Agri Robot Mapping and Pathing Progress Report

Toby Towler

registration: 100395626

1. Introduction

This "Agri Robot" was created by several masters students in a previous year, I am working in a team to continue its development into a fully functioning product. The robot is designed to function as an automated lawnmower, taking in a virtual map and mowing the entire area efficiently and accurately. The robot will use an RTK chip to determine its location, as well as wheel odometry. It will be able to detect its surroundings using a combination of LiDAR and camera technology, using these inputs to alter its path when necessary, for example if there is an animal in its inteded path. By the end of this project, the robot should be able to know where it is, what's around it and where it's going next to a high degree of accuracy. The final aim of this project is to have a finished, publishable and sellable product that could be used in the real world.

The tasks I have been working on so far are:

1. Random map generation
2. Path generation
3. Map generation from an aerial image

My motivation for this project is to make it run fast and efficiently, on the lowest specification hardware possible. The algorithms produced during this project should be efficient in terms of both memory and processing power while still completing the task to an acceptable standard. In a real world business context this would mean the production cost could be lowered, allowing either a lower selling price or higher profits.

2. Progress

2.1. Mapping

The robot should be able to cover a full area regardless of the shape of the given input, So there must be a way to generate varying maps to test all steps of the process on.

Firstly, I had to build an algorithm to generate maps to represent fields. The maps are made up of the outline of the main field, and can have a series of "holes" these would be trees or telegraph poles for example in real life, areas in the field that must be avoided by the robot.

My solution was the following:

1. Generate n random (x, y) points
2. Select origin point with lowest y value
3. Sort remaining points by polar angle to the origin point

This always produces a full polygon that never overlaps and can have any number of vertices where n < 2 For simplicity, I used a fixed origin of (0, 0) and generate points with positive x and y coordinates. Although the algorithm could be adapted to work with all random points. The polar angle method comes from the Graham Scan for a convex hull GeeksforGeeks (2023). My algorithm is based on the first part of this idea. This algorithm is repeated for the outline and the holes in the field, due to the varying number of vertices and random point generation, all polygons in the field can be different.

2.1.1. Fields2Cover Mapping

Fields2Cover (F2C) is a Complete Coverage Path Planning (CCPP) open source library, see next section. I used this library primarily for CCPP, but it also has methods to generate fields of its own, Which I used this for testing the F2C pathing solutions.

This method words by taking parameters for the area of the field and the number of vertices required. There are no essential differences to using the F2C maps or my own algorithm as they both return a collection of 2-dimensional points meaning that the CCPP solution should be able to work with either map as input.

2.2. Complete Coverage Path Planning

Complete coverage path planning (CCPP) is the process of making a path for an object of width x to traverse an area while covering all points of the given area. This is important for the agri robot as it needs to mow an entire area of grass leaving no area uncut to be a successful product.

My advisor showed me a paper tackling this problem with an open source library linked to it. The library is called Fields2Cover (F2C) Fields2Cover (2023), it is written in C++ and available as a C++ or python library. At the beginning of trying to use the library, I had several compilation errors when trying to build from source. I was able to fix the issues myself and then submit a GitHub pull request that was later approved, meaning I am now a contributor of this open source project.

After fixing these issues, I was able to use the library to generate CCPP solutions to inputs of any map. There are several options to differ the final path making this a very flexible solution for a number of different applications This library also allows easy configuration of the sizing, turning circle and velocities of the object to follow the path.

2.3. Map Generation From An Aerial Image

The idea of this task is that a drone could fly above an area and a suitable map be automatically generated with no user input.

As I have currently only just completed the CCPP problem, I have very little to show for this task. I found some papers to read while doing previous research for mapping, reading these will be my next step as I try to find an appropriate algorithm.

3. Progress Evaluation and Risk Analysis

A big problem I have found so far during this project is finding algorithms or real world solutions that also apply to this problem. Specifically for the CCPP aspect. There were very few open source algorithms or even discussions about a possible solution. This was especially bad when the F2C source code did not compile, as this is the only way to use the library. I spent a lot of time changing header files as well as C++ and Python build files in order to get it to finally work, because of this it took me a lot longer to implement some features than I would have liked.

If the papers I have found so far for the aerial mapping problem do not prove very useful to this application, This issue could repeat causing me to spend more time researching instead of adding features to the system. Should this occur, I will analyze and talk to my team about if it would be better to move onto another feature where I could use my time more efficiently. If the issue proves too hard to implement, I will also do this.

One concern I have about the future of this project is that additions to the code base will require libraries too big for the onboard memory of the Raspberry Pi, or that some actions may be too complex for the processor to handle in real time. If this does happen, I will need to change the way that my solution is implemented or change pre-existing parts of the code base to allow for greater consumption.

4. Gantt Chart

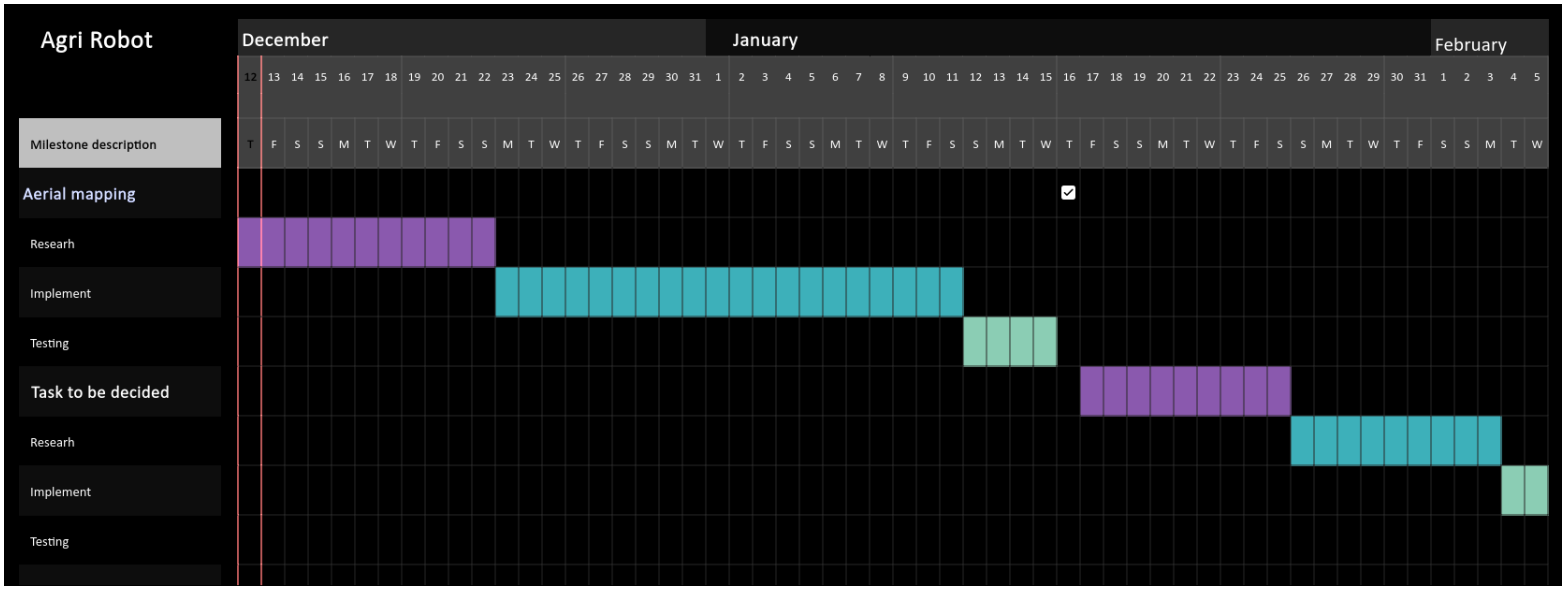

Figure 1: Gantt chart of current plan

Note: I am not sure which feature I will be working on once I have completed the aerial mapping task so i have inserted a placeholder in the Gantt chart

A. Appendix: 2.1 Mapping

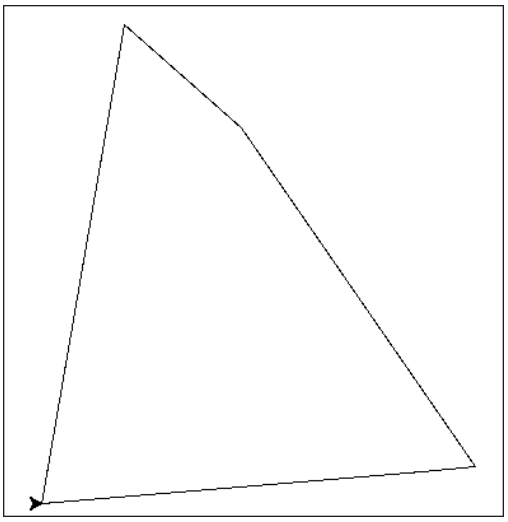

Figure 2: Polar angle method with 4 vertices

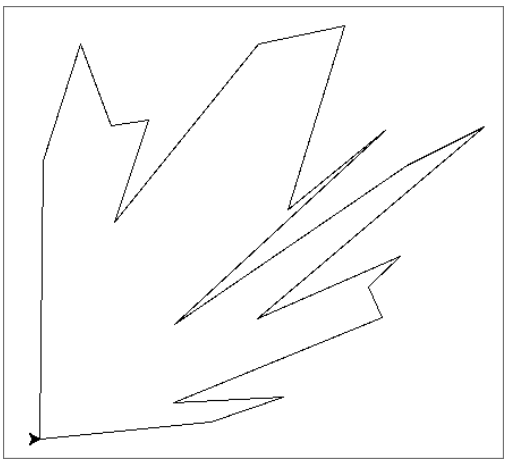

Figure 3: Polar angle method with 20 vertices

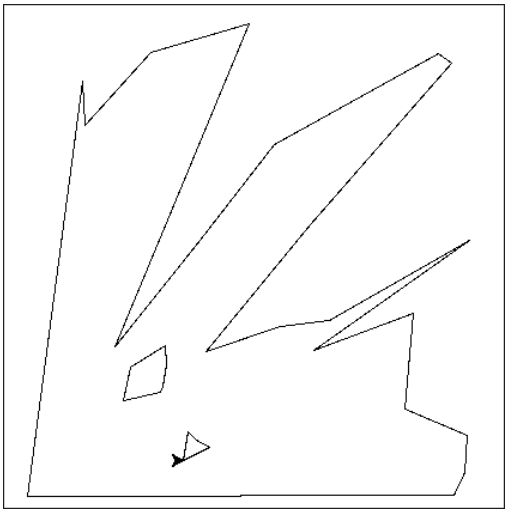

Figure 4: Polar angle method with 2 holes

B. Appendix: 2.1.1 F2C Mapping

Note: Fields2Cover has its own drawing function which is used here

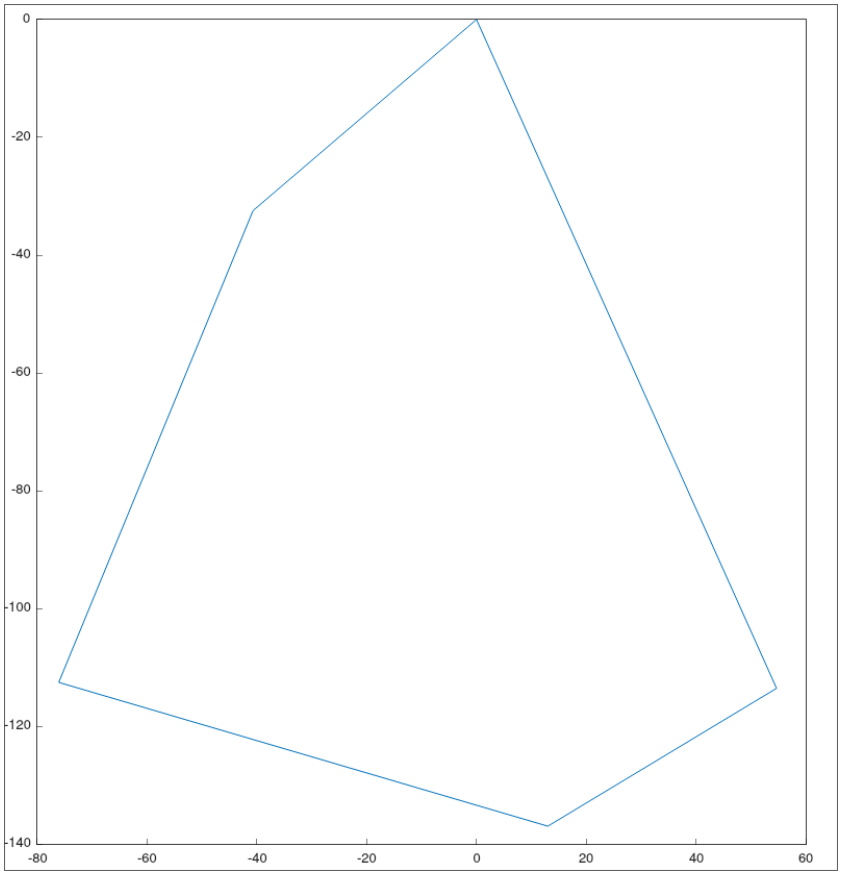

Figure 5: F2C randomly generated field with area 1e5 and 5 vertices

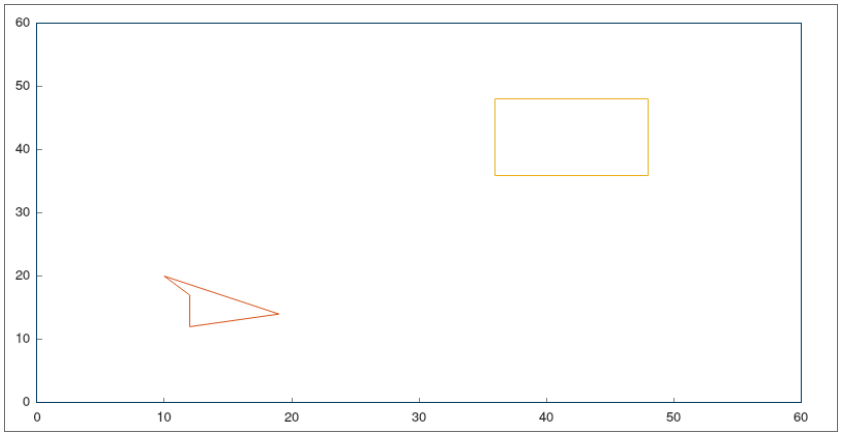

Figure 6: F2C field plotted with 2 holes

C. Appendix: 2.2 F2C Pathing

Note: Fields2Cover has its own drawing function which is used here

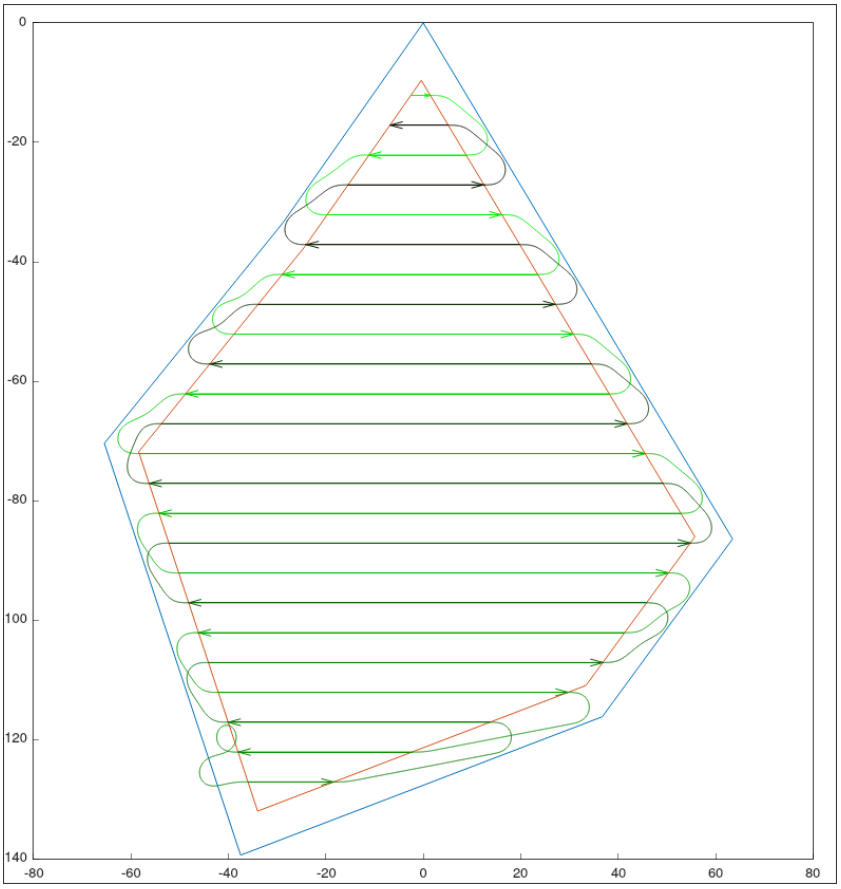

Figure 7: Path generated by F2C on a basic field

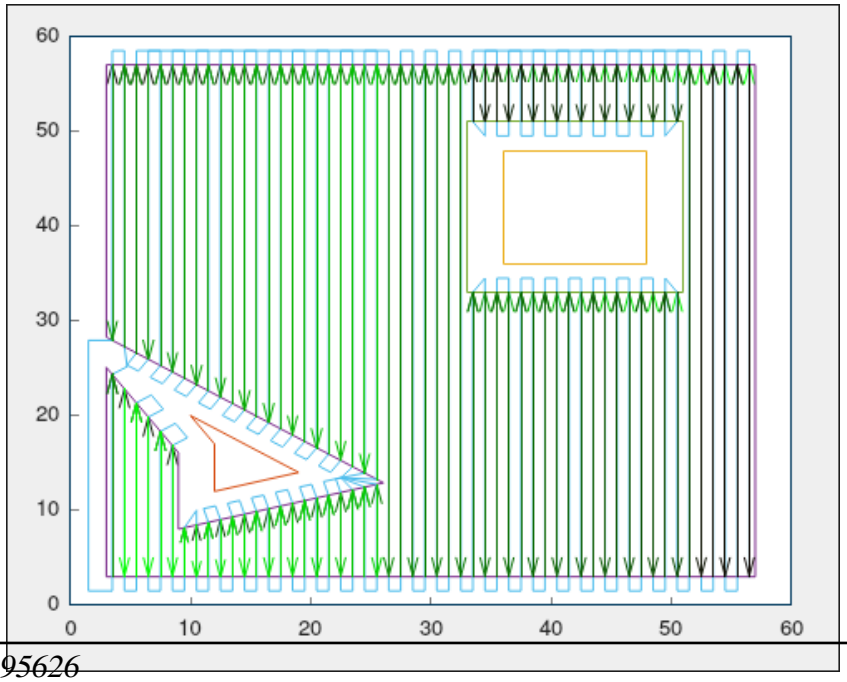

Figure 8: Path generated by F2C on a field with holes

References

Fields2Cover (2023). Github - fields2cover/fields2cover: Robust and efficient coverage paths for autonomous agricultural vehicles. a modular and extensible coverage path planning library. GitHub. Available at: *https://github.com/Fields2Cover/Fields2Cover*.

GeeksforGeeks (2023). Convex hull using graham scan. Available at: *https://www.geeksforgeeks.org/convex-hull-using-graham-scan/*. Accessed: 2023-12-11.