Read me

Multi-agent Path Planning using Artificial Potential Fields in Cluttered Environments

Firine Bugenhagen, Rens Galesloot, Andrei Stefan

May 25, 2025

1 Project Overview

The implementation of autonomous robots is becoming increasingly widespread, with applications for instance for factories, search and rescue missions and delivery systems. Navigation is one of the biggest challenges that these autonomous systems face, especially in areas where the terrain is highly cluttered and unknown. The problem becomes even more complex when several agents are required to work together in planning a path. Lunar Zebro, a lunar micro rover developed by the Delft University of Technology, is aiming to apply such a multi-agent strategy for traversing the harsh lunar environment. Due to the limited power, memory, and communication capabilities, a lightweight and robust real-time multi-agent path planning algorithm must be developed.

The artificial potential field (APF) method is an effective and low-complexity method for single-agent path planning; however, little research has been conducted on the application for multiple agents. This project aims to build upon the existing methods that have been developed for single-agent path planning, by making an assessment of the improvements that swarms can yield.

This project was conducted at Delft University of Technology as part of the Honours Bachelor's programme, 'Next Generation Robotics'.

2 Simulation Overview

The project is divided into multiple levels to investigate different aspects regarding swarm-based navigation. In Level 1 the influence of swarm size and initial distribution of the different agents is analysed. In Level 2, different APF algorithms (CAPF, BAPF, CR-BAPF* and RAPF) and A* are implemented and compared for a swarm setting. In Level 3, three different inter-agent collision avoidance methods are tested. Level 4 analyses the communication range needed within a swarm.

The performance parameters used to assess each simulation are reachability of the target, (effective) path length and computational complexity.

An overview of the file structure of the GitHub can be seen in Figure 1.

Simulation

Setup:
- Stores settings and Metadata for current run
- Used to initialise agents and environment

Environment:
- Stores data (e.g. obstacle positions) for current run

Agent:
- Class for each single agent

Agent algorithms (folder):
- A_star, bapf, capf, cr_bapf, cr_bapf_star, rapf
- Multi_agent_rapf (includes collision methods)
- L4_rapf, grid_node

Communication:
- Protocol for communication between agents

Hyperparameter tuner:
- Used to find appropriate hyperparameter for simulation

Run Simulations

Analysis_settings_levels:
- Stored settings for simulations of the different levels

Main files:
- Single run to visualize simulation:
  - main
- Monte Carlo files for different levels:
  - mc_main_scattering_L1
  - mc_main_swarm_size_L1
  - mc_main_algorithm_L2
  - mc_main_algorithm_L3
  - main_L4

Evaluation:
- Find performance parameter for single run

Arrays_Storage (folder):
- Store .npy files with results of each run for simulations

Arrays_Storage_Control:
- Handle .npy result files

Analyse Simulations

Plot_results:
- Create graphs of results of different levels

Results (folder):
- Stores created result graphs for each level

Table_generator:
- Transforms results from .npy to table form

Figure 1: Overview Code file structure