Institute for Cognitive Systems
Technische Universität München
Prof. Gordon Cheng

Multi-sensory Based Robot Dynamic Manipulation
ROS Tutorial 4: Interfacing Matlab with ROS

Homework

Exercise 1: ROS-Matlab Interface

Consider the RPPR robot model of Fig.1, used during the last tutorials. So far, we were able to visualize the D-H and the URDF frames of this robot on separated interfaces. In this tutorial, we will learn how to interface ROS with Matlab in order to display these frames within the same virtual environment.

$[mm]$
$L_1 = 300$
$L_2 = 340$
$L_3 = 583$
$L_4 = 80$
$L_5 = 80$
$L_6 = 115$
$L_7 = 82$
$L_8 = 108$
$L_9 = 123$
$L_{10} = 167$
$L_{11} = 28$
$L_{12} = 50$
$c = d = 320$
$\theta_1 = 45°$
$\theta_2 = 30°$

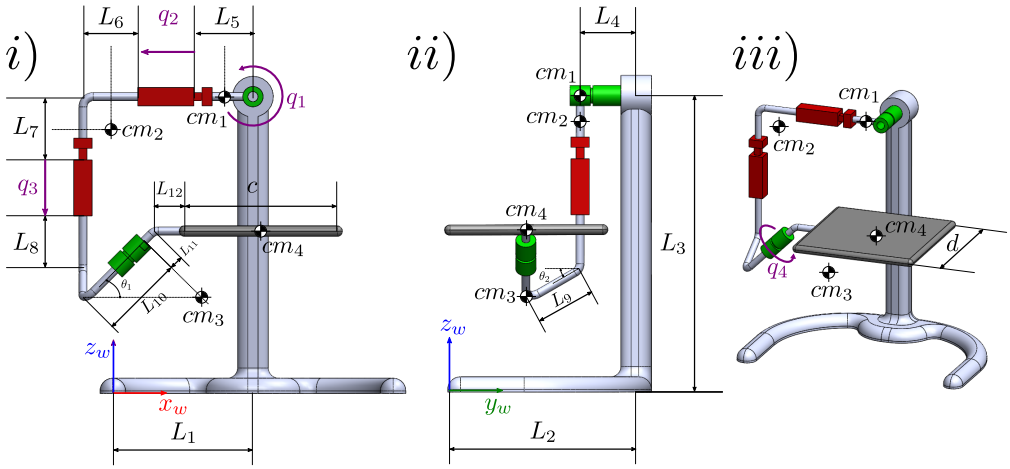

Figure 1: 4 DOF Rehabilitation Robot. $i)$ Front view, $ii)$ Right view, and $iii)$ Isometric view. The figure also shows the *center of mass* of each link cm.

The Matlab "*Robotics System Toolbox*" provides an interesting interface between Matlab-Simulink and ROS. This interface allows bidirectional communication between a Matlab simulation and a set of ROS nodes. A full documentation is provided on the mathworks website: `https://de.mathworks.com/help/robotics/examples/get-started-with-ros.html`.

1. Using the ROS-Matlab interface, publish a `sensor_msgs/JointState` topic containing the joint values to be applied to your robot model. You can check the definition of the `JointState` massage at the following address: `http://docs.ros.org/api/sensor_msgs/html/msg/JointState.html`.
2. Modify the launch file of the previous tutorial, in order to animate your robot model using the joint values published by Matlab, instead of the GUI. Visualize the resulting motions on rviz.

3. Using the D-H transforms calculated in the second tutorial, compute the positions and orientations of each joint frame of your robot with respect to the world frame of the URDF. The result must be presented in the form of a 3-dimensional position vector, associated with an orientation quaternion[1]. You can use the built-in Matlab function "`dcm2quat()`" `https://de.mathworks.com/help/aerotbx/ug/dcm2quat.html`.
4. Now compute in the same way the positions and orientations of each center of mass of your robot with respect to the URDF world frame.
5. Using the ROS-Matlab interface, publish a `geometry_msgs/TransformStamped` topic containing your transforms. You can check the definition of the `TransformStamped` message at following address: `http://docs.ros.org/api/geometry_msgs/html/msg/TransformStamped.html`.
6. On rviz, visualize the D-H frames for each joint and each center of mass of the robot alongside with the URDF frames. Do not forget to save your rviz configuration file.
7. In the Linux terminal, use the "`rosrun tf tf_echo`" command in order to extract the static relative transform $\mathbf{H}_{i_{DH}}^{i_{URDF}}$ between each D-H and URDF joint frames. Describe your reasoning in the "`Readme.txt`" file and write the corresponding transforms into Matlab. **Hint:** you can type "`rosrun tf tf_echo -h`" in order to get access to the help for this command.
8. On Matlab, compute each URDF transform with respect the URDF world frame as a product of the form $\mathbf{H}_{W}^{i_{URDF}} = \mathbf{H}_{W}^{i_{DH}} \mathbf{H}_{i_{DH}}^{i_{URDF}}$. Then extract the corresponding positions and orientations as a set of 3-dimensional position vectors, and orientation quaternions.
9. Using the ROS-Matlab interface, publish a `geometry_msgs/TransformStamped` topic containing the corresponding transforms.
10. On rviz, visualize at the same time (see Fig. 2):
    - the D-H frames of each joint and each center of mass of your robot (published from Matlab.)
    - the URDF frames (generated by the `joint_state_publisher` and `robot_state_publisher` nodes)
    - the new URDF frames, computed from the D-H frames (published from Matlab.)

    Do not forget to save your rviz configuration file.
11. What do you observe ?

Delivery format and due date

You should deliver a compressed "`.zip`" file with your implementations using the instructions of the exercise sheet and of the given template. Include a "`Readme.txt`" file to indicate how to run your nodes and to answer the questions. Please, name the compressed file as follows: "`Name_lastName_MSRDM_ROStutorial4.zip`".

[1] Quaternion are a common alternative to Euler angles, allowing to represents rotations at a low computational cost. Mathematically speaking, a quaternion can as a four-dimensional complex number with three different "imaginary" parts: $\mathbf{q} = a\mathbf{1} + b\mathbf{i} + c\mathbf{j} + d\mathbf{k} = \cos(\frac{\theta}{2})\mathbf{1} + \sin(\frac{\theta}{2})(r_x\mathbf{i} + r_y\mathbf{j} + r_z\mathbf{k})$.

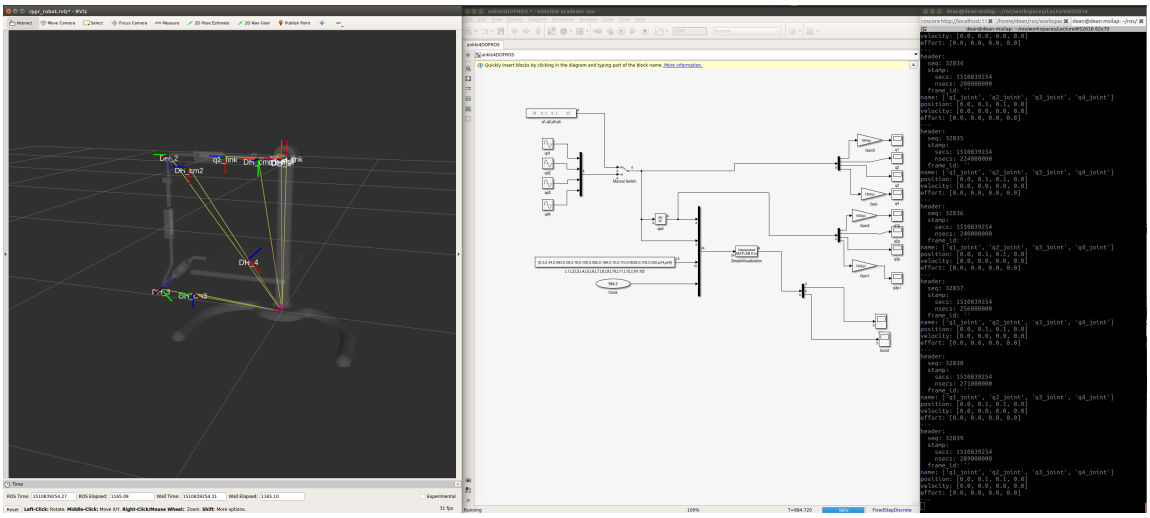

Figure 2: Expected result from this tutorial.