0/1 Questions Answered

PA3 - Coordination mechanisms in ROS

Q1 The problem

7.5 Points

Goal

The purpose of this assignment is to introduce you to the Robot Operating System (ROS) and explicit communication protocol with the robots. In particular, with this assignment, you will learn how to:

- use of ROS tf to place robots in a common reference frame.
- use ROS custom message.
- use ROS services to let robots communicate.
- use ROS actions to let robots synchronize and be aware of the status of other robots.

Instructions

Please read carefully the following tasks and write the program(s) accordingly. *You should do this assignment individually*. Feel free to reach out on Slack in the #help-assignments channel for questions.

Please look also at the general guidelines for writing code:

https://canvas.dartmouth.edu/courses/46138/pages/notes-on-coding-for-the-assignments

Getting ready

Please revise the material we covered so far, in particular the latest about ROS custom messages, services, actions, and tf. You can find the files used as well as additional readings on the description of

each day in the calendar.

It is assumed that the ROS environment is already set up with both Gazebo and `stage_ros`.

The task

The robots should coordinate to get to specific locations. In particular, one robot will be the leader and the other robots will be the follower. You can assume that the robot with lower ID will be the leader, while the others will be the followers. The test can be done in an empty world with 3 robots, where they have to go to waypoints that are vertices of a triangle.

Here a guideline that you can follow to achieve this behavior.

Setting a common reference frame for all robots

1. Modify the robot model `turtlebot3_waffle_pi` (same file modified for PA2) and change from `world` to `encoder` which sets the odometry information coming from the wheel (https://github.com/ROBOTIS-GIT/turtlebot3/blob/master/turtlebot3_description/urdf/turtlebot3_waffle_pi.gazebo.xacro#L66).
2. Create a ROS node that has a tf broadcaster that will set tfs for each robot according to their initial pose, from a global reference frame called `world` to each respective `odom`. Each tf will be set whenever there is a message on `/initialpose` topic, message type `geometry_msgs/PoseWithCovarianceStamped`. To transform Euler angles to quaternion, you can still use tf (http://wiki.ros.org/tf2/Tutorials/Quaternions#Think_in_RPY_then_convert_to_quaternion).

Leader

Create another ROS node for the leader that has the following capabilities:

a. It gets a number of waypoints as input. You can decide how to pass the waypoints, manually, through a ROS message (e.g., http://docs.ros.org/en/api/nav_msgs/html/msg/Path.html), automatically calculated using the regular polygon as implemented

for PA1. You can assume that the number of waypoints is the same as the number of robots available (leader+followers).
b. It has a publisher to publish a custom message to other robot subscribers, to initiate the coordinated behavior, once waypoints are given from step a. Note that the waypoints do not need to be shared at this point with the other robots.
c. It has a ROS service server to handle the registration of followers to the team. The service will contain the robot name so that the leader can retrieve the pose of that robot in the world reference frame through a tf listener.
d. It performs the allocation to specific waypoints in the world. For this assignment, you can determine that in a greedy way, based on the distance between the robots and those waypoints. The leader should be assigned to a waypoint as well.
e. It has a ROS action client to send a goal to each follower. A custom action needs to be created. You don't need to have a callback to get the status, but if you want to include it you are welcome to do so, where the information shared is the current distance to the assigned waypoint.
f. Once robots get to the locations, the leader waits for another list of waypoints and can repeat.

Follower

Create another ROS node for a follower that has the following capabilities:

1. It has a subscriber to receive a custom message type from the leader to indicate that the leader wants to initiate an allocation.
2. It has a ROS service client sending a request to register to the team interacting with the leader service server in the point c for the leader.
3. It has a ROS action server that receives a request to move from the current location to the goal set by the Leader (point e. for the leader, using the custom action created).
4. It waits then for another allocation initiation by the leader (point a).

Report

Please write:

1. a report in markdown in a file called `report.md` to discuss your implementation. Then use pandoc or something similar to generate a pdf. Include both files in your submission, as well as all figures (as pdf). The report should contain the following information:

   a. Header with the code of the class, the term, and year, the assignment number, and your name.
   b. A short typewritten report (at least one full page of complete sentences) describing your results. The report should contain three sections:
   (a) Method description: How does your program work? What design decisions did you make?
   (b) Evaluation: Does your program actually work? How well? If it doesn't work, can you tell why not?

2. a README file in markdown with the instructions on how to execute the program.

Video

Please record also a short video showing the robots coordination in action.

Q1.1 Grading rubric and submission

7.5 Points

Grading

Your programs will be evaluated based on both their functionality and their coding style. In the notes for writing programs, you can find an informal style guide to help give you an idea of what is expected together with the coding style that you should follow. In particular, the following aspects are considered:

Performance (5.5):

- Correct setting up of the tf.
- Correct setting up of the leader.
- Correct setting up of the follower.

Style (1):

- No duplication of executable code?
- No magic numbers?
- Names match functionality?
- Adequate comments?
- Comments match code?
- Consistent formatting?

Documentation (1):

- Report is complete and clear.
- Required sections exist under readily identifiable headings.
- Free of typos and grammatical errors.
- Video included

Submission

Please submit a single zip file containing the full ROS package, README, report, and video. You are welcome to write any comment about the assignment/submission in the following text box.

Enter your answer here

Please select file(s) Select file(s)

Save Answer

Save All Answers Submit & View Submission >