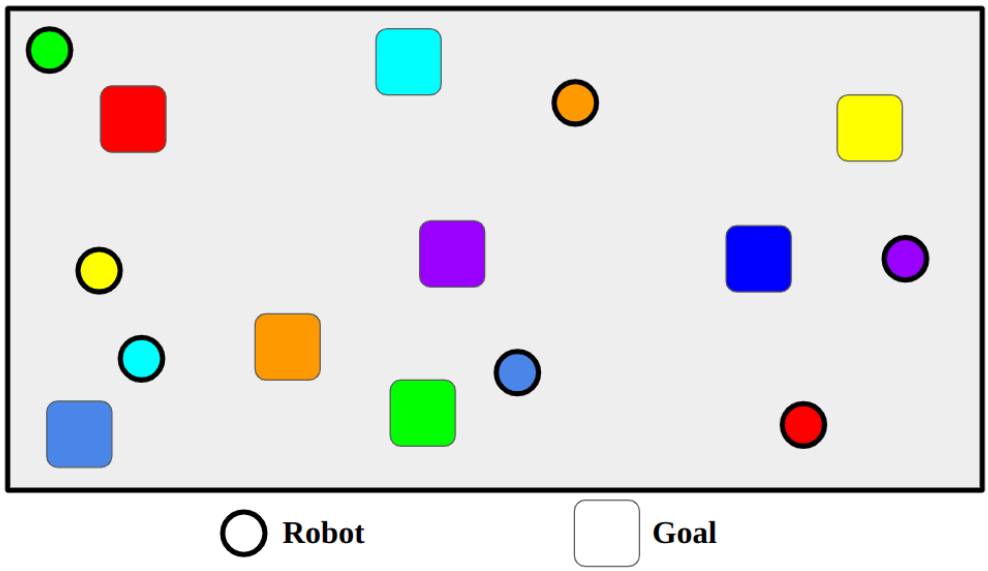
ENPM808X Final Project Proposal: Dynamic Fleet Management (DFM)

Vikram Setty (119696897)
Vinay Lanka (120417665)

Overview

To assist Acme Robotics' swarm team, we (Vikram and Vinay) are proposing **Dynamic Fleet Management (DFM)**, a swarm robotics platform that incorporates multi-agent collision avoidance and path-planning capabilities to lead multiple robots to their goal positions in a shared obstacle-ridden environment. A simplified simulation example is shown below. With robots and goals sharing the same color as shown, the robots would plan paths and simultaneously avoid collisions to reach their goal.



The applications of DFM include use in Acme Robotics' warehouses where goods can be transported over short or long distances in real-time by customizing goal positions and swarm size. With warehouse goods often becoming disproportionately large and heavy, transporting them through an autonomous swarm system would lead to a big save in time and cost for Acme Robotics.

Methodology

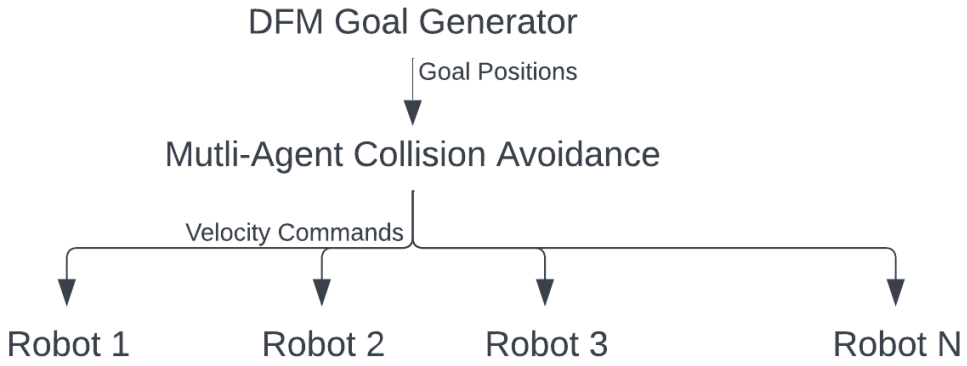
1. Technique

The swarm robotics stack would take the approach of spawning vehicles in the Gazebo simulator with their respective robot namespaces for individual control. The goal as specified is to spawn and control 20 such vehicles. We're planning to use the Turtlebot3 with different namespaces and give them various goals and have them carry the tasks out with multi-agent collision avoidance.

The main challenge in this project is the multi-agent collision avoidance and planning. We intend to check out a couple of approaches for this particular problem.

1. Open-RMF - The Open Robotics Middleware Framework (Open-RMF) provides a framework for robot fleet management with tasks, goals, and collision avoidance being a core component of the same.
2. RVO2 - RVO2 is a multi-agent collision avoidance library that we can leverage for collision avoidance while giving out our goals and can be integrated into our solution with a greater level of flexibility in usage.
3. Custom Swarm Control - We can write a crude version of the goal planning and collision avoidance library keeping the positions and trajectories of all the spawned robots in mind which offers us the greatest level of flexibility.

Each method has its merits and we intend to flush out each idea and go forward with the best approach for our use case.



System Architecture Overview

2. **Algorithms and Software Tools**
   We use the following tools and libraries to develop our fleet management system. The final system is developed to be a ROS 2 package.
   - Gazebo Classic Version 11 - Simulator.
   - Turtlebot3 - Robot platform used for deployment.
   - OpenRMF/RVO2 - MultiAgent Collision Avoidance.
   - ROS2 Humble Hawksbill - Robotics middleware.
   - Colcon - Build system.
   - Doxygen - Documentation.
3. **Development Process**
   The development process would take place using the Agile Iterative Process (AIP) methodology. Using two iterations/sprints and a well-maintained product and iteration backlog, the DFM would be ready to be rolled out in two weeks. Each sprint would last one week and would have a sprint plan and review meetings that are well documented. The development process at the most basic level would make use of pair programming, with each task in the product and iteration backlog being assigned a driver and navigator. While the driver would work on the implementation of the task, the navigator would ensure that the code devised by the driver conforms with the rest of the project and other design principles.

Potential Bottlenecks

Some potential bottlenecks and possible solutions we think we might encounter are listed below.

- The Gazebo simulation is highly resource intensive and could be a major bottleneck during development as we might not be able to spawn 20 robots with the current development hardware setups. A possible solution would be to run the simulation in headless mode and visualize the position of each robot with a custom plot.
- The various approaches for multi-agent collision avoidance might prove to be extremely tricky to integrate especially in the given time frame of the project. We intend to plan around this by quickly testing out each approach with the estimated timeframe as a parameter and consider an approach keeping our three-week time frame in mind.

References

1. https://gamma.cs.unc.edu/RVO2/
2. https://openrmf.readthedocs.io/en/latest/
3. https://docs.ros.org/en/humble/index.html