Robert Nick Vandemark
ENPM690 Project Proposal
10 Mar 2023
nickvand@umd.edu

Project Name

RON, an acronym for ‘Robot of Nodes’.

Team Members

None, just myself.

Abstract / Introduction

RON is an idea inspired by a video I saw on LinkedIn that I wanted to replicate a version of, but it uses robot learning techniques to optimize higher-level orchestration and custom motion planning and control techniques to implement lower-level commands. The ‘robot’ consists of N independent robotic nodes and M independent bridge nodes (which are really just magnetic spheres that the robotic nodes use to at/detach to, not robotic actors themselves). The system can then learn how to orchestrate motion of the robotic nodes, utilizing the other robotic nodes and the bridge nodes, to reach a goal with some quantifiable metric that can be optimized. For example, there can be simple tasks such as the nodes stacking to reach a certain height by any means, or more complex tasks such as the collection of nodes creating a general shape and then following a path as a conglomerate mobile robot.

Objective

I imagine three main goals for myself:

1. Learning how to integrate machine learning into a large-scale project by supporting both an automated ‘learning’ mode, where it learns how to optimize its behavior autonomously, and a ‘production’ mode, where it executes the behavior it has learned.
2. Improving my skills with ROS2 control in general, such as deploying controllers and visualization with a variable number of robotic actors, and dynamically (de)activating controllers depending on the current control problem (differential-drive control, omni-directional control, etc).
3. Exploring the performance limits of the chosen machine learning algorithm as the complexity of the desired behavior increases.

Methodology

Development began earlier this semester in anticipation of pursuing this task for this project, and is hosted on GitHub here. The approach will be built on top of ROS2, and will largely use C++ and perhaps Python where necessary. Visualization will be supported via Gazebo and RViz for ROS2. Some low-level control will be accomplished with open source solutions such as the ROS2 control

differential drive controller found here, but some will likely be custom solutions (for example, multiple robotic nodes attached to a single bridge node can be modeled as a conglomerate omni-directional robot, but this singleton controller would have to know about all the robotic nodes' command interfaces to be able to dynamically switch between them, and there is no open-source solution to this).

As mentioned previously, machine learning will be used to optimize the higher-level requests, and the lower-level controllers are responsible for executing this intelligence's higher-level commands. As to what exactly these will be are still to be determined (and perhaps could be dynamically configurable), but they might be at a level of abstraction comparable to the following:

- At/Detach robot node *i* to bridge node *j*
- Move a robot node *i* that is not currently attached to a bridge node to a 3-DoF Cartesian pose (*x*, *y*, *gamma*)
- Move robot node *i* to the 2-DoF polar coordinates (*theta*, *psi*) of the bridge node it is currently attached to

To accomplish this learning, a genetic algorithm will be employed. When this system is in its 'learning' mode, the exact implementation is to be determined, but could be a ROS2 node which hosts a server for a custom action that simulates the performance of a given generation of solutions. Then perhaps this same ROS2 node hosts a server for a custom service which handles the crossover and mutation operations. The configuration of these operations can be tuneable via the parameters of the ROS2 node, enabling learning for any set of state parameters and corresponding fitness function. The details of the implementation are to be determined, but definitely fits into an application based on ROS2.

With all of that said, this is obviously a rather ambitious goal, especially when considering the motion planning and navigation control utilities that would be required for a full-fledged simulation. So, the bare minimum objective is to implement the machine learning strategy in a demonstrable manner (learning and applying the learned behavior to some sort of simulated system).

Tools

Many of the tools that are being employed have been discussed above. Notably, these are the ROS2 stack, the ROS2 control packages, and Gazebo and RViz for ROS2. For now, the scope of this project will remain entirely in simulation, so now expected use of hardware.

Links

The LinkedIn video URL: https://www.linkedin.com/posts/richard-hulskes_freesn-a-freeform-strut-node-structured-activity-7022073359785967616-n129?utm_source=li_share&utm_content=feedcontent&utm_medium=g_mb_web&utm_campaign=copy

GitHub repository: https://github.com/rnvandemark/ron.git

ROS2 control differential drive controller:
https://github.com/ros-controls/ros2_controllers/tree/galactic/diff_drive_controller