

Toby William Towler

Registration number 100395626

2025

Robot Mower Mapping and Pathing

Supervised by Edwin Ren



University of East Anglia
Faculty of Science
School of Computing Sciences

Abstract

The abstract of your report summarises your entire work () in no more than half a page. It should include the context of your work including its main objective, what methods you employed, how you implemented these, what the outcomes were and a final statement as a conclusion. It should not contain acronyms, abbreviations, elements of a literature review (though a statement of related work is permissible if it is crucial to your work) or future work. The abstract should be written when everything else has been written up and the project is finished! is this workng

Acknowledgements

Edwin Ren, Eden Attleborough

Contents

1. Introduction	4
2. Background	4
3. Methodology	5
3.1. Map Generation	5
3.2. Complete Coverage Path Planning	5
3.2.1. Robot sizing	6
3.2.2. Headland Generation	6
3.2.3. Swath Generation	6
3.2.4. Route Planning	6
3.2.5. Path Planning	6
3.2.6. Cell Decomposition	6
3.3. Aerial Map Generation	6
4. Implementation and Evaluation	6
5. Conclusion and Future Work	7
References	8
A. Map Generation	9

1. Introduction

The robot mower is an already existing project developed by previous masters students from the University of East Anglia. Physically, the mower has 2 tracks for movement on the sides of a metal frame, it is controlled by raspberry pi 4 running the Robot Operating System (Macenski et al., 2022). Sensor wise, the robot is equipped with a 4G dongle, lidar and a GPS chip that was upgraded to an RTK chip in this iteration of the project. The existing code base was mostly written in python with very small amounts of C++. Because of this, all of my code will be written in python to slot into existing code without issue.

My contribution to this project this year will be, regarding the overall movement and guidance of the robot. For ease of planning, I have broken this down into 3 sections:

1. Basic map generation
2. complete coverage path planning
3. map generation from an aerial image

These sections are all modular meaning they can be developed, tested and function independently but still easily be integrated together for the final product. The specified use case of the robot will now be to cut golf courses, this is particularly relevant to the aerial map generation section of my work which will likely use a machine learning model and require relevant training data, while the other sections are not concerned with a real world use case as their algorithms will work be able to tweaked for any applicable use case of this robot.

2. Background

Another section that is essential and should keep its title as is! Although you could perhaps call it “Literature Review” instead, this is not advisable as at this stage of your project we do not expect an extensive literature review since this was already done in the second formative assignment. The rationale is simply because you will lose valuable pages that could be used better in the next two sections that will cover the preparation and implementation of actual work done. So just provide the context in which your project operates here, and then provide a brief overview of similar work that is directly

relevant to yours. Try to avoid blatant copying and pasting from the formative literature review as it is bound to read awkwardly.

3. Methodology

3.1. Map Generation

For the basic map generation, a simple shape generation algorithm should suffice. The algorithm should also take two parameters, K then number of angles in the desired shape and N, the number of "obstacles" in the field.

"Obstacles" can be thought to represent real world obstructions for example trees or telephone poles in a real field. As the outputs of this section will mostly be used for testing the path planning algorithm on regions with differing area, number of corners and complexity, no excessive algorithmic complexity is needed. This program should also be able to create n obstacles within the main field, such areas would represent obstructions in the mowers desired path, for example trees or telephone poles in the real world. This means we need a function with 2 parameters:

- K, number of angles in the outer field
- N, number of obstacles within the field, since it would not be sensible to take a parameter for the number of corners for every hole, we can generate them randomly assuming 3-8 corners staying inline with the complexity of the rest of the field without being unreasonably over engineered.

3.2. Complete Coverage Path Planning

Complete coverage path planning(CCPP) is "the task of determining a path that passes over all points of an area or volume of interest while avoiding obstacles" Zhao and Hwang (2023)

3.2.1. Robot sizing

3.2.2. Headland Generation

3.2.3. Swath Generation

3.2.4. Route Planning

3.2.5. Path Planning

3.2.6. Cell Decomposition

3.3. Aerial Map Generation

4. Implementation and Evaluation

Could be a section each for implementation and evaluation if this suits you better or you could use subsections instead. The difference between this section and the previous "Methodology" section is that this one covers "action" or in other words your active contributions to the project. These may include:

- Implementation of programming code: Describe your final code architecture using for example (UML) diagrams and code snippets. Make sure that code snippet (figure) captions are self-explanatory which means that you should not have to consult the text body to understand what is shown in the figure. Many code snippets of the same kind should end up in an appendix instead.
- Results from experiments run, including testing (user and software). Use figures and tables with self-explanatory captions (see earlier statement). Multiple figures and tables that cover several pages should be put in an appendix.
- Analysis of results: Discuss your experimental and/or test findings in depth. Compare them against other studies and/or benchmarks, and point out limitations of your work (that could be due to limited time) and elaborate on scope for improvement.

5. Conclusion and Future Work

Another essential section that should keep its title as suggested. Briefly discuss your main findings, outcomes, results; what worked and what could have been done differently. Then summarise your work in a concluding statement by comparing your outcomes against the main and sub-objectives and/or MoSCoW requirements (if used) and suggest potential future work that could be done if more time would be available.

References

- Macenski, S., Foote, T., Gerkey, B., Lalancette, C., and Woodall, W. (2022). Robot operating system 2: Design, architecture, and uses in the wild. *Science Robotics*, 7(66):eabm6074.
- Zhao, S. and Hwang, S.-H. (2023). Complete coverage path planning scheme for autonomous navigation ros-based robots. *ICT Express*, 9(3):361–366.

A. Map Generation

Algorithm 1 Point Class Definition

```
1: procedure CLASS POINT
2:    $X \leftarrow -1$                                 ▷ X-coordinate initialized to -1
3:    $Y \leftarrow -1$                                 ▷ Y-coordinate initialized to -1
4:    $angle \leftarrow -10$                             ▷ Angle initialized to -10
5:   procedure CONSTRUCTOR( $x, y$ )
6:      $this.X \leftarrow x$ 
7:      $this.Y \leftarrow y$ 
8:      $this.angle \leftarrow -10$                     ▷ Default angle value
9:   end procedure
10: end procedure
```

Algorithm 2 Generate random points

```
1: function GENPOINTS( $num, P, size$ )
2:    $points \leftarrow []$ 
3:   for  $i \leftarrow 0$  to  $num - 1$  do
4:      $randX \leftarrow \text{random\_integer}(P.X + 1, P.X + size)$ 
5:      $randY \leftarrow \text{random\_integer}(P.Y + 1, P.Y + size)$ 
6:      $points.append(\text{Point}(randX, randY))$ 
7:   end for
8:   return  $points$ 
9: end function
```

Algorithm 3 Sort points by polar angle to origin

```
1: function SORTPOINTS(points, origin)
2:   hull  $\leftarrow$  [origin]
3:   Sort points by Y-coordinate
4:   for i  $\leftarrow$  0 to length(points) – 1 do
5:     points[i].angle  $\leftarrow$  CALCANGLE(origin, points[i])
6:   end for
7:   Sort points by angle
8:   Append points to hull
9:   return hull
10: end function
```

Algorithm 4 Main function

```
1: function MAIN
2:   hull  $\leftarrow$  []
3:   origin  $\leftarrow$  Point(20, 20)
4:   field  $\leftarrow$  GENPOINTS(20, origin, 400)
5:   hull.append(SORTPOINTS(field, origin))
6: end function
```

Algorithm 5 Main function with holes in shape

```
1: procedure MAIN
2:   hull  $\leftarrow$  empty list
3:   origin  $\leftarrow$  Point(20, 20)
4:   field  $\leftarrow$  GenPoints(20, origin, 400)
5:   add SortPoints(field, origin) to hull
6:   hole1Base  $\leftarrow$  Point(100, 100)
7:   hole1Points  $\leftarrow$  GenPoints(5, hole1Base, 50)
8:   add SortPoints(hole1Points, hole1Base) to hull
9:   hole2Base  $\leftarrow$  Point(150, 50)
10:  hole2Points  $\leftarrow$  GenPoints(3, hole2Base, 30)
11:  add SortPoints(hole2Points, hole2Base) to hull
12: end procedure
```
