COMP 514 Assignment 2
Flying the Firefly
Sept 26

In this assignment, you will provide a trajectory for Firefly to follow. Your trajectory will be generated from cubic polynomial (3rd order polynomial), and you will provide the positions for the robot.

1 Installation

- Download the package
  `https://gitlab.cs.mcgill.ca/applied-robotics/robots/rotors_simulator`
- You might need to install extra packages:
  ```
  > sudo apt-get install libeigen3-dev
  > sudo apt-get install ros-noetic-octomap-ros
  > sudo apt-get install libgoogle-glog-dev
  ```
- Add `rotors_simulator` to your catkin workspace.
- Build the package by running
  ```
  > catkin build rotors_gazebo
  ```
- Once the new packages are built, you can launch the Firefly by
  ```
  > roslaunch rotors_gazebo a2.launch
  ```

2 How to submit

Create a new ROS package under your Gitlab repository `robotic-coursework-f2022`. Your new ROS package should be called `cubic_polynomial_planner`. When you are ready to submit, push your change to `robotic-coursework-f2022`. Do not create a new repository. Do not add your catkin workspace. When you visit your repository, you should only see 2 ROS packages:

```
husky_teleop_controller
cubic_polynomial_planner
```

3 Useful Tips

- Make sure that you understand the tutorial before working on the assignment.
- Make sure that you add all dependencies to your CMakeLists.txt and package.xml
- If you are not sure how to use a topic or a service, the following commands should help you

  ```
  > rostopic type [topic name]
  > rosmsg info [topic name]
  > rosservice list
  > rosservice type [service name]
  > rosservice info [service name]
  > rossrv list
  > rossrv info [service name]
  ```

4 Steps

4.1 Define Your Own Service

- Create a customized service (i.e., your own *.srv). Your service should have 4 inputs and 1 output. The inputs are the target position (x,y,z) and the target time (T). The output can be a boolean flag or a message to be displayed.
- There are a few things you need to add in CMakeLists.txt. Use `hello_service` as your reference.
  - Add `genmsg` to `CMakeList.txt` and `package.xml`. This is a required package if you have customized services.
  - You need to add `add_service_files`, `generate_messages` before building it, and add dependency on `cubic_polynomial_planner_gencpp`.
- Try "catkin build" at this step to ensuret the service is created. You should find an automatically generated hearder file in `devel/include/cubic_polynomial_planner/[my_service_name].h`. When you run `rossrv info cubic_polynomial_planner/[my_service_name]`, you should see the service you created. **Do not continue if it fails at this step**.

4.2 Create a Service Server

- Write a **class** for the ROS service server and advertise a service called `move_robot`
- When the service is called, the robot should move to the target position specified by the user.
- Provide a call back function for your service. In the callback function, you need to save the following information
  - The current position of the robot in the "world" frame. You have two options
    - Call the service `/gazebo/get_model_state`
    - Subscribe to the topic `/gazebo/model_states`
  - The target position and the target time. You should be able to read it from the service request.
  - The current time. You can get this by `ros::Time::now().toSec()`
  - The duration can be computed by
    `ros::Duration duration = ros::Time::now() - ros_start_time`

4.3 Update function

- Compute the time scale for 3rd order polynomial
- Compute the position at time t. You can keep the orientation as a constant
- Publish the desired pose in the "world" frame to `/firefly/command/pose`

4.4 Launch File

- Create a launch file called "a2.launch" to launch your node
- Check that you can start your node by `roslaunch cubic_polynomial_planner a2.launch`

5 Evaluation

We will test your implementations as follow:

1. Open a terminal and run
   `> roslaunch rotors_gazebo a2.launch`
2. Open another terminal and run
   `> roslaunch cubic_polynomial_planner a2.launch`
3. Open another terminal and run
   `> rosservice call /cubic_polynomial_planner/move_robot 1 2 3 5`
   The robot should move to position (1,2,3) in 5 seconds.
4. We will test your code by trying a few different combinations of inputs

Marks:

[2 points] Correctly create a costomized service

[2 points] Create a class for the service server

[2 points] Correct callback function from the service server

[2 points] Reading correct feedback from Gazebo

[2 points] Robot reaches the target with cubic polynomial