Running the CWRU Abby Arm Simulator

Wyatt Newman
March, 2015

The package folder “abby_arm_simu” within cwru_376_student of our repository contains a Gazebo simulation model of the ABB IRB120 robot arm, which we will be using on our mobile manipulator, “Abby.”

Modelling Issues: The Gazebo model is described primarily in the in the package “abby_description”, which is a sub-folder of “abby_arm_simu.” The URDF (in sub-folder “urdf”) references the “mesh” models contained in a separate folder, “abby_common.” This model was built up based on the rrbot tutorial here: http://gazebosim.org/tutorials/?tut=ros_urdf, plus work by Ed Venator from his M.S. Thesis at CWRU, “A Low-cost Mobile Manipulator for Industrial and Research Applications.” (See thesis posted on Blackboard) and by SouthWest Research Institute (SWRI) via ROS-Industrial (see https://github.com/ros-industrial/abb and https://github.com/ros-industrial/industrial_core).

Initial attempts to make a Gazebo model of the IRB120 resulted in exploding components. It appears this problem is traceable to the defined collision models. The collision models originally were chosen to be mesh models, identical to the visualization models. This resulted in unnecessarily slow code (due to the high resolution of the models), and it appears this was responsible for Gazebo's failures.

This problem has been bypassed by defining crude collision models for each of the links. *However*, these collision models are too crude; they should be improved.

Kinematic modeling is believed to be accurate, including all Denavit-Hartenberg parameters and joint ranges of motion.

Dynamic modelling, however, is not to be trusted. The URDF requires specifying values for mass properties: link mass, link center of mass, and rotational inertias about specified axis directions. Numbers were filled in for these, but no attempt was made to make them realistic. Transmissions are also specified, and these are specified to simply be 1:1 ratios. This is certainly not true. Effort limits are specified for each joint. It is unknown how far off these values are from realistic values.

To make the robot move in Gazebo via user position commands, position-based PID controllers must be specified. The PID gains are declared in a YAML file in the folder abby_control/config/irb120_control.yaml. Numbers were chosen that resulted in acceptable motion. However, there is no reason to believe these values reflect the controller dynamics of the ABB robot. In fact, with presumed 1:1 transmissions and the stated PID gains, the gazebo model of the irb120 experiences significant gravity droop that is not realistic.

In short, there is much room for improvement of the dynamic model.

Running the Simulator: The command to launch the irb120 simulation is
roslaunch abby_gazebo abby_world.launch

This launch command performs multiple actions. It loads the robot model onto ROS's parameter server, and this allows both Gazebo and Rviz to access the same model. In a separate action of this launch file, the robot model is “spawned” within Gazebo (via reference to the parameter server).

The controllers (emulated within Gazebo) are launched via reference to the controller yaml file in abby_control/config/irb120_control.yaml. A "robot_state_publisher" node is also started by this launch file, which results in the robot joint values being published on topic "/abby/joint_states" as messages of type sensor_msgs::JointState. This topic emulates the same behavior as a corresponding ROS-Industrial node that publishes the same message type to topic "joint_states."

Finally, this launch file starts up a node from our "example_robot_interface" package. This node emulates the counterpart of ROS-Industrial, accepting trajectory messages on topic "joint_path_command" which carries messages of type "trajectory_msgs::JointTrajectory." In ROS-Industrial, a corresponding ROS node accepts trajectories messages of this type on this topic name and relays them to the target robot controller. For the irb120 simulation, the same is performed. However, the trajectory points are sent to the robot and executed without interpolation. An industrial robot controller further subdivides these points to produce smooth motion. The Gazebo simulator will thus exhibit coarser dynamics than the real robot.

ROS-Industrial control interface: ROS nodes can drive the simulated arm with identical to driving the physical arm. *However*, care must be taken to be sure the programmed trajectories are dynamically safe and kinematically safe. Possible collisions with the physical arm are not included in the Gazebo simulation. Further, high-speed motions in simulation are not a concern, but high-speed motions on the physical arm should be avoided.

Example nodes for commanding trajectories are contained in multiple packages, including:
example_robot_interface/test_ik_traj_sender
example_joint_space_planner/test_ik_traj_sender2

An interactive command interface is illustrated as follows:

1) roslaunch abby_gazebo abby_world.launch
2) rosrun example_interactive_marker IM_6dof_example
    1. in rviz, add an "InteractiveMarker" display, and select topic "example_marker/update."
3) rosrun example_irb120_IM_interface example_irb120_IM_interface joint_states:= abby/joint_states
4) In rviz, move the 6-DOF marker to a desired pose for the tool flange
5) In another window, trigger motion with: rosservice call move_trigger 1

The irb120 interactive-marker interface window will display output regarding whether the desired pose is achievable, how many solutions were found, and what the target pose is. If there is at least one IK solution, the first solution will be used to generate a trajectory.

In the example interface node, the joint_states topic is subscribed to, and this is used as the first command of the trajectory. This should be done with the physical robot as well. For the example interface, the second trajectory point is arbitrarily chosen to be the home pose. The third trajectory point is chosen to be the first IK solution for the interactively-selected flange pose.

In practice, the joint trajectory should be assembled to be something more intelligent, with a sequence of joint-space poses leading to a desirable solution. The sequence of poses should be assigned time stamps that produce desirable acceleration and deceleration. The target pose should be chosen intelligently among the solution options.

Use of the interactive marker is a step towards hand/eye coordination. Later, the pose implied by the interactive marker will be replaced by a pose implied by point-cloud processing.