ARIAC Tutorial

C. Schlenoff, W. Harrison, and Z. Kootbally

February 3, 2018

Table of contents

Before we start...

- Tutorial will be easier with a USB mouse.
- T1: Open a Terminator terminal.
- T1.1: Open another Terminator terminal from the first one (Shift + Ctrl + i).
- Split T1.1 into two terminals (Shift + Ctrl + o) → T2 and T3.

Gazebo Environment for Agile Robotics (GEAR) Interface

- Provides a ROS interface to the teams for:
  - controlling all available actuators
  - reading sensor information
  - sending/receiving notifications

Running GEAR

- To launch GEAR with a sample work cell environment configuration that contains a UR10 arm and some sensors in various locations T1

```
rosrun osrf_gear gear.py \
--development-mode -f `catkin_find \
--share osrf_gear`/config/sample.yaml
```

- Minimize T1

To paste a command in the terminal

- right-click → paste
- click mouse's middle button (wheel)
- `Ctrl + Shift + V`

Gazebo Environment for Agile Robotics (GEAR) Interface

Starting the Competition

- When GEAR is started, the various competition elements will be in an inactive state.
- Call the following service to start the competition. T2

```
rosservice call /ariac/start_competition
```

Gazebo Environment for Agile Robotics (GEAR) Interface

Receiving Orders

- An order is composed of a set of kits to prepare.
- Orders are communicated to the teams via ROS messages on the topic `/ariac/orders`.
- Teams should subscribe to this topic to receive the initial order, as well as any future order updates.
- Run the following command to see the last order that was published, once the competition has been started. T2

```
rostopic echo /ariac/orders
```

- `Ctrl + C`

Gazebo Environment for Agile Robotics (GEAR) Interface

Querying storage locations of parts and trays

- Parts T2
  - To determine where in the work cell parts may be found:

```
rosservice call /ariac/material_locations "material_type: piston_rod_part"
```

- Trays T2
  - This service can also be used for locating trays:

```
rosservice call /ariac/material_locations tray
```

Gazebo Environment for Agile Robotics (GEAR) Interface

Gripper control and status

- Each arm has a simulated pneumatic gripper attached to the arm's end effector.
- When the suction is enabled and the gripper is making contact with an object, the contacting object will be attached to the gripper. T2
  - Enable the gripper:

```
rosservice call /ariac/gripper/control "enable: true"
```

  - Check the state of the gripper:

```
rostopic echo /ariac/gripper/state
```

  - `Ctrl + C` when done.
- At any point, teams will also be able to disable the suction, causing the detachment of the object if it was previously attached. T2
  - Disable the gripper:

```
rosservice call /ariac/gripper/control "enable: false"
```

Gazebo Environment for Agile Robotics (GEAR) Interface

Submitting trays/AGV communication

- Kits from orders are to be built on trays which are stored on the automated guided vehicles (AGVs).
- When contestants have completed a kit, they should notify the relevant AGV to deliver the tray to the delivery area.
- The ROS service `/ariac/agv{N}` is used for this purpose, where `N` is 1 or 2. T2

Gazebo Environment for Agile Robotics (GEAR) Interface

Submitting trays/AGV communication

- Kits from orders are to be built on trays which are stored on the automated guided vehicles (AGVs).
- When contestants have completed a kit, they should notify the relevant AGV to deliver the tray to the delivery area.
- The ROS service `/ariac/agv{N}` is used for this purpose, where `N` is 1 or 2. T2
  - Spawn a model on the tray on AGV1 (just for demonstration purpose):

```
rosrun gazebo_ros spawn_model \
 -sdf -x 0.0 -y 0.15 -z 0.1 -R 0 -P 0 -Y 0 \
 -file `catkin_find osrf_gear \
 --share`/models/piston_rod_part_ariac/model.sdf \
 -reference_frame agv1::kit_tray_1::kit_tray_1::tray \
 -model piston_rod_part_1
```

Gazebo Environment for Agile Robotics (GEAR) Interface

Submitting trays/AGV communication

- Kits from orders are to be built on trays which are stored on the automated guided vehicles (AGVs).
- When contestants have completed a kit, they should notify the relevant AGV to deliver the tray to the delivery area.
- The ROS service `/ariac/agv{N}` is used for this purpose, where `N` is 1 or 2. T2
  - Spawn a model on the tray on AGV1 (just for demonstration purpose):

```
rosrun gazebo_ros spawn_model \
 -sdf -x 0.0 -y 0.15 -z 0.1 -R 0 -P 0 -Y 0 \
 -file `catkin_find osrf_gear \
 --share`/models/piston_rod_part_ariac/model.sdf \
 -reference_frame agv1::kit_tray_1::kit_tray_1::tray \
 -model piston_rod_part_1
```

  - Call the service to submit the tray for evaluation.

```
rosservice call /ariac/agv1 "kit_type: order_0_kit_0"
```

Gazebo Environment for Agile Robotics (GEAR) Interface

Submitting trays/AGV communication

- Kits from orders are to be built on trays which are stored on the automated guided vehicles (AGVs).
- When contestants have completed a kit, they should notify the relevant AGV to deliver the tray to the delivery area.
- The ROS service `/ariac/agv{N}` is used for this purpose, where `N` is 1 or 2. T2
  - Spawn a model on the tray on AGV1 (just for demonstration purpose):

```
rosrun gazebo_ros spawn_model \
 -sdf -x 0.0 -y 0.15 -z 0.1 -R 0 -P 0 -Y 0 \
 -file `catkin_find osrf_gear \
 --share`/models/piston_rod_part_ariac/model.sdf \
 -reference_frame agv1::kit_tray_1::kit_tray_1::tray \
 -model piston_rod_part_1
```

  - Call the service to submit the tray for evaluation.

```
rosservice call /ariac/agv1 "kit_type: order_0_kit_0"
```

  - Take a look at T1

Gazebo Environment for Agile Robotics (GEAR) Interface

Faulty parts

- There are quality control sensors above each AGV that publish the pose of faulty parts that they see on the tray
  - They are positioned above each AGV in pre-defined locations: users cannot specify the locations of these sensors.
  - They report only the presence of faulty parts: they do not report any information about non-faulty parts.
  - They will only detect faulty parts once they are in the trays on AGVs.

Gazebo Environment for Agile Robotics (GEAR) Interface

Faulty parts

- There are quality control sensors above each AGV that publish the pose of faulty parts that they see on the tray
  - They are positioned above each AGV in pre-defined locations: users cannot specify the locations of these sensors.
  - They report only the presence of faulty parts: they do not report any information about non-faulty parts.
  - They will only detect faulty parts once they are in the trays on AGVs.
- Spawn a part that is known to be faulty on the tray on the AGV T2

```
rosrun gazebo_ros spawn_model \
-sdf -x 0.1 -y 0.1 -z 0.05 -R 0 -P 0 -Y 0 \
-file `catkin_find osrf_gear \
--share`/models/piston_rod_part_ariac/model.sdf \
-reference_frame agv1::kit_tray_1::kit_tray_1::tray \
-model piston_rod_part_5
```

Gazebo Environment for Agile Robotics (GEAR) Interface

Faulty parts

- There are quality control sensors above each AGV that publish the pose of faulty parts that they see on the tray
  - They are positioned above each AGV in pre-defined locations: users cannot specify the locations of these sensors.
  - They report only the presence of faulty parts: they do not report any information about non-faulty parts.
  - They will only detect faulty parts once they are in the trays on AGVs.
- Spawn a part that is known to be faulty on the tray on the AGV T2

```
rosrun gazebo_ros spawn_model \
-sdf -x 0.1 -y 0.1 -z 0.05 -R 0 -P 0 -Y 0 \
-file `catkin_find osrf_gear \
--share`/models/piston_rod_part_ariac/model.sdf \
-reference_frame agv1::kit_tray_1::kit_tray_1::tray \
-model piston_rod_part_5
```

- Check the output of the quality control sensor above AGV T2

```
rostopic echo /ariac/quality_control_sensor_1
```

Gazebo Environment for Agile Robotics (GEAR) Interface

Viewing kit contents

- The `/ariac/trays` topic can be used during development for seeing the pose of parts on the trays in the same frame as that which will be used for kit evaluation.
  - For example, spawn a part on a tray T2

```
rosrun gazebo_ros spawn_model \
-sdf -x 0.0 -y 0.15 -z 0.1 -R 0 -P 0 -Y 0 \
-file `catkin_find osrf_gear \
--share`/models/disk_part_ariac/model.sdf \
-reference_frame agv2::kit_tray_2::kit_tray_2::tray \
-model disk_part_0
```

  - See the reported part poses on the trays T2

```
rostopic echo /ariac/trays
```

  - `Ctrl + C` after a few seconds.

Gazebo Environment for Agile Robotics (GEAR) Interface

Sensor Interface

- A team can place sensors around the environment. Each sensor has a cost that factors into the final score.
  - Break beam: reports when a beam is broken by an object. It does not provide distance information.
  - Laser scanner: provides an array of distances to a sensed object.
  - Cognex logical camera: provides information about the pose and type of all models within its field of view.
  - Proximity: detects the range to an object.
- More information at `http://wiki.ros.org/ariac/Tutorials/SensorInterface`

Gazebo Environment for Agile Robotics (GEAR) Interface

Sensor Interface: Logical camera

- A simulated camera with a built-in object classification and localization system.
- The sensor reports the position and orientation of the camera in the world, as well as a collection of the objects detected within its frustum.
- In the sample environment, there is a logical camera above the bins that store parts.
- Subscribe to the logical camera topic T2

```
rostopic echo /ariac/logical_camera
```

- Ctrl + C
- Take a look at T2

Gazebo Environment for Agile Robotics (GEAR) Interface

Controlling the arm: Command-line

- The arm's controller is subscribed to the `/ariac/arm/command`.
- Use this topic to control all the joints of the arm.
  - First, disable the gripper T2

```
rosservice call /ariac/gripper/control false
```

  - Move the arm over a part in the bin T2

```
rostopic pub /ariac/arm/command trajectory_msgs/JointTrajectory \
 "{joint_names: ['elbow_joint', 'linear_arm_actuator_joint', \
 'shoulder_lift_joint', 'shoulder_pan_joint', \
 'wrist_1_joint', 'wrist_2_joint', 'wrist_3_joint'], points: \
 [{time_from_start: {secs: 1}, \
 positions: [1.85, 0.35, -0.38, 2.76, 3.67, -1.51, 0.00]} ]}" -1
```

  - Enable the suction gripper T2

```
rosservice call /ariac/gripper/control true
```

  - Check that the gripper is enabled and has a part T2

```
rostopic echo /ariac/gripper/state
```

  - `Ctrl + C`

Gazebo Environment for Agile Robotics (GEAR) Interface

Controlling the arm: Command-line

- To move the part over AGV:
  - Not sufficient to send a single point to the arm controller.
  - Needs to include obstacle avoidance.
  - Send a sequence of points to be reached over a course of a few seconds T2

```
rostopic pub /ariac/arm/command trajectory_msgs/JointTrajectory " {joint_names: ['elbow_joint', \
'linear_arm_actuator_joint', 'shoulder_lift_joint', 'shoulder_pan_joint', 'wrist_1_joint',\
'wrist_2_joint', 'wrist_3_joint'], points: [ {time_from_start: {secs: 1}, \
positions: [1.76, 0.28, -1.38, 2.76, 3.27, -1.51, 0.00]}, {time_from_start: {secs: 2}, \
positions: [1.76, 0.38, -1.38, 1.5, 3.27, -1.51, 0.0]}, {time_from_start: {secs: 3}, \
positions: [1.76, 2.06, -1.38, 1.5, 3.27, -1.51, 0.0]}, {time_from_start: {secs: 4}, \
positions: [1.76, 2.06, -0.63, 1.5, 3.27, -1.51, 0.0]}]}" -1
```

- Disable the suction gripper T2

```
rosservice call /ariac/gripper/control false
```

Gazebo Environment for Agile Robotics (GEAR) Interface

Controlling the arm: MoveIt!

- Run the following command to launch the MoveIt nodes that enable motion planning T2

```
roslaunch ur10_moveit_config \
ur10_moveit_planning_execution.launch sim:=true
```

- Interfacing with MoveIt! using the MoveIt RViz plugin T3

```
roslaunch ur10_moveit_config moveit_rviz.launch config:=true
```

1. Select the “Planning” tab
2. Click on “Select Start State”
3. Choose <current>
4. Click “Update”
5. Click on “Select Goal State”
6. Choose <random valid>
7. Click “Update”
8. Click the “Plan” button
9. Click the “Execute” button
10. Resize Gazebo and RViz windows so you can see both
11. Watch the robot move in Gazebo

- Ctrl + C in T2 and T3

Gazebo Environment for Agile Robotics (GEAR) Interface

Controlling the arm: Programming

- We refer you to the MoveIt! tutorials page for details on interfacing with MoveIt programmatically.
- Teams are free to use alternative motion planning and execution strategies entirely.

Gazebo Environment for Agile Robotics (GEAR) Interface

Trial end

- When all orders have been filled, or the time limit for the trial has been exhausted, the competition state published on /ariac/competition_state will change to done.
- If you wish to request that the trial end early for whatever reason, you can do so with the following command (in T2)

```
rosservice call /ariac/end_competition
```

- Check the state with: (in T2)

```
rostopic echo /ariac/competition_state
```