Autonomous Robotic Systems Engineering (AURO)

Week 4 practical - Control architectures (continued)

Example code

Example solutions for last week's practical can be found here:
https://github.com/alanmillard/auro-practicals/tree/main

Specifically, there are two new ROS packages:
https://github.com/alanmillard/auro-practicals/tree/main/week_3
https://github.com/alanmillard/auro-practicals/tree/main/auro_interfaces

Downloading the code

If you have previously cloned the GitHub repository into a workspace, you should be able to run "`git pull`" from the `src` directory to pull the changes.

If you are starting from scratch, create a workspace and download the packages as follows:

```
mkdir -p ~/auro_ws/src
cd ~/auro_ws/src
git clone https://github.com/alanmillard/auro-practicals.git .

git clone https://github.com/DLu/tf_transformations.git
pip3 install transforms3d
```

Building the new packages

To build the packages, run these commands:

```
cd ~/auro_ws
colcon build --symlink-install && source install/local_setup.bash
```

Running the examples

IMPORTANT: Before you can run the example code, you will need to set an environment variable to tell Gazebo where to find the TurtleBot3 models. You will need to run this command every time you start a new terminal, or you can add it to the end of your `~/.bashrc` file.

```
export
GAZEBO_MODEL_PATH=$GAZEBO_MODEL_PATH:/opt/ros/humble/york/install/turtlebot3_gazeb
o/share/turtlebot3_gazebo/models/
```

You can then run the example launch file as follows:

```
source ~/auro_ws/install/local_setup.bash
ros2 launch week_3 turtlebot3_simulation_launch.py
```

By default, this script will launch Gazebo, RViz, and run the example robot controller called `turtlebot3_random_walk`. This controller code doesn't subscribe to the `/scan` topic, so the robot will bump into walls. Here's how you can run it in the empty world instead:

```
ros2 launch week_3 turtlebot3_simulation_launch.py world:=empty_world
```

To see all of the possible command line arguments for this launch script, run it with the `--show-args` option:

```
ros2 launch week_3 turtlebot3_simulation_launch.py --show-args
```

Try experimenting with the different command line arguments for this launch script, to see how it affects the simulation.

RViz text markers

The `turtlebot3_random_walk` controller publishes its current state and pose to a custom node called `rviz_text_marker`, which in turn, publishes a RViz Marker message that can be visualised: http://wiki.ros.org/rviz/DisplayTypes/Marker

You can add a visualisation for this as follows:

Add > By topic > `/marker_output`

This marker visualisation will be easier to see if you disable the visualisations for TF and Odometry.

If you don't want to manually set up RViz every time, you could save the configuration to a file, and then specify its location with the launch script command line argument `rviz_config_file`.