# Worksheet 4 – Forward and Inverse Kinematics

In this worksheet we will take a step further and, using the previously created robot description from Worksheet 3, compute the forward and inverse kinematics for the TR5 arm manipulator. The following ROS ecosystem will be used in this work:

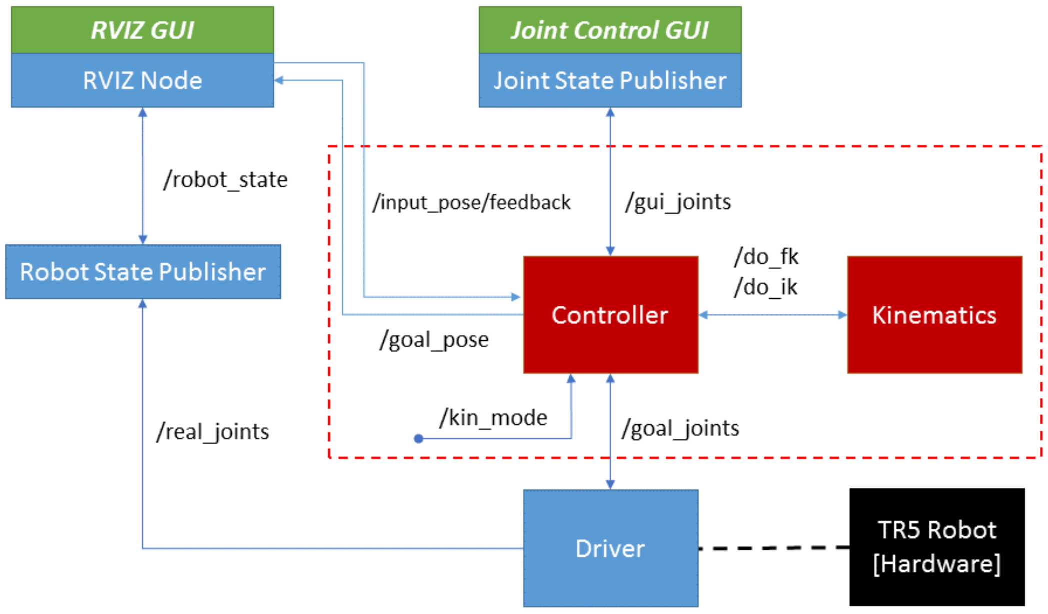

*Figure 1 – Proposed ROS ecosystem to control the TR5 arm manipulator.*

To finish this work only the nodes inside the red dashed rectangle, depicted in Figure 1, must be completed (the ones outside the rectangle will be provided). In addition, the **controller_node** is already partially implemented whereas the **kinematics_node** needs to be created from scratch.

## Workflow Explained

The debugging and interaction with TR5 robot is done with the help of the RVIZ node. In particular, both robot state and joint state publishers are loaded with the previously created **TR5 description** (xacro files). The RVIZ node subscribes to the robot state publisher in order to constantly render a visual representation of the TR5's current state. The joint state publisher launches its own graphical user interface (GUI) with a set of configurable input slides according to the joints defined in the **TR5 description** file (see Figure 2). The joint state publisher node is constantly broadcasting the desired joint angles in **"/gui_joints"** according to its sliders' settings.

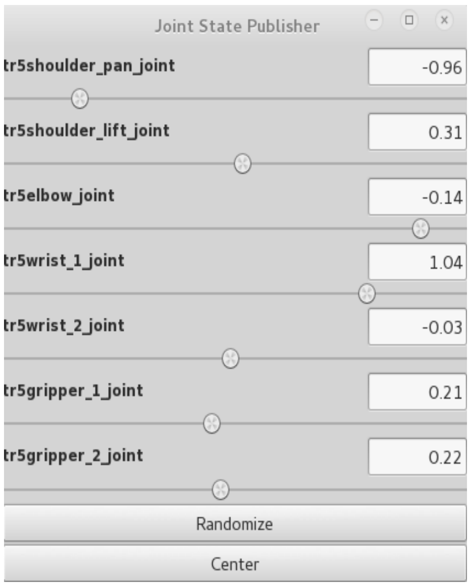

*Figure 2 – Graphical user interface launched by the joint state publisher.*

## *Forward Kinematics*

The user controls the TR5 robot by interacting with the joint state publisher's GUI. The **controller_node** receives the joints information (sensor_msgs::JointState) in the **"/gui_joints"** topic and retransmits that information into **"/goal_joints"** topic used by the TR5 **driver_node**. Moreover, the **controller_node** needs to publish a visualization marker in **"/goal_pose"** topic used by the RVIZ node. That marker will be displayed in RVIZ's GUI and represents the 3D coordinate of the TR5's end effector (see the red sphere depicted in Figure 3).

The marker's coordinates must be computed using the forward kinematics equations. In this sense, the **kinematics_node** node will handle the kinematics calculations and expose their results as two separate services: topic **"/do_fk"** for the forward kinematics and topic **"/do_ik"** for the inverse one. To change between kinematics modes (forward and inverse), the **controller_node** advertises a specific ROS service in **"/kin_mode"** topic. These ROS services are defined as follows:

| ***Service Type*** | ***Input*** | ***Output*** |
|---|---|---|
| DoForwardKinematics | sensor_msgs::JointState | geometry_msgs::Pose |
| DoInverseKinematics | geometry_msgs::Pose | sensor_msgs::JointState |
| KinematicMode | uInt8 | bool |

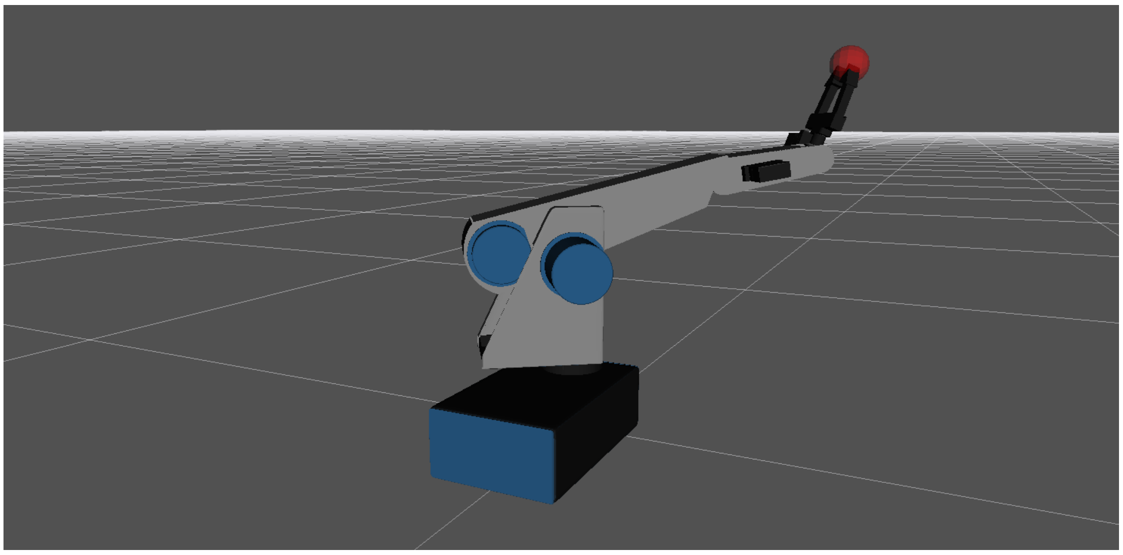

*Figure 3 – RVIZ's GUI rendering the TR5 arm manipulator. The red sphere represents the end effector's coordinates that results from the direct kinematics equations.*

## *Inverse Kinematics*

The user can control the position of the TR5's end effector by moving an interactive marker, a blue sphere model in the RVIZ GUI (see Figure 4). The coordinates of this marker are published by the RVIZ node in **"/input_pose/feedback"** topic and received by the **controller_node**. According to the desired end effector position, the latter node computes the inverse kinematics via **"/do_ik"** service and publishes the resulted joint values in the **"/goal_joints"** topic. During this kinematics mode, the information published by the joint state publisher's GUI must be ignored by the **controller_node**.

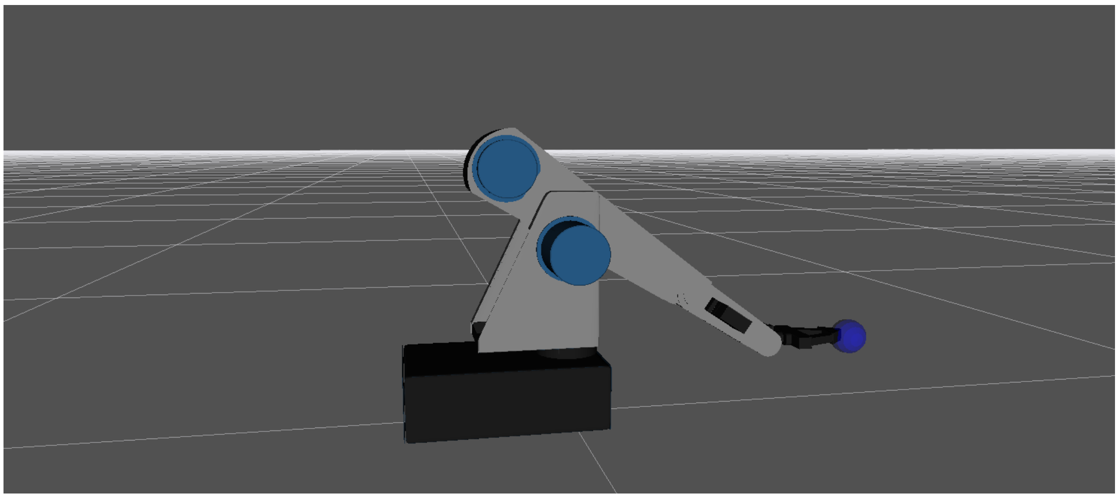

*Figure 4 - RVIZ's GUI rendering the TR5 arm manipulator. The blue sphere represents the end effector's coordinates that the user can displace in the virtual scenario. The TR5 3D model moves according to the joint angles that results from applying the inverse kinematics equations from the end effector's coordinates.*

## Tasks Overview

1. Compute the TR5's direct kinematics equations, implement and expose them a ROS service in the **kinematics_node**;
2. Complete the **controller_node** in order to use the direct kinematics according to the explained workflow for the forward kinematics scenario;
3. Test and debug your implementation;
4. Repeat the previous steps for the inverse kinematics.

## Testing Forward Kinematics

1. The user must first instruct the **controller_node** to change its kinematics mode to forward kinematics by calling its ROS service advertised in **"/kin_mode"** topic.

2. By interacting with the **Joint Control GUI**, the user should change the TR5's joint angles as desired. The visualization marker published by the **controller_node** should appear as a blue sphere in the RVIZ GUI, representing the TR5 end effector's position;

3. To validate the forward kinematics, the blue sphere should match the position of the end effector of the TR5's 3D model that is being displayed in the RVIZ;

## Testing Inverse kinematics

1. The user must first instruct the **controller_node** to change its kinematics mode by calling its ROS service advertised in **"/kin_mode"** topic. In **RVIZ**'s screen the red sphere should now disappear, indicating that we are not anymore in forward kinematics mode;

2. By interacting with the **RVIZ** node, the user should move a blue sphere in the virtual scene (see Figure 4). This sphere represents the TR5 end effector's desired position. These coordinates are constantly being sent to the **controller_node**, via **"/input_pose/feedback"** topic;

3. Upon receiving the end effector’s desired position, the **controller_node** must calculate the inverse kinematics using the **“/do_ik”** service topic and publish the resulted joint values in **“/goal_joints”** message topic;