CS-7630: Autonomous Robotics

Homework

Support: cedric.pradalier@georgiatech-metz.fr
Office hours: 10:00 to 18:00.

Step 0: infrastructure

All PCs are setup with Ubuntu 22.04, ROS humble and CoppeliaSim.

More details about ROS can be found on
https://docs.ros.org/en/humble/Tutorials.html

More details about CoppeliaSim can be found on
https://manual.coppeliarobotics.com/

If the PCs are running windows, you can access linux by rebooting and pressing Escape during the PXE boot process. This will trigger a menu in which you can chose to start linux.

You can login with your normal user account and password, which will make your windows home space available. Note that, when you are logged in, you cannot see the home folders of your project group partners. To share files in a proper way, it is recommended to use the GT github service on github.gatech.edu. To proceed:

1. Login on github.gatech.edu with your GT account (jsmith8) and password.
2. Create a new private repository (e.g. cs8803_jsmith8)
3. Add your instructor (cpradalier7) as collaborators, and potentially your group partner for the larger projects.
4. Use git clone, git add, git commit, git push and git pull to recover your files.

Check the following GIT cheat sheet for reference:
http://www.git-tower.com/blog/git-cheat-sheet/

Note: any other version control system would be acceptable (we also have a gitlab server at GT-Europe).

Step 1: teach yourself ROS

For this set of project we will use the Humble version of ROS2.

Go to https://docs.ros.org/en/humble/Tutorials.html. Within the following list (copy-pasted from the web site), you should specifically do the tutorials highlighted in yellow, and you can probably ignore the barred ones.
Beginner: CLI tools

- Configuring environment
- Using `turtlesim`, `ros2`, and `rqt`

    - Understanding nodes
    - Understanding topics
    - Understanding services
    - Understanding parameters
    - ~~Understanding actions~~
    - ~~Using `rqt_console` to view logs~~
    - Launching nodes
    - Recording and playing back data
- Beginner: Client libraries
    - Using `colcon` to build packages
    - Creating a workspace
    - Creating a package
    - Writing a simple publisher and subscriber (C++)
    - Writing a simple publisher and subscriber (Python)
    - Writing a simple service and client (C++)
    - Writing a simple service and client (Python)
    - Creating custom msg and srv files
    - ~~Implementing custom interfaces~~
    - Using parameters in a class (C++)
    - Using parameters in a class (Python)
    - ~~Using `ros2doctor` to identify issues~~
    - ~~Creating and using plugins (C++)~~

Get familiar with BOTH Python and C++: we will need them during the class

As a suggestion, edit the file .bashrc in your group home folder and add the following line:
`source /opt/ros/humble/setup.bash`
(in case it is not yet so).

Note that you don't need “sudo” access. Any command suggested by the tutorials that requires root access can be skipped.

Step 2: start CoppeliaSim

CoppeliaSim is installed in /cs-share/pradalier/CoppeliaSim.
Launch it by running the following line:
```
cd /cs-share/pradalier/CoppeliaSim
./coppeliaSim.sh
```
Run roscore in an other terminal **before** CoppeliaSim to make it export all its variable to ROS.

Once CoppeliaSim is running, open the scene `rosControlKinect.ttt` in
`/cs-share/pradalier/ros2_ws/src/scenes`
Start playing it by clicking on the triangular icon in the menu bar. The first time a scene is launched, the pre-processing takes some time. The simulation is running once you see the red laser marks on the ground.

V-REP exports the following topics:

`/rosout` `/rosout_agg`	Ignore, used in ROS logging system
`/tf` `/tf_static`	The tree of transformation describing the relation between different objects in the scene. Ignore for now.
`/vrep/depthSensor`	The kinect data points (PointCloud2).
`/vrep/hokuyo`	The laser scanner data (LaserScan)
`/vrep/twistCommand` `/vrep/twistStatus`	Commanded velocity vector (6D) and status. Use /vrep/twistCommand to make the robot move.
`/vrep/visionSensor`	The output of the on-board camera.
`/vrep/visionSensorInfo`	The camera calibration.

Use `ros2 topic list` and `ros2 topic echo` to check the data published by CoppeliaSim but do not spend too much time on it for this homework, will work on the data later. You can also use `Rviz2` to visualize all the 3D data, but this can wait until next homework.

If you want to fully visualize the 3D data, you need to create a ROS workspace and include the `vrep4_helpers` package. Starting the included launch file will produce a 3D point cloud that can be visualized in Rviz2.

Step 3: Implement a joystick control

A set of joystick is available on demand. Please borrow them when you need them. You can keep them during the semester, but please bring them back after the final project.

ROS provides tools to read the joystick state. Check the web for details. If you are working remotely and do not have access to a joystick, a keyboard controller is possible (check the links on canvas) but unpractical. Buying a Logitech joystick may be a relevant option.

Create a small ros package with a node that will receive the joystick data and send corresponding commands to the simulated robot.

Submit the ros package (tar.gz or zip) by email before next Thursday, using CANVAS.