Lab05 - URDF

MTRN4231 - UNSW School of Mechanical and Manufacturing Engineering

Introduction

In this lab, you will learn how to create your own robot model based off a simple end effector. This robot model can then be visualised using RViz and controlled via a ROS2 node.

Task 1 - Understand the description package

Read through the `pizza_cutter_description` directory and understand the package structure.

- `launch` folder contains launch files that spin-up nodes and processes.
- `mesh` folder contains STL files that describe the topology of a physical component.
- `rviz` folder contains RViz config settings i.e. the layout of the RViz window and tools used.
- `urdf` folder contains XML descriptions/models of the robot.
- Note that `package.xml`, `CMakeLists.txt`, and `*.launch.py` have been minimally done for you.

Task 2 - Model a pizza cutter with URDF

Write a `urdf` file to describe a simple pizza cutter in `pizza_cutter_description`.

The pizza cutter should have the following links:

- `pizza_cutter_handle`
- `pizza_cutter_blade`

The pizza cutter should have the following joints:

- `cutting_joint` (continuous joint) between `pizza_cutter_handle` and `pizza_cutter_blade`

After building and sourcing, the launch command is:

```
ros2 launch pizza_cutter_description display.launch.py
```

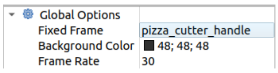
URDF XML joints doc

URDF XML links doc

Task 3 - Setup RViz configuration

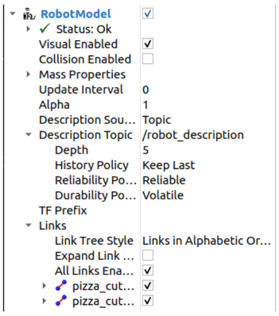
Launching `pizza_cutter_description` should open a new RViz window. Configure RViz so that the URDF can be viewed.

`\Global Options > Fixed Frame"` should be set to `pizza_cutter_handle`.



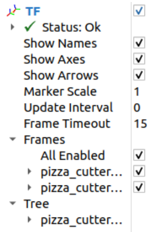
Global Options	
Fixed Frame	pizza_cutter_handle
Background Color	■ 48; 48; 48
Frame Rate	30

Add the `RobotModel` plugin and edit the `Description Topic` to view the `/robot_description` topic.



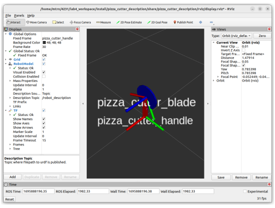
RobotModel	☑
✓ Status: Ok	
Visual Enabled	☑
Collision Enabled	□
Mass Properties	
Update Interval	0
Alpha	1
Description Sou…	Topic
Description Topic	/robot_description
Depth	5
History Policy	Keep Last
Reliability Po…	Reliable
Durability Po…	Volatile
TF Prefix	
Links	
Link Tree Style	Links in Alphabetic Or…
Expand Link …	□
All Links Ena…	☑
pizza_cut…	☑
pizza_cut…	☑

The `TF` plugin can optionally be added to view the transform frames.



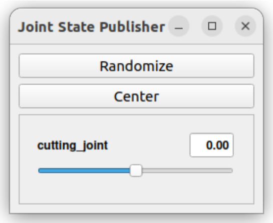
TF	☑
✓ Status: Ok	
Show Names	☑
Show Axes	☑
Show Arrows	☑
Marker Scale	1
Update Interval	0
Frame Timeout	15
Frames	
All Enabled	☑
pizza_cutter…	☑
pizza_cutter…	☑
Tree	
pizza_cutter…	

Save the config as `pizza_cutter_description/rviz/pizza_cutter.rviz`. Your final RViz configuration may look like below:



Task 4 - Control the URDF via GUI

Check that the URDF joints can be controlled with `joint_state_publisher_gui` by changing the joint values in the following GUI:



cutting_joint 0.00

You should be able to see the URDF joints updating in RViz.

Note that `joint_state_publisher` or `joint_state_publisher_gui` is required in the launch file but only the latter will create a GUI.

Task 5 - Control the URDF via publishing

Enable controllable URDF joints by adding desired joint names to the `source_list` parameter of the `joint_state_publisher` node (in the launch file). This will create a topic with the same name as the joint e.g. `/cutting_joint`:

```
parameters=[{'robot_description' : xacro_raw_description, 'source_list': ['cutting_joint']}]
```

Source list allows URDF to listen to topics for joint state value changes.

Control the URDF by publishing to `/cutting_joint` which has the JointState message type:

```
ros2 topic pub --once /cutting_joint sensor_msgs/JointState "{name: ['cutting_joint'], position: [2.5]}"
```

Debug that the message was successfully received by listening to the topic (or by watching RViz):

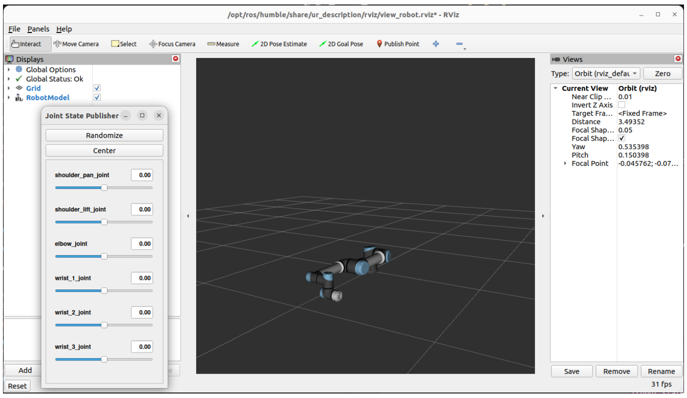
```
ros2 topic echo /cutting_joint
```

Task 6 - Visualise the UR5e

Visualise just the robot arm by running:

```
ros2 launch ur_description view_ur.launch.py ur_type:=ur5e
```

You should be able to see and control the UR5e:



Grid ☑
RobotModel ☑

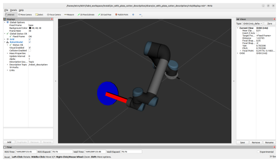
shoulder_pan_joint 0.00
shoulder_lift_joint 0.00
elbow_joint 0.00
wrist_1_joint 0.00
wrist_2_joint 0.00
wrist_3_joint 0.00

Type: Orbit (rviz_defau
Current View Orbit (rviz)
Near Clip ... 0.01
Invert Z Axis □
Target Fra... <Fixed Frame>
Distance 3.49352
Focal Shap... 0.05
Focal Shap... ☑
Yaw 0.535398
Pitch 0.150398
Focal Point -0.045762; -0.07...

Note that the UR5e description package is already installed.

Task 7 - Attach the end effector to UR5e

Create a new package called `ur_with_pizza_cutter_description` which contains a `ur_with_pizza_cutter.xacro` file which attaches `pizza_cutter` to `ur_robot`. The attachment should be a fixed joint between an appropriate link on the end effector to the `tool0` link from the arm.



`urdf` or `xacro` files can be included into `xacro` files. Inclusion is a simple copy-and-paste of the file's contents into the inclusion line - and can be done like so:

```
<xacro:include filename="$(find package_name)/urdf/example.urdf">
```

After building and sourcing, the launch command for the UR5e robot with the pizza cutter is:

```
ros2 launch ur_with_pizza_cutter_description display.launch.py
```

Extensions

- Apply the steps in this lab (and read the documentation) to visualise your own custom end effector with the UR5e.
- Look through the `claw_example` packages for an in-depth and commented example of another (more complex) end-effector.
- This URDF was obtained by using Solidworks URDF Exporter however still needed edits to correct the links and joints.