
# Software Engineering Work Term Report

---

## Choosing an Optimal Path Planning Method for an Autonomous Lawn Mower Prototype

---

Author: Samuel Noguchi

Relevant Courses:
MTE 140: Data Structures and Algorithms

# 1 Background

For my second co-operative work term I was hired as a software designer at the Canadian Tire Innovations Digital Garage. The "garage" as it's called is a research and innovation lab where new technology is designed and built with the hope that it might one day reach the shelves of Canadian Tire stores country wide. Essentially, everything made at the garage is a prototype. We were not writing production ready code, nor were we making any final design decisions for the company. What we were doing was exploring new ideas and experimenting with implementation.

## 1.1 An Autonomous Lawn Mower?

The idea of a lawn mower that operates on its own is not a new idea. Companies like Husqvarna are dedicated to making these machines, which operate like one might imagine: place the small, industrial-looking rover on your front lawn and press go. The rover then proceeds to drive around your lawn cutting all of your grass without the need for any human interference or supervision. Husqvarna has many autonomous lawn mowers currently on the market, offering features like full weatherproofing, silent operation and optimal cut results. The problem with the current implementation is the need to install a wire perimeter around your lawn, which in many cases can involve an installation crew digging up parts of your lawn and garden. Clearly, this is a pain point for many customers, and a solution to this problem could be highly valuable to Canadian Tire.

## 1.2 Removing the Need for a Perimeter Wire

At the garage, the previous co-op student had begun working on an implementation of a rover that could traverse an area of land without the the need for any perimeter wiring. Using a newly-accessible technology called Real Time Kinematic positioning, or RTK positioning for short, the co-op had been able to receive GPS positioning data on the rover with centimeter-level accuracy. This high level of accuracy meant that if the rover knew the bounds of the lawn, it could move within the lawn knowing when it was approaching the perimeter. This implementation essentially removes the need for a perimeter wire by using high precision GPS positioning instead. The last piece of this project was to implement a path planning algorithm that could be embedded on the rover.

## 1.3 Path Planning Constraints

The path planning algorithm is required to generate a path such that the entire lawn area is eventually covered. The input to the algorithm will be a single shape representing the lawn to be mowed, given as a series of latitudinal and longitudinal coordinates. Additionally, multiple "obstacles" within the lawn may also be provided in the same manner. These obstacles represent objects to avoid while mowing such as trees or gardens. The algorithm

should output a series of points forming the path in which the rover should travel.

Optimally, the path created should form an aesthetically-pleasing route, as the mowed grass will leave a pattern depending on how it's cut. An aesthetically-pleasing cut may be straight lines, spirals or some other deliberate pattern. The algorithm also needs to be reliable and adaptable since any lawn shape could be given as input, with any set of obstacles within. Whatever input shapes are given, the algorithm must be able to output a path. Finally, the path created by the algorithm would optimally be as short as possible. Taking a shorter path to traverse the entire lawn would mean shorter mow time.

Constraints regarding the operation of the program implementing the algorithm were also considered. The program should run without failure on a Raspberry Pi 3B+, which has a 1.4 GHz processor with 1 GB of RAM. The computation of the path should be done in a reasonable (less than one minute) amount of time.

## 2 Semi-Random Traversal

A common use for floor coverage algorithms can be seen in autonomous vacuum cleaners like those made by iRobot for their Roomba products. When researching the approach taken by this manufacturer, I found that the path taken by the rover was semi-random. Floor Coverage usually began with a spiral like motion, then ended with many straight zig-zagging paths. According to Deconstructing Product Design, "Roombas rely on a few simple algorithms, such as spiral cleaning (spiraling), room crossing, wall-following and random walk angle-changing after bumping into an object or wall" [1]. Seen below is long exposure shot of a floor being traversed by a roomba with mounted LED's demonstrating the path taken by these cleaners.

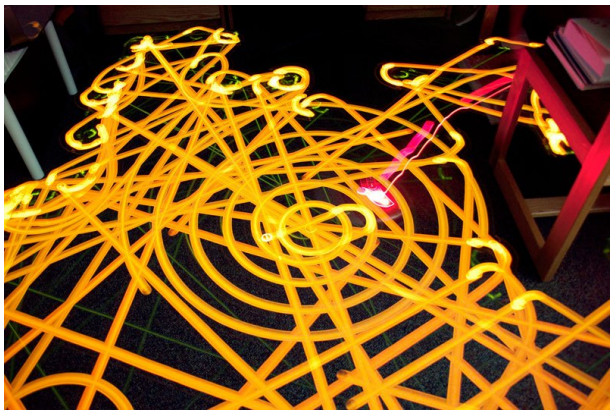


**Figure 1:** Roomba Path (Source: [2])

This approach has some clear advantages, such as ease of implementation, adaptability and low computational requirements. Writing a few commands instructing the rover to drive straight, turn to a random angle and follow a wall would not be difficult. These commands could be called in real time depending on environmental events like reaching a border. Also, since nothing is pre-planned, the approach can be considered highly adaptable. The rover will call movement instructions depending on the environment and can even adapt to unexpected changes in the environment since decisions are computed in real time. A final advantage of real time traversal is the elimination of preliminary path computation, and a low number of computationally heavy calculations.

Despite these advantages, this approach has many obvious downfalls, making it an unfit solution for an autonomous lawn mowing application. Many of the design constraints are not fulfilled, such as forming an aesthetically-pleasing route and minimizing the path length taken to traverse the entire area. The path taken using this approach would appear very random, which would not look pleasing on a mowed lawn. Finally, the path taken by the rover could cover the same area multiple times and some areas may not be covered at all! Given that mowing the entire lawn is an extremely important constraint in this problem, semi-random traversal was discarded as a feasible solution.

# 3 Boustrophedon Motion

While researching common floor coverage algorithms in robotics, the term boustrophedon motion kept appearing. According to Choset in "Coverage Path Planning: The Boustrophedon Cellular Decomposition", the word "boustrophedon" has roots in ancient Greece, meaning "the way of the ox". Later, Choset describes boustrophedon motion:

> "Typically, when an ox drags a plow in a field, it crosses the full length of the field in a straight line, turns around, and then traces a new straight line path adjacent to the previous one. By repeating this procedure, the ox is guaranteed to cover (and thereby to plow) the entire field" [3]

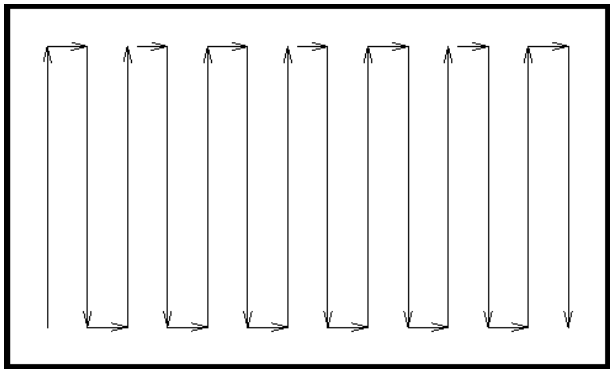
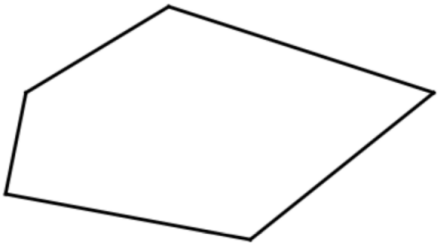


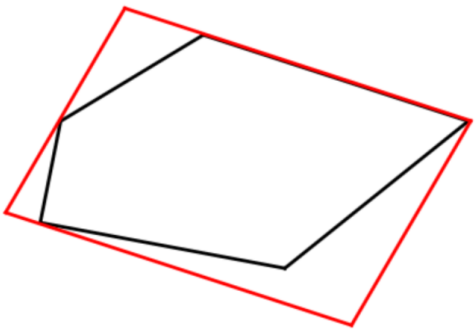
**Figure 2:** Boustrophedon motion (Source: [3])

This motion is commonly used in robotics for floor coverage as it is intuitive and relatively easy to implement. This method of floor coverage has many benefits over random or semi-random traversal. Importantly, the back and forth movement would generate an aesthetically pleasing cut pattern in the grass. Additionally, the path taken while traversing simple shapes such as a rectangle would be of minimal distance, since no spot is visited twice. Implementing this motion programmatically is very easy for convex shapes. The algorithm for doing so is outlined below.

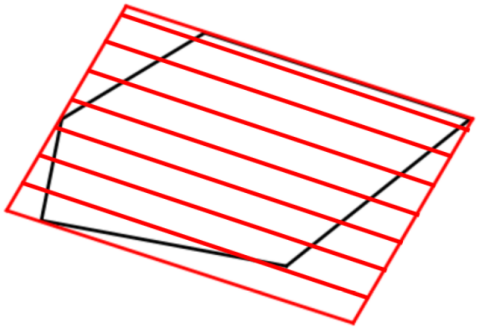
1. Generate a convex shape that represents the shape of the lawn to be traversed. It is important that the lawn is convex so that a line drawn across it can only ever intersect with the shape twice. This is a basic property of all convex shapes.



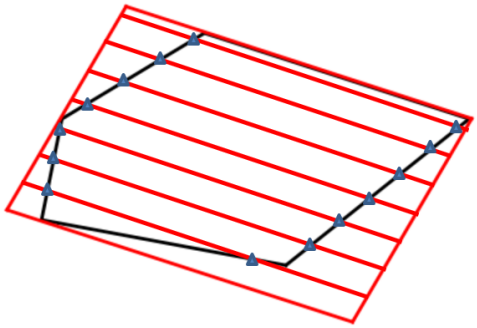
2. Now create a minimum area bounding box around the shape. In computational geometry, a minimum area bounding box is the smallest rectangle enclosing a set of points. In this case, the points are the vertices of the shape. An important property of all minimum area bounding boxes encompassing a shape is that at least one side of the box will be coincident with the shape. This means that mowing lanes generated along the side of this box will appear aligned with the shape.



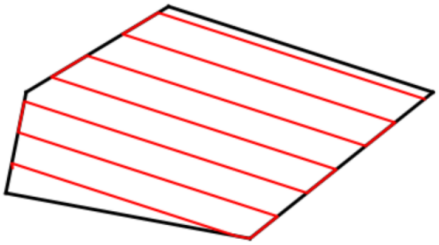
3. Using the longest side of the bounding box as reference, generate a series of parallel line segments, each offset by a set distance. In this example, the bottom most side of the box was used as reference and subsequent line segments were generated upward. The offset of each line is determined by the width of the lawn mowing rover to be used.



4. For each line segment that was generated, compute the intersect points between the line and the shape. In the figure below these intersections are indicated with small triangles. These points will act as way points for the rover.



5. Finally, generate the path using these intersection points. This is achieved by iterating through the generated line segments and adding the intersection points to a list representing the path. The order in which each pair of intersection points is added alternates in order to achieve the back and forth boustrophedon motion. Additionally, if shape vertices exist between two line segments, logic must be implemented to correctly add these vertices to the path if necessary.



Clearly, this approach generates an aesthetic, distance optimal route for traversing convex shapes; however, this approach fails for most concave shapes. This algorithm relies heavily on the fact that each line intersects the shape only twice, giving each mow lane a definite start and end point. Lines drawn across concave shapes could intersect multiple times, breaking this type of algorithm. Since many lawns form concave shapes, this method of traversal is not optimal. Although this approach fits many of the listed constraints, it fails to be adaptable and is therefore a poor candidate for an autonomous lawn mower.

# 4 Boustrophedon Cellular Decomposition

A common method for traversing more complex environments, including obstacles, can be achieved by using boustrophedon cellular decomposition. This method breaks shapes down into "cells" which are guaranteed to be traversable by boustrophedon motion. As illustrated in the diagram below, the trapezoidal environment is decomposed into multiple cells, with the dark grey shapes representing obstacles. An important characteristic of boustrophedon cellular decomposition is that all cell walls generated are parallel to one another.

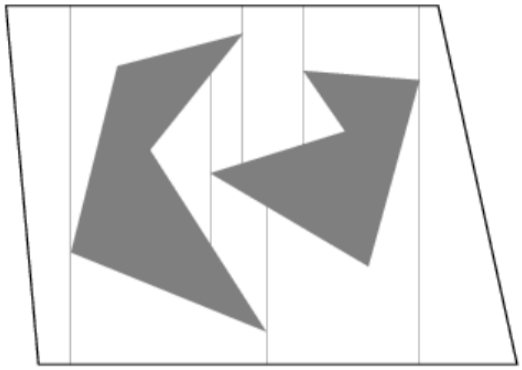


**Figure 3:** Boustrophedon cellular decomposition (Source: [4])

Although these cells are not all convex, they are all traversable by boustrophedon motion if the mowing lanes are oriented parallel to the generated cell walls. An algorithm to implement this decomposition is outlined in Coverage Path Planning: The Boustrophedon Cellular Decomposition [3]. This algorithm involves sweeping a line across the environment. If the connectivity of the line decreases, an obstacle has been encountered. The current cell is closed and two new cells are created, one above the obstacle and one below. If connectivity of the line increases, an obstacle has ended, and the two current cells are closed and one new cell is created. Once the environment is decomposed into these cells, each cell is mowed individually using the method discussed in the previous section.

This method has many advantages, such as aesthetics and built-in obstacle avoidance. Firstly, since all cells are created and mowed parallel to one another, the end path forms a very uniform and aesthetic mow pattern. Additionally, since cells are built around obstacles, there is no need to add any logic to avoid obstacles.

This algorithm was designed for environments with many complex obstacles. The algorithm decomposes the environment into cells based on the obstacles present. Unfortunately, the algorithm does not decompose based on the perimeter itself. Complex concave lawn perimeters would need additional decomposition to ensure all cells are traversable by boustrophedon motion. Another issue with this method is that once the cells have been created, an algorithm is needed to generate an efficient path between cells. Below is a diagram of a vertically-decomposed environment with connections drawn between adjacent cells.

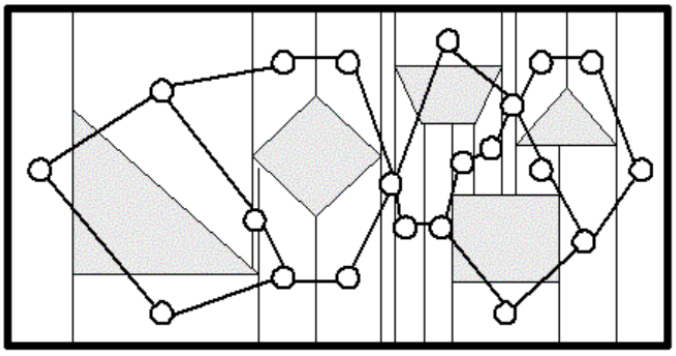


**Figure 4:** Vertical cell decomposition (Source: [5])

Typically, once shapes are decomposed for path planning, a graph data structure is generated to represent the environment, with cells being vertices and cell adjacencies representing edges. For the diagram above, the environment has been removed to reveal the underlying graph structure, pictured in the next figure.

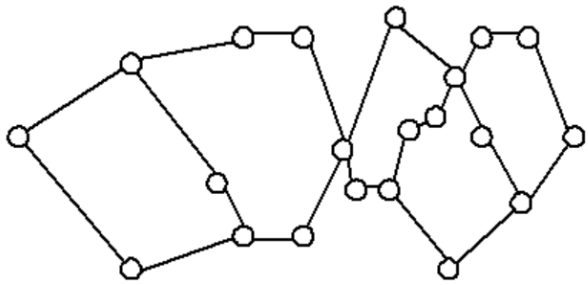
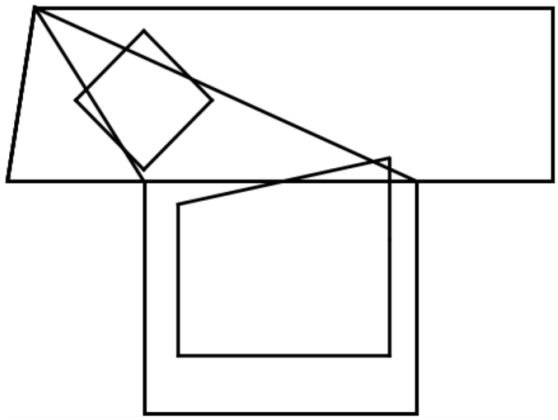


**Figure 5:** Graph data structure (Source: [5])

The resulting data structure is clearly a cyclical graph. For the application of an autonomous lawn mower, the rover should traverse the vertices in such a way that all vertices are visited, the resulting path is as short as possible and the rover returns to the start point after traversal. This is a classic problem known more commonly as “The travelling salesman problem”. This problem is classified as NP-complete, with the computational solution time growing exponentially with the number of vertices in worst case scenarios. Although this method has many clear advantages, the complex computations needed to generate an optimal path are a major downfall.

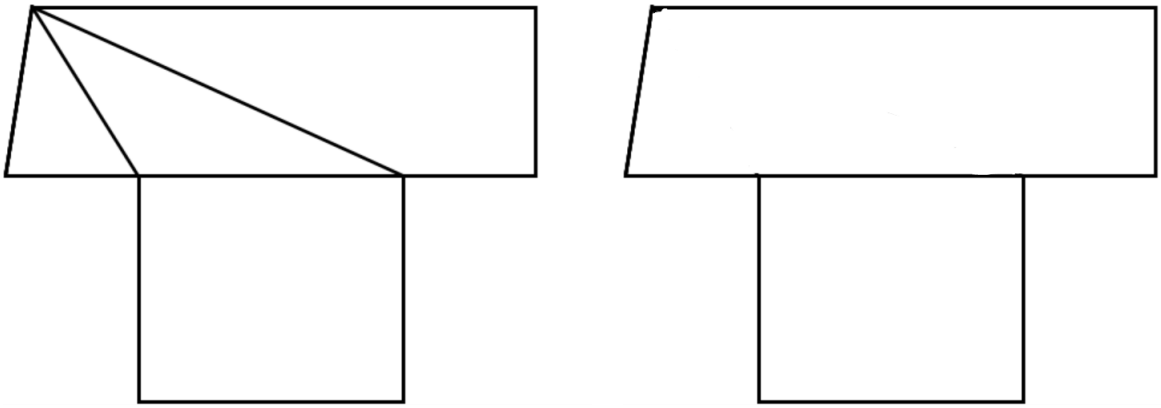
# 5 Convex Polygon Decomposition

Using convex polygon decomposition would be similar to using boustrophedon decomposition. However, instead of decomposing the environment vertically into parallel cells separated by obstacles, the environment would be decomposed into convex polygons, ignoring the presence of any obstacles. The following figure shows the decomposition of a T shaped lawn, clearly not being affected by the obstacles within it.



The primary advantage of using this method over using boustrophedon decomposition is the reduction in complexity for planning cell traversal. Creating an optimal path between cells generated using the boustrophedon decomposition method was an exponential problem. In comparison, the algorithm for planning a path about a graph generated by polygonal decomposition would only be $O(V + E)$ time, where $V$ is the number of vertices in the graph and $E$ the number of edges. This complexity reduction can be attributed to the fact that any graph generated using this method will not be cyclical and can therefore be traversed optimally using a simple searching algorithm such as a depth-first search.

Of course, some trade-offs come with this method of path planning. Since obstacles are not inherently avoided in decomposition, obstacles must be avoided while the rover is traversing within each cell. Implementing obstacle avoidance within these cells is a complex task since they may appear in any shape and an obstacle may span multiple cells. Additionally, the resulting cells of a convex polygon decomposition may not be ideal. Often, shapes are decomposed into too many polygons. Seen below on the left is the resulting decomposition using convex polygon decomposition, when optimally, the decomposition would look like that of the figure on the right.



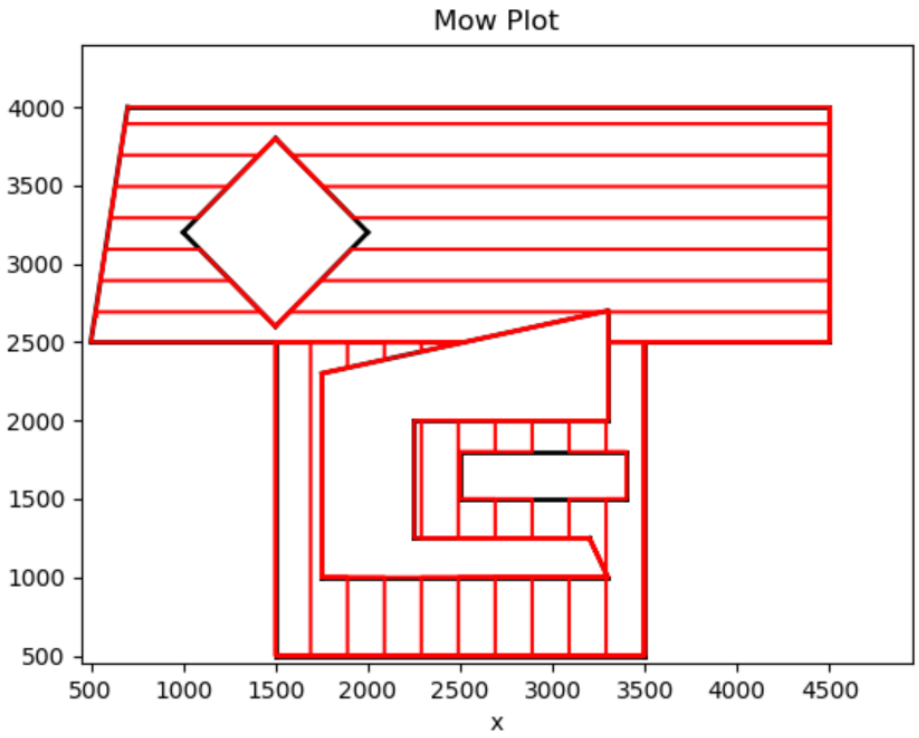
# 6 Selecting a Method

Given the multiple proposed methods of generating a path for an autonomous lawn mowing application, convex polygon decomposition was selected to be implemented on the rover. Each method had individual advantages and disadvantages, however most methods had weaknesses that could not be overcome. Semi-random traversal and simple boustrophedon motion were not adaptable enough, as they could not be applied to all lawn shapes. Boustrophedon cellular decomposition and convex polygon decomposition were both viable options, however, additional decomposition and computations with exponential time solutions were required to produce viable results using the boustrophedon decomposition

method. Even though the use of convex cellular decomposition would require implementing more rigorous obstacle avoidance and cell merging algorithms, the reduced complexity of cell traversal made it a more viable solution.

# 7 Conclusion

When presented with the problem of implementing a path finding algorithm for an autonomous lawn mowing application, several methods were thoroughly examined. By comparing the advantages and disadvantages of 4 different options, an informed and supported engineering design decision could be made. A path finding algorithm based on convex polygon decomposition was built and implemented on the lawn mower prototype, and after several iterations of bug fixes and improvements the final product was complete.




The final software release was able to handle all environments it was tested against, demonstrating successful polygonal decomposition and obstacle avoidance as pictured in the figure above. The process of designing and building this complex piece of software was an immensely rewarding and educational experience as I was required to spend most of my time researching many viable approaches and techniques for robotic pathfinding. In the future, I hope that my work can be successfully implemented in an autonomous lawn mower product for the Canadian Tire brand.

# References


[1] W. Lidwell and G. Manacsa, “Deconstructing product design”, Rockport Publishers, Tech. Rep., 2001.

[2] Zim2411. (2013). This is what happens when you put leds on a roomba.

[3] H. Choset, “Coverage path planning: The boustrophedon cellular decomposition”, Carnegie Mellon University, Report, 1997.

[4] S. M. LaValle. (2006). Planning algorithms: Defining the roadmap.

[5] H. Choset, “General robotics, lecture 9”, Carnegie Mellon University, Lecture, 2005.