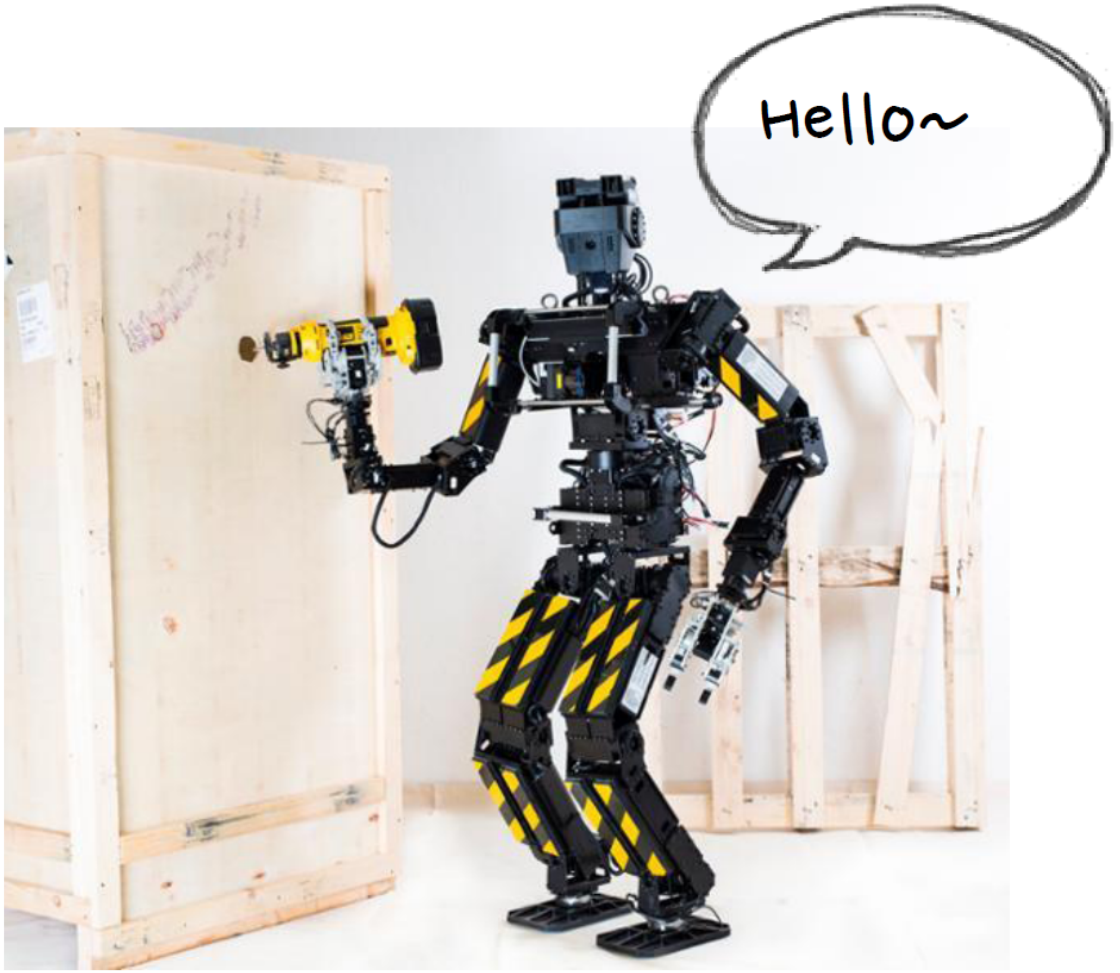


Vertical Articulated
Manipulator

Manipulator



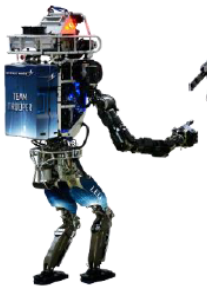
http://www.robotis.com/



http://www.theroboticschallenge.org/

Manipulator in DRC Finals 25

Hercules
(Team TRAC Labs)

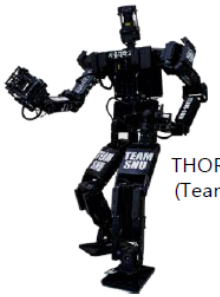


LEO
(Team TROOPER)

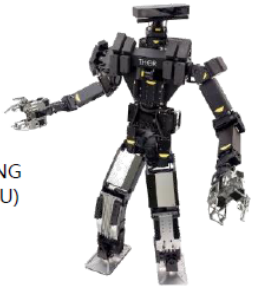
Helios (Atlas)
(Team MIT)

Florian
(Team ViGIR)

THORMANG
(Team ROBOTIS)



THORMANG
(Team SNU)



THOR-RD
(Team THOR)

Johnny 05
(TEAM HECTOR)

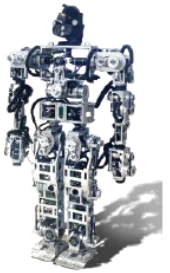
Running Man
(Team IHMC Robotics)

Atlas
(TEAM HKU)

WARNER
(Team WPI-CMU)

Momaro
(Team NimbRo Rescue)

Aero DRC
(TEAM AERO)



Xing Tian
(Team Intelligent Pioneer)

DRC-HUBO
(Team KAIST)

DRC-Hubo
(TEAM DRC-HUBO AT UNLV)

HRP2+
(TEAM AIST-NEDO)

HRP-2
(Team HRP2-Tokyo)

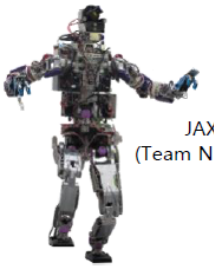
RoboSimian
(Team RoboSimian)

Cog-Burn
(TEAM GRIT)

CHIMP
(TARTAN RESCUE)

WALK-MAN
(Team WALK-MAN)

ESCHER
(Team VALOR)



JAXON
(Team NEDO-JSK)

Hydra
(Team NEDO-Hydra)

Manipulator Control

- Sensing
  - Position, Torque
  - Surrounding
  - Environment,
  - Objects, …
  -
- Plan
  - Manual / Autonomous
  - Collision Avoidance
  - Grasping
  - Trajectory Generation
  -
- Action
    - Control Position, Velocity, Torque

https://www.youtube.com/watch?v=T9JKTa1wzQ0

Controlling Manipulator?
It was no picnic to do!
However, the situation is changed!

They help us! |(^^)/

Why ROS is Useful for Manipulator?

• URDF(Unified Robot Description Format) helps you to do modelling, visualizing and simulating a robot without difficulties.

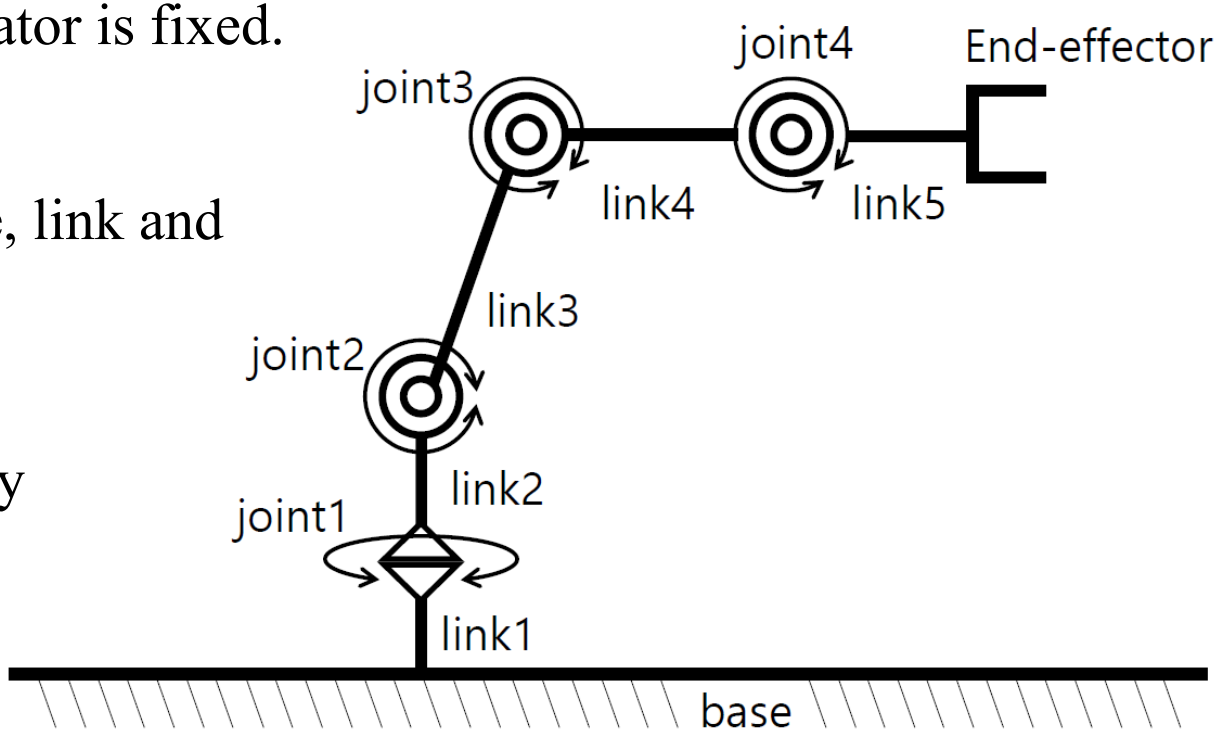
You can write **URDF** easily by utilizing **XML** (Extensive Markup Language)

• It supports **'3D Simulator Gazebo'** which helps you to construct simulation environment similar to the real one.
Gazebo includes many plug-ins related to sensor and robot control.

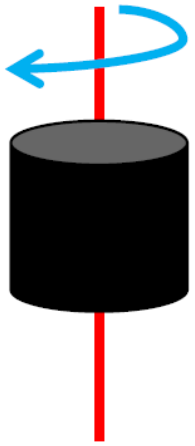
• You can utilize **'MoveIt!'** , the combined library for manipulator
  • It provides open libraries such as Kinematics and Dynamics Library(KDL) and the Open Motion Planning Library(OMPL)
  • Many functions related to manipulators are available. For example, collision avoidance, motion planning, pick and place, and so on.

Components of Manipulator

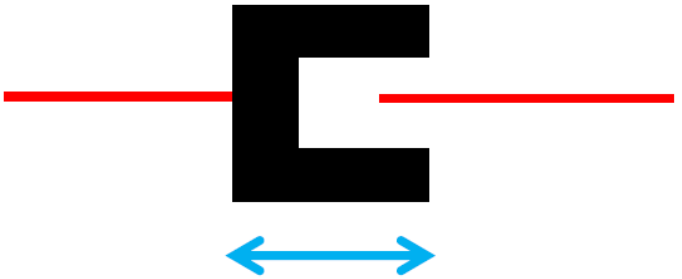
- **base**
  - A place where a manipulator is fixed.
- **Link**
  - Connectors between base, link and end-effector.
- **joint**
  - Related to robot's motility
  - Revolute
  - Prismatic
- **End-effector**
  - Similar to the human hand



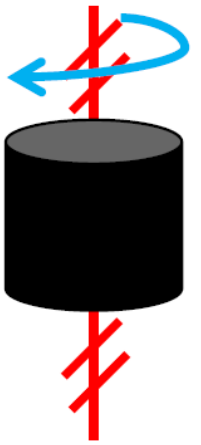
Types of Joints



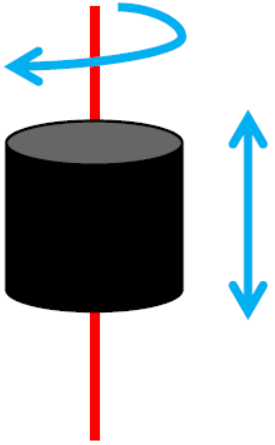
Revolute Joint

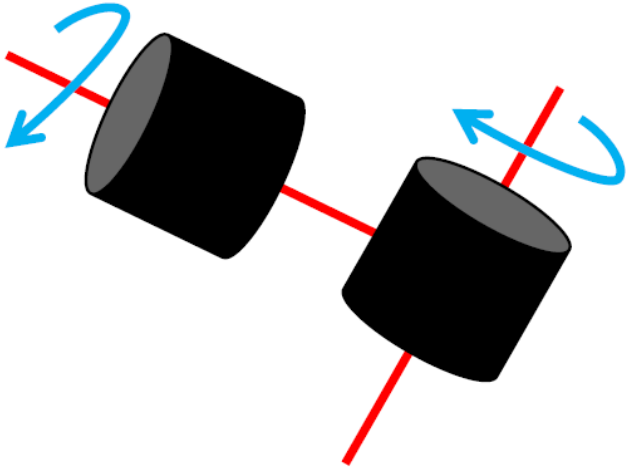


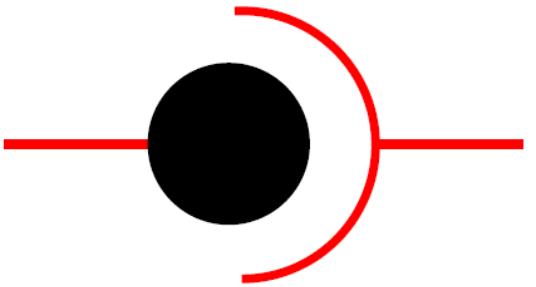
Prismatic Joint



Screw Joint

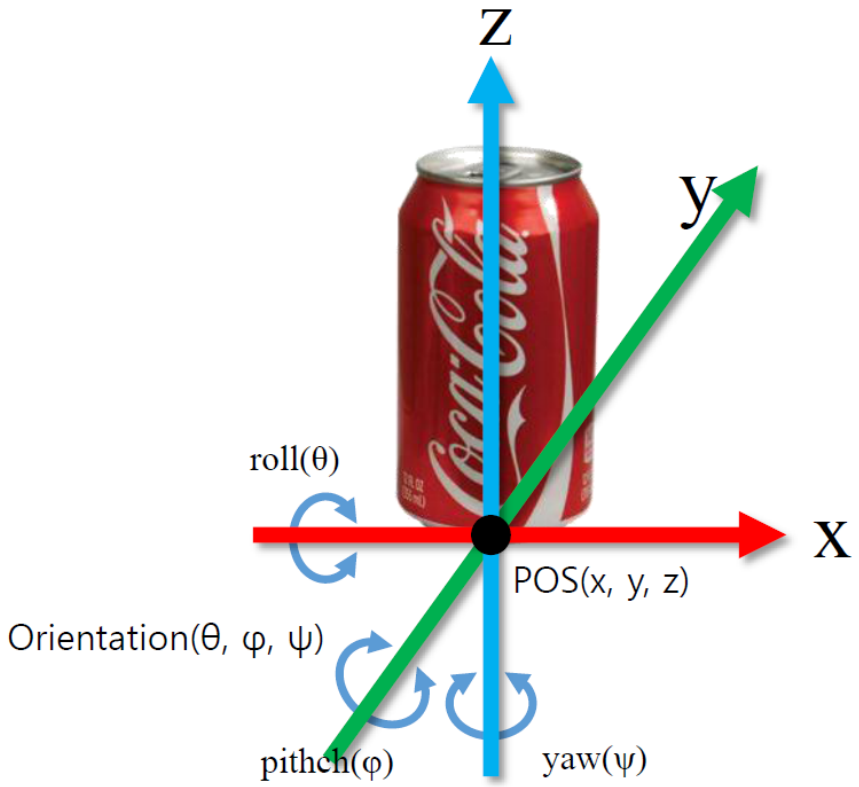


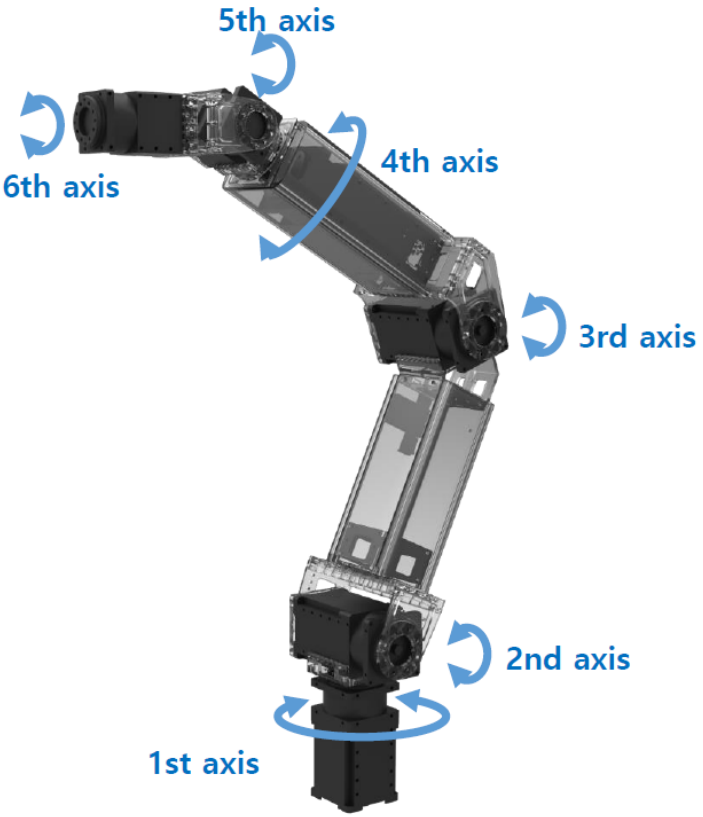




DOF, degrees of freedom

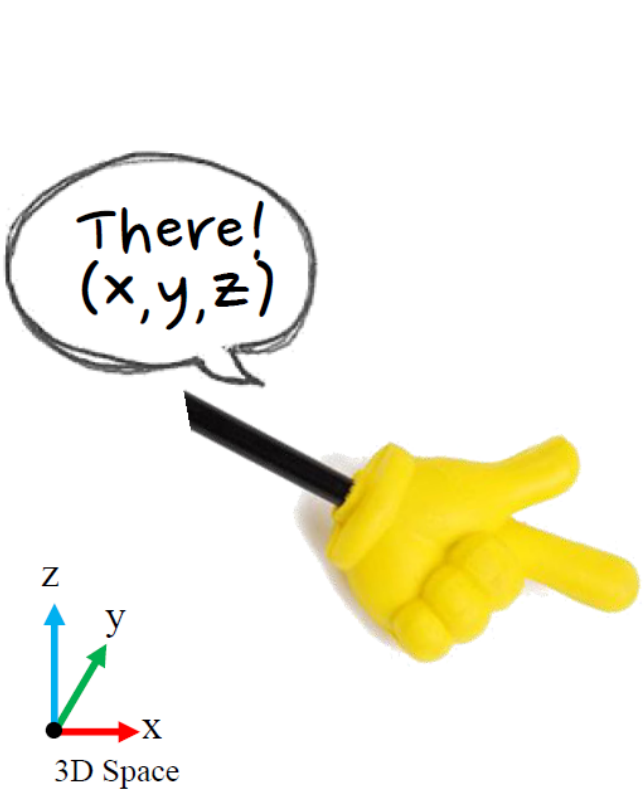
Degree of Freedom
= The number of parameters of the system that may vary.
= If 6 parameters are required to describe object's position&orientation ->It has 6 D.o.F.

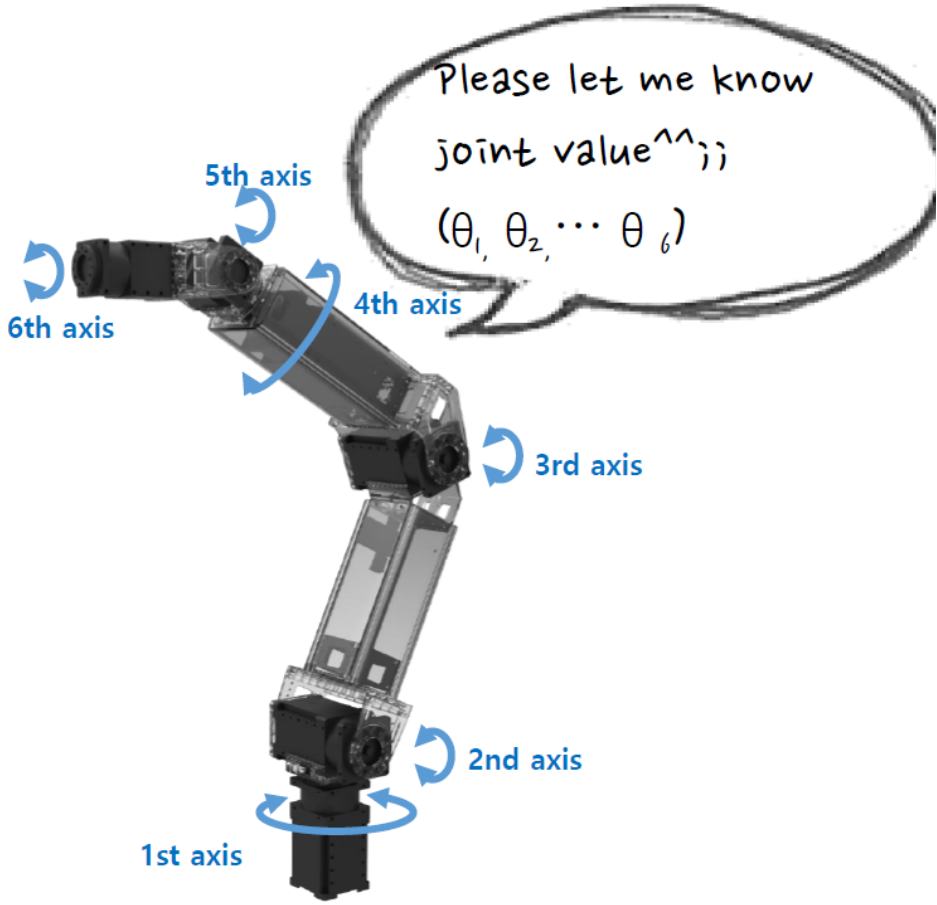




http://www.robotis.com http://www.tbotech.com/sodacansafe.htm

Workspace & Jointspace





Work space : A volume where end-effector of a manipulator can move. Described with (x, y, z, θ, φ, ψ)

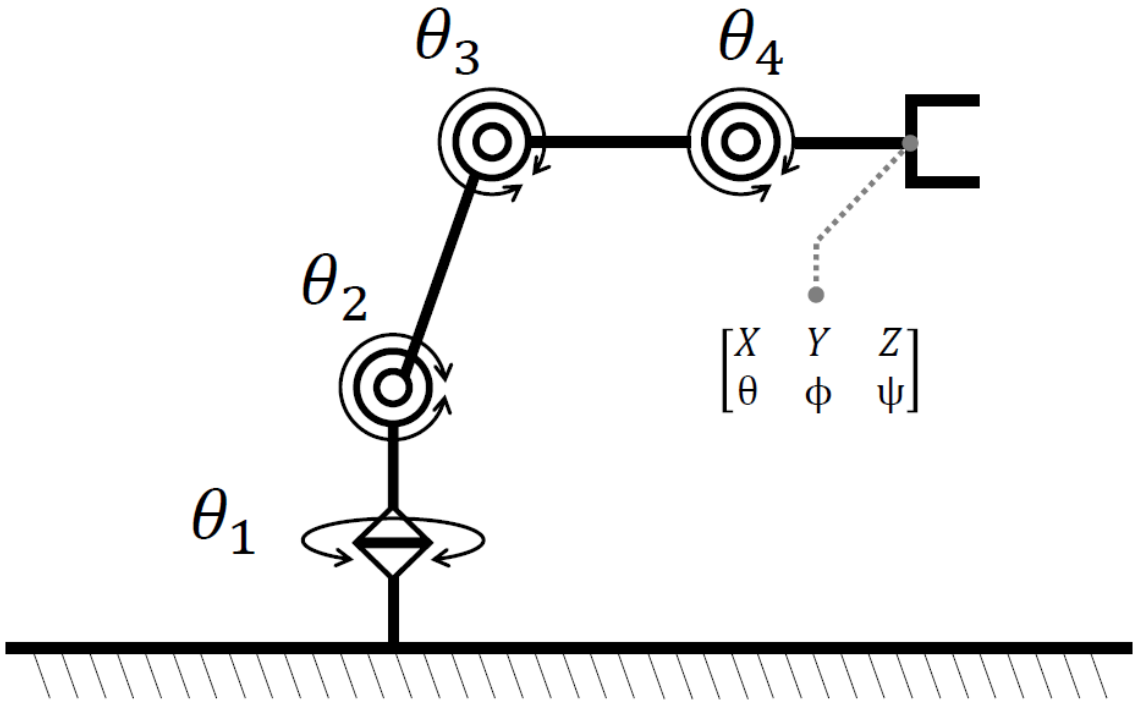
Joint space : A space described with angles of joints(θ1,θ2,θ3,...)

Forward Kinematics & Inverse Kinematics

- **Things required to control a robot as you want**
  - Planning based on geometric information(Kinematics)
  - Planning based on how forces and torques will affect to motion(Dynamics),
  - Give commands to each joint based on planning result

Forward Kinematics & Inverse Kinematics

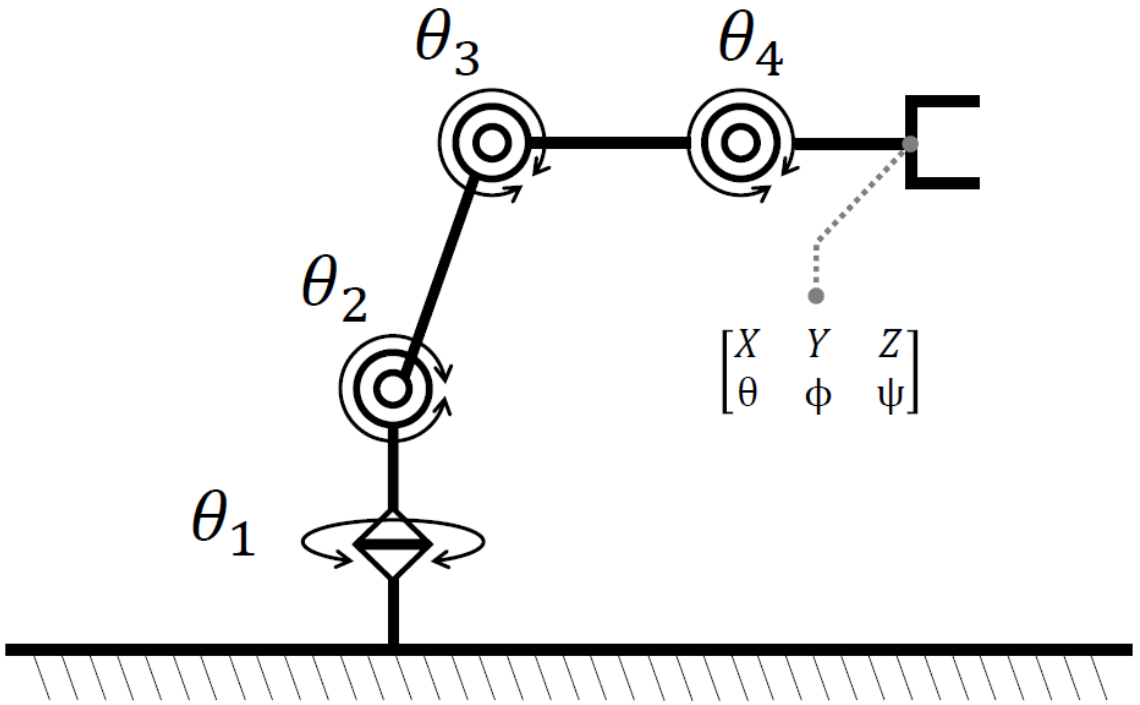
- **Forward Kinematics**
  - Joints angles are given, The position & orientation of end-effector will be computed
  - (θ 1 , θ 2 , θ 3 , ..., θ n) → (x, y, z), (θ, φ, ψ)



$$\begin{matrix}\theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4\end{matrix} \Rightarrow \begin{bmatrix} X & Y & Z \\ \theta & \phi & \psi \end{bmatrix}$$

Forward Kinematics & Inverse Kinematics

- **Inverse Kinematics**
  - Position & Orientation of end-effector are given, joint angles will be computed
  - (x, y, z), (θ, φ, ψ) → (θ 1 , θ 2 , θ 3 , ..., θ n)



$$\begin{bmatrix} X & Y & Z \\ \theta & \phi & \psi \end{bmatrix} \Rightarrow \begin{matrix} \theta_1 \\ \theta_2 \\ \theta_3 \\ \theta_4 \end{matrix}$$

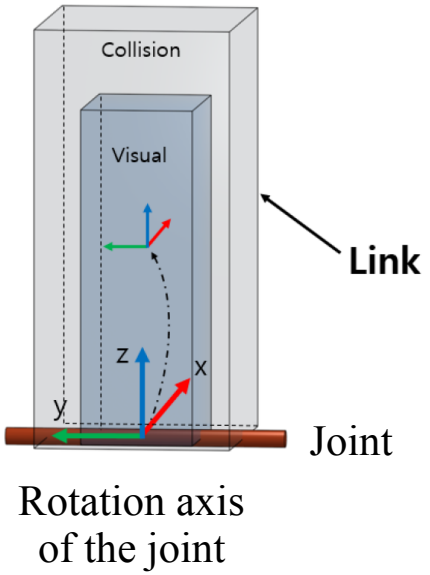
MoveIt! (A combined library for manipulator)

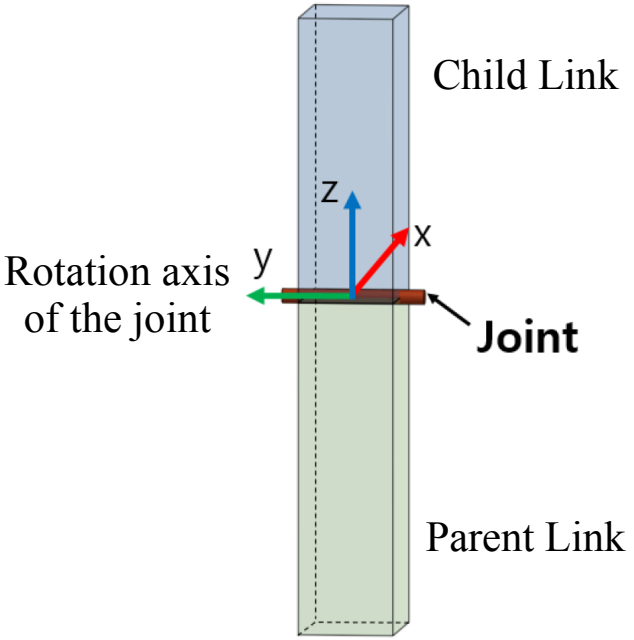
https://youtu.be/0og1SaZYtRc

- Manipulator Introduction
- **OpenManipulator Modeling and Simulation**
- MoveIt!

Information you should give to MoveIt!

- **Link**
  - Geometric Information
  - STL or DAE(COLLDA); 3d mesh file
  - Mass
  - Moment of Inertia
- **Joint**
  - Child/Parent Links
  - Type of joint: Revolute or Prismatic
  - Revolute Axis(or Prismatic Axis)
  - Limit Values(Torque limit, Min/Max Angle, Velocity limit)





Information you should give to MoveIt!

- **Link**
  - Geometric Information
  - STL or DAE(COLLDA); 3d mesh file
  - Mass
  - Moment of Inertia
- **Joint**
  - Child/Parent Links
  - Type of joint: Revolute or Prismatic
  - Revolute Axis(or Prismatic Axis)
  - Limit Values(Torque limit, Min/Max Angle, Velocity limit)

URDF

includes all of these!

URDF(Unified Robot Description Format) → RViz
SRDF(Semantic Robot Description Format)→ MoveIt!
SDF(Simulation Description Format) → Gazebo

http://wiki.ros.org/ROS/Patterns/RobotModelling

Practice

<3-link Manipulator>

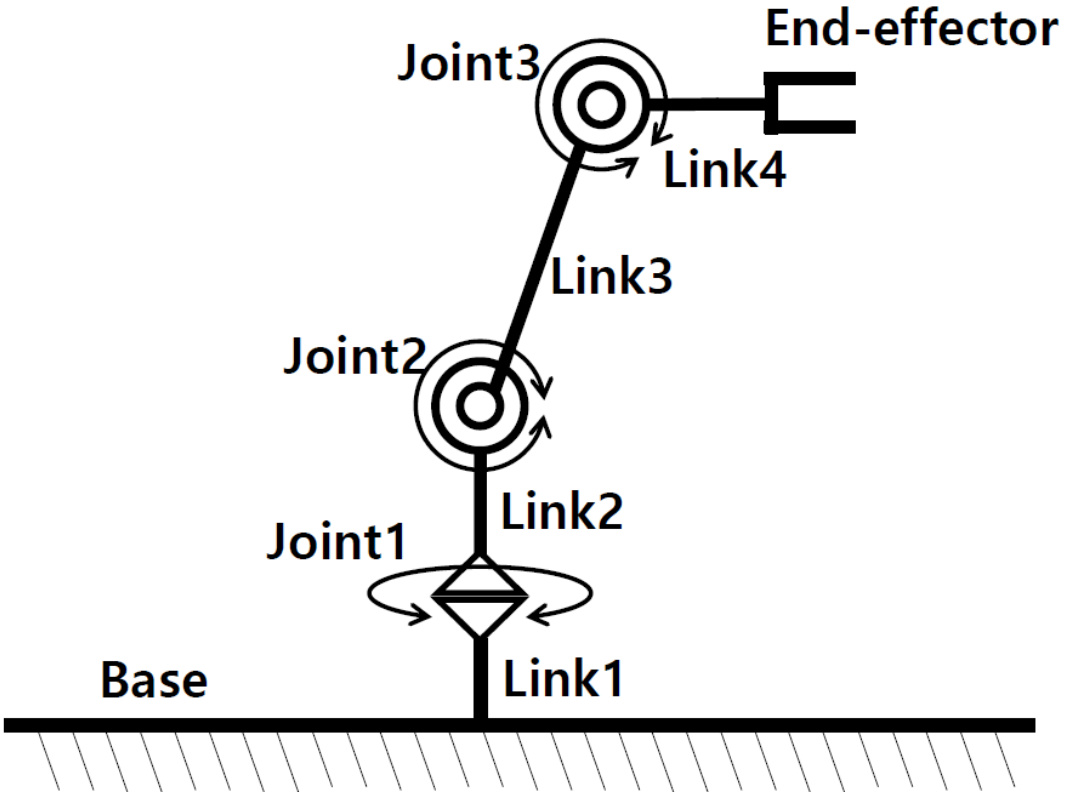
https://github.com/ROBOTIS-GIT/ros_tutorials/tree/master/testbot_description

Practice! 3-link Manipulator Modelling

- **Components of 3-link Manipulator**
  - 3 joints, 4 links

- **URDF**

```
$ cd ~/catkin_ws/src
$ catkin_create_pkg
testbot_description urdf
$ cd testbot_description
$ mkdir urdf
$ cd urdf
$ gedit testbot.urdf
```

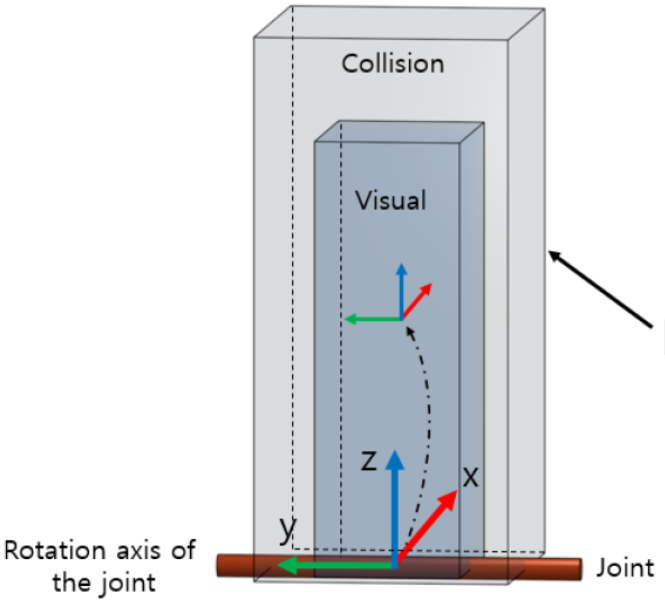


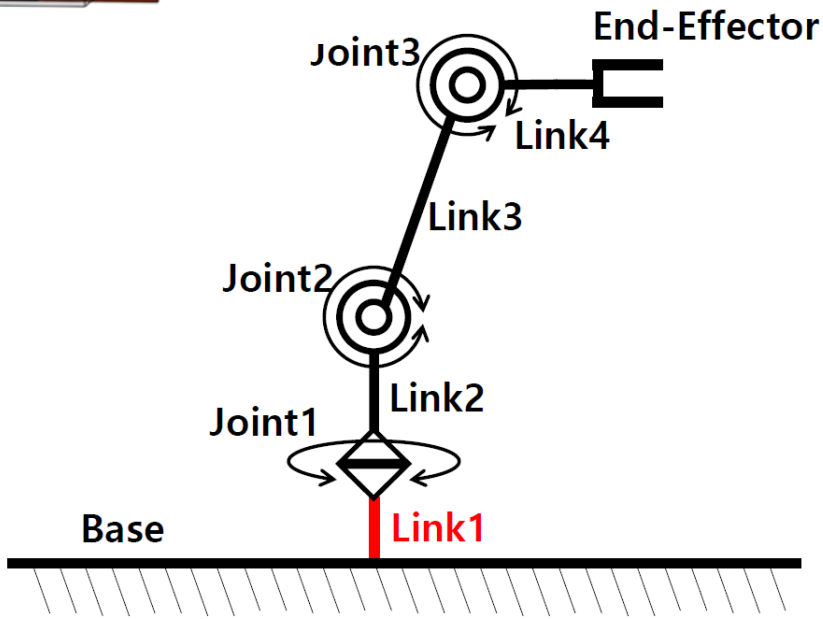
https://github.com/ROBOTIS-GIT/ros_tutorials/blob/master/testbot_description/urdf/testbot.urdf

Practice! 3-link Manipulator Modelling

- In urdf, (link1)

```
<link name="link1">

<collision>
<origin xyz="0 0 0.25" rpy="0 0 0"/>
<geometry>
<box size="0.1 0.1 0.5"/>
</geometry>
</collision>

<visual>
<origin xyz="0 0 0.25" rpy="0 0 0"/>
<geometry>
<box size="0.1 0.1 0.5"/>
</geometry>
<material name="black"/>
</visual>

<inertial>
<origin xyz="0 0 0.25" rpy="0 0 0"/>
<mass value="1"/>
<inertia ixx="1.0" ixy="0.0" ixz="0.0"
         iyy="1.0" iyz="0.0"
                   izz="1.0"/>
</inertial>

</link>
```

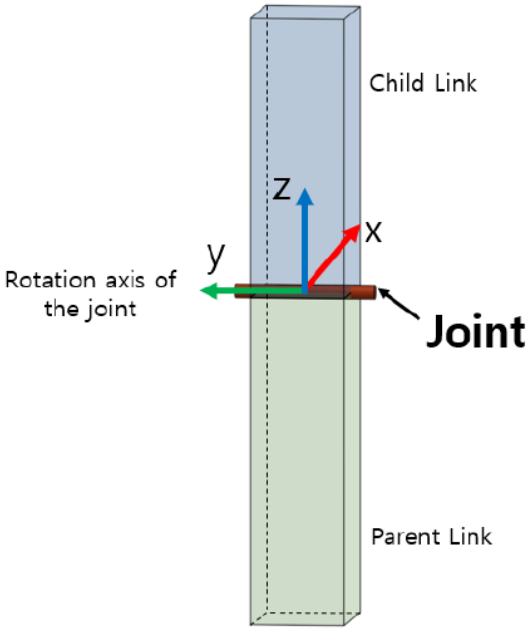


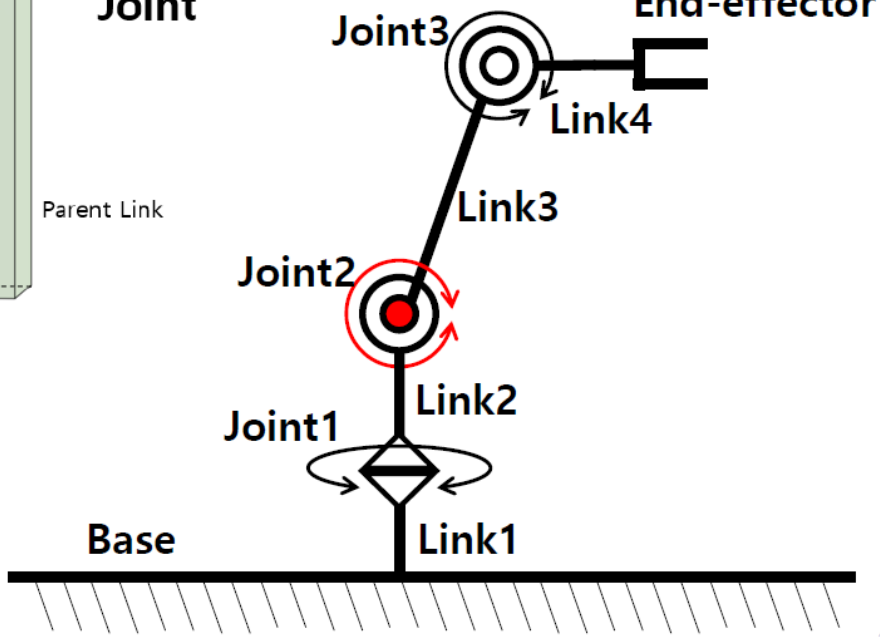


Practice! 3-link Manipulator Modelling

- In urdf, (joint 2)

```
<joint name="joint2" type="revolute">
<parent link="link2"/>
<child link="link3"/>
<origin xyz="0 0 0.5" rpy="0 0 0"/>
<axis xyz="0 1 0"/>
<limit effort="30" lower="-2.617"
       upper="2.617" velocity="1.571"/>
</joint>
```





https://github.com/ROBOTIS-GIT/ros_tutorials/blob/master/testbot_description/urdf/testbot.urdf

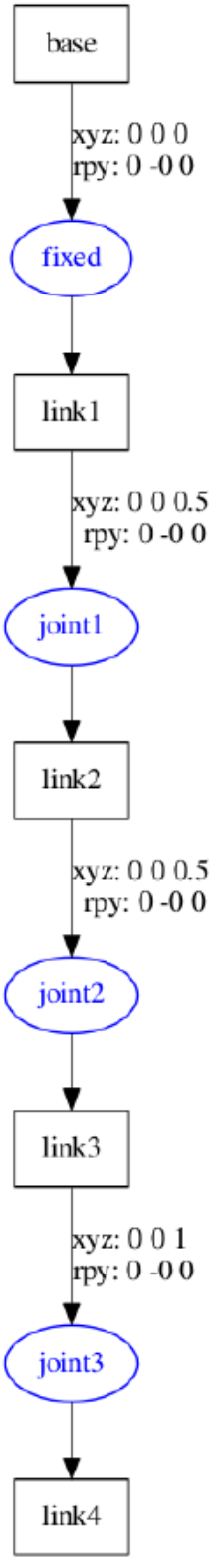
URDF Grammar Check & Graph

- **check_urdf (Grammar Check)**

```
$ check_urdf testbot.urdf
robot name is: test_robot
---------- Successfully Parsed XML ---------------
root Link: base has 1 child(ren)
      child(1): link1
            child(1): link2
                  child(1): link3
                        child(1): link4
```

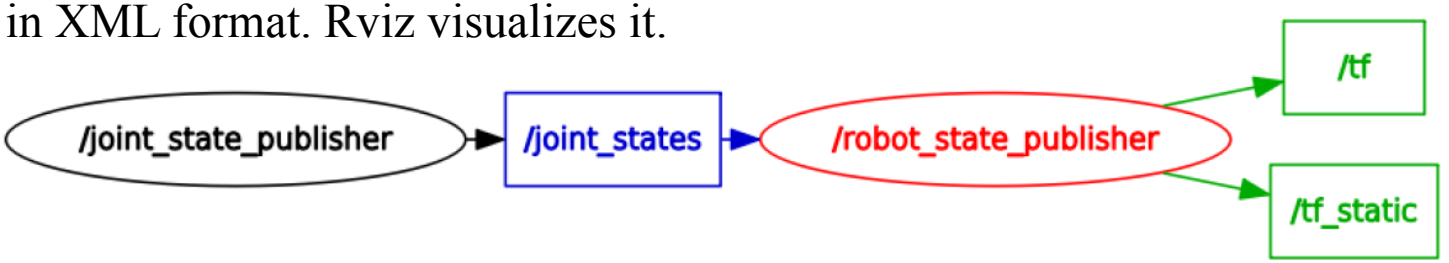
- **urdf_to_graphiz (Graph)**

```
$ urdf_to_graphiz testbot.urdf
Created file test_robot.gv
Created file test_robot.pdf
```



URDF Model Check

- **You can check URDF model by using RViz**
  - robot_description
  : Robot model is described in XML format. Rviz visualizes it.
  - joint_state_publisher
  : ROS node that publishes joint values
  - robot_state_publisher
  : ROS node that subscribes joint values, and than publishes relations between links&joints as /tf(transformation)

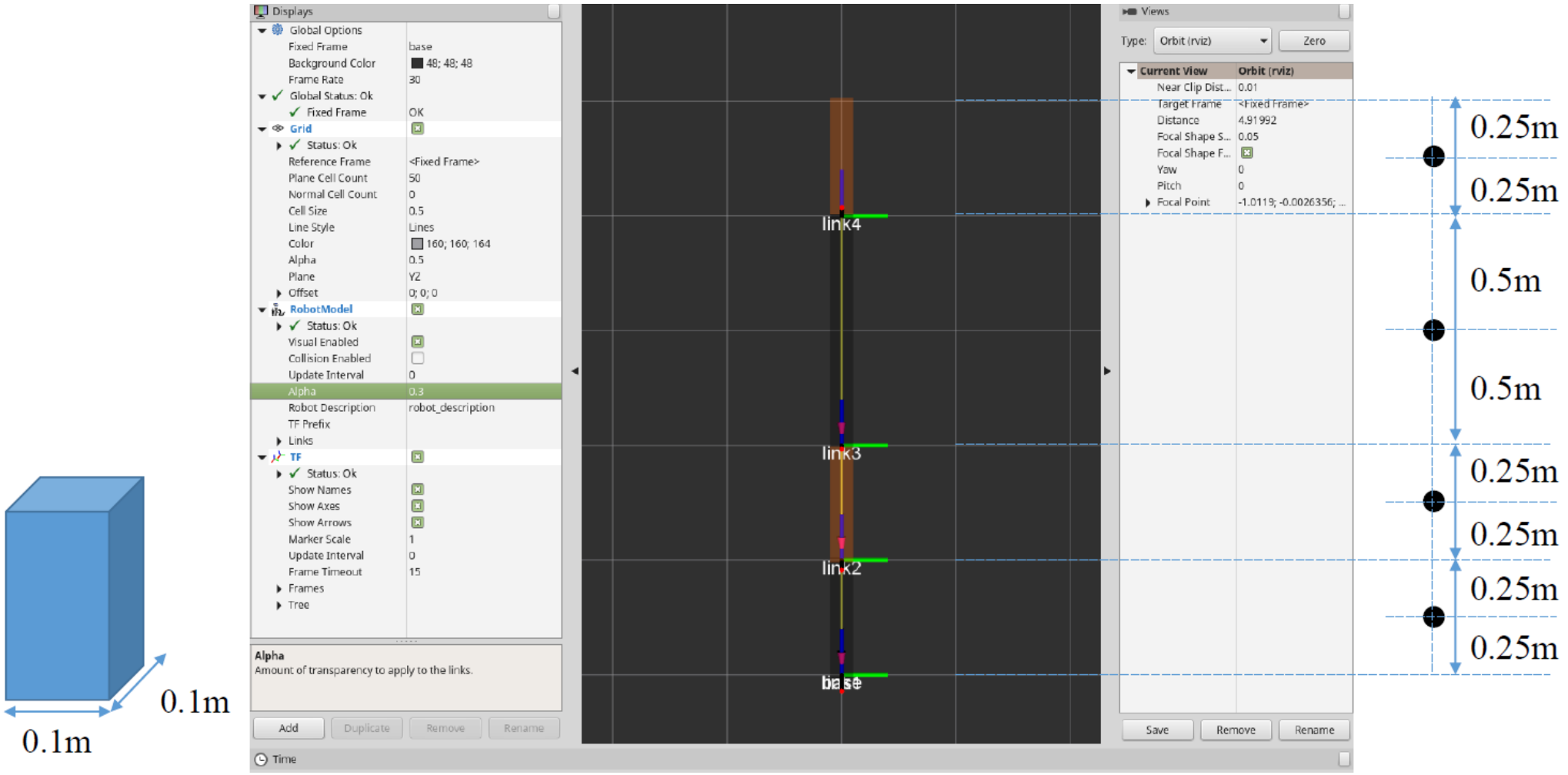



```
$ cd ~/catkin_ws/src/testbot_description
$ mkdir launch
$ cd launch
$ gedit testbot.launch
```

```
<launch>
<arg name="model" default="$(find testbot_description)/urdf/testbot.urdf" />
<arg name="gui" default="True" />
<param name="robot_description" textfile="$(arg model)" />
<param name="use_gui" value="$(arg gui)"/>
<node pkg="joint_state_publisher" type="joint_state_publisher" name="joint_state_publisher" />
<node pkg="robot_state_publisher" type="state_publisher" name="robot_state_publisher" />
</launch>
```

```
$ roslaunch testbot_description testbot.launch
$ rviz
```

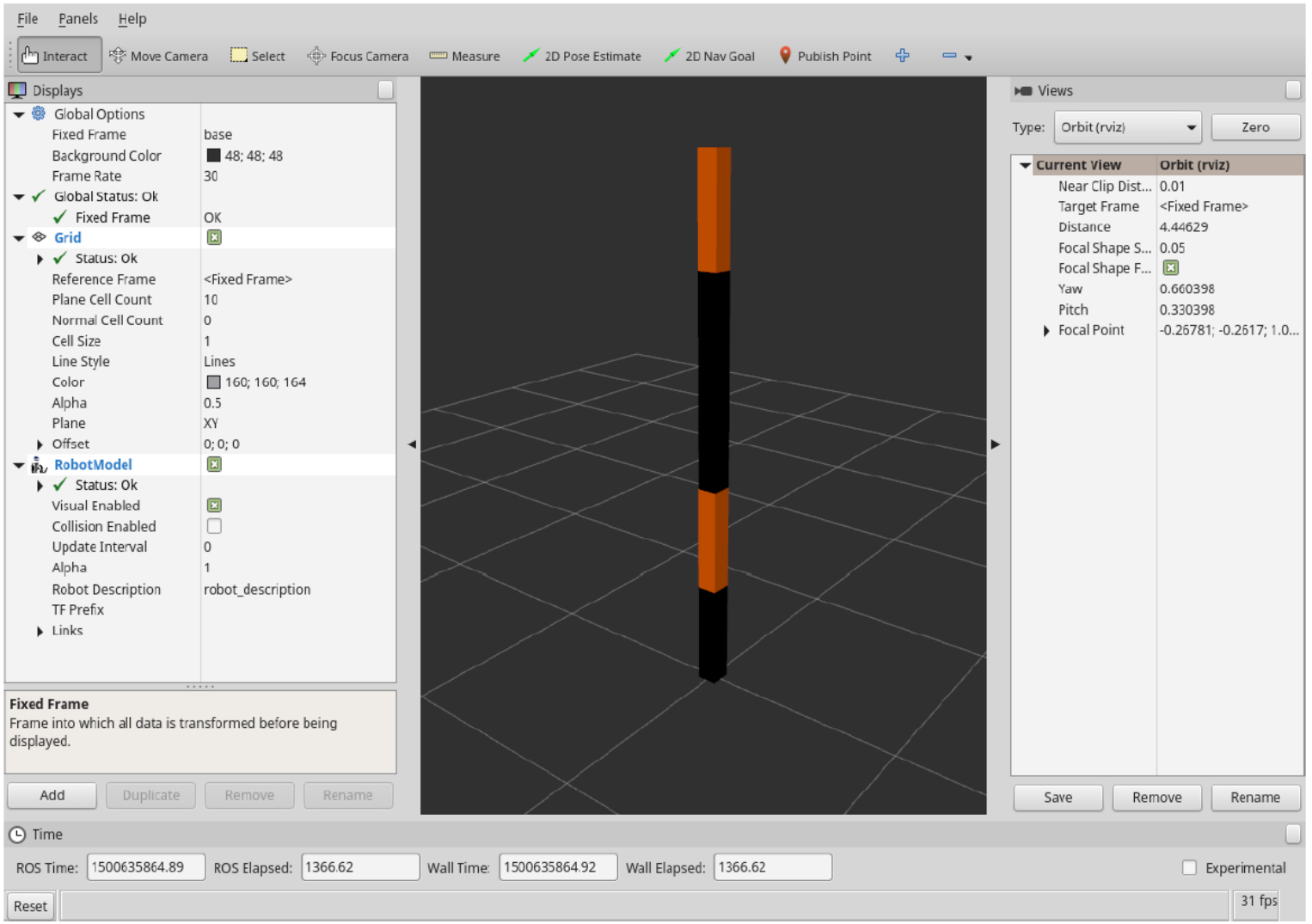
RViz


Displays
Global Options
Fixed Frame base
Background Color 48; 48; 48
Frame Rate 30
Global Status: Ok
Fixed Frame OK
Grid
Status: Ok
Reference Frame <Fixed Frame>
Plane Cell Count 50
Normal Cell Count 0
Cell Size 0.5
Line Style Lines
Color 160; 160; 164
Alpha 0.5
Plane XY
Offset 0; 0; 0
RobotModel
Status: Ok
Visual Enabled
Collision Enabled
Update Interval 0
Alpha 0.3
Robot Description robot_description
TF Prefix
Links
TF
Status: Ok
Show Names
Show Axes
Show Arrows
Marker Scale 1
Update Interval 0
Frame Timeout 15
Frames
Tree
Alpha
Amount of transparency to apply to the links.
Add
Duplicate
Remove
Rename
link4
link3
link2
base
Views
Type: Orbit (rviz)
Zero
Current View Orbit (rviz)
Near Clip Dist... 0.01
Target Frame <Fixed Frame>
Distance 4.91992
Focal Shape S... 0.05
Focal Shape F...
Yaw 0
Pitch 0
Focal Point -1.0119; -0.0026356; ...
Save
Remove
Rename
Time
0.25m
0.25m
0.5m
0.5m
0.25m
0.25m
0.25m
0.25m
0.1m
0.1m

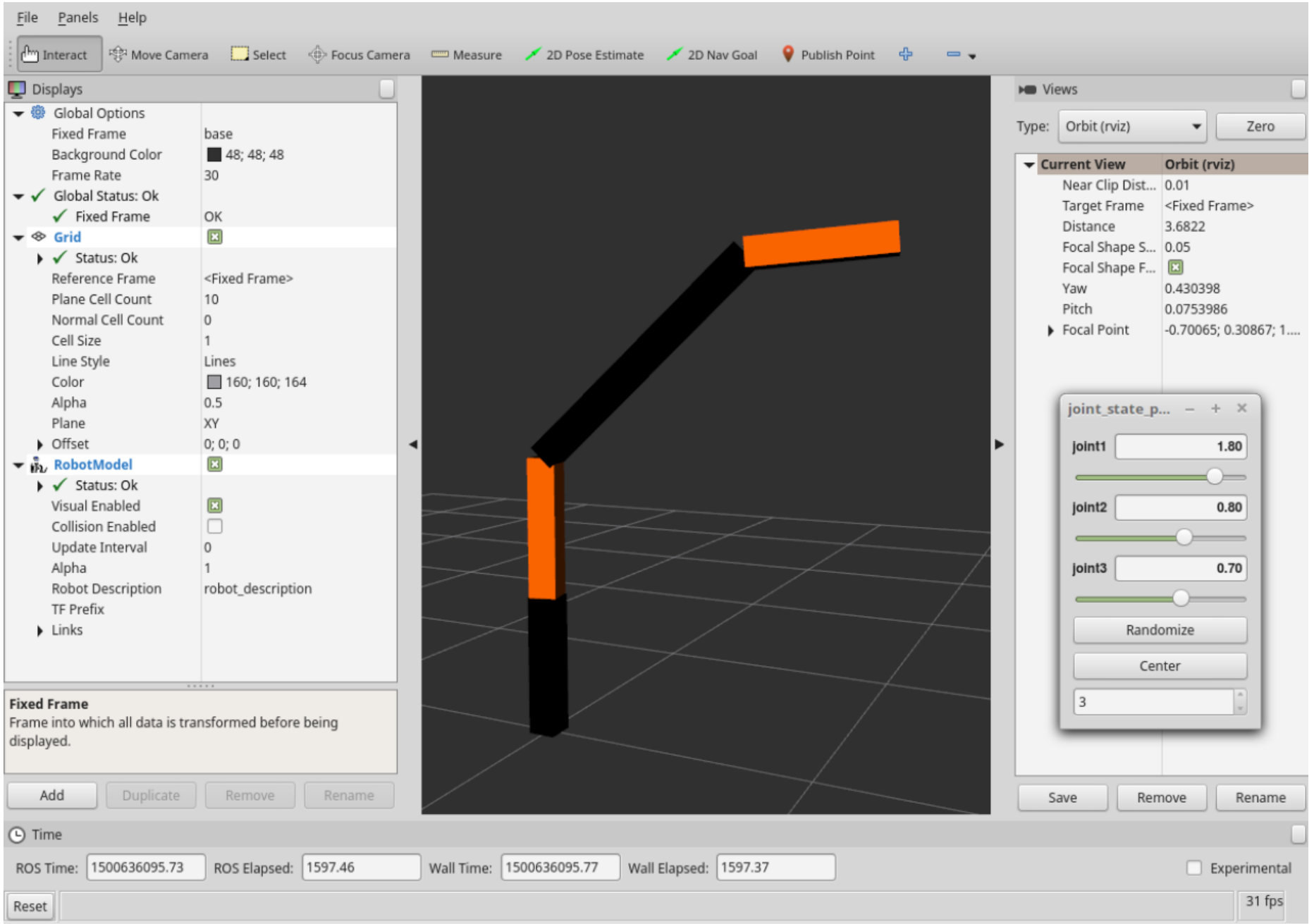
RViz

change

joint values!



RViz

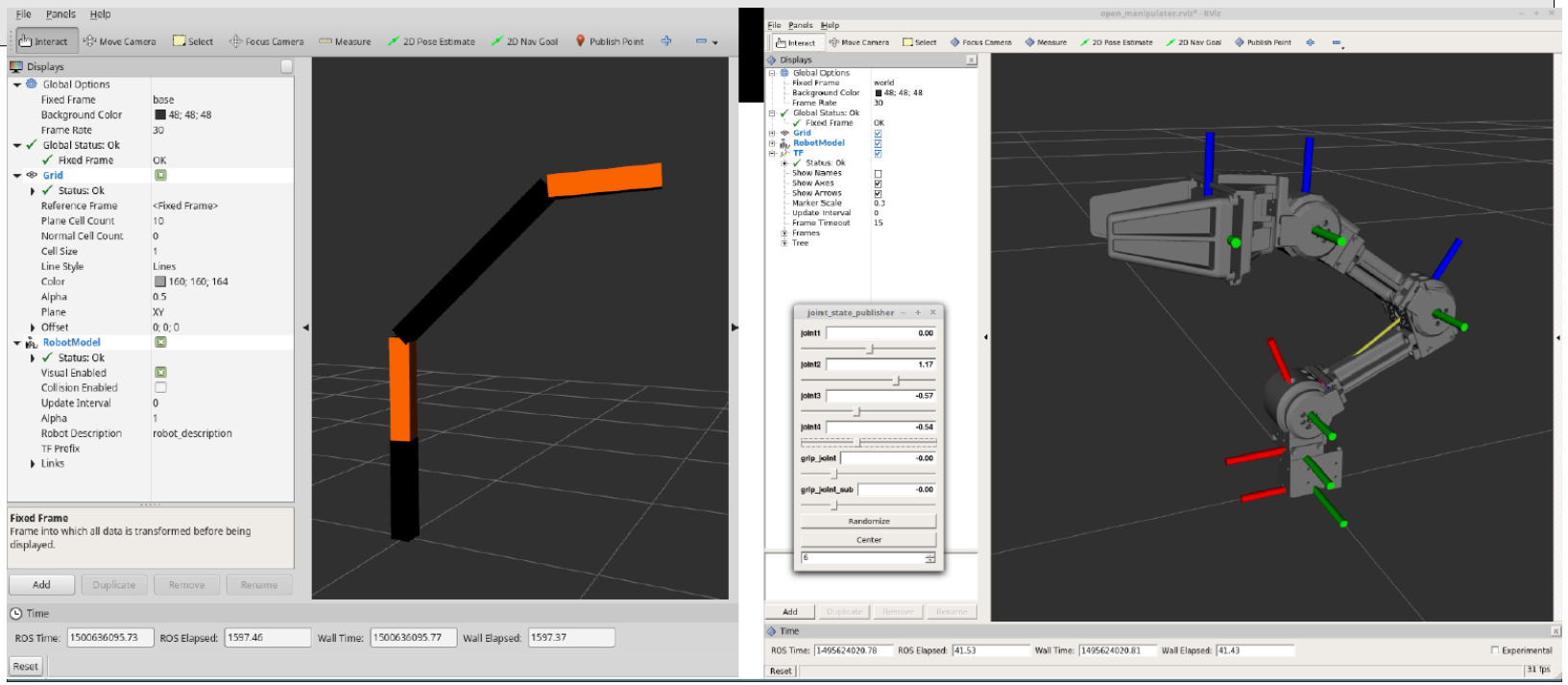

File
Panels
Help
Interact
Move Camera
Select
Focus Camera
Measure
2D Pose Estimate
2D Nav Goal
Publish Point
Displays
Global Options
Fixed Frame
base
Background Color
48; 48; 48
Frame Rate
30
Global Status: Ok
Fixed Frame
OK
Grid
Status: Ok
Reference Frame
<Fixed Frame>
Plane Cell Count
10
Normal Cell Count
0
Cell Size
1
Line Style
Lines
Color
160; 160; 164
Alpha
0.5
Plane
XY
Offset
0; 0; 0
RobotModel
Status: Ok
Visual Enabled
Collision Enabled
Update Interval
0
Alpha
1
Robot Description
robot_description
TF Prefix
Links
Fixed Frame
Frame into which all data is transformed before being displayed.
Add
Duplicate
Remove
Rename
Views
Type:
Orbit (rviz)
Zero
Current View
Orbit (rviz)
Near Clip Dist...
0.01
Target Frame
<Fixed Frame>
Distance
3.6822
Focal Shape S...
0.05
Focal Shape F...
Yaw
0.430398
Pitch
0.0753986
Focal Point
-0.70065; 0.30867; 1....
joint_state_p...
joint1
1.80
joint2
0.80
joint3
0.70
Randomize
Center
3
Save
Remove
Rename
Time
ROS Time:
1500636095.73
ROS Elapsed:
1597.46
Wall Time:
1500636095.77
Wall Elapsed:
1597.37
Experimental
Reset
31 fps

If you have 3D mesh file of your robot, you can use it

- **URDF: in link>visual>geometry, you can load STL or DAE file**

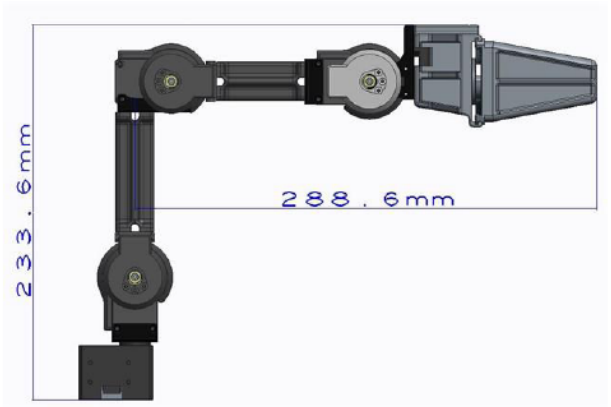
Example) open_manipulator/open_manipulator_description/urdf/open_manipulator_chain.xacro

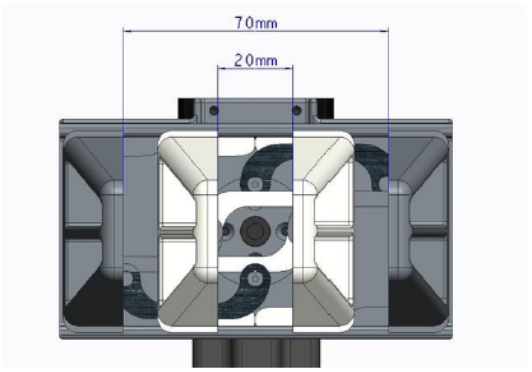
```
<visual>
<origin xyz="0.0 0.0 0.0" rpy="0 0 0"/>
<geometry>
<mesh filename="package://open_manipulator_description/meshes/chain_link1.stl" scale="0.001 0.001 0.001"/>
</geometry>
<material name="grey"/>
</visual>
```

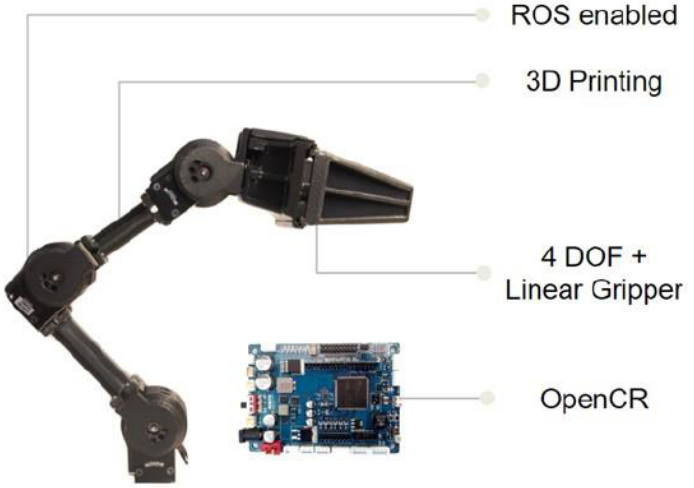


OpenManipulator

- Purpose: ROS based Manipulator platform that can replace expensive manipulators
- Specialties: Open Source S/W, H/W
  - Support Dynamixel X series. You can choose any actuators to construct a robot.
  - 3D design examples are provided(vertical articulated type, SCARA type, Delta and so on)
  - Support ROS and MoveIt! example
  - Support OpenCR which is embedded system for realtime control.
  - Support Arduino IDE, Processing IDE
  - You can use it with TurtleBot3

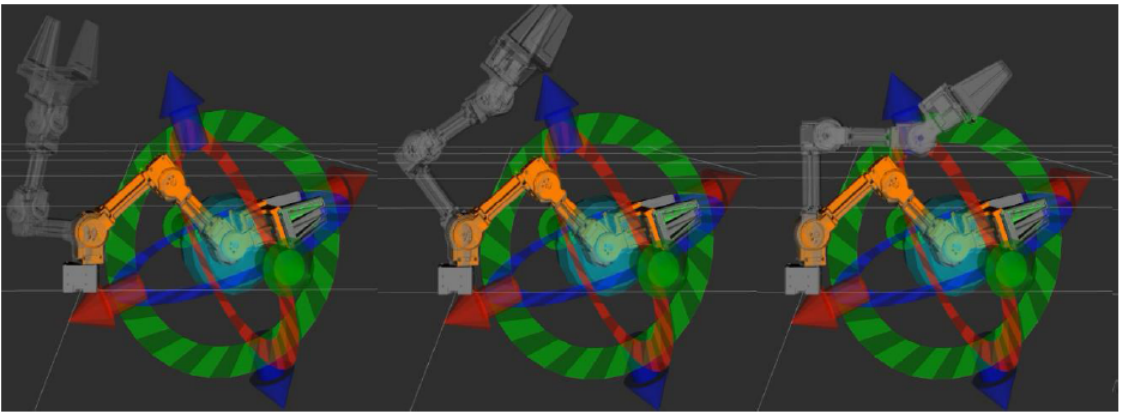
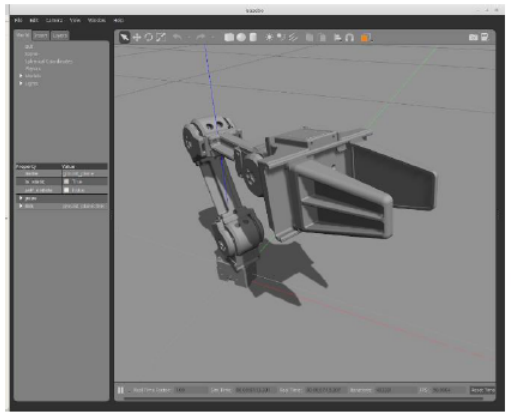
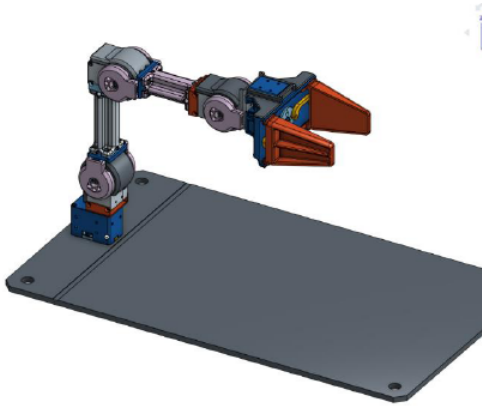






OpenManipulator

- Wiki page: http://emanual.robotis.com/docs/en/platform/openmanipulator/
- Open SW: https://github.com/ROBOTIS-GIT/open_manipulator
- Open HW
  - TurtleBot3 + OpenManipulator Chain
  - OpenManipulator Chain
  - OpenManipulator SCARA
  - OpenManipulator Link

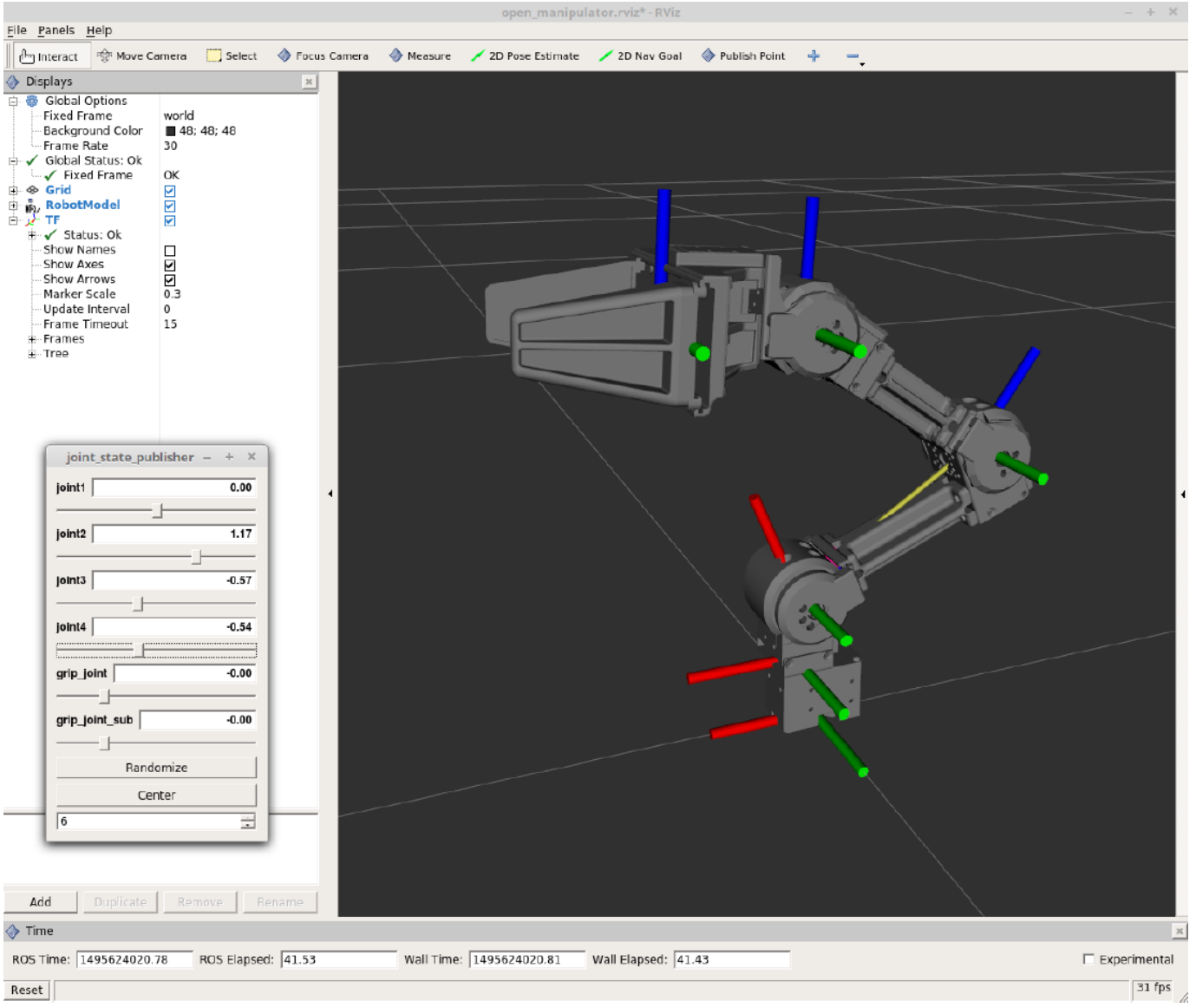
OpenManipulator & MoveIt! Install Dependencies

```
$ sudo apt-get install ros-kinetic-moveit*
$ sudo apt-get install ros-kinetic-gazebo*
$ sudo apt-get install ros-kinetic-industrial-core
$ sudo apt-get install ros-kinetic-dynamixel-sdk
$ sudo apt-get install ros-kinetic-ros-controllers
$ sudo apt-get install ros-kinetic-qt-build
$ sudo apt-get install ros-kinetic-robotis-math
```

```
$ cd ~/catkin_ws/src
$ git clone https://github.com/ROBOTIS-GIT/dynamixel-workbench.git
$ git clone https://github.com/ROBOTIS-GIT/dynamixel-workbench-msgs.git
$ git clone https://github.com/ROBOTIS-GIT/open_manipulator.git
$ cd ~/catkin_ws && catkin_make
```

OpenManipulator Modelling & Control Test

```
$ roslaunch open_manipulator_description open_manipulator_chain_rviz.launch
```

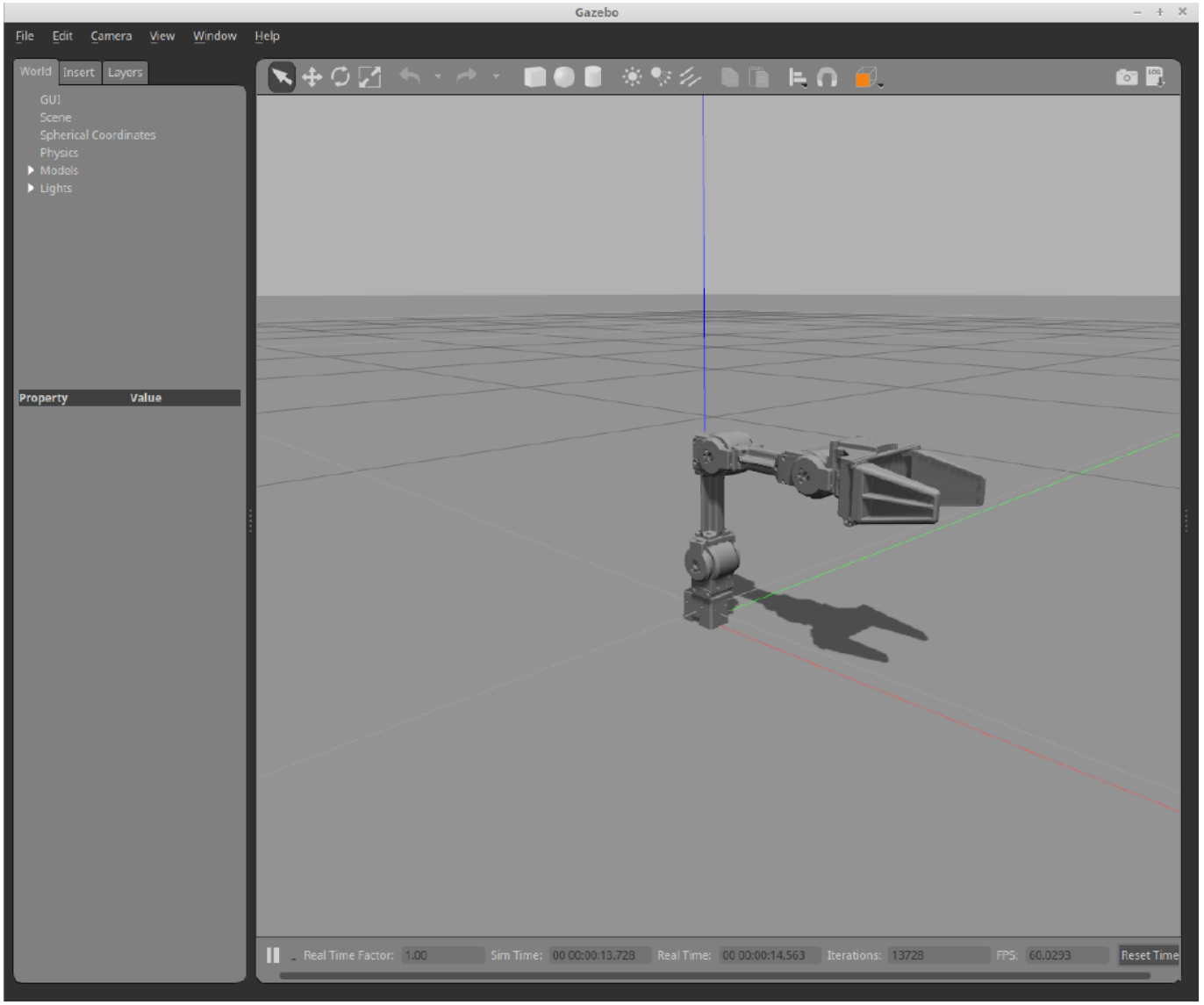


OpenManipulator Code Structure

```
$ cd ~/catkin_ws/src/open_manipulator
$ ls
Arduino                          → Library for Arduino
open_manipulator                 → MetaPackage
open_manipulator_description     → Modeling package
open_manipulator_dynamixel_ctrl  → Dynamixel control package
open_manipulator_gazebo          → Gazebo package
open_manipulator_moveit          → MoveIt! package
open_manipulator_msgs            → Message package
open_manipulator_position_ctrl   → Position control package
open_manipulator_with_tb3        → OpenManipulator and TurtleBot3 package

$ roscd open_manipulator_description/urdf && ls
materials.xacro                  → Material info
open_manipulator_chain.xacro     → Manipulator modeling
open_manipulator_chain.gazebo.xacro → Manipulator Gazebo modeling

$ roscd open_manipulator_description/launch && ls
open_manipulator.rviz            → RViz configuration file
open_manipulator_chain_ctrl.launch  → File to execute manipulator state info Publisher node
open_manipulator_chain_rviz.launch  → File to execute manipulator modeling info visualization node
```

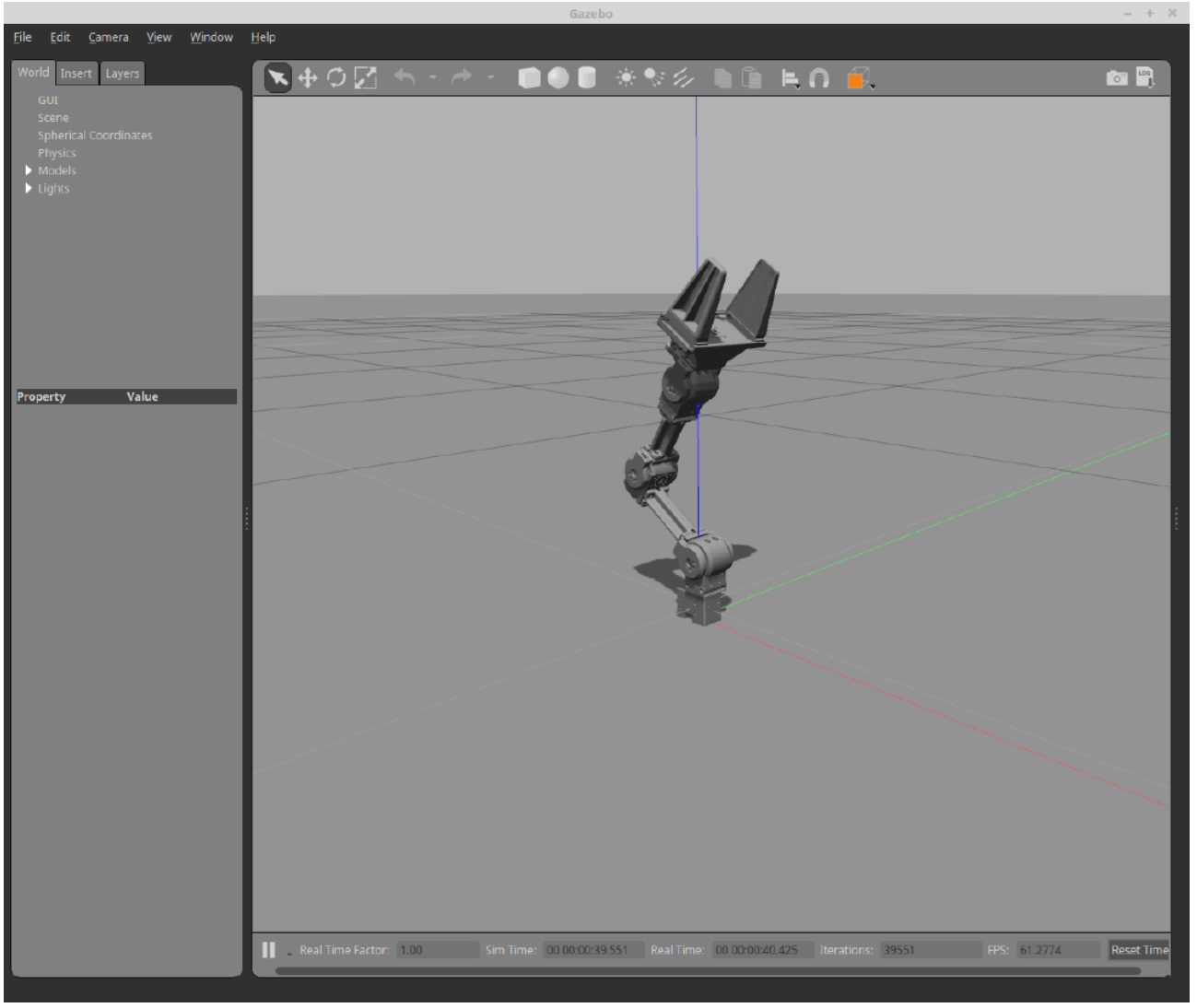
Run OpenManipulator in Gazebo

```
$ roslaunch open_manipulator_gazebo open_manipulator_gazebo.launch
```



Run OpenManipulator in Gazebo

```
$ rostopic pub /open_manipulator_chain/joint2_position/command
std_msgs/Float64 "data: 1.0" --once
```



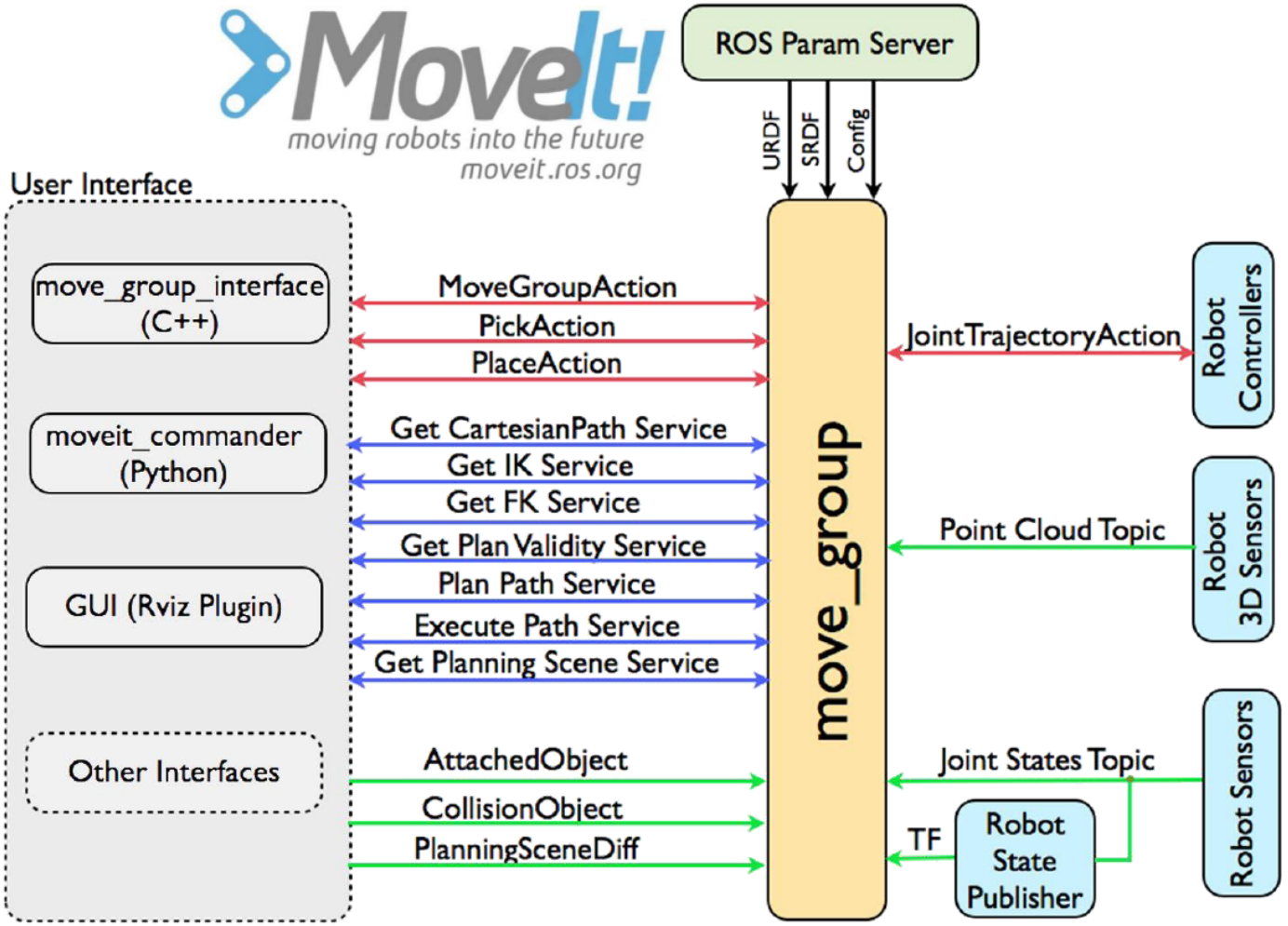
- Manipulator Introduction
- OpenManipulator Modeling and Simulation
- **MoveIt!**

What is ‘MoveIt!’?

- **Combined libraries for manipulator**
- It provides many functions like FIK(Fast Inverse Kinematics) for motion planning, advanced algorithm for manipulation, robot hand control, dynamics, controller, and so on.
- Robot description is loaded via **URDF** & **SRDF**
- It provides **GUI tool**
- **C++ based** move_group interface
- **Python based** move_commander
- **Motion Planning plugin** to RViz
- Many open-sourced libraries like OMPL, KDL, FCL, RRT

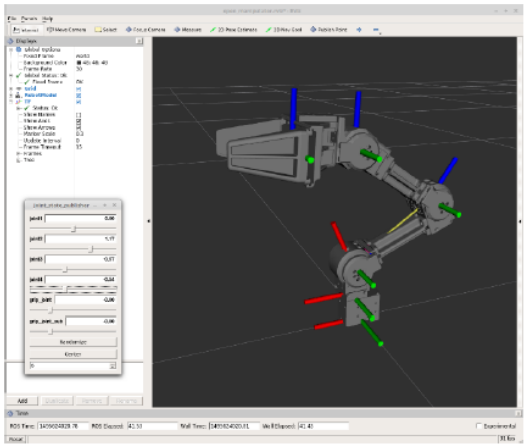
MoveIt! Includes S, P, and A.

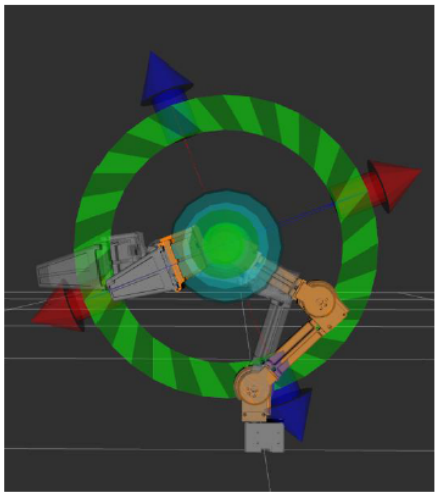
- Sensing
  - Joint Angle
  - Transformation
  - 3D sensing
- Plan
  - FK, IK
  - Grasping
  - Collision Avoidance
  - Motion Planning
- Action
  - POS/VEL control
  - Pick and Place



http://moveit.ros.org/documentation/concepts/

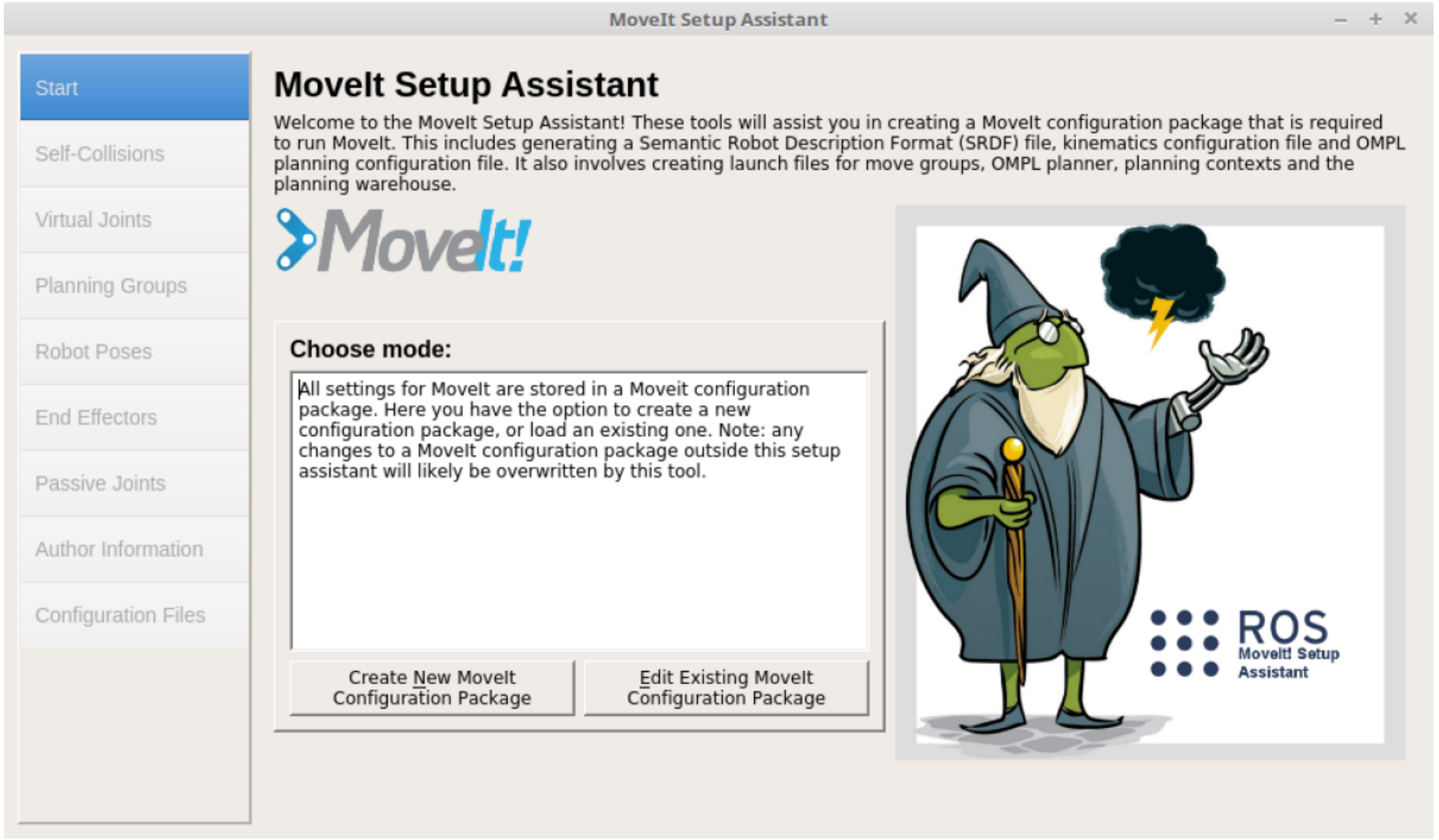
You can create MoveIt! Package by just providing URDF





MoveIt Setup Assistant

$ roslaunch moveit_setup_assistantsetup_assistant.launch



MoveIt Setup Assistant

- Start
- Self-Collisions
- Virtual Joints
- Planning Groups
- Robot Poses
- End Effectors
- Passive Joints
- Author Information
- Configuration Files

MoveIt Setup Assistant

Welcome to the MoveIt Setup Assistant! These tools will assist you in creating a MoveIt configuration package that is required to run MoveIt. This includes generating a Semantic Robot Description Format (SRDF) file, kinematics configuration file and OMPL planning configuration file. It also involves creating launch files for move groups, OMPL planner, planning contexts and the planning warehouse.

MoveIt!

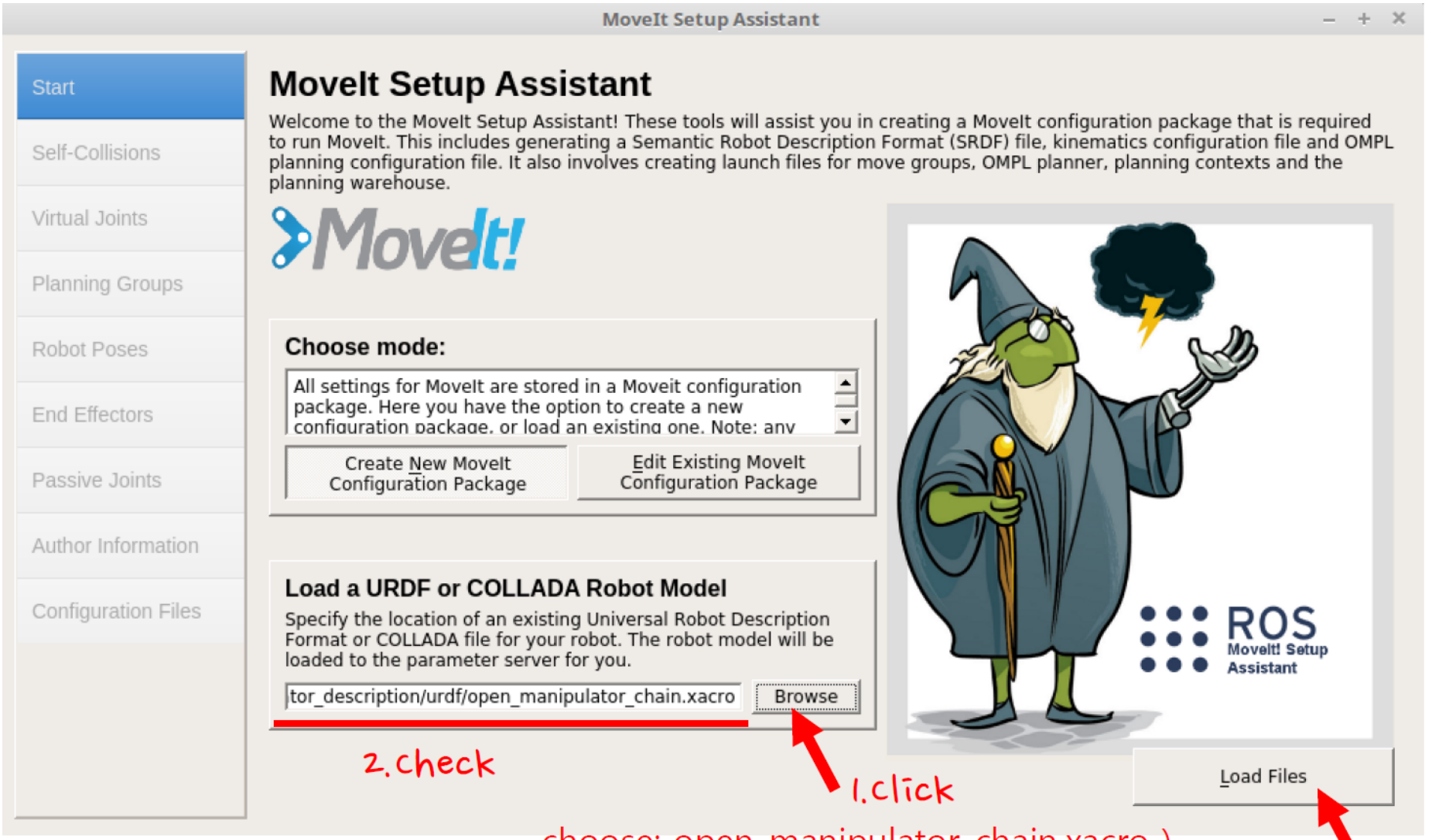
Choose mode:

All settings for MoveIt are stored in a Moveit configuration package. Here you have the option to create a new configuration package, or load an existing one. Note: any changes to a MoveIt configuration package outside this setup assistant will likely be overwritten by this tool.

Create New MoveIt Configuration Package | Edit Existing MoveIt Configuration Package

click

This wizard will help you^^


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Author Information
Configuration Files
MoveIt Setup Assistant
Welcome to the MoveIt Setup Assistant! These tools will assist you in creating a MoveIt configuration package that is required to run MoveIt. This includes generating a Semantic Robot Description Format (SRDF) file, kinematics configuration file and OMPL planning configuration file. It also involves creating launch files for move groups, OMPL planner, planning contexts and the planning warehouse.
MoveIt!
Choose mode:
All settings for MoveIt are stored in a MoveIt configuration package. Here you have the option to create a new configuration package, or load an existing one. Note: any
Create New MoveIt Configuration Package
Edit Existing MoveIt Configuration Package
Load a URDF or COLLADA Robot Model
Specify the location of an existing Universal Robot Description Format or COLLADA file for your robot. The robot model will be loaded to the parameter server for you.
tor_description/urdf/open_manipulator_chain.xacro
Browse
ROS
MoveIt! Setup Assistant
Load Files
2. check
1. click
choose: open_manipulator_chain.xacro)
3. click

MoveIt Setup Assistant

- Start
- Self-Collisions
- Virtual Joints
- Planning Groups
- Robot Poses
- End Effectors
- Passive Joints
- Author Information
- Configuration Files

Optimize Self-Collision Checking

The Default Self-Collision Matrix Generator will search for pairs of links on the robot that can safely be disabled from collision checking, decreasing motion planning processing time. These pairs of links are disabled when they are always in collision, never in collision, in collision in the robot's default position or when the links are adjacent to each other on the kinematic chain. Sampling density specifies how many random robot positions to check for self collision. Higher densities require more computation time.

1. click

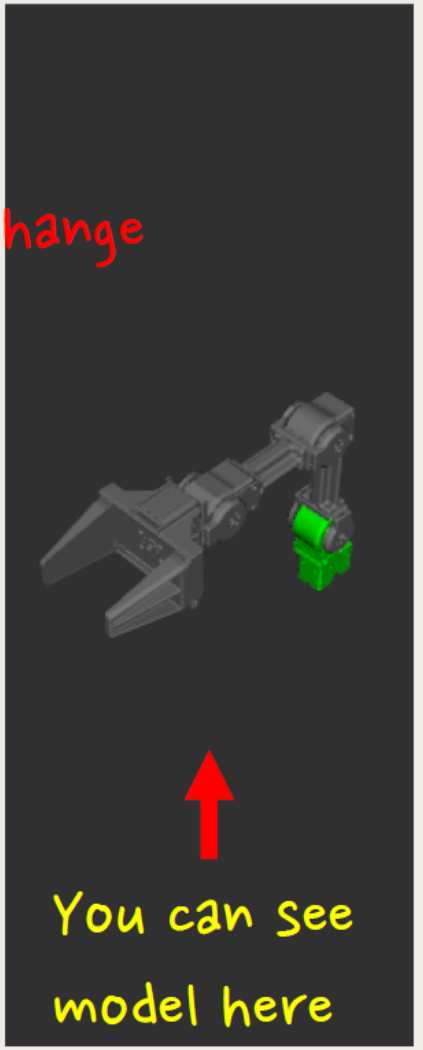
2. change

Sampling Density: Low — High 100000

Min. collisions for "always"-colliding pairs: 95% Generate Collision Matrix

3. click

	Link A	Link B	Disabled	eason to Disabl
1	link1	link2	☑	Adjacent Links
2	link2	link3	☑	Adjacent Links
3	link3	link4	☑	Adjacent Links
4	link4	link5	☑	Adjacent Links
5	link5	grip_link	☑	Adjacent Links
6	link5	grip_link_sub	☑	Adjacent Links



It will compute adjacent links and links that will never collide in advance.

MoveIt Setup Assistant

- Start
- Self-Collisions
- Virtual Joints
- Planning Groups
- Robot Poses
- End Effectors
- Passive Joints
- Author Information
- Configuration Files

Virtual Joints

Define a virtual joint between a robot link and an external frame of reference (considered fixed with respect to the robot).

Virtual Joint Name	Child Link	Parent Frame	Type

1. click

If there is no virtual joints, just skip it!

Delete Selected | Add Virtual Joint

MoveIt Setup Assistant

- Start
- Self-Collisions
- Virtual Joints
- Planning Groups
- Robot Poses
- End Effectors
- Passive Joints
- Author Information
- Configuration Files

Planning Groups

Create and edit planning groups for your robot based on joint collections, link collections, kinematic chains and subgroups.

Current Groups

1. click

2. click

Expand All Collapse All

Add Group

MoveIt Setup Assistant

- Start
- Self-Collisions
- Virtual Joints
- Planning Groups
- Robot Poses
- End Effectors
- Passive Joints
- Author Information
- Configuration Files

Planning Groups

Create and edit planning groups for your robot based on joint collections, link collections, kinematic chains and subgroups.

Create New Planning Group

Group Name:	arm
Kinematic Solver:	kdl_kinematics_plugin/KDLKinematicsPlugin
Kin. Search Resolution:	0.005
Kin. Search Timeout (sec):	0.005
Kin. Solver Attempts:	3

1.Input

Next, Add Components To Group:

Recommended:

Add Joints

2. click

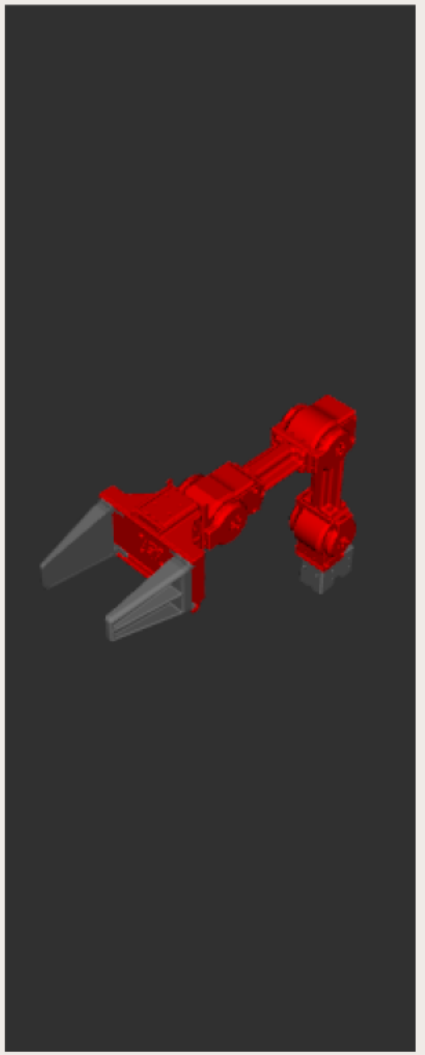
Advanced Options:

Add Links

Add Kin. Chain

Add Subgroups

Cancel



MoveIt Setup Assistant

- Start
- Self-Collisions
- Virtual Joints
- Planning Groups
- Robot Poses
- End Effectors
- Passive Joints
- Author Information
- Configuration Files

Planning Groups

Create and edit planning groups for your robot based on joint collections, link collections, kinematic chains and subgroups.

Edit 'arm' Joint Collection

Available Joints

	Joint Names
1	world_fixed
2	joint1
3	joint2
4	joint3
5	joint4
6	grip_joint
7	grip_joint_sub

1.choose

> 2. click

<

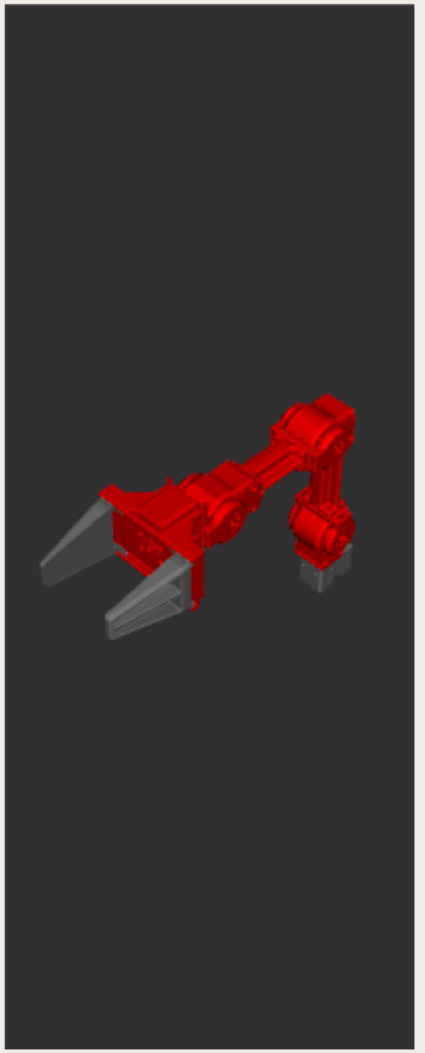
Selected Joints

	Joint Names
1	joint1
2	joint2
3	joint3
4	joint4

3.check

4. click

Save Cancel



MoveIt Setup Assistant

- Start
- Self-Collisions
- Virtual Joints
- Planning Groups
- Robot Poses
- End Effectors
- Passive Joints
- Author Information
- Configuration Files

Planning Groups

Create and edit planning groups for your robot based on joint collections, link collections, kinematic chains and subgroups.

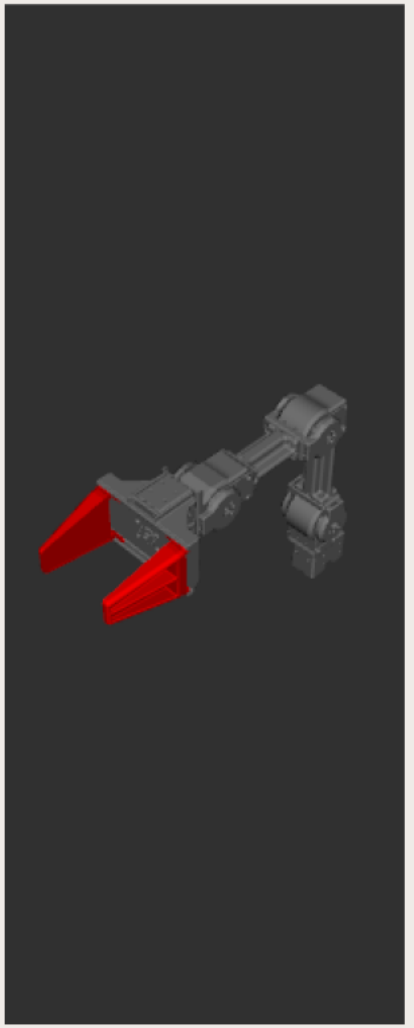
Current Groups

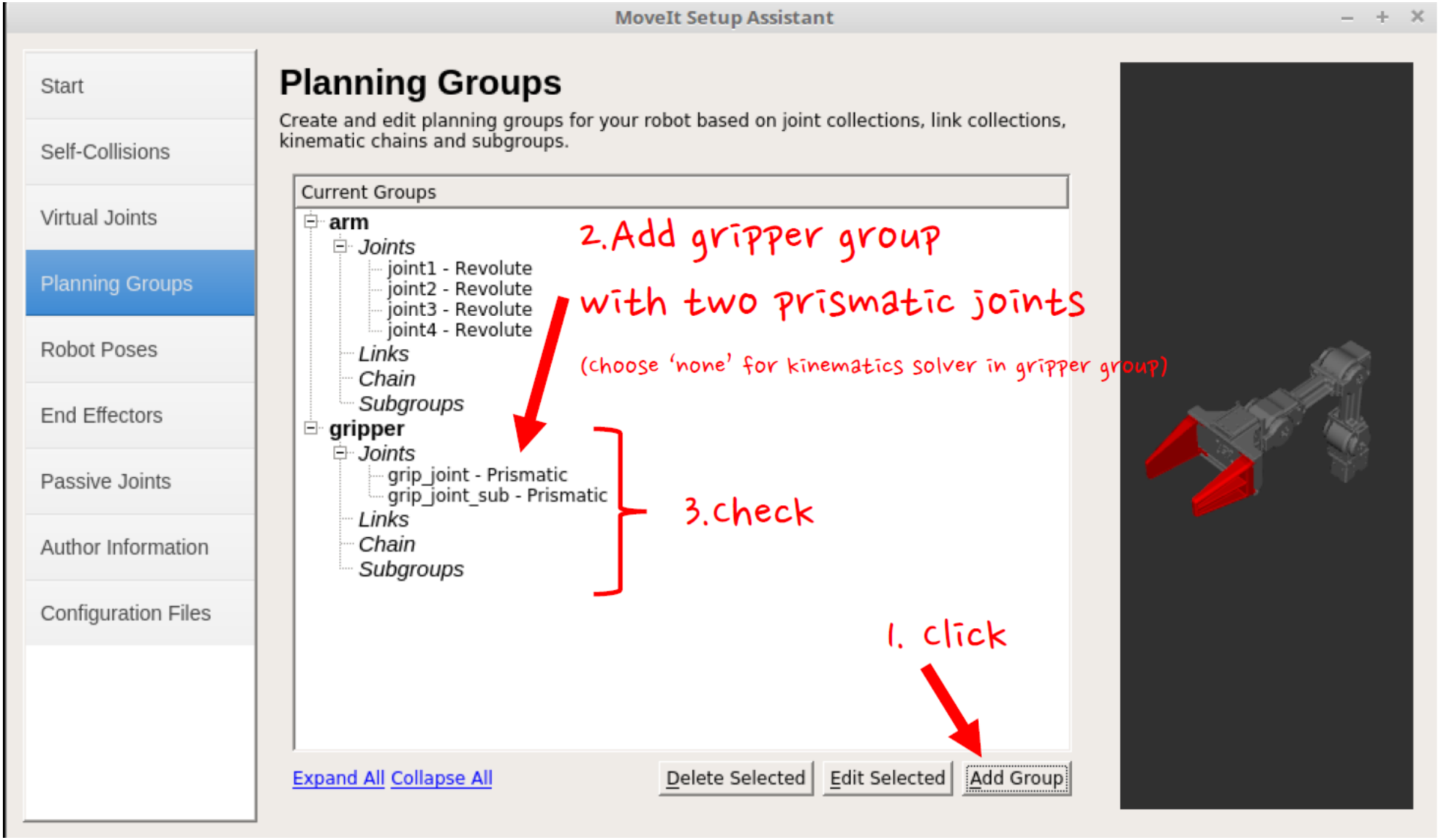
- **arm**
  - *Joints*
    - joint1 - Revolute
    - joint2 - Revolute
    - joint3 - Revolute
    - joint4 - Revolute
  - *Links*
  - *Chain*
  - *Subgroups*

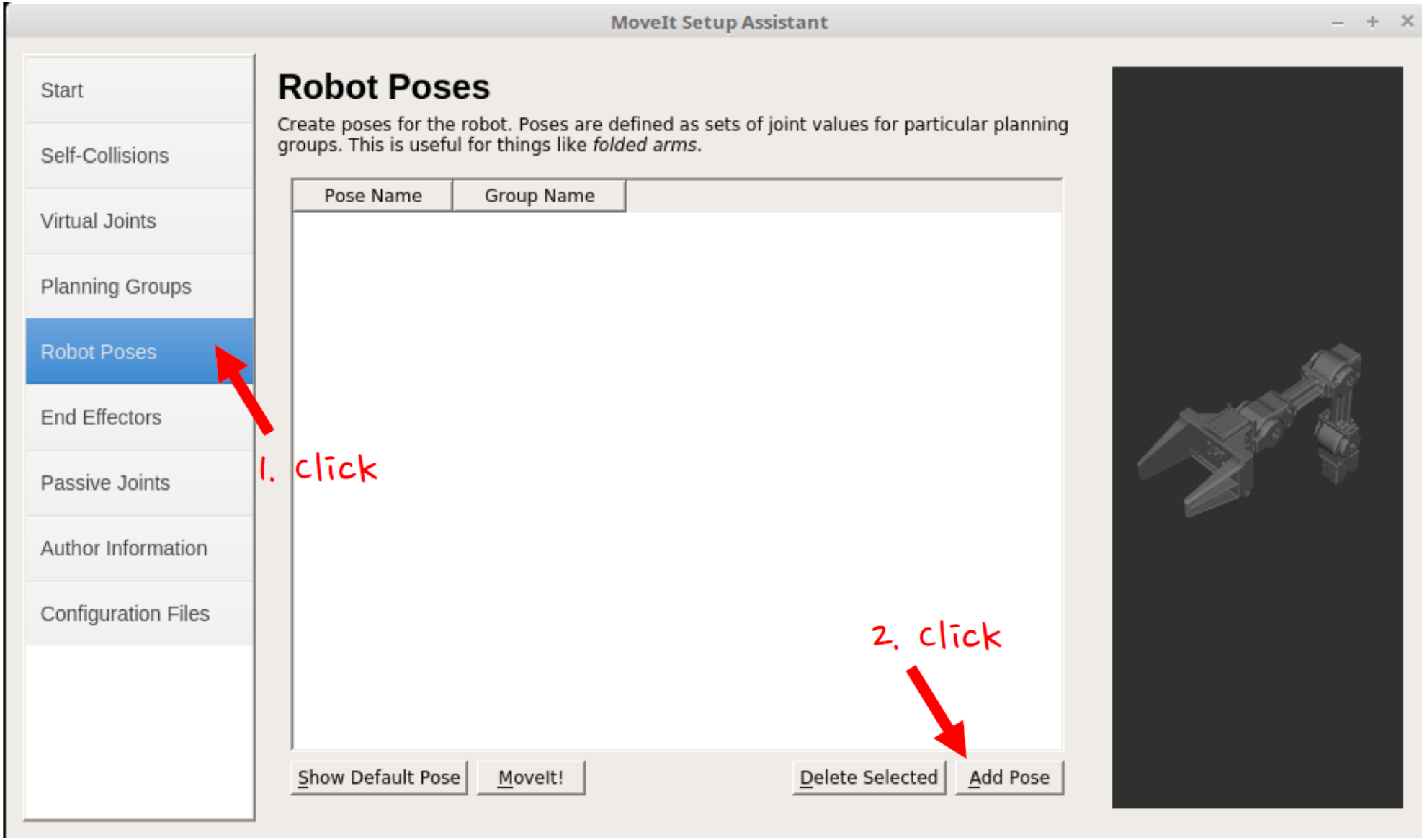
check

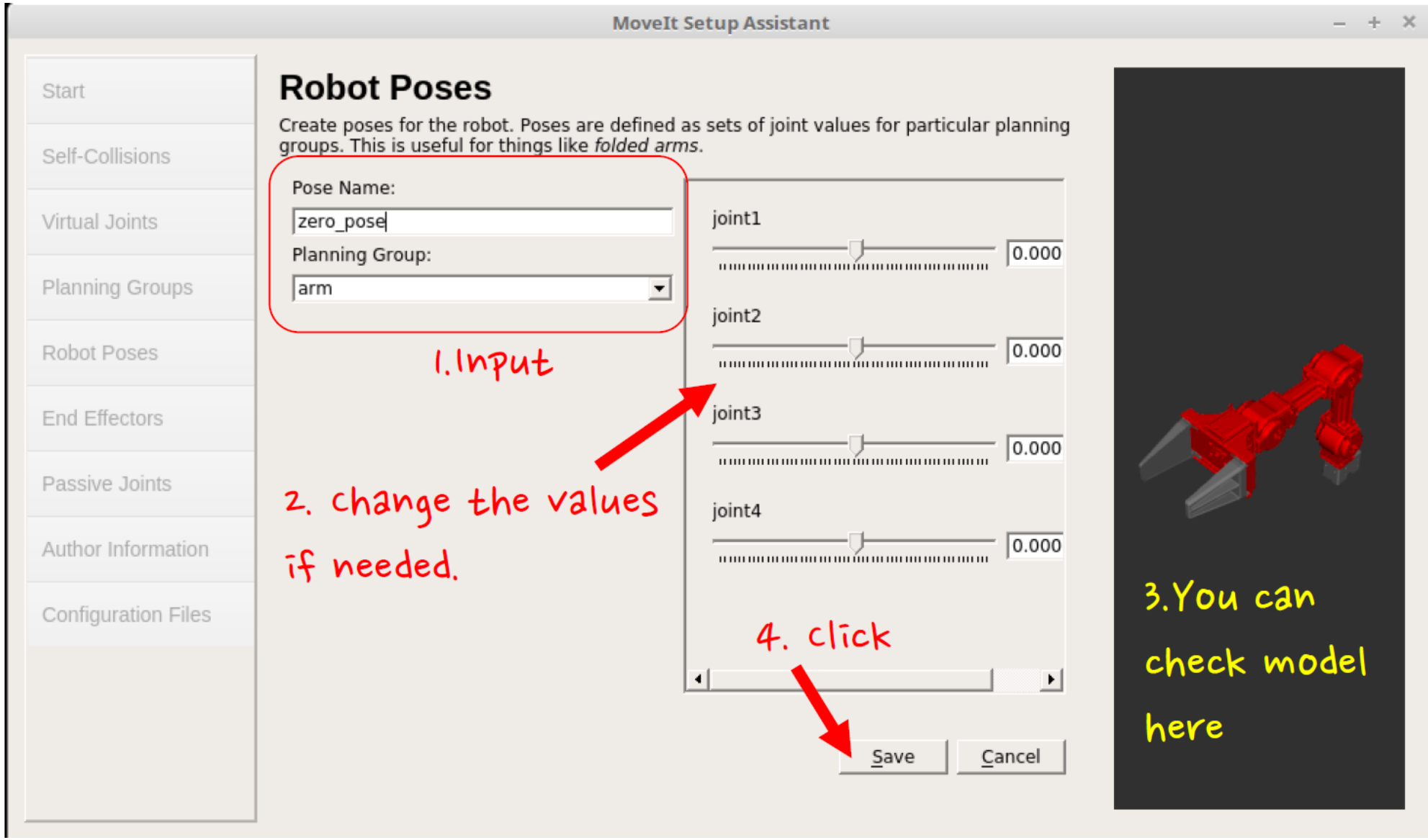
Expand All Collapse All

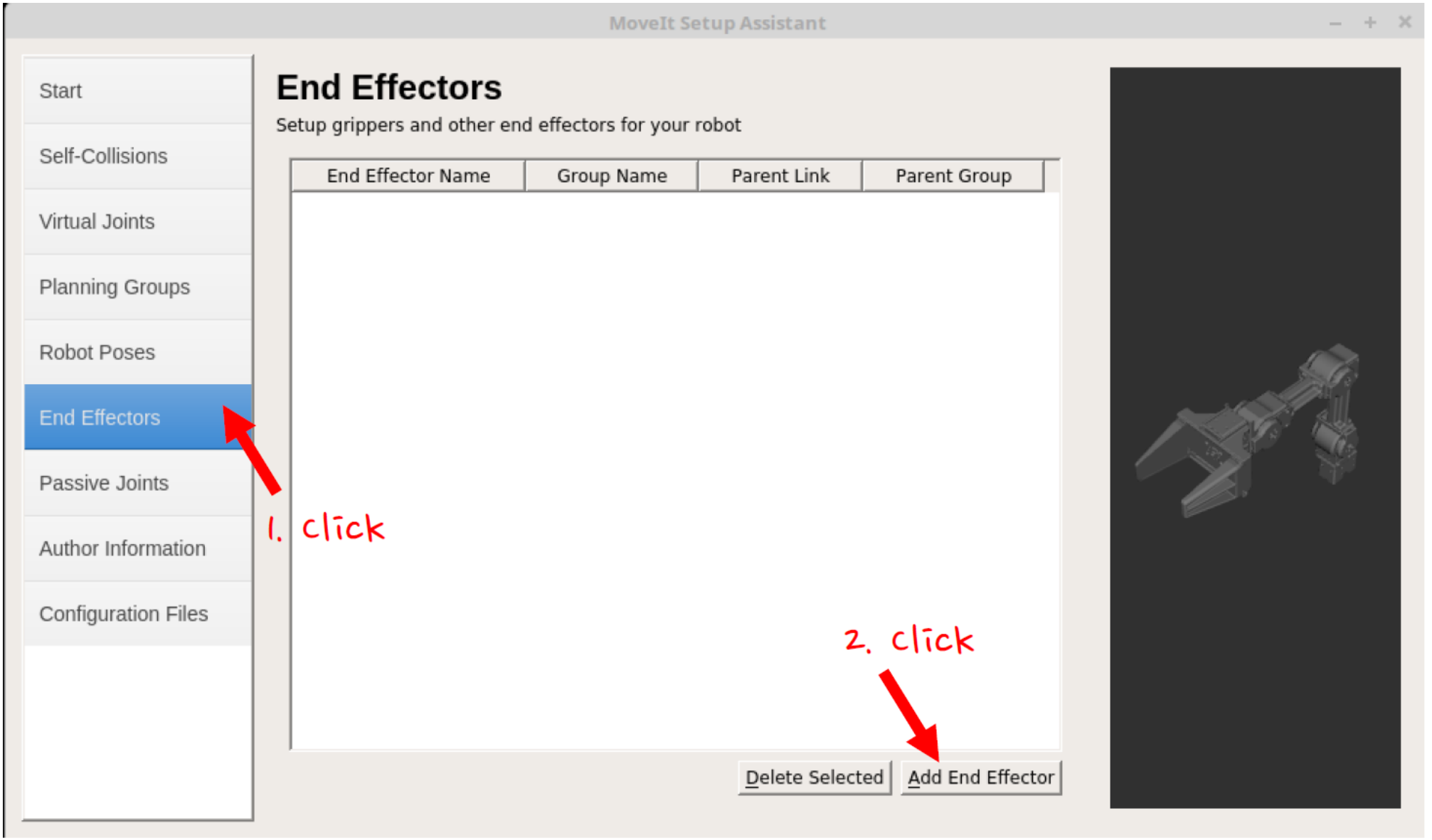
Delete Selected | Edit Selected | Add Group




MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Author Information
Configuration Files
Planning Groups
Create and edit planning groups for your robot based on joint collections, link collections, kinematic chains and subgroups.
Current Groups
arm
Joints
joint1 - Revolute
joint2 - Revolute
joint3 - Revolute
joint4 - Revolute
Links
Chain
Subgroups
gripper
Joints
grip_joint - Prismatic
grip_joint_sub - Prismatic
Links
Chain
Subgroups
2.Add gripper group
with two prismatic joints
(choose 'none' for kinematics solver in gripper group)
3.check
1. click
Expand All
Collapse All
Delete Selected
Edit Selected
Add Group


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Author Information
Configuration Files
Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like folded arms.
Pose Name
Group Name
1. click
2. click
Show Default Pose
MoveIt!
Delete Selected
Add Pose


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Author Information
Configuration Files
Robot Poses
Create poses for the robot. Poses are defined as sets of joint values for particular planning groups. This is useful for things like folded arms.
Pose Name:
zero_pose
Planning Group:
arm
1.Input
joint1
0.000
joint2
0.000
joint3
0.000
joint4
0.000
2. change the values if needed.
4. click
Save
Cancel
3.You can check model here


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Author Information
Configuration Files
End Effectors
Setup grippers and other end effectors for your robot
End Effector Name
Group Name
Parent Link
Parent Group
1. click
2. click
Delete Selected
Add End Effector

MoveIt Setup Assistant

- Start
- Self-Collisions
- Virtual Joints
- Planning Groups
- Robot Poses
- End Effectors
- Passive Joints
- Author Information
- Configuration Files

End Effectors

Setup grippers and other end effectors for your robot

End Effector Name:

linear_gripper

End Effector Group:

gripper

Parent Link (usually part of the arm):

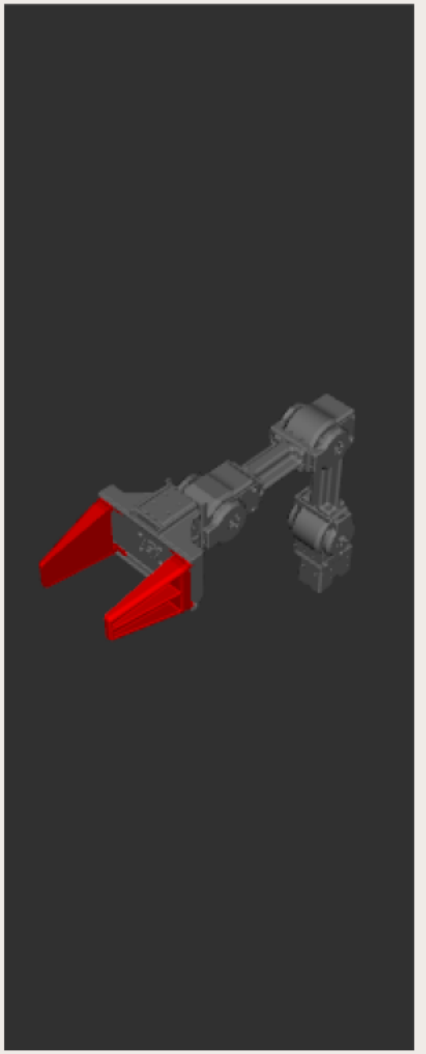
link5

Parent Group (optional):

1. Input

2. click

Save Cancel



MoveIt Setup Assistant

- Start
- Self-Collisions
- Virtual Joints
- Planning Groups
- Robot Poses
- End Effectors
- Passive Joints
- Author Information
- Configuration Files

Passive Joints

Specify the set of passive joints (not actuated). Joint state is not expected to be published for these joints.

Active Joints

	Joint Names
1	joint1
2	joint2
3	joint3
4	joint4
5	grip_joint
6	grip_joint_sub

>

<

Passive Joints

Joint Names

1. click

2. If there is no passive joint, then skip it!

MoveIt Setup Assistant

- Start
- Self-Collisions
- Virtual Joints
- Planning Groups
- Robot Poses
- End Effectors
- Passive Joints
- Author Information
- Configuration Files

Author Information

Specify contact information of the author and initial maintainer of the generated package. catkin requires valid details in the package's package.xml

Name of the maintainer this MoveIt! configuration:

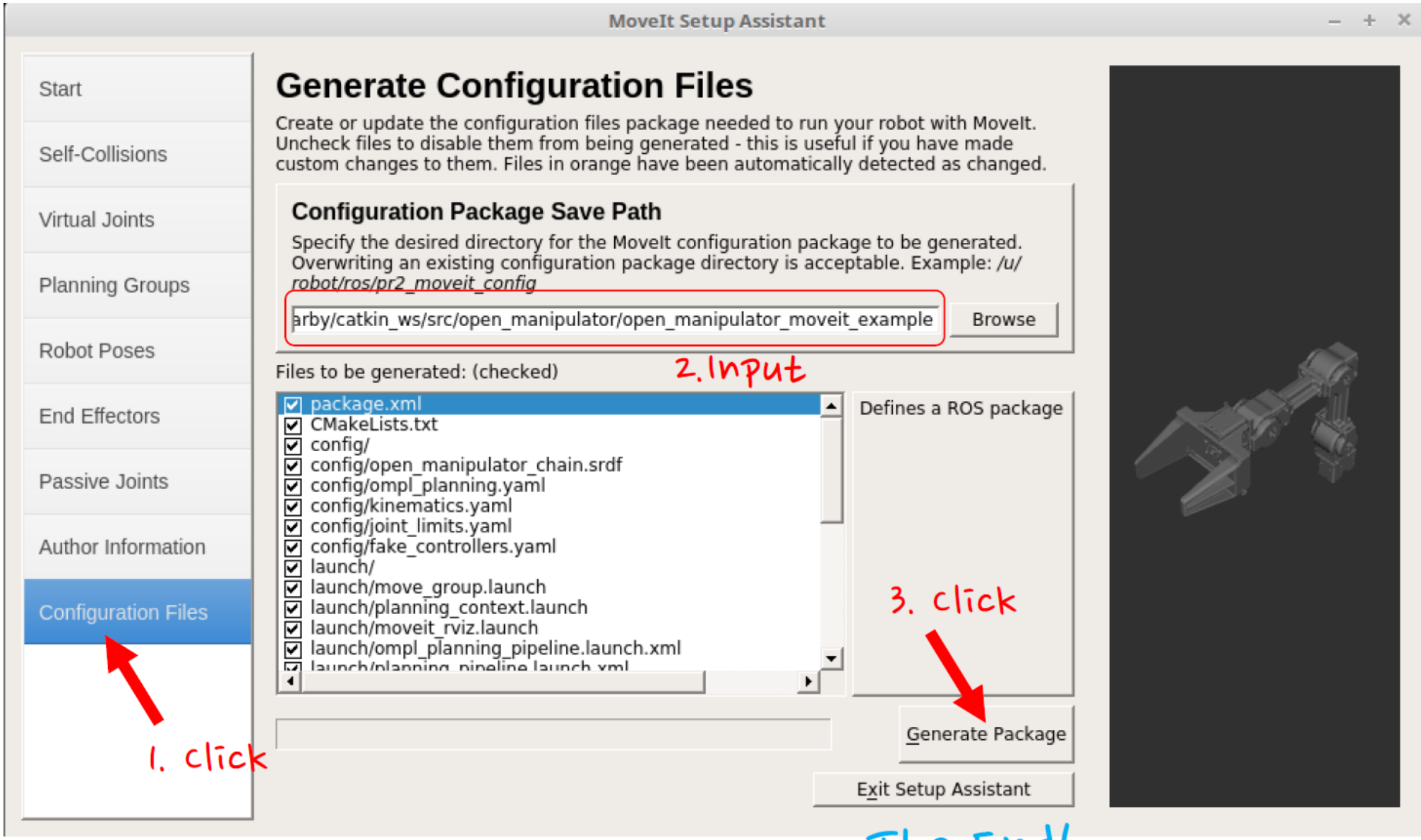
User

Email of the maintainer of this MoveIt! configuration:

User@User.com

2. Input

1. click


MoveIt Setup Assistant
Start
Self-Collisions
Virtual Joints
Planning Groups
Robot Poses
End Effectors
Passive Joints
Author Information
Configuration Files
1. click
Generate Configuration Files
Create or update the configuration files package needed to run your robot with MoveIt.
Uncheck files to disable them from being generated - this is useful if you have made
custom changes to them. Files in orange have been automatically detected as changed.
Configuration Package Save Path
Specify the desired directory for the MoveIt configuration package to be generated.
Overwriting an existing configuration package directory is acceptable. Example: /u/
robot/ros/pr2_moveit_config
arby/catkin_ws/src/open_manipulator/open_manipulator_moveit_example
Browse
2. Input
Files to be generated: (checked)
package.xml
CMakeLists.txt
config/
config/open_manipulator_chain.srdf
config/ompl_planning.yaml
config/kinematics.yaml
config/joint_limits.yaml
config/fake_controllers.yaml
launch/
launch/move_group.launch
launch/planning_context.launch
launch/moveit_rviz.launch
launch/ompl_planning_pipeline.launch.xml
Defines a ROS package
3. click
Generate Package
Exit Setup Assistant
The End!

Run OpenManipulator Demo

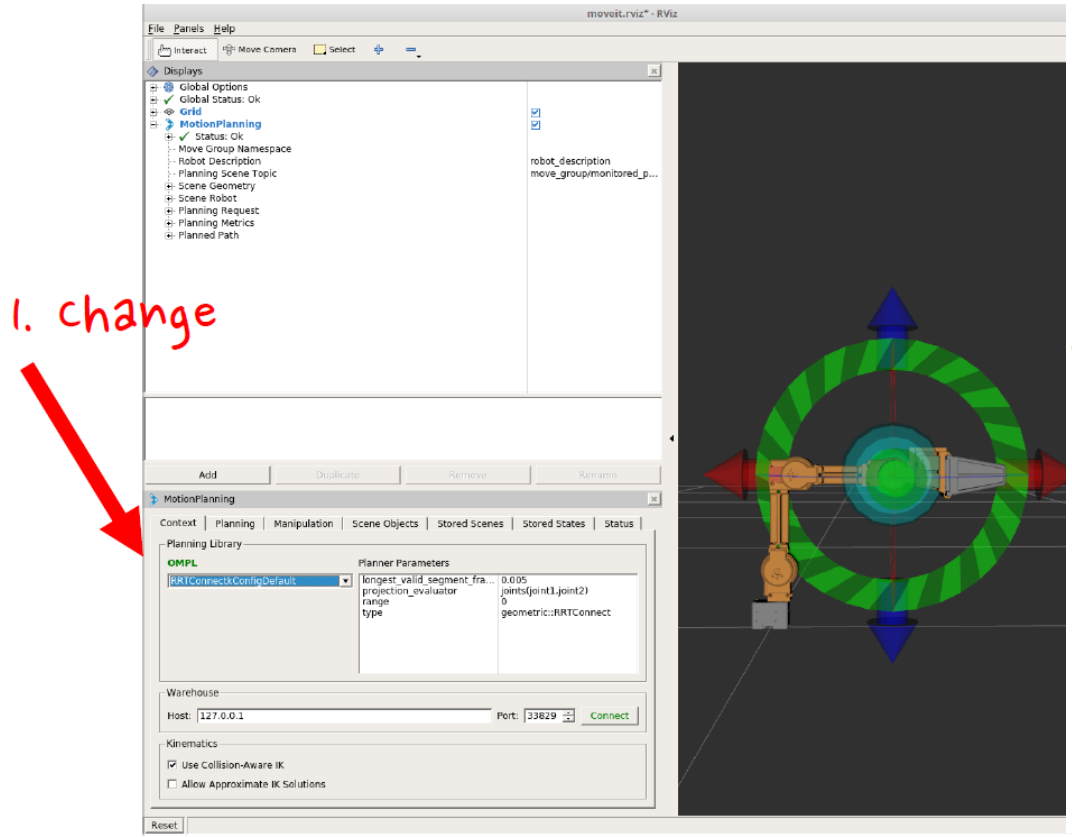
```
$ cd ~/catkin_ws/src/open_manipulator/open_manipulator_moveit_example
$ ls
Config            → MoveIt! yaml & SRDF files for robot configuration
Launch            → launch file
.setup_assistant  → Package information created by setup assistant
CmakeLists.txt    → CMake file
package.xml       → Package configuration file

$ cd ~/catkin_ws && catkin_make
```

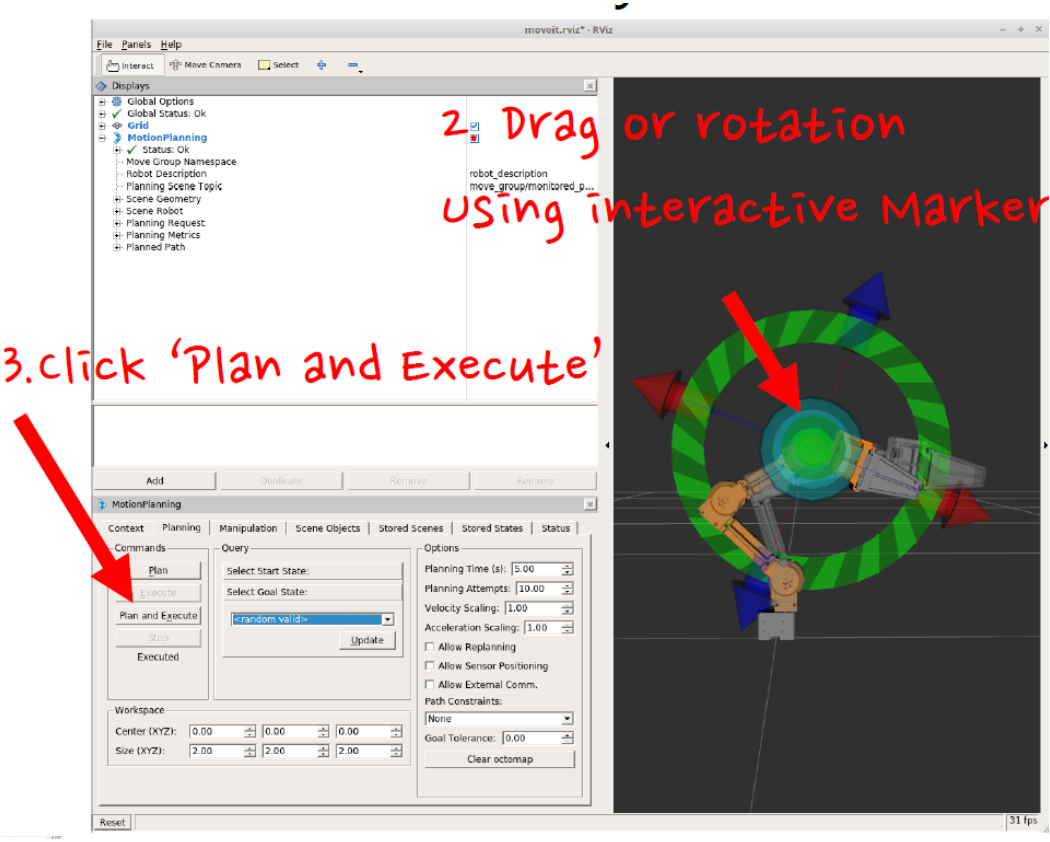
```
$ roslaunch open_manipulator_moveit_example demo.launch
```

Run OpenManipulator Demo

- Context > Planning Library > OMPL
- You can choose planning library
- Choose RRTConnectkConfigDefault

- Locate end-effector
- Click ‘Plan & Excute’ button
- Check how they move

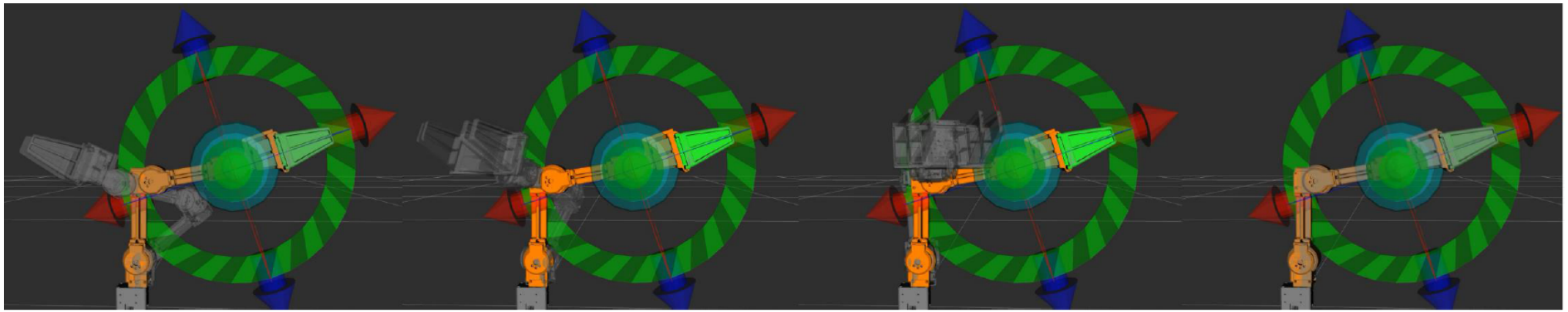


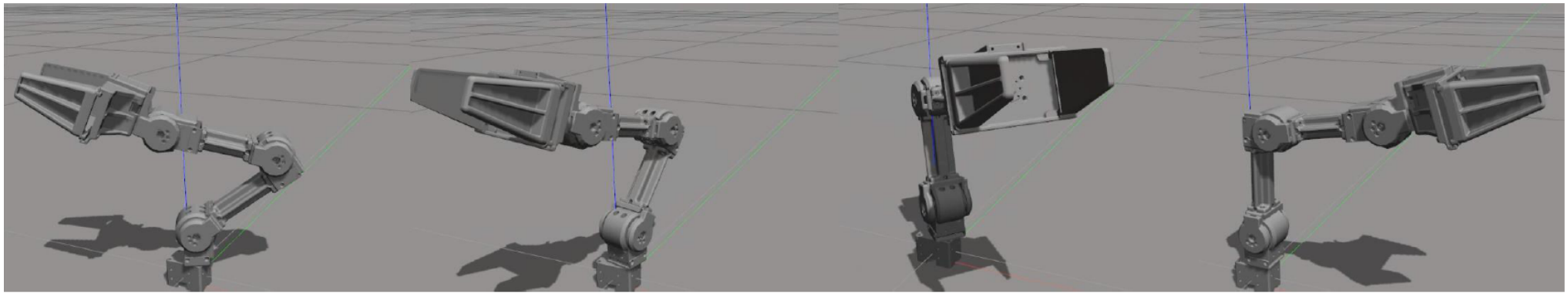


MoveIt! + Gazebo : Move to the goal position

```
$ roslaunch open_manipulator_gazebo open_manipulator_gazebo.launch
```

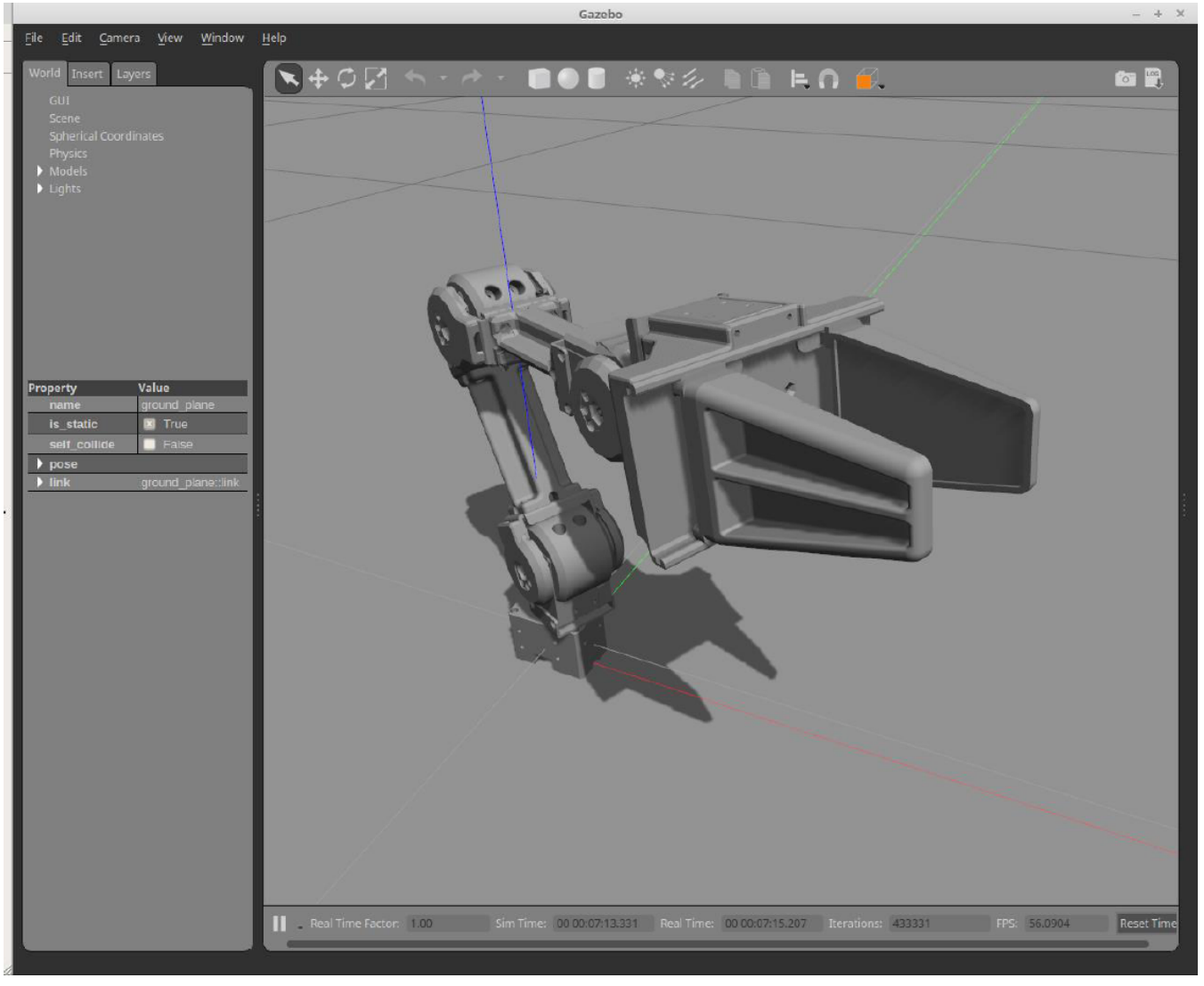
```
$ roslaunch open_manipulator_moveit open_manipulator_demo.launch use_gazebo:=true
```





MoveIt! + Gazebo : Gripper control

```
$ rostopic pub /robotis/open_manipulator/gripper std_msgs/String "data: 'grip_on'" --once
```



OpenManipulator + TurtleBot3 Waffle

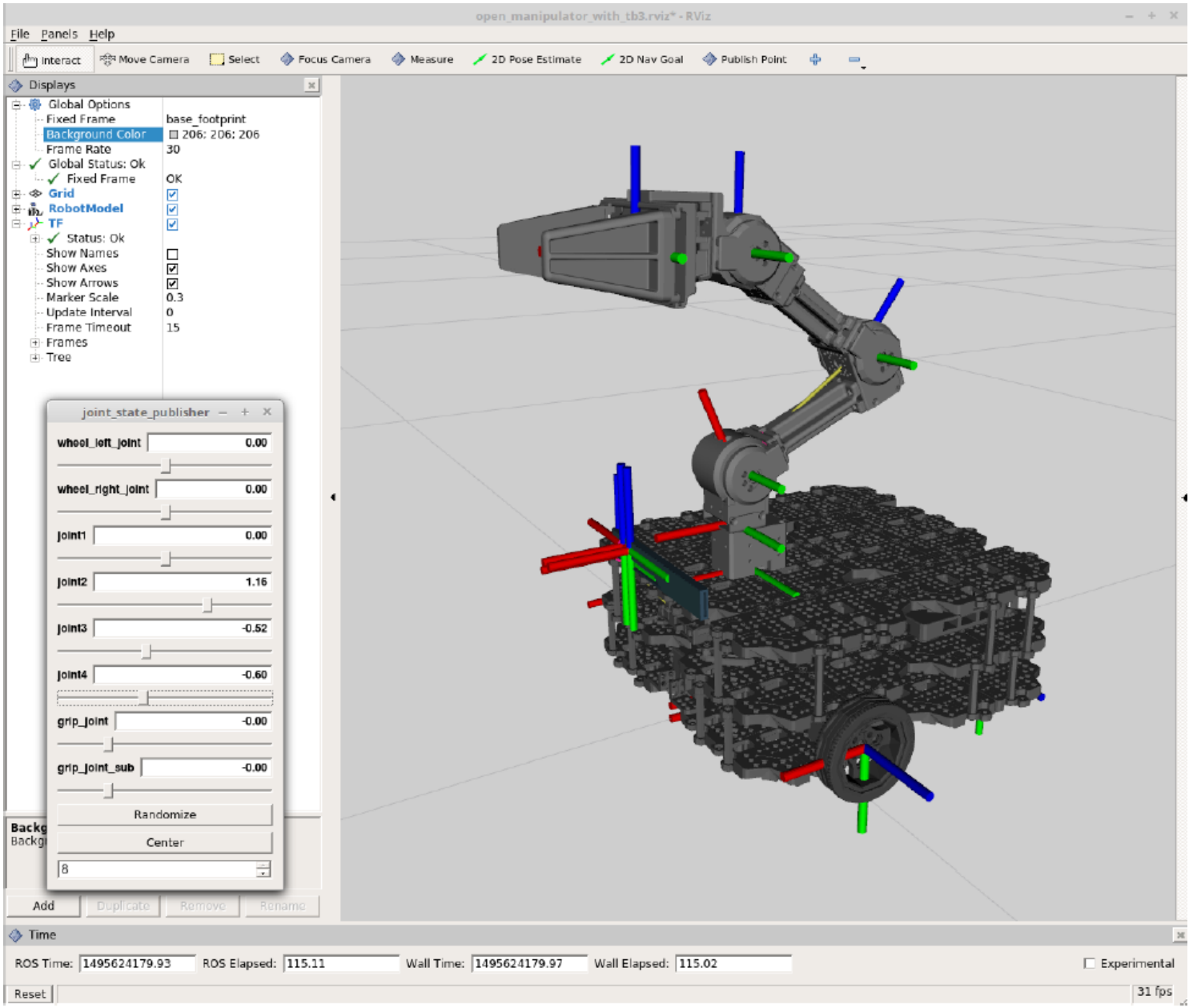
```
$ roscd open_manipulator_with_tb3/urdf
$ gedit open_manipulator_chain_with_tb3.xacro
```

```
<!-- Include TurtleBot3 Waffle URDF -->
<xacro:include filename="$(find
turtlebot3_description)/urdf/turtlebot3_waffle_naked.urdf.xacro" />

<!-- Base fixed joint -->
<joint name="base_fixed" type="fixed">
  <origin xyz="-0.005 0.0 0.091" rpy="0 0 0"/>
  <parent link="base_link"/>
  <child link="link1"/>
</joint>
```

Insert ‘OpenManipulator’ to TurtleBot3

```
$ roslaunch open_manipulator_with_tb3
open_manipulator_chain_with_tb3_rviz.launch
```

OpenManipulator + TurtleBot3 Waffle



Reference

- **Book**

  - ROS_Robot_Programming_EN
- http://wiki.ros.org/Documentation
- http://wiki.ros.org/ROS/Tutorials

Question Time!