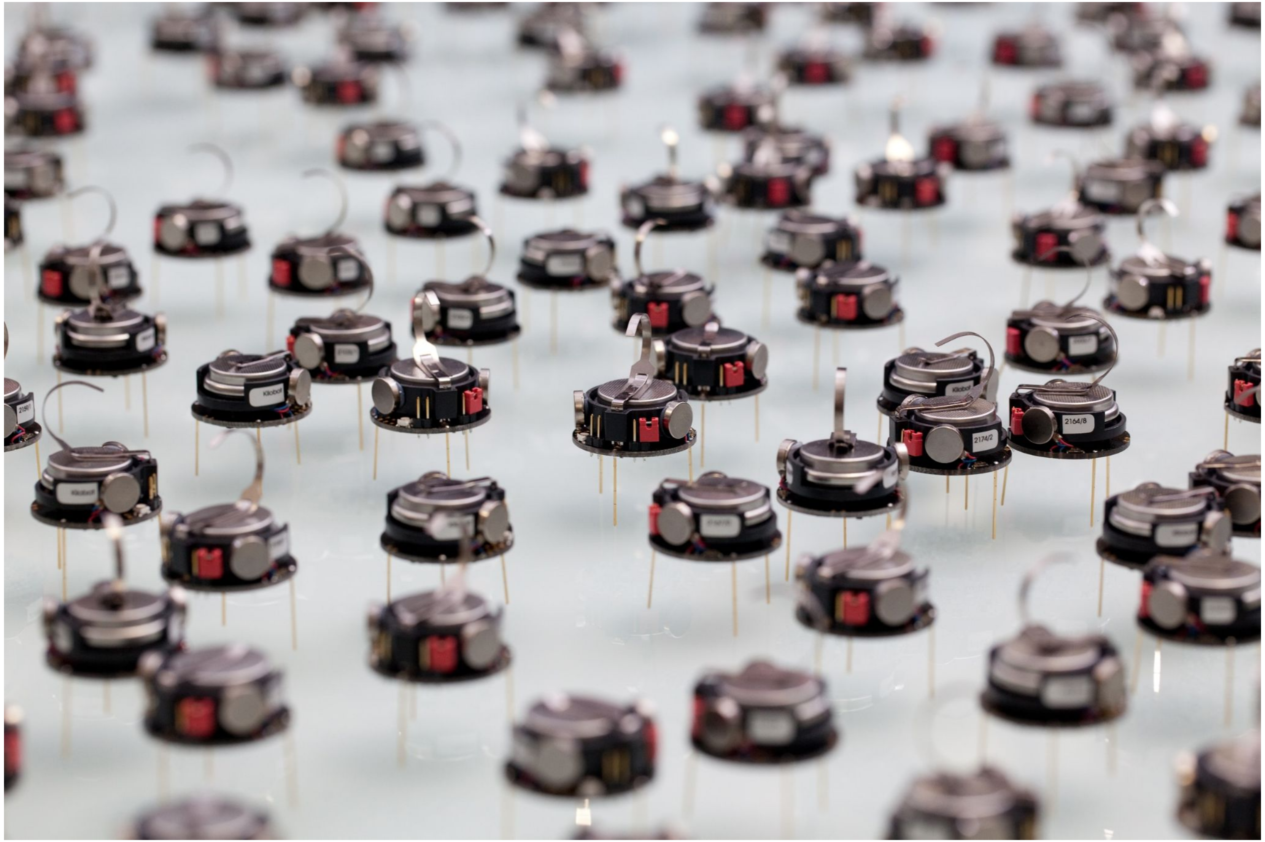
Decentralized Task and Path Planning for Swarm Robots

About Me?

Swarm Robotics, the Market and Design Challenges



What is the problem?

Bottlenecks in deployment:

- Path Planning
- Task Assignment

Centralized systems don’t scale to 5,000 to 10,000 agents!

Potential Solutions

1. Seed Based Synchronization:
   - The same way minecraft generates an exact world using a seed.
   - Given a common map, poses of all robots, and a common seed, simultaneously path plan for all agents in a swarm.

2. Temporal Graph Path planning:
   - Ensure each target pose is reached at a specific time.

Potential Solutions

3. Decentralized Localization:
- The agent knows it’s own location, and the ArUco marker of another, it will localize that robot relative to itself.
- When you globally localize one robot across the swarm, it yields the positions of all robots without centralized computation.

4. Real Time Status Sharing:
- Achieving this with minimal dependency on network infrastructure.
- All tasks should be communicated and acknowledged in **~5 seconds**, for **5,000-10,000** robots.

Potential Solutions

5. Distributed Job Execution Framework:

Build a framework in each agent using:

A **DistributedJobExecutor** class/object:

- Owns unique pointers to locations with agent status data (updated via HTTP syncing).
- In-memory cache.
- `tick()` method to update statuses every second.
- `compute(function_pointer)` method to perform computations on accessible data.

Potential Solutions

6. Avoid Compute Offloading (Initially):

Build algorithms that

- Accept a pool of tasks and compute the **same assignments** for every agent *on every agent*.
- Accept a map/graph, current pose, and target pose, and compute paths ***for every agent on every agent.***

What Does “For Every Agent, On Every Agent” Mean?

- Given the same input, each function must output the same result on every agent.
- Then each agent can act, confident that others have computed non-conflicting tasks/paths.

Constraints

- Algorithms must **scale to 5,000–10,000 agents**.
- Run all **behavior-control applications in Docker containers** (as done in industry).
- Use **C++17 standard library** exclusively.
- Prefer **mathematically derived solutions** before coding.

Constraints

- **No AI solutions**.
  1% margin of error on physical control systems can result in human death.

- Strict documentation and development pipeline:
  - Write a feature doc
  - Divide work and split tasks
  - Implement
  - Test
  - Integrate
  - Demonstrate

Important Questions - Intuitive and from peer review

- How can we decentralize path planning and task assignment?
- Can/should these tasks be done under a unified robot job decentralization framework?
- For Decentralized path planning, why would there be a need for a random seed if there is a common goal to be performed?
- Do collaborative robots in general typically share their entire path or just a destination and some vectors and the receiving bot needs to compute the path of the peers?
  How is path typically represented in existing systems today when they are being processed by a robot? (technical perspective)

AI Usage Guidelines

Vibe coding and mounting tech debt can kill your project within a week.
Do not rely on AI, or indulge in the use of AI, beyond using it for menial tasks.

Timeline

Weekly plans

1. Requirement definition and architecture - what does it mean to be real time
2. Making the code deployable and setting the design patterns and data structures
3. Filling in and refining the logic
4. Testing against the performance indicators determined in week 1
5. Reports and documentation

Intern Requirements

Number of interns: 3

Necessary skills/prerequisites for the interns:

C++

Nice to have skills for the interns:

STL containers and smart pointers, ROS

Resources Required for the project

Laptops that can run turtlebot simulations

Mentor Requirements

Nice to have mentor skills:

C++, STL, Gazebo, ROS

Number of mentors required: 3

Current mentors: Aditya Naskar, DL Rameshwar, Sriram Radhakrishna

Thank you