begindocument/before

Inverse Kinematics Module Proposal

Fabrizzio Coronado* and Kautilya Chappidi†
Maryland Applied Graduate Engineering
University of Maryland
(Dated: November 29, 2023)

I. OVERVIEW

A. Purpose, Objective, Scope

Swarm robots are a specialized type of robotic system crafted to collaborate autonomously, drawing inspiration from the coordinated behaviors observed in natural swarms like groups of fish or birds. The primary aim of swarm robots is to harness the advantages of decentralized control, allowing them to take on complex tasks that would be difficult for an individual robot. These tasks often involve exploration, mapping, monitoring, or manipulating environments. Swarm robotics seeks to amplify the overall efficiency, adaptability, and resilience of robotic systems by enabling collaboration, collective intelligence, and adapting dynamically to changes in their surroundings. The potential applications of swarm robotics are diverse, ranging from search and rescue operations to environmental monitoring, agriculture, and industrial automation. In these fields, the scalability and versatility of swarm systems offer innovative solutions to intricate challenges.

The aim of this project is to create a demo where 20 or more turtlebot3 to follow each other in a line like fashion. This objective is applicable for situations where many robots are needed like search and rescue, care package delivery and more. The scope of this project is strictly limited to having all turtlebots follow one leader. If there is extra time available, attempts will be made to map an area using all the robots. This proposed project will be code named as project Hive. The project will follow a master slave framework. The master node will broadcast a command one at a time to each robot. The robot will receive, complete the command and report to the master node that it has

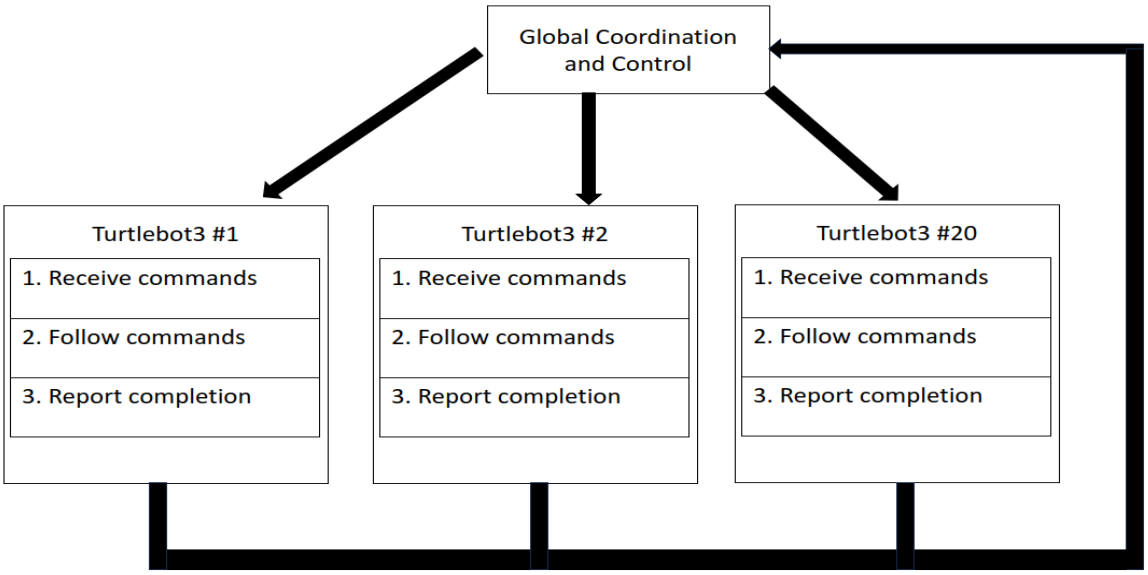

FIG. 1. Block Diagram

* fcoronad@umd.edu
† kautilya@umd.edu

completed the command. The master node will then send the next command. This will continue until the sequence of commands is completed by each robot. An illustration of this concept is shown in figure 1.

B. Technologies and Assumptions

This project will be written primarily in C++ but may also include python, yaml, XML and more. For libraries, we will be using: tf2, tf2_ros, rclcpp, geometry_msgs, and possibly more. CMake will be used to build the package on Ubuntu 22.04 with ROS 2 Humble. Github is where the module will reside and be updated. Code coverage will be generated using Github CI along with CodeCov for automatically running unit tests and generating the reports. The unit tests will be conducted using the GoogleTest module. Additionally, formatting and bug checking will be performed using cppcheck and cpplint. We will be using the Apache 2 license to disclaim liability and keep the module open source. Lastly, doxygen will be used for documenting all class members. We assume the user has C++ 14 installed.

C. Risks and Unknowns

Possible risks associated with project Hive include collisions between turtlebots, communication break down and resource limitations. Collisions will be mitigated by spawning the robots at a safe distance from each other and having them move at a slow speed so sliding when stopping is minimized. Communication breakdown will be avoided by having a low publishing rate from the central node to all nodes. Additionally, we can have a more than substantial buffer to accommodate for lag. Lastly, the hardware on which the demo is run attribute to resource limitations. We cannot assume everyone's machine has the performance to run the demo however low publishing rates, small to moderate buffers, following OOP principles and leveraging CPP memory management.

II. PROCESS AND ORGANIZATION

During the design, integration, and testing phases of this module, we will adhere to an agile iterative process, ensuring that development is test-driven to deliver a high-quality product to Acme. Following the Agile Integration Process (AIP), we will implement a driver and navigator approach. Before pushing substantial commits to the repository, the navigator will review and refine the work accomplished by the driver. Once both parties reach consensus on the changes, the commit is approved. The two pairs will collectively assess the modifications during the iteration meeting and subsequently update the product and iteration backlog. Each team member will be responsible for documenting the time spent on task completion in the time log.

This process will be following during the development of the project and will be split into 3 phases: Phase 0, 1 and 2. Phase 0 is the current phase where we develop this proposal. Phase 1 will consist of sprints where the driver and navigator get together to plan what work needs to be done for the day. Phase 2 will be a continuation of Phase 1 plus refactoring, verification testing, finalizing the module design, repository and more.

While AIP typically consists of 5-10 people, our team consists of two members: Fabrizzio Coronado and Kautilya Chappidi. Each member will be switching roles as driver and navigator. Roles will be switched at the end of each iteration, which will typically last between 3-4 days. All pair programming work will be docuented in the product backlog, iteration backlog, time log and meeting notes.

A. Deliverables

The final deliverables submitted to Acme will be the following:
1. Proposal, 5 minute presentation, UML diagram
2. Demo involving ROS and Gazebo
3. AIP items: time, product and iteration logs, meeting notes
4. Code stubs and unit tests already integrated with Github CI and code coverage with CodeCov
5. Doxygen documentation
6. Github repository containing all the items listed above

The initial proposal and UML Class diagrams which are subject to change.