

CSC398 Autonomous Robots

- Introduction into ROS (3) -

Ubbo Visser

Department of Computer Science
College of Arts and Sciences
University of Miami

September 2024



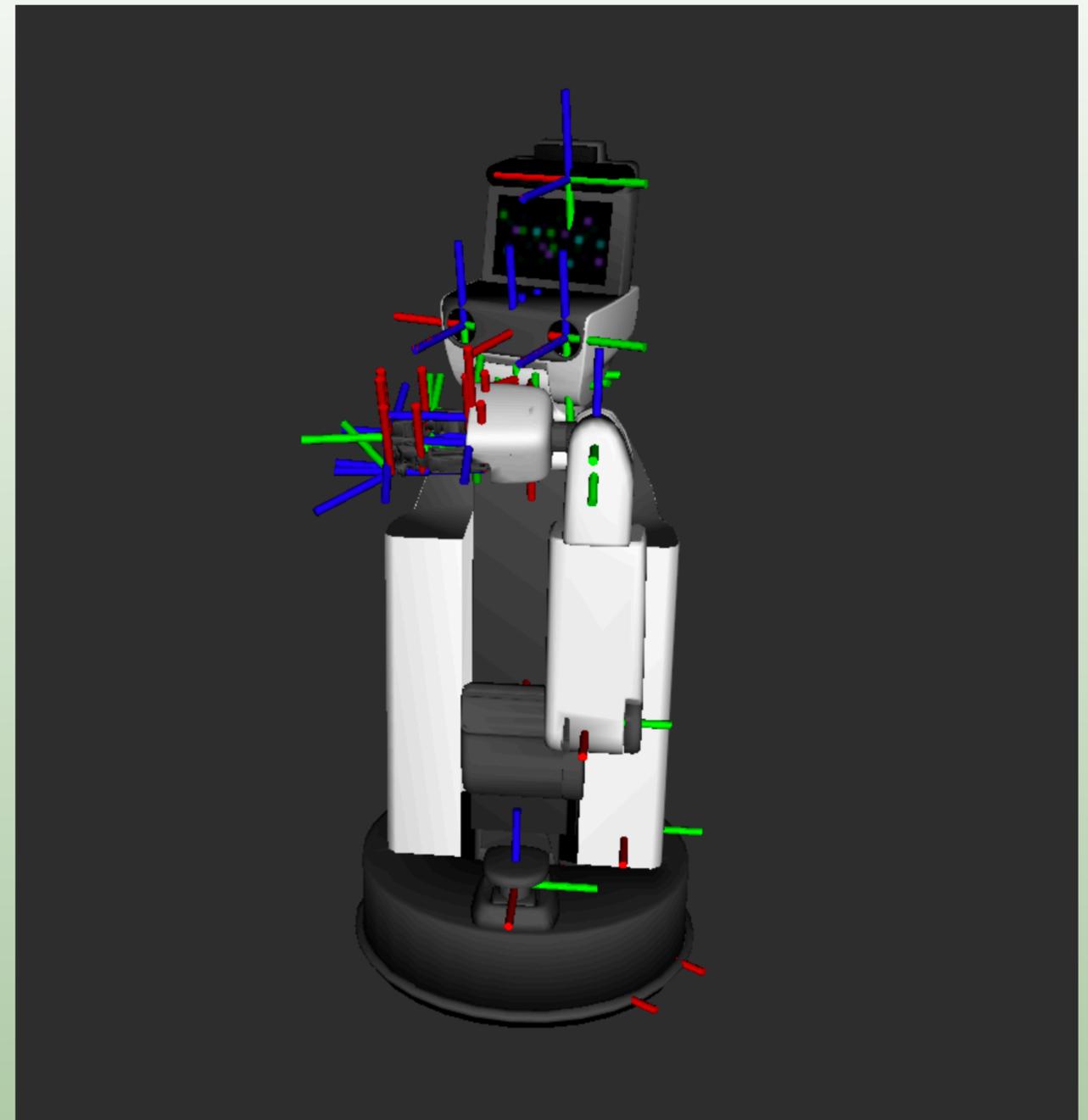
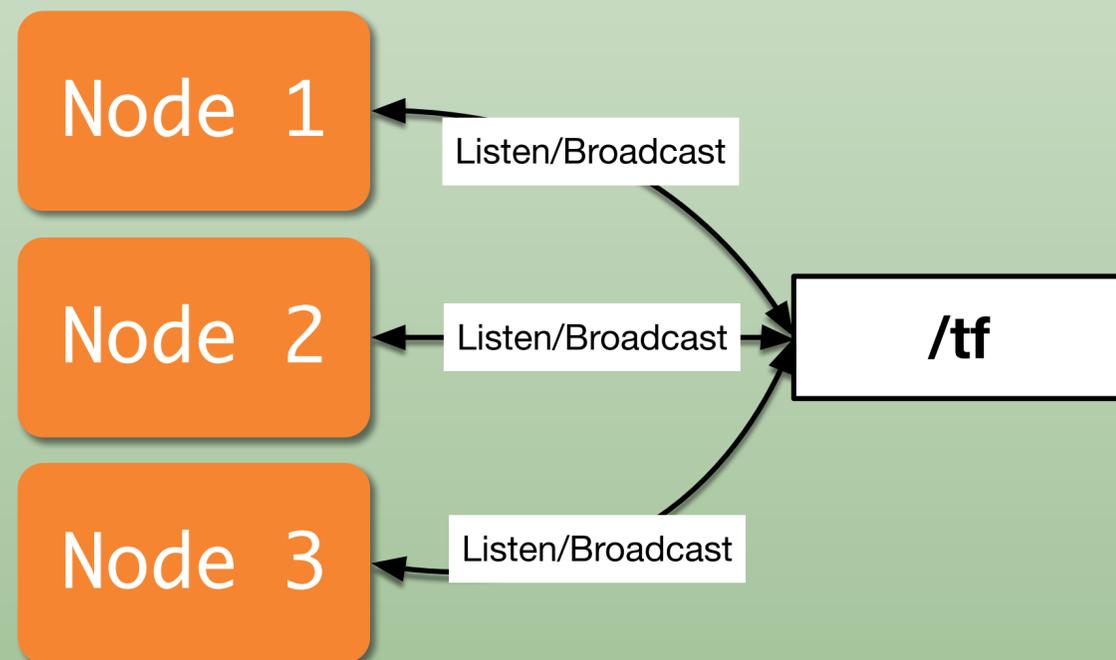
UNIVERSITY OF MIAMI
ROBOCANES



- ▶ TF Transformation System
- ▶ rqt User Interface
- ▶ Robot models (URDF)
- ▶ Simulation descriptions (SDF)



- ▶ Tool for keeping track of coordinate frames over time
- ▶ Maintains relationship between coordinate frames in a tree structure buffered in time
- ▶ Lets the user transform points, vectors, etc. between coordinate frames at desired time
- ▶ Implemented as publisher/subscriber model on the topics `/tf` and `/tf_static`



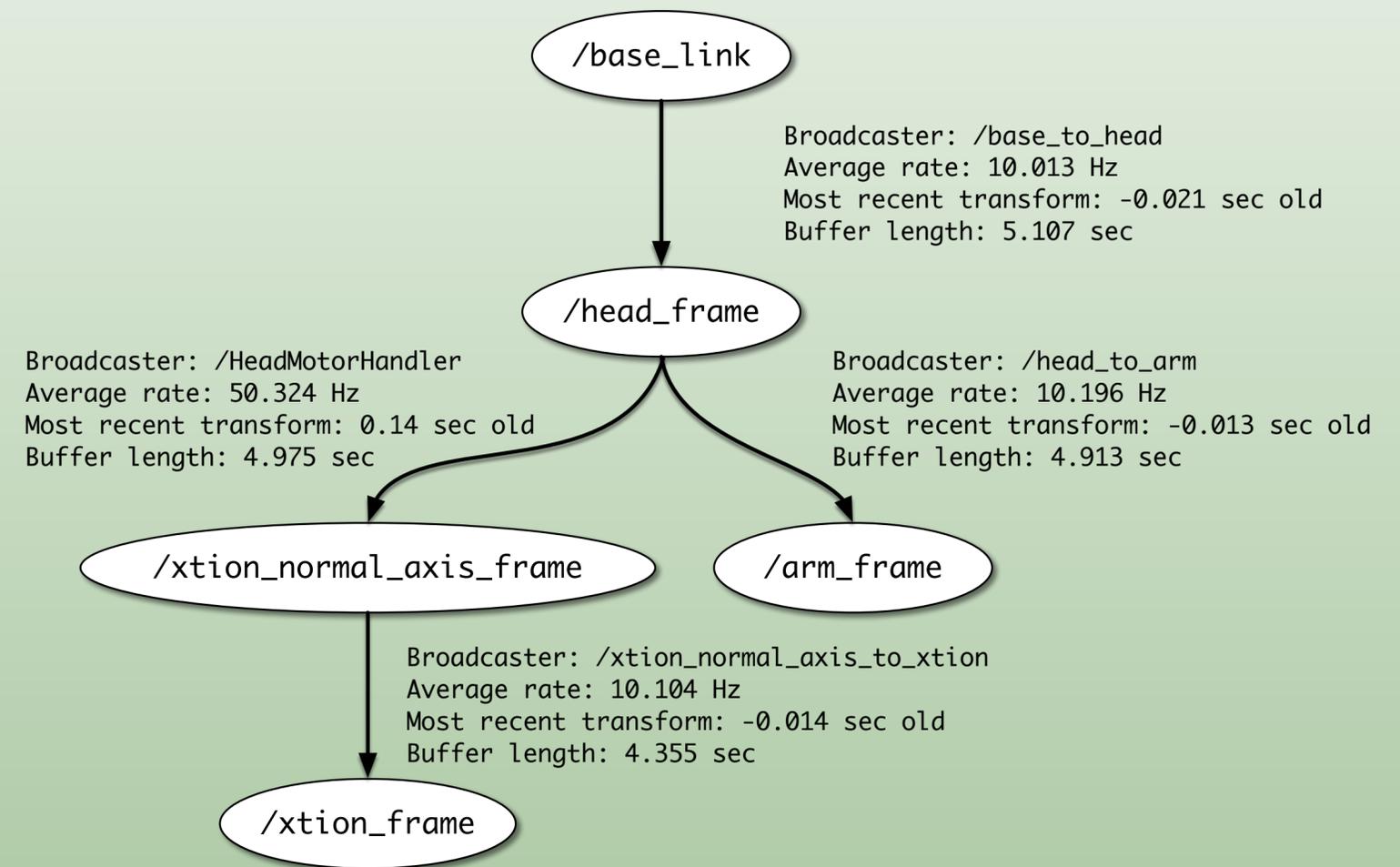
Details at: <http://wiki.ros.org/tf2>

TF TRANSFORMATION SYSTEM - TRANSFORM TREE

- ▶ TF listeners use a buffer to listen to all broadcasted transforms
- ▶ Query for specific transforms from the transform tree

tf2_msgs/TFMessage.msg

```
geometry_msgs/TransformStamped[] transforms
  std_msgs/Header header
    uint32 seqtime stamp
  string frame_id
  string child_frame_id
  geometry_msgs/Transform transform
    geometry_msgs/Vector3 translation
    geometry_msgs/Quaternion rotation
```



TF TRANSFORMATION SYSTEM - TRANSFORM TREE

- ▶ TF listeners use a buffer to listen to all broadcasted transforms
- ▶ Query for specific transforms from the transform tree

tf2_msgs/TFMessage.msg

```
geometry_msgs/TransformStamped[] transforms
  std_msgs/Header header
    uint32 seqtime stamp
  string frame_id
  string child_frame_id
  geometry_msgs/Transform transform
    geometry_msgs/Vector3 translation
    geometry_msgs/Quaternion rotation
```

Partial link tree from HSRB, torso

```
▼ torso_lift_link
  ▸ Details
  ▼ head_pan_link
    ▸ Details
    ▼ head_tilt_link
      ▸ Details
      ▼ head_center_camera_frame
        ▸ Details
        ▼ head_center_camera_gazebo_frame
          ▸ Details
        ▼ head_l_stereo_camera_link
          ▸ Details
          ▼ head_l_stereo_camera_gazebo_frame
            ▸ Details
        ▼ head_r_stereo_camera_link
          ▸ Details
          ▼ head_r_stereo_camera_gazebo_frame
            ▸ Details
        ▼ head_rgbd_sensor_link
          ▸ Details
          Alpha 
          Show Trail 
          Show Axes 
          Position
          X
          Y
          Z
          Orientation
          X
          Y
          Z
          W
          ▼ head_rgbd_sensor_gazebo_frame
            ▸ Details
```

1

-0.059954; 0.022008; 0.99209
-0.0599536
0.0220083
0.992095
-0.49975; 0.49977; -0.50025; 0.50023
-0.499747
0.499769
-0.500253
0.500231

Details at: http://docs.ros.org/jade/api/tf2_msgs/html/msg/TFMessage.html

Terminal

Get information about the current transform tree

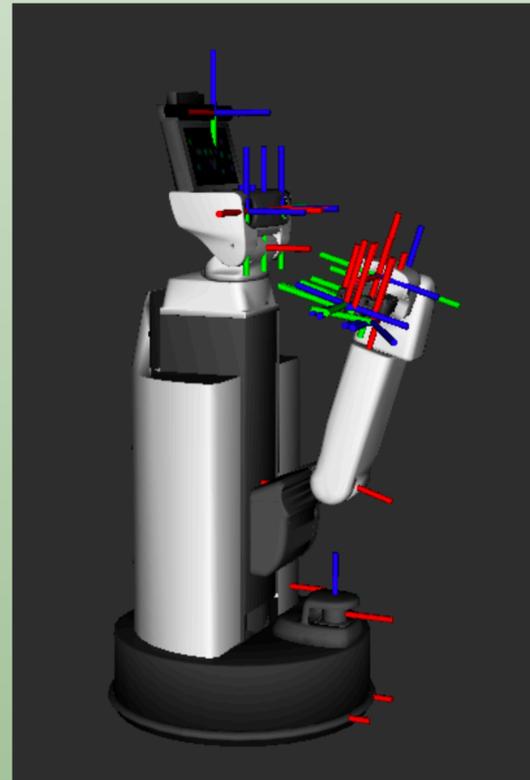
```
~$roslaunch tf tf_monitor
```

Get information about the transform between two frames

```
~$roslaunch tf tf_echo source_frame target_frame
```

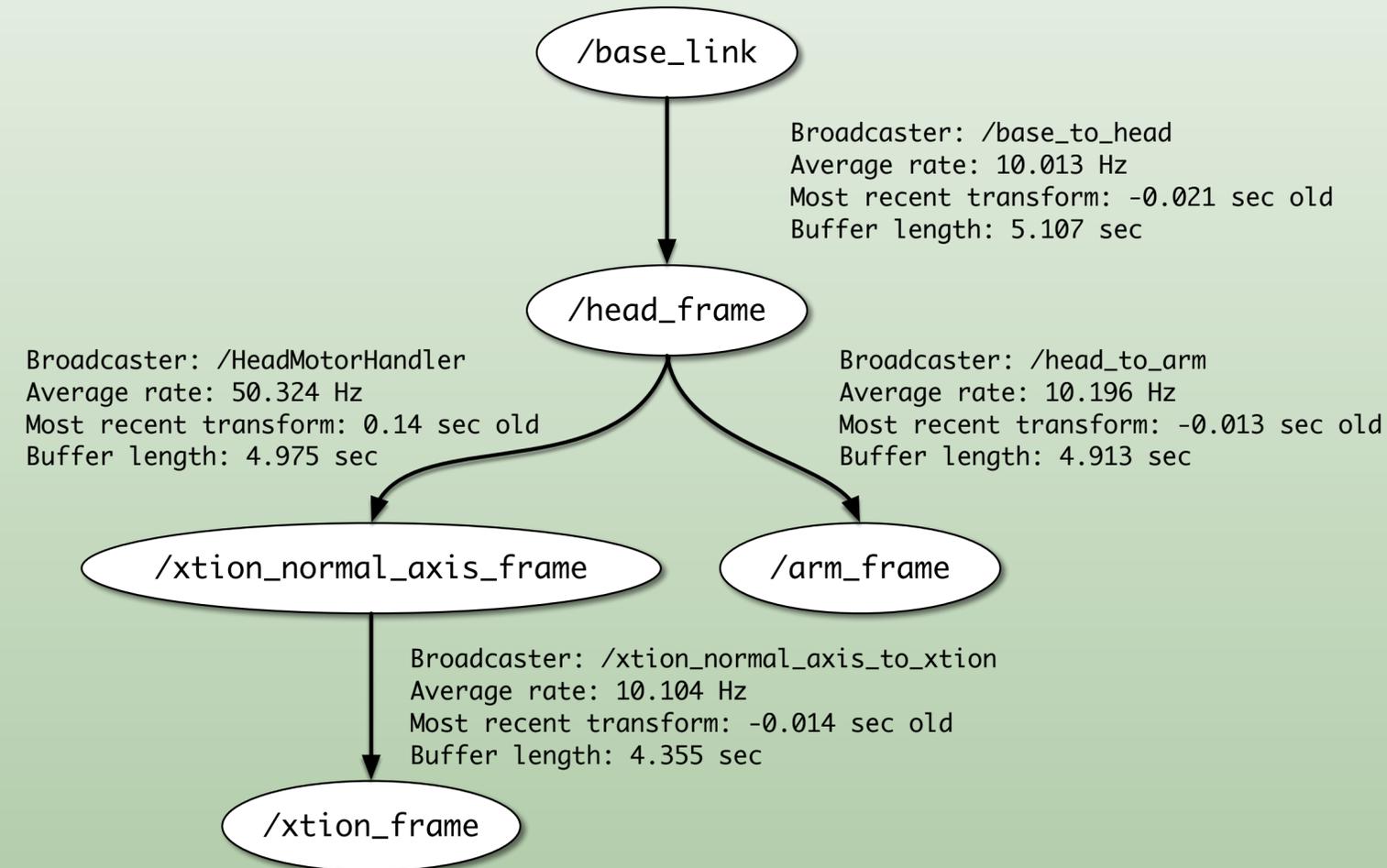
RViz

3D visualization of the transforms



View frames

Visual graph of the transform tree

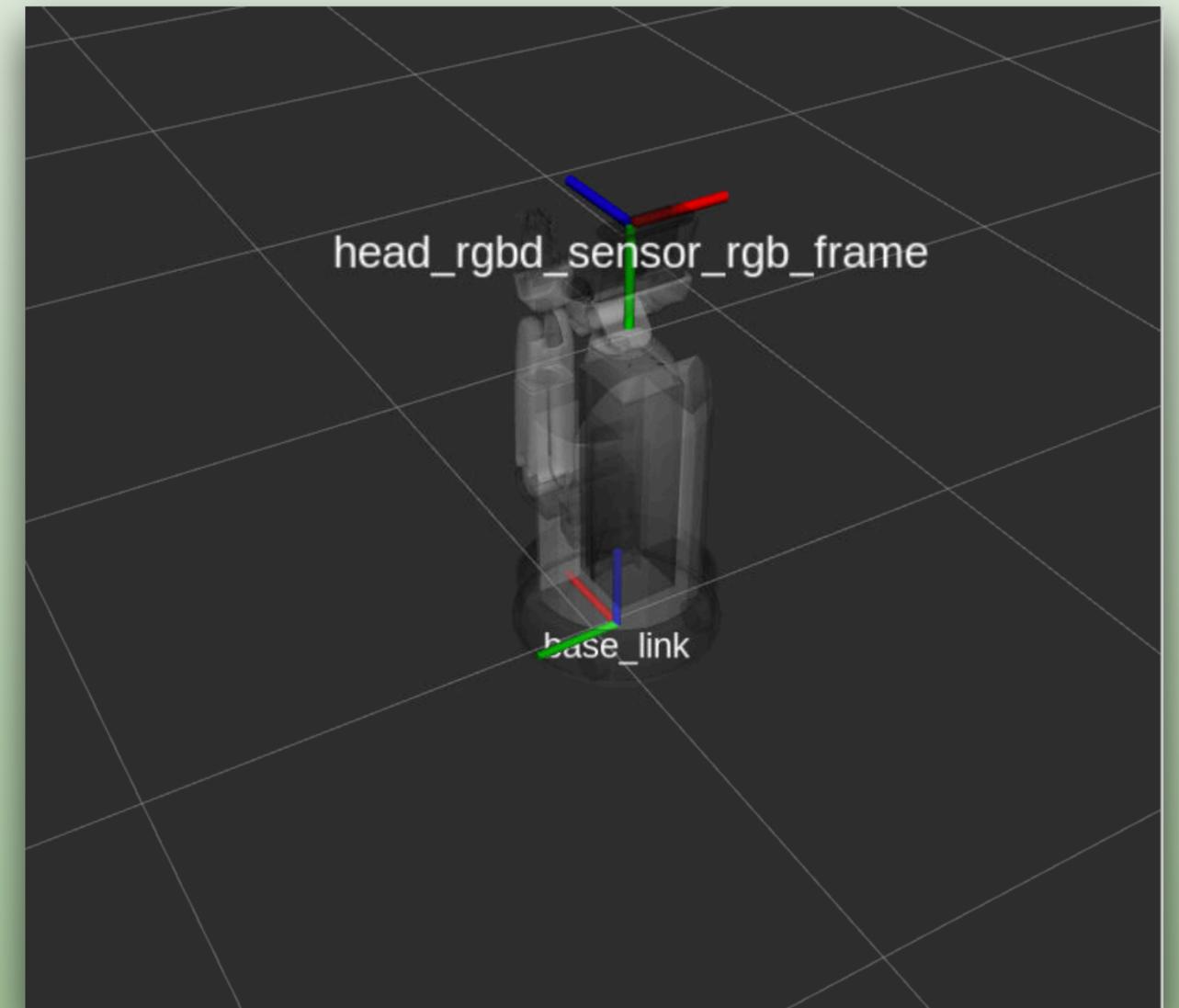


Terminal

Example

```
~$roslaunch tf_echo base_link head_rgbd_sensor_rgb_frame
```

```
At time 489.133
- Translation: [-0.060, 0.022, 0.967]
- Rotation: in Quaternion [0.500, -0.500, 0.500, -0.500]
            in RPY (radian) [-1.571, -0.000, -1.571]
            in RPY (degree) [-90.000, -0.000, -90.000]
At time 490.133
- Translation: [-0.060, 0.022, 0.967]
- Rotation: in Quaternion [0.500, -0.500, 0.500, -0.500]
            in RPY (radian) [-1.571, -0.000, -1.571]
            in RPY (degree) [-90.000, -0.000, -90.000]
At time 491.133
- Translation: [-0.060, 0.022, 0.967]
- Rotation: in Quaternion [0.500, -0.500, 0.500, -0.500]
            in RPY (radian) [-1.571, -0.000, -1.571]
            in RPY (degree) [-90.000, -0.000, -90.000]
At time 492.133
- Translation: [-0.060, 0.022, 0.967]
- Rotation: in Quaternion [0.500, -0.500, 0.500, -0.500]
            in RPY (radian) [-1.571, -0.000, -1.571]
            in RPY (degree) [-90.000, -0.000, -90.000]
```



TF TRANSFORMATION SYSTEM RVIZ PLUGIN

The screenshot displays the RVIZ (Robot Visualization) interface. The top toolbar includes icons for Interact, Move Camera, Select, Focus Camera, Measure, 2D Pose Estimate, 2D Nav Goal, and Publish Point. The main window is divided into a configuration panel on the left and a 3D visualization area on the right.

Configuration Panel (Left):

- Displays:**
 - Global Options
 - Fixed Frame: map
 - Background Color: 48; 48; 48
 - Frame Rate: 30
 - Default Light:
 - Global Status: Ok
 - Fixed Frame: OK
 - Grid:
 - Map:
 - Map:
 - RobotModel
 - Status: Ok
 - Visual Enabled:
 - Collision Enabled:
 - Update Interval: 0
 - Alpha: 1
 - Robot Description: robot_description
 - TF Prefix:
 - Links
 - Link Tree Style: Links in Alphanumeric Order
 - Expand Link Details:
 - All Links Enabled:
 - arm_flex_link
 - Alpha: 1
 - Show Trail:
 - Show Axes:
 - Position: 0.14091; 0.077817; 0.38997
 - Orientation: 5.7811e-05; 0.15136; -0.00...
 - arm_lift_link
 - Alpha: 1
 - Show Trail:
 - Show Axes:
 - Position: -0.0001456; -7.4902e-05; 0...
 - Orientation: 0; 0; -0.00038194; 1
 - arm_roll_link
 - Alpha: 1
 - Show Trail:
 - Show Axes:
 - Position: 0.24892; 0.077735; 0.71767
 - Orientation: -0.10693; 0.10712; -0.6988...
 - base_b_bumper_link
 - Alpha: 1
 - Show Trail:
 - Show Axes:
 - Position: -0.00014461; 0.0012251; 0
 - Orientation: 0; 0; 1; 0.00038194
 - base_f_bumper_link
 - Alpha: 1
 - Show Trail:
 - Show Axes:
 - Position: -0.00014461; 0.0012251; 0
 - Orientation: 0; 0; 1; 0.00038194

3D Visualization Area (Right):

Shows a 3D model of a robot with a white body and a black base. The robot's right arm is extended, and its joints are highlighted with red, green, and blue coordinate axes. The robot is positioned on a dark circular base.

Time Panel (Bottom):

ROS Time: 280.27 ROS Elapsed: 56.06 Wall Time: 1599487395.75 Wall Elapsed: 234.66 Experimental

Reset Left-Click: Rotate. Middle-Click: Move X/Y. Right-Click/Mouse Wheel: Zoom. Shift: More options. 9 fps

TF TRANSFORMATION SYSTEM - TRANSFORM LISTENER C++ API

```
#include <ros/ros.h>
#include <tf2_ros/transform_listener.h>
#include <geometry_msgs/TransformStamped.h>

int main(int argc, char** argv)
{
    ros::init(argc, argv, "tf2_listener");
    ros::NodeHandle nodeHandle;

    tf2_ros::Buffer tfBuffer;
    tf2_ros::TransformListener tfListener(tfBuffer);

    ros::Rate rate(10.0);
    while (nodeHandle.ok())
    {
        geometry_msgs::TransformStamped transformStamped;
        try
        {
            transformStamped = ("base", "odom", ros::Time(0));
        }
        catch (tf2::TransformException &exception)
        {
            ROS_WARN("%s", exception.what());
            os::Duration(1.0).sleep();
            continue;
        }
        rate.sleep();
    }
    return 0;
}
```

- ▶ Create a TF listener to fill up a buffer

```
tf2_ros::Buffer tfBuffer;
tf2_ros::TransformListener tfListener(tfBuffer);
```

- ▶ Beware of scope!
- ▶ Lookup transformations use this:

```
geometry_msgs::TransformStamped transformStamped =
    tfBuffer.lookupTransform(target_frame_id,
                             source_frame_id, time);
```

- ▶ For time: use **ros::Time(0)** to get latest available transform

Details at:

[http://wiki.ros.org/tf2/Tutorials/Writing%20a%20tf2%20listener%20\(C++\)](http://wiki.ros.org/tf2/Tutorials/Writing%20a%20tf2%20listener%20(C++))

TF TRANSFORMATION SYSTEM - TRANSFORM LISTENER PYTHON API

```
#!/usr/bin/env python

import rospy
import tf
from geometry_msgs.msg import TransformStamped

def get_transform():
    try:
        # Lookup the transformation from base_link to head_rgb_sensor_rgb_frame
        (trans, rot) = listener.lookupTransform('base_link', 'head_rgb_sensor_rgb_frame', rospy.Time(0))
        rospy.loginfo("Translation: %s" % str(trans))
        rospy.loginfo("Rotation: %s" % str(rot))
    except (tf.LookupException, tf.ConnectivityException, tf.ExtrapolationException):
        rospy.logwarn("Could not get transform from base_link to head_rgb_sensor_rgb_frame")

# Initialize the ROS node
rospy.init_node('tf_listener')

# Create a tf listener
listener = tf.TransformListener()

# Set up the timer to call 'get_transform' every second
rospy.Timer(rospy.Duration(1), lambda event: get_transform())

# Keeps the node running until it is manually shut down
rospy.spin()
```

- ▶ Create a TF listener to fill up a buffer

```
listener = tf.TransformListener()
```

- ▶ Lookup transformations, `rospy.Time(0)` for latest frame

```
(trans, rot) = listener.lookupTransform('base_link', 'head_rgb_sensor_rgb_frame', rospy.Time(0))
```

Details at:

<http://wiki.ros.org/tf2/Tutorials/Writing%20a%20tf2%20listener%20%28Python%29>

RQT USER INTERFACE

- ▶ User interface based on Qt
- ▶ Custom interfaces possible
- ▶ Use existing plugins
- ▶ Create your own plugins

Run RQT

```
~$roslaunch rqt_gui rqt_gui
```

Alternative

```
~$rqt
```

The screenshot displays the RQT (Robot Query Tool) interface. The main window is titled "Joint trajectory controller" and shows the following configuration:

- controller manager ns: `/hsrb/controller_manager`
- controller: `arm_trajectory_controller`
- A green power button is visible below the controller selection.
- Under the "joints" section, there are five sliders with numerical values:
 - arm_flex_joint: -0.71
 - arm_lift_joint: 0.00
 - arm_roll_joint: 0.34
 - wrist_flex_joint: 0.00
 - wrist_roll_joint: 0.00
- Under the "speed scaling" section, there is a slider set to 50%.

On the right side, there are three panels for monitoring:

- "Error Device" and "Message" panel (empty).
- "Warned Device" and "Message" panel (empty).
- "All devices" and "Message" panel (empty).

At the bottom right, there is a "Pause" button and a message indicator: "`<- old to messages receive new ->`".

In the background, a 3D visualization of a robot arm is visible on a grid floor.

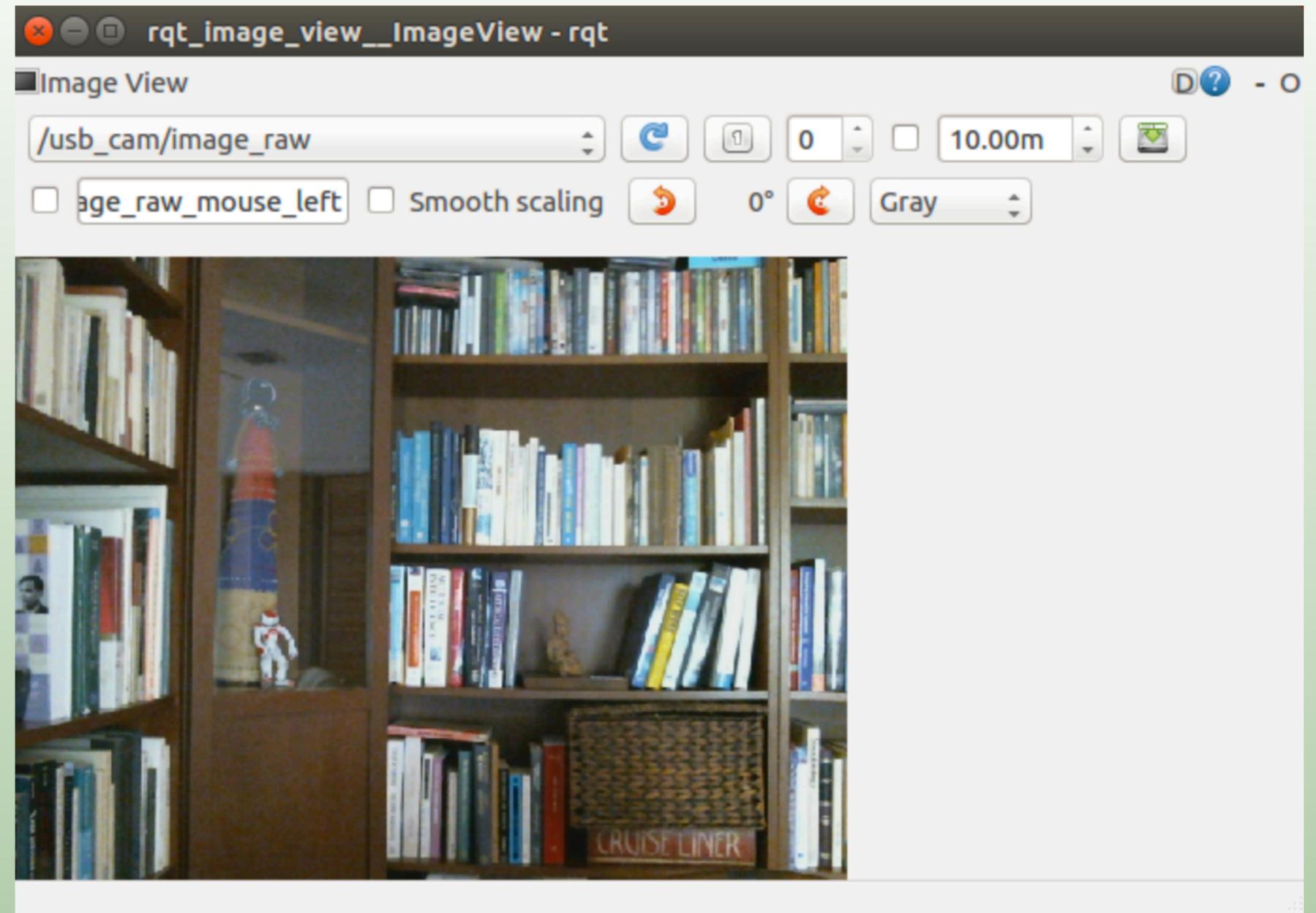
Details at: <http://wiki.ros.org/rqt/Plugins>

RQT USER INTERFACE IMAGE VIEW

► Visualizing images

Run `rqt_image_view`

```
~$roslaunch rqt_image_view rqt_image_view
```

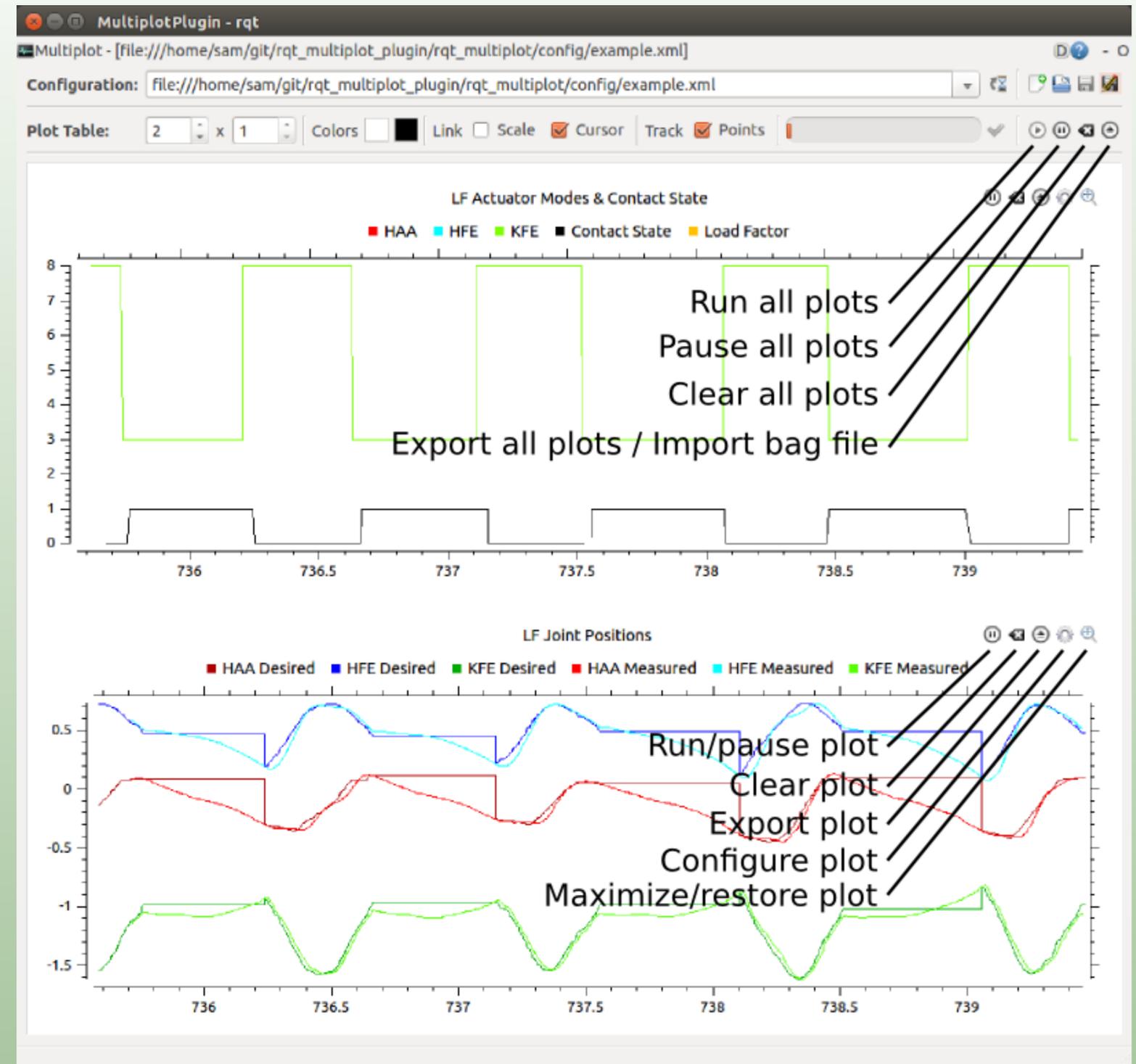


RQT USER INTERFACE RQT_MULTIPLOT

- ▶ Visualizing numeric values in 2D plots

Run rqt_multiplot

```
~$rosrun rqt_multiplot rqt_multiplot
```



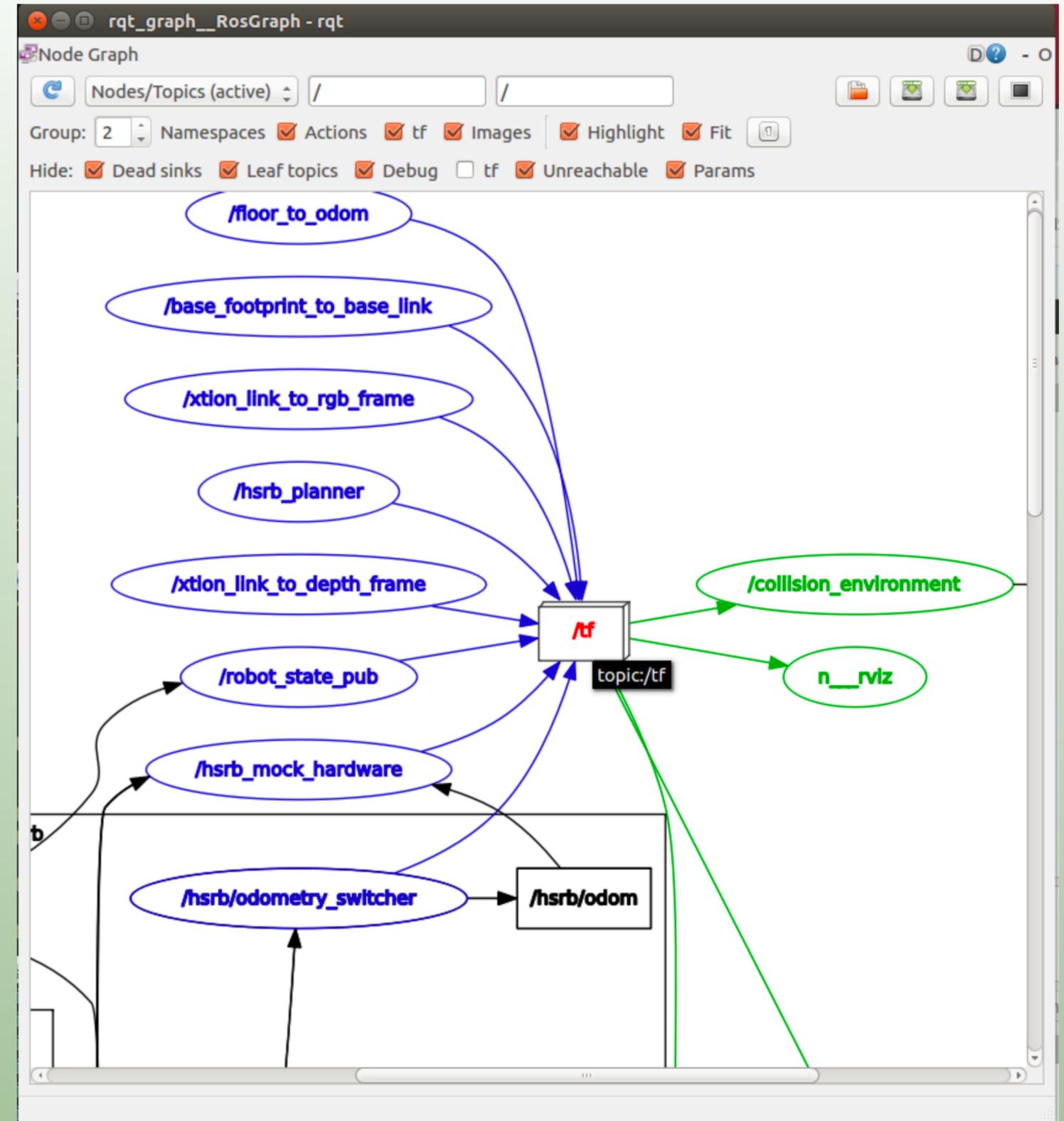
Details at: http://wiki.ros.org/rqt_multiplot

RQT USER INTERFACE RQT_GRAPH

- ▶ Visualizing the ROS computation graph

Run rqt_graph

```
~$rosrun rqt_graph rqt_graph
```

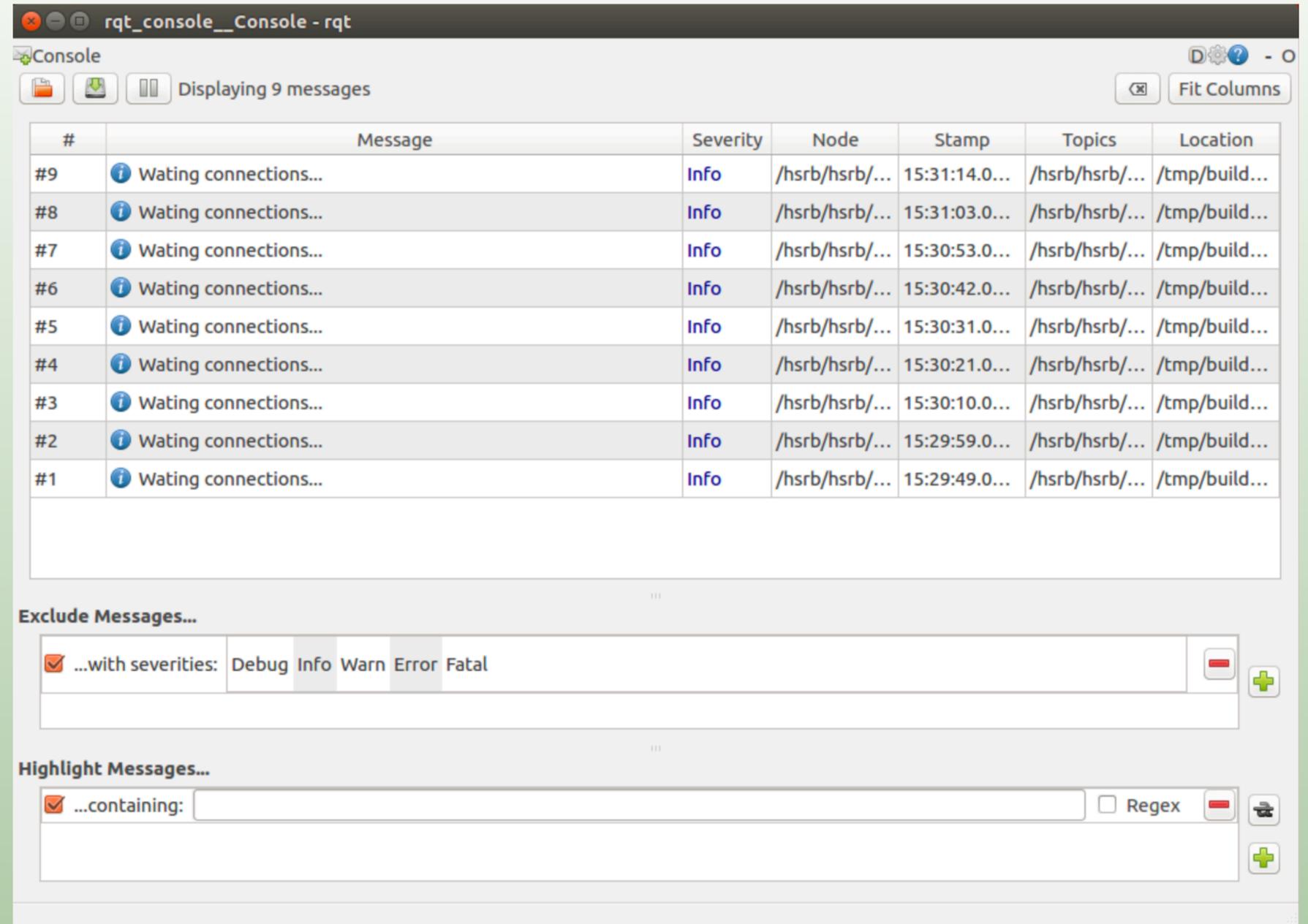


Details at: http://wiki.ros.org/rqt_graph

▶ Displaying and filtering ROS messages

Run `rqt_console`

```
~$rosrun rqt_console rqt_console
```



The screenshot shows the RQT Console window with the following data:

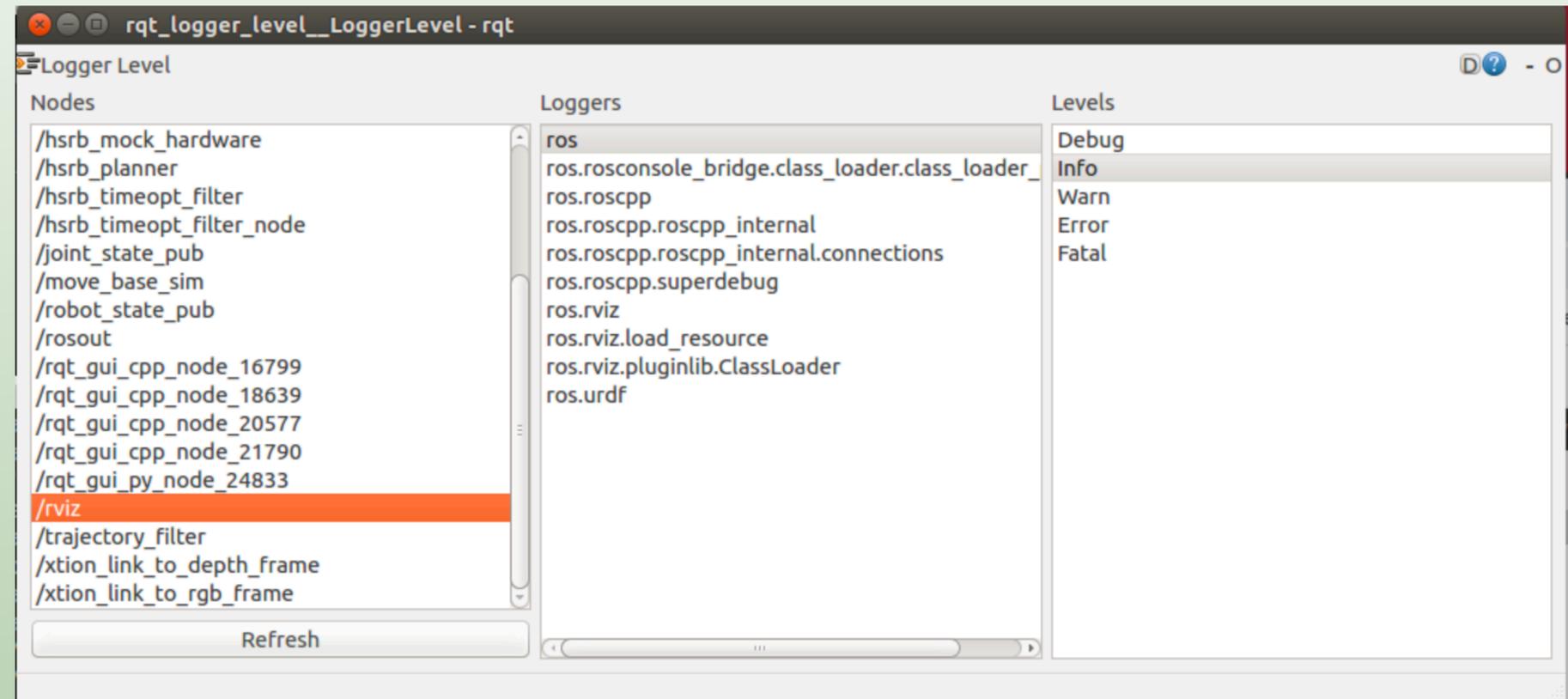
#	Message	Severity	Node	Stamp	Topics	Location
#9	Wating connections...	Info	/hsrb/hsrb/...	15:31:14.0...	/hsrb/hsrb/...	/tmp/build...
#8	Wating connections...	Info	/hsrb/hsrb/...	15:31:03.0...	/hsrb/hsrb/...	/tmp/build...
#7	Wating connections...	Info	/hsrb/hsrb/...	15:30:53.0...	/hsrb/hsrb/...	/tmp/build...
#6	Wating connections...	Info	/hsrb/hsrb/...	15:30:42.0...	/hsrb/hsrb/...	/tmp/build...
#5	Wating connections...	Info	/hsrb/hsrb/...	15:30:31.0...	/hsrb/hsrb/...	/tmp/build...
#4	Wating connections...	Info	/hsrb/hsrb/...	15:30:21.0...	/hsrb/hsrb/...	/tmp/build...
#3	Wating connections...	Info	/hsrb/hsrb/...	15:30:10.0...	/hsrb/hsrb/...	/tmp/build...
#2	Wating connections...	Info	/hsrb/hsrb/...	15:29:59.0...	/hsrb/hsrb/...	/tmp/build...
#1	Wating connections...	Info	/hsrb/hsrb/...	15:29:49.0...	/hsrb/hsrb/...	/tmp/build...

Below the table, the 'Exclude Messages...' section has a checked checkbox for '...with severities:' and radio buttons for 'Debug', 'Info', 'Warn', 'Error', and 'Fatal'. The 'Highlight Messages...' section has a checked checkbox for '...containing:', a text input field, a 'Regex' checkbox, and a printer icon.

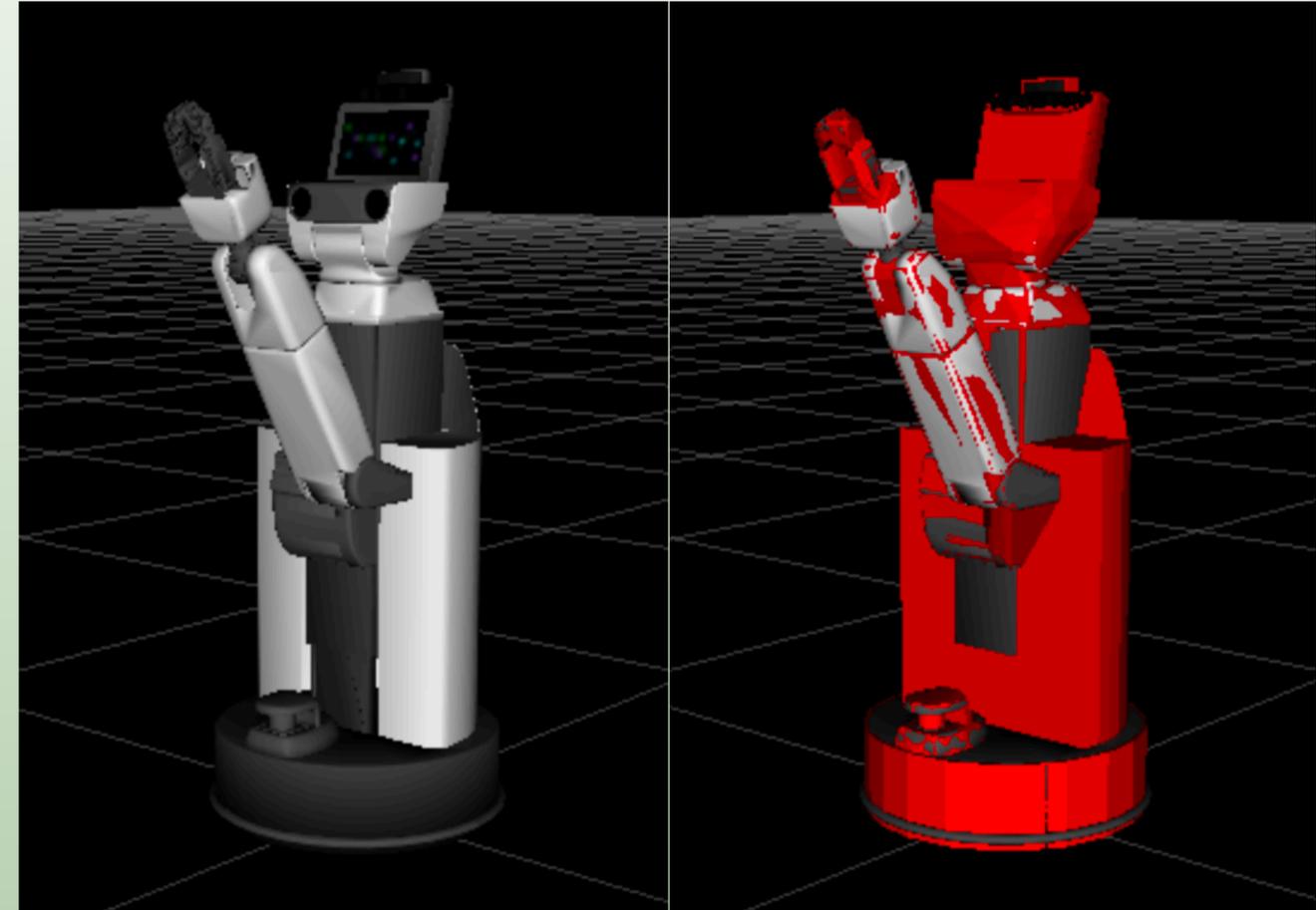
- ▶ Configuring the logger level of ROS nodes

Run `rqt_logger_level`

```
~$roslaunch rqt_logger_level rqt_logger_level
```

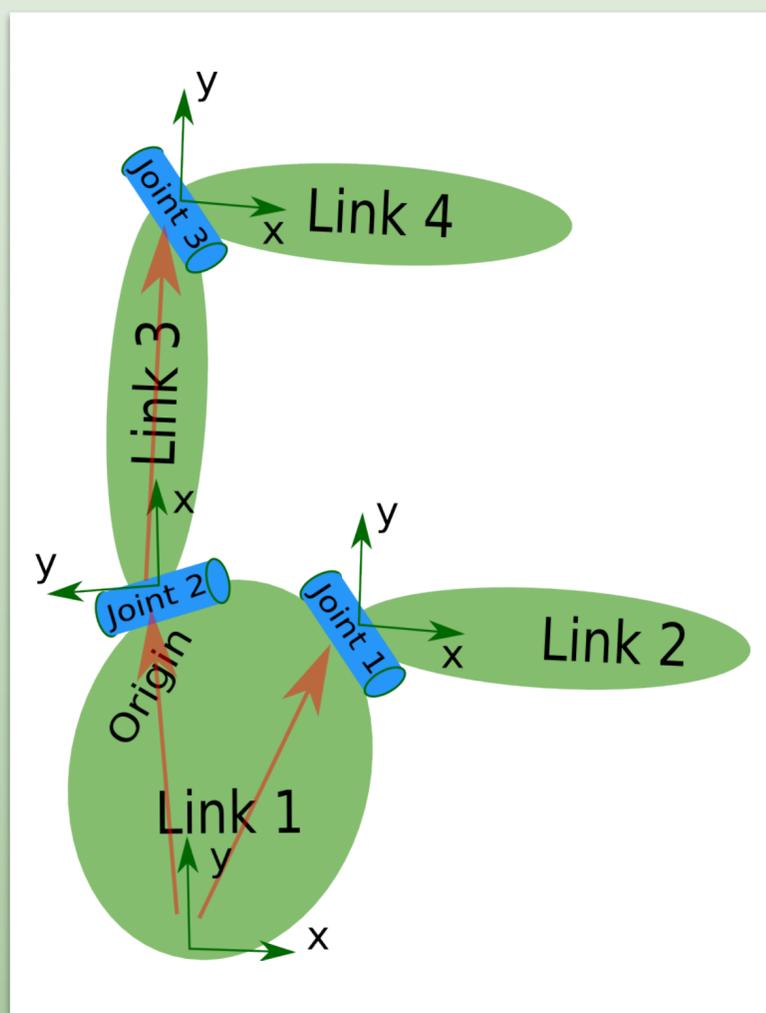


- ▶ Unified Robot Description Format (URDF)
- ▶ Defines robot model in XML format
 - ▶ Kinematic description
 - ▶ Dynamic description
 - ▶ Visual representation (left figure of HSR)
 - ▶ Collision model (right figure of HSR)
- ▶ URDF generation can be scripted using XACRO



ROBOT MODELS - URDF

- ▶ Description consists of a set of **link** elements and a set of **joint** elements
- ▶ Joints connect the link elements together



```
<robot name="hsr">
  <link> ... </link>
  <link> ... </link>
  <link> ... </link>

  <joint> .... </joint>
  <joint> .... </joint>
  <joint> .... </joint>
</robot>
```

```
<link name="my_link">
  <inertial>
    <origin xyz="0 0 0.5" rpy="0 0 0"/>
    <mass value="1"/>
    <inertia ixx="100" ixy="0" ixz="0" iyy="100"
      iyz="0" izz="100" />
  </inertial>

  <visual>
    <origin xyz="0 0 0" rpy="0 0 0" />
    <geometry>
      <box size="1 1 1" />
    </geometry>
    <material name="Cyan">
      <color rgba="0 1.0 1.0 1.0"/>
    </material>
  </visual>

  <collision>
    <origin xyz="0 0 0" rpy="0 0 0"/>
    <geometry>
      <cylinder radius="1" length="0.5"/>
    </geometry>
  </collision>
</link>
```

Details at:
<http://wiki.ros.org/urdf/XML/model>

ROBOT MODELS - USAGE IN ROS

- ▶ The robot description (URDF) is stored on the parameter server (typically) under /**robot_description**
- ▶ You can visualize the robot model in RViz with the RobotModel plugin

husky.urdf.xacro

```
<robot name="husky" xmlns:xacro="http://ros.org/wiki/xacro">
  <xacro:arg name="laser_enabled" default="false" />
  <xacro:arg name="laser_xyz" default="$(optenv HUSKY_LMS1XX_XYZ 0.2206 0.0 0.00635)" />
  <xacro:arg name="laser_rpy" default="$(optenv HUSKY_LMS1XX_RPY 0.0 0.0 0.0)" />

  <xacro:arg name="kinect_enabled" default="false" />
  <xacro:arg name="kinect_xyz" default="$(optenv HUSKY_KINECT_XYZ 0 0 0)" />
  <xacro:arg name="kinect_rpy" default="$(optenv HUSKY_KINECT_RPY 0 0.18 3.14)" />

  <xacro:arg name="realsense_enabled" default="false" />
  <xacro:arg name="realsense_xyz" default="$(optenv HUSKY_REALSENSE_XYZ 0 0 0)" />
  <xacro:arg name="realsense_rpy" default="$(optenv HUSKY_REALSENSE_RPY 0 0 0)" />
  <xacro:arg name="realsense_mount" default="$(optenv HUSKY_REALSENSE_MOUNT_FRAME sensor_arch_mount_link)" />

  <xacro:property name="husky_front_bumper_extend" value="$(optenv HUSKY_FRONT BUMPER_EXTEND 0)" />
  <xacro:property name="husky_rear_bumper_extend" value="$(optenv HUSKY_REAR BUMPER_EXTEND 0)" />

  <xacro:arg name="robot_namespace" default="/" />
  <xacro:arg name="urdf_extras" default="empty.urdf" />
</robot>
```

spawn_husky.launch

```
<launch>
  <arg name="multimaster" default="false" />
  <arg name="robot_namespace" default="/" />

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

  <arg name="laser_enabled" default="$(optenv HUSKY_LMS1XX_ENABLED false)" />
  <arg name="kinect_enabled" default="$(optenv HUSKY_KINECT_ENABLED false)" />
  <arg name="realsense_enabled" default="$(optenv HUSKY_REALSENSE_ENABLED false)" />
  <arg name="urdf_extras" default="$(optenv HUSKY_URDF_EXTRAS)" />

  <group ns="$(arg robot_namespace)">
    <group if="$(arg multimaster)">
      <include file="$(find husky_description)/launch/description.launch" >
        <arg name="robot_namespace" value="$(arg robot_namespace)" />
        <arg name="laser_enabled" default="$(arg laser_enabled)" />
        <arg name="kinect_enabled" default="$(arg kinect_enabled)" />
        <arg name="realsense_enabled" default="$(arg realsense_enabled)" />
        <arg name="urdf_extras" default="$(arg urdf_extras)" />
      </include>

      <include file="$(find multimaster_launch)/launch/multimaster_gazebo_robot.launch">
        <arg name="gazebo_interface" value="$(find husky_control)/config/gazebo_interface.yaml" />
        <arg name="robot_namespace" value="$(arg robot_namespace)" />
      </include>

      <!-- For multimaster bringup, need to load the controller config -->
      <rosparam command="load" file="$(find husky_control)/config/control.yaml" />
    </group>
  </group>
</launch>
```

ROBOT MODELS - USAGE IN ROS

- ▶ The robot description (URDF) is stored on the parameter server (typically) under `/robot_description`
- ▶ You can visualize the robot model in RViz with the RobotModel plugin

hsrb4s.urdf.xacro

```
<robot name="hsrb"
  xmlns:controller="http://playerstage.sourceforge.net/gazebo/xmllschema/#controller"
  xmlns:interface="http://playerstage.sourceforge.net/gazebo/xmllschema/#interface"
  xmlns:sensor="http://playerstage.sourceforge.net/gazebo/xmllschema/#sensor"
  xmlns:xacro="http://ros.org/wiki/xacro">

  <!-- common xacro -->
  <xacro:include filename="$(find hsr4_description)/urdf/common.xacro" />
  <xacro:include filename="$(find hsr4_description)/urdf/materials.urdf.xacro" />

  <!-- links and joints -->
  <xacro:include filename="$(find hsr4_description)/urdf/base_v2/base.urdf.xacro" />
  <xacro:include filename="$(find hsr4_description)/urdf/torso_v0/torso.urdf.xacro" />
  <xacro:include filename="$(find hsr4_description)/urdf/head_v2/head.urdf.xacro" />
  <xacro:include filename="$(find hsr4_description)/urdf/arm_v0/arm.urdf.xacro" />
  <xacro:include filename="$(find hsr4_description)/urdf/wrist_v0/wrist.urdf.xacro" />
  <xacro:include filename="$(find hsr4_description)/urdf/hand_v0/hand.urdf.xacro" />

  <xacro:arg name="personal_name" default="" />
  <xacro:arg name="loopback_hardware" default="false" />

  <!-- constant -->
  <xacro:property name="personal_name" value="$(arg personal_name)" />
  <xacro:property name="robot_name" value="hsrb" />

  <!-- create robot -->
  <xacro:hsrb_base prefix="base" personal_name="${personal_name}" robot_namespace="${robot_name}" robot_name="${robot_name}" />

  <xacro:hsrb_torso prefix="torso" parent="base_link" mimic_joint="arm_lift_joint">
    <origin xyz="0.0 0.0 0.752" rpy="0.0 0.0 0.0" />
  </xacro:hsrb_torso>

  <xacro:hsrb_head prefix="head" parent="torso_lift_link">
    <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
  </xacro:hsrb_head>

  <xacro:hsrb_arm prefix="arm" parent="base_link">
    <origin xyz="0.0 0.0 0.340" rpy="0.0 0.0 0.0" />
  </xacro:hsrb_arm>

  <xacro:hsrb_wrist prefix="wrist" parent="arm_roll_link" robot_namespace="${robot_name}">
    <origin xyz="0.0 0.0 0.0" rpy="0.0 0.0 0.0" />
  </xacro:hsrb_wrist>

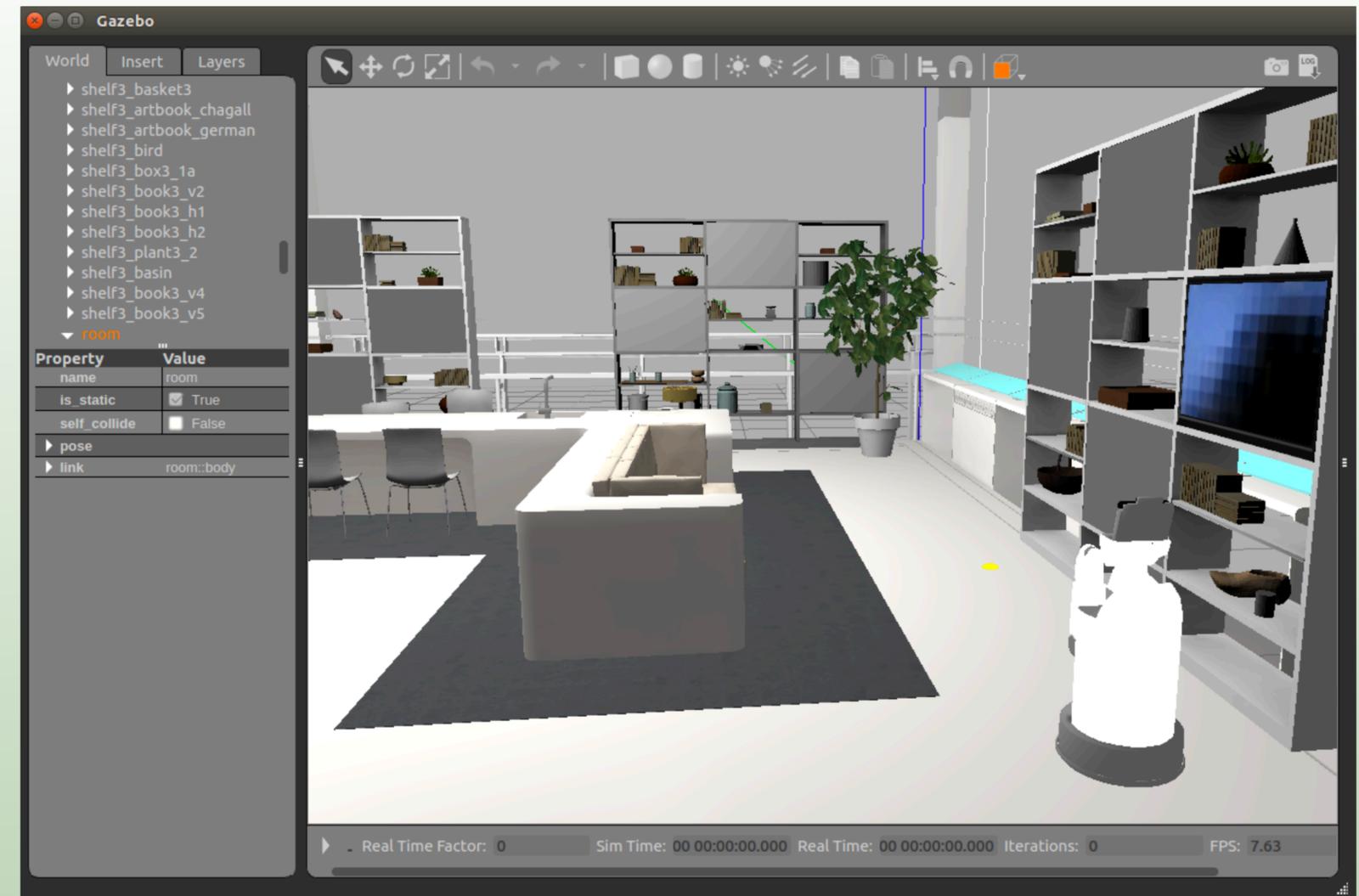
  <xacro:hsrb_hand prefix="hand" parent="wrist_roll_link">
    <origin xyz="0.012 0.0 0.1405" rpy="0.0 0.0 ${pi}" />
  </xacro:hsrb_hand>
</robot>
```

hsrb4s.urdf

```
<robot name="hsrb" xmlns:controller="http://playerstage.sourceforge.net/gazebo/xmllschema/#controller" xmlns:interface="http://playerstage.sourceforge.net/gazebo/xmllschema/#interface" xmlns:sensor="http://playerstage.sourceforge.net/gazebo/xmllschema/#sensor" xmlns:xacro="http://ros.org/wiki/xacro">
  <material name="body_main">
    <color rgba="1.0 0.0 0.0 1.0" />
  </material>
  <material name="body_sub">
    <color rgba="0.33 0.33 0.33 1.0" />
  </material>
  <material name="wheel">
    <color rgba="0.2 0.2 0.2 1.0" />
  </material>
  <material name="black">
    <color rgba="0.1 0.1 0.1 1.0" />
  </material>
  <material name="tablet">
    <color rgba="0.1 0.1 0.2 1.0" />
  </material>
  <!--
  ASUS Xtion PRO LIVE
  Specs from:
  http://www.asus.com/Multimedia/Xtion_PRO_LIVE/#specifications
  -->
  <link name="base_footprint">
  </link>
  <joint name="base_footprint_joint" type="fixed">
    <parent link="base_footprint" />
    <child link="base_link" />
  </joint>
  <link name="base_link">
    <inertial>
      <mass value="50.00" />
      <!-- use dummy weight to stabilize gazebo move base (real value: 11.017971)-->
      <origin xyz="-0.025978 -0.005498 0.17633" />
      <inertia ixx="0.3742" ixy="0.000434172172" ixz="0.03088" iyy="0.3436" iyz="0.01775" izz="0.1509" />
    </inertial>
    <visual>
      <geometry>
        <mesh filename="package://hsrb_meshes/meshes/base_v2/base_light.dae" />
      </geometry>
    </visual>
    <visual>
      <geometry>
        <mesh filename="package://hsrb_meshes/meshes/base_v2/body_light.dae" />
      </geometry>
    </visual>
  </link>
</robot>
```

UNIVERSAL SCENE DESCRIPTION (USD)

- ▶ For large-scale, physically accurate digital twins
- ▶ OpenUSD is foundational to NVIDIA Omniverse
- ▶ Alliance for OpenUSD (AOUSD)—including Pixar, Adobe, Apple, and Autodesk—to evolve OpenUSD as it becomes one of the building blocks in the era of AI and industrial digitalization
 - ▶ Environments (incl. gravity, lights etc)
 - ▶ Objects (both static and dynamic)
 - ▶ Sensors
 - ▶ Robots
- ▶ USD is standard for Isaac Sim

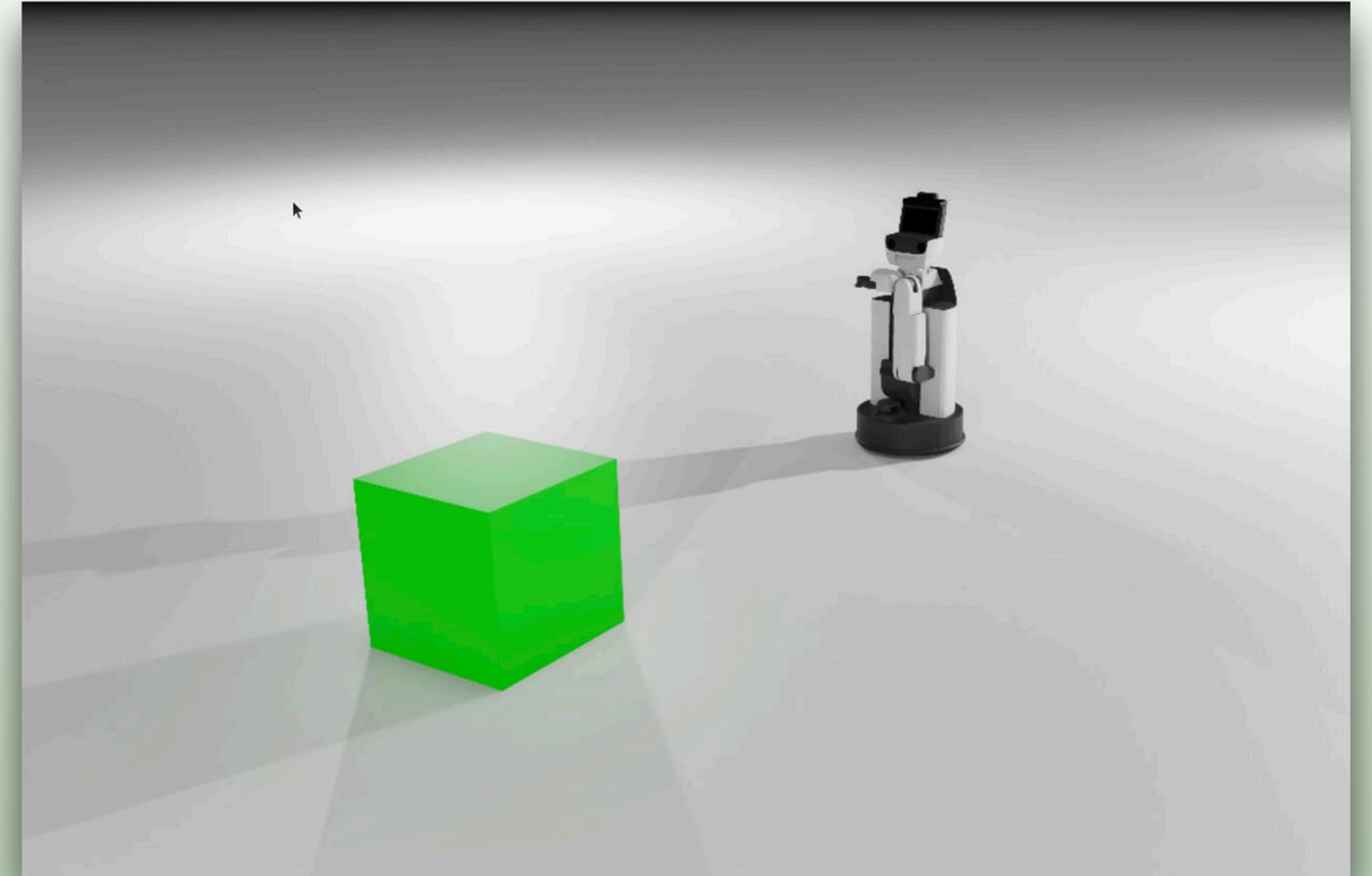


```
<?xml version="1.0" ?>
<sdf version="1.4">
  <model name="shelf3_book3_v4">
    <link name="body">
      <pose>0 0 0 0 0 0</pose>
      <inertial>
        <pose> 0.001798 0.085315 0.005927 0 0 0</pose>
        <mass>0.5</mass>
        <inertia>
          <ixx>0.01</ixx>
          <ixy>0</ixy>
          <ixz>0</ixz>
          <iyy>0.01</iyy>
          <iyz>0</iyz>
          <izz>0.01</izz>
        </inertia>
      </inertial>
      <collision name="collision">
        <geometry>
```

Details at:
<http://sdformat.org/>

UNIVERSAL SCENE DESCRIPTION (USD)

- ▶ Encoding scalable, hierarchically organized, static and time-sampled data
- ▶ Rich, extensible language
 - ▶ Environments (incl. gravity, lights etc)
 - ▶ Objects (both static and dynamic)
 - ▶ Sensors
 - ▶ Robots
- ▶ USD is standard for Isaac Sim
- ▶ Converters to USD available



```
from pxr import Usd, UsdGeom

stage = Usd.Stage.CreateNew('/path/to/HelloWorld.usda')
xformPrim = UsdGeom.Xform.Define(stage, '/hello')
spherePrim = UsdGeom.Sphere.Define(stage, '/hello/world')
# generic_spherePrim = stage.DefinePrim('/hello/world_generic', 'Sphere')
stage.GetRootLayer().Save()
```

```
#usda 1.0
def Xform "hello"
{
  def Sphere "world"
  {
  }
}
```

```
#usda 1.0
(
  defaultPrim = "bookshelf"
  upAxis = "Y"
)

def Xform "bookshelf"
{
  token ui:displayGroup = "Material Graphs"
  token ui:displayName = "bookshelf"
  int ui:order = 1024
  float3 xformOp:rotateXYZ = (0, -0, 0)
  float3 xformOp:scale = (1, 1, 1)
  double3 xformOp:translate = (0, 0, 0)
  uniform token[] xformOpOrder = ["xformOp:translate", "xformOp:rotateXYZ", "xformOp:scale"]

  def Xform "bookshelf" (
    apiSchemas = ["SemanticsAPI:Semantics_rm0w", "SemanticsAPI:Semantics_zY6Z",
      "SemanticsAPI:Semantics_Ql3t", "SemanticsAPI:QWQQ", "SemanticsAPI:QWQL",
      "SemanticsAPI:QWQC", "PhysicsRigidBodyAPI", "PhysicsMassAPI"]
```

Details at:

https://docs.omniverse.nvidia.com/isaacsim/latest/open_usd.html

FURTHER REFERENCES

- ▶ ROS Wiki

- ▶ <http://wiki.ros.org/>

- ▶ Installation

- ▶ <http://wiki.ros.org/ROS/Installation>

- ▶ Tutorials

- ▶ <http://wiki.ros.org/ROS/Tutorials>

- ▶ Packages

- ▶ <https://www.ros.org/browse/list.php>

- ▶ ROS Cheat Sheet

- ▶ <https://www.clearpathrobotics.com/ros-robot-operating-system-cheat-sheet/>

- ▶ https://kapeli.com/cheat_sheets/ROS.docset/Contents/Resources/Documents/index

- ▶ ROS Best Practices

- ▶ https://github.com/leggedrobotics/ros_best_practices/wiki

- ▶ ROS Package Templates

- ▶ https://github.com/leggedrobotics/ros_best_practices/tree/master/ros_package_template

Material is based on ROS Wiki and ETH Zürich ROS Introduction (<https://rsl.ethz.ch/>)