Project Chakravyu

Jay Prajapati, Shail Shah, and Shantanu Parab

I. Introduction

Simple robots working together in a multi-robot system to accomplish predetermined objectives. Robots in these systems are much less powerful when acting alone, but the actual power comes from their ability to work together. Industrial robots with several uses can automate a variety of manufacturing processes. A single industrial robot frequently has the capacity to carry out several different kinds of tasks. The scope of this study encompasses all of robotics' related subfields, including robotic control systems, planning, and perception. These systems are widely used in warehouse robotics, cooperative robotics, swarm robotics, unmanned aerial vehicles, etc. In this project, we'll create a software module that uses C++ and ROS to control more than twenty ROS TurtleBots at once to create geometric patterns.

II. Keywords

Multi-Robots, Swarm Robotics, Swarm Robotics Algorithms, ROS2, C++, TurtleBot, AIP, TDD

III. Process

Twenty ROS TurtleBots will be simulated in Gazebo using ROS2 Humble Hawksbill, and geometric patterns will be generated utilizing the individual robots as points to build, for example, rectangles, triangles, and squares. To accomplish the objective, we will use C++14 and higher to develop the node structure for every robot with to a chosen swarm robotics algorithm. For the creation and implementation of our software concept for ACME Robotics, we will use an upfront class of models referred to as "designs". This will go over the primary traits/features of the software we'll be developing. Using the Test-Driven Development methodology, we will combine our class of models with the Agile Iterative Software Process (AIP). Given that the Agile Iterative Process involves consensus decision-making, this would lead to a comprehensive evaluation of every step of the software development process and prevent any misinterpretations of customer requirements. Additionally, the Google Test Framework will be used to write the test cases for the unit testing technique, and the Cpplint and Cppcheck tools will be used to review the code. GitHub CI will be utilized to implement Continuous Integration, and Coveralls will be used to provide the code coverage report.

We will be using our robot swarm to generate a geometric pattern in an empty field. For this we will be using a fleet of 20 TurtleBots 3.0 from ROS these robots will be controlled by a master to perform the geometric pattern formation. We will be using the pose estimation data from each of the robots and then giving them new coordinates to be reached depending upon their current position. We will be testing different tools to provide an optimal solution which will be tested and improved throughout the process. We will be trying the following approaches

1. Using MATLAB Swarm Robot Environment.

   Here we will be using the MATLAB Swarm Robot environment to design the behavior of the robots. We will then simulate the behavior in the Mobile robotics toolbox in MATLAB. Then we will generate the C++ code for the same and migrate it in ROS. The system will then be simulated in Gazebo using multiple TurtleBots.

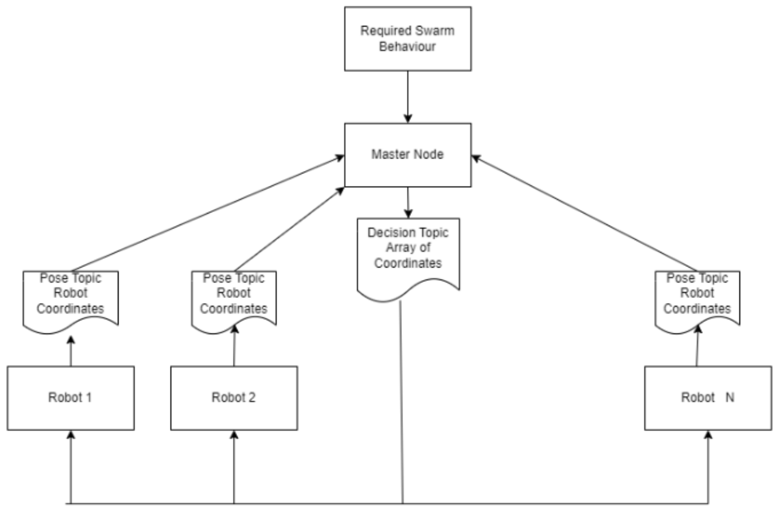
2. Using ROS2 RMF

   ROS2 has provided a package for robot fleet management which allows us to manage a fleet of robots in a particular area. We will be using this package to manage our fleet of robots. We will first build the blueprint of the field and expert it as an image. This image will be used as the base map in the traffic editor tool to plan the behavior of the robot fleet. The traffic editor provides us with a yaml file that is used to simulate the behavior of the robot in Gazebo.

3. Crude Method Using Swarm Algorithm

   We plan on implementing a crude and simplistic swarm algorithm in which the decision for the robots is made by a master based on the required pattern and all the robots get their own goal coordinates through respective topics.

The architecture of our system will be as follows,


Required Swarm Behaviour
Master Node
Decision Topic Array of Coordinates
Pose Topic Robot Coordinates
Pose Topic Robot Coordinates
Pose Topic Robot Coordinates
Robot 1
Robot 2
Robot N

IV. Risk Identification and Mitigation

The fleet of robots while performing the prescribed task of aligning to form a geometrical pattern might collide during movement. Communication error due to frequency mismatch between nodes might lead to robots going haywire and can disrupt the process. Disconnection of master node or slave robots from the network might lead to inconsistent geometric patterns being formed or a complete standstill of entire process.

Mitigation:

To avoid the risks mentioned, we intend to implement a collision avoidance logic to avoid collisions. The message buffers for the reception and publishing of information will be sufficiently big enough to accommodate the frequency mismatch in the master and slave nodes. The network will be such that it will accommodate the loss of a few slaves in the geometric pattern.

V. References

[1] A. Barciś, M. Barciś and C. Bettstetter, "Robots that Sync and Swarm: A Proof of Concept in ROS 2," 2019 International Symposium on Multi-Robot and Multi-Agent Systems (MRS), 2019, pp. 98-104, doi: 10.1109/MRS.2019.8901095.

[2] https://osrf.github.io/ros2multirobotbook/