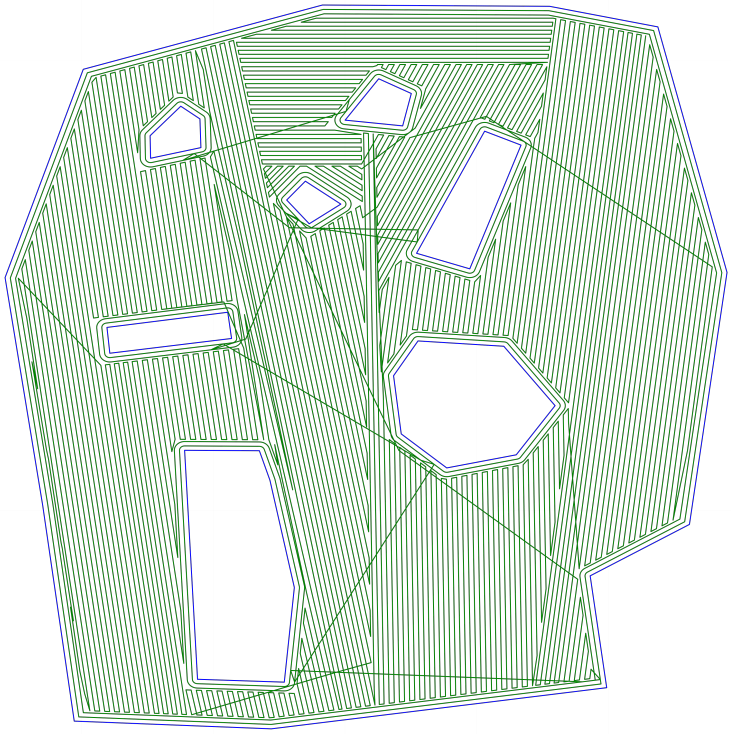


Coverage path planning for autonomous lawnmowers using exact cellular decomposition

Master's thesis in Systems, Control, and Mechatronics

LOGI SIGURDARSON

DEPARTMENT OF ELECTRICAL ENGINEERING

CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden 2023
www.chalmers.se

Master's thesis 2023

Coverage path planning for autonomous lawnmowers using exact cellular decomposition

Logi Sigurdarson

CHALMERS
UNIVERSITY OF TECHNOLOGY

Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
Gothenburg, Sweden 2023

Coverage path planning for autonomous lawnmowers using exact cellular decomposition
LOGI SIGURDARSON

Supervisor: Henric Cronholm, Husqvarna Group AB
Examiner: prof. Martin Fabian, Department of Electrical Engineering

Master's Thesis 2023
Department of Electrical Engineering
Division of Systems and Control
Chalmers University of Technology
SE-412 96 Gothenburg
Telephone +46 31 772 1000

Cover: An example map planned with the optimized BCD algorithm

Typeset in LaTeX
Gothenburg, Sweden 2023

Coverage path planning for autonomous lawnmowers using exact cellular decomposition
LOGI SIGURDARSON
Department of Electrical Engineering
Chalmers University of Technology

Abstract

The rise of autonomous lawnmowers equipped with real-time kinematic (RTK) GPS has given access to accurate absolute localization during operation. Accurate localization has made it possible for mowers to follow predefined systematic paths and avoid relying on inefficient pseudo-random patterns. In this thesis, coverage path planning (CPP) algorithms using exact cell decomposition applicable to RTK-GPS enabled autonomous lawnmowers are investigated. Specifically Oksanens split and merge algorithm is implemented and evaluated with the objective to minimize the coverage time for complex maps, together with a short study of the constriction decomposition method applied to complex maps. The split and merge cell decomposition algorithm is implemented using a simple objective based on the cell geometry along with a complex objective using cutting time estimations. For the connecting transport paths between cells, a simple sub-optimal greedy algorithm is used. The evaluations are based on an estimated time consumption along with the simulated time. The simulations are performed in a hybrid system simulation environment for a differential drive lawnmower using a path following pure pursuit controller adapted to follow the path with the mower cutting deck. The results showcase the cell decompositions, planning and cutting times for four different maps for the adapted split and merge algorithm. The thesis finds that the adapted split and merge algorithm cannot guarantee a faster coverage pattern compared to the baseline given the tested objective functions, although it can be preferable in certain cases.

Keywords: Coverage Path Planning, Boustrophedron Cell Decomposition, Exact Cell Decomposition, Pure Pursuit Controller, Autonomous Mobile Robot, Differential Drive Robot.

Acknowledgements

First, I would like to thank Elias Josefsson at Husqvarna to give me the opportunity to write this thesis, who together with my supervisor Henric Cronholm have helped me with valueable knowledge and feedback. Also a great thanks to all Husqvarna employees i have met along the way, who have all shown excitement and curiosity during my work and who i look forward to call my co-workers.
A special thanks goes to my academic supervisor and examiner Martin Fabian, who has been a great support and helped keep my work on track and giving valuable input during my weekly ramblings. I am also grateful of the amount of time you have spent reading and helping adjust the final report with your meticulous attention to detail.
And last but not least, thanks to my wonderful wife Malin, who has shown nothing but never ending support and understanding during my six years of studying.

Logi Sigurdarson, Gothenburg, Juli 2023

Contents

List of Figures

List of Tables

1
Introduction

Until recently, most autonomous lawnmowers have been relying on pseudo-random coverage patterns in conjunction with physical boundary wires dug into the ground to cover the area to be mowed. A recent improvement in terms of localization is RTK-GPS [1] (Real-Time Kinematic GPS), which is available in high end autonomous lawn mowers [2]. The RTK-GPS gives the possibility for centimeter scale localization precision and has made it feasible to follow a predetermined systematic path. Systematic paths can greatly increase the efficiency compared to random traversal of the area, and double the area capacity of a mower [3].

Current commercial implementations of systematic cutting in autonomous lawnmowers rely on a back and forth pattern with evenly spaced straight parallel lines. These patterns can be sub optimal in respect to cutting efficiency when handling complex maps with tight passages perpendicular to the cutting lines. If the path would be adapted to the topography of the map instead, potential gains in respect to cutting efficiency might be achieved.

This thesis investigates and implements systematic paths for autonomous lawnmowers while taking the constraints and dynamics of the lawnmowers into account.

1.1 Background

The problem of planning a path that covers an area goes under the topic of *coverage path planning* (CPP) [4]. Coverage path planning is variation of the traveling salesman problem, and has been shown to be a NP-hard if an optimal solution is to be guaranteed [5]. The hard nature of the problem leads to using approximations, assumptions, and simplifications. Coverage path planning is a vast research topic used for robotic mowers, robotic vacuum cleaners, unmanned aerial vehicles, and autonomous underwater vehicles [4], [6].

In the literature there exist several interesting CPP algorithms applicable to autonomous mowers [4]. A common approach is the divide and conquer strategy which decomposes the map into smaller parts using *exact cellular decomposition*, these cells are then planned and connected separately.

A common coverage path planning algorithm is the *boustrophedon cellular decomposition* (BCD) algorithm [7]. The word *boustrophedon* originates from ancient greek and translates to “the way of the ox”, referring to the back and forth parallel pattern occurring when plowing a field. A predecessor of the boustrophedon cellular

decomposition is the trapezoidal cellular decomposition which follows a similar procedure but generates an unnecessary amount of cells and possibly leading to longer covering times [8]. Coverage path planning can further be solved by using Genetic algorithms, Neural networks, and Reinforcement learning [4], however none of those methods are investigated in this thesis.

Despite of the research on CPP being vast, this thesis contributes by evaluating and implementing the algorithms with constraints and limitations of two different types of autonomous lawnmowers. The algorithms are evaluated in the context of realistic scenarios given complex maps such as gardens with multiple odd shaped obstacles.

1.2 Purpose and objective

The purpose of this thesis is to implement and evaluate suitable offline[1] coverage path planning algorithms taking the mower kinematics and constraints into account, with the objective to minimize the cutting time in complex environments.

1.3 Scope and delimitations

The scope of the thesis is to investigate, test and develop exact cellular decomposition coverage path planning algorithms suitable for small and midsize autonomous lawnmowers with one or three cutting blades. The thesis focuses on offline coverage path planning algorithms using exact cell decomposition given a predefined 2-dimensional map with obstacles.

1.4 Contributions

The contributions of this thesis are:

- An implementation and evaluation of an adaption of the split and merge exact cell decomposition using boustrophedon cell decomposition in the domain of small and midsize automatic lawnmowers.
- A detailed description of a Boustrophedon cell decomposition implementation, taking duplicate events into account.
- An evaluation of the constriction decomposition method exact cell decomposition when applied to a complex map.
- An implementation of a pure pursuit path following controller using an external reference adhering to constraints in angular velocity with the possibility to reverse.

[1] Referring to computations done in advance with full knowledge.

2

Background theory

In the following chapter the background theory used in the coming chapters is presented. The theory covers the use of geometry, mower modeling and dynamics.

2.1 Mower modeling

In this thesis two different mower models are taken into account, a smaller mower with a single disc cutting deck and a bigger mower with a three disc cutting deck; these will be referred to as the single disc and three disc mowers, respectively. Aside from the number of cutting discs, the mowers have similar dynamics and only differ in the parameter values. The mowers have differential drives where the cutting discs are in front of the wheels. A schematic covering both mowers is shown in Figure 2.1.

This section presents the kinematic, dynamics and constraints of the mowers.

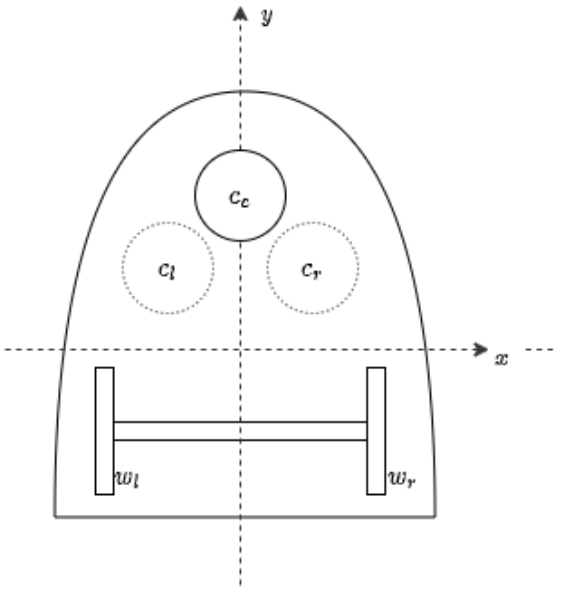


Figure 2.1: Simplified schematics of an autonomous mower with the possibility of multiple cutting discs.

2.1.1 Mower dynamics

The dynamics of the mowers can be modeled by the following ordinary differential equation system for a differential drive robot,

$$\begin{aligned}\dot{x}_m &= \frac{r}{2}[\omega_l + \omega_r]\cos(\theta)\\ \dot{y}_m &= \frac{r}{2}[\omega_l + \omega_r]\sin(\theta)\\ \dot{\theta} &= \frac{r}{L}[\omega_l - \omega_r],\end{aligned}$$

where ω_l and ω_r are the rotational velocities of the left and right wheel, and the coordinates (x_m, y_m, θ) define the mower frame located at the middle of the virtual drive axle with the angle θ perpendicular to the virtual axle in the global frame. The differential drive model can be converted into a unicycle model by using the mowers velocity and angular velocity as inputs

$$v_u = \frac{r}{2}[\omega_l + \omega_r], \quad \dot{\theta}_u = \frac{r}{L}[\omega_r - \omega_l],$$

which gives

$$\begin{aligned}\dot{x} &= v_u\cos(\theta)\\ \dot{y} &= v_u\sin(\theta)\\ \dot{\theta} &= \dot{\theta}_u.\end{aligned}$$

To avoid fast turns and comply with safety regulations a constraint on the angular velocity is to be enforced, for evaluation purposes the following constraint is chosen:

$$|v(t)| \leq \begin{cases} v_s, & \text{if} \quad |\dot{\theta}| \leq \epsilon \ \wedge \ v(t) > 0 \\ v_t\left(1 - \dfrac{|\dot{\theta}(t)|}{\dot{\theta}_{max}}\right), & \text{otherwise.}\end{cases} \tag{2.1}$$

Here v_s and v_t are the maximum speeds when driving straight and turning respectively. ϵ defines the interval for which the mower can be assumed to drive straight and $\dot{\theta}_{max}$ is the maximum angular velocity when turning in place. The allowed velocities are illustrated in Figure 2.2.

2.1.2 Reference point kinematics

Given a point $(x_r,\, y_r)$ in the mower frame the corresponding point $(x_p,\, y_p)$ in global coordinates can be described as

$$\begin{bmatrix}x_p\\ y_p\end{bmatrix} = \begin{bmatrix}x_m\\ y_m\end{bmatrix} + \begin{bmatrix}\cos(\theta) & -\sin(\theta)\\ \sin(\theta) & \cos(\theta)\end{bmatrix}\begin{bmatrix}x_r\\ y_r\end{bmatrix}, \tag{2.2}$$

and when differentiated the dynamics are given by

$$\begin{bmatrix}\dot{x}_p\\ \dot{y}_p\end{bmatrix} = \begin{bmatrix}\dot{x}_m\\ \dot{y}_m\end{bmatrix} + \begin{bmatrix}-\sin(\theta) & -\cos(\theta)\\ \cos(\theta) & -\sin(\theta)\end{bmatrix}\begin{bmatrix}x_r\\ y_r\end{bmatrix}. \tag{2.3}$$

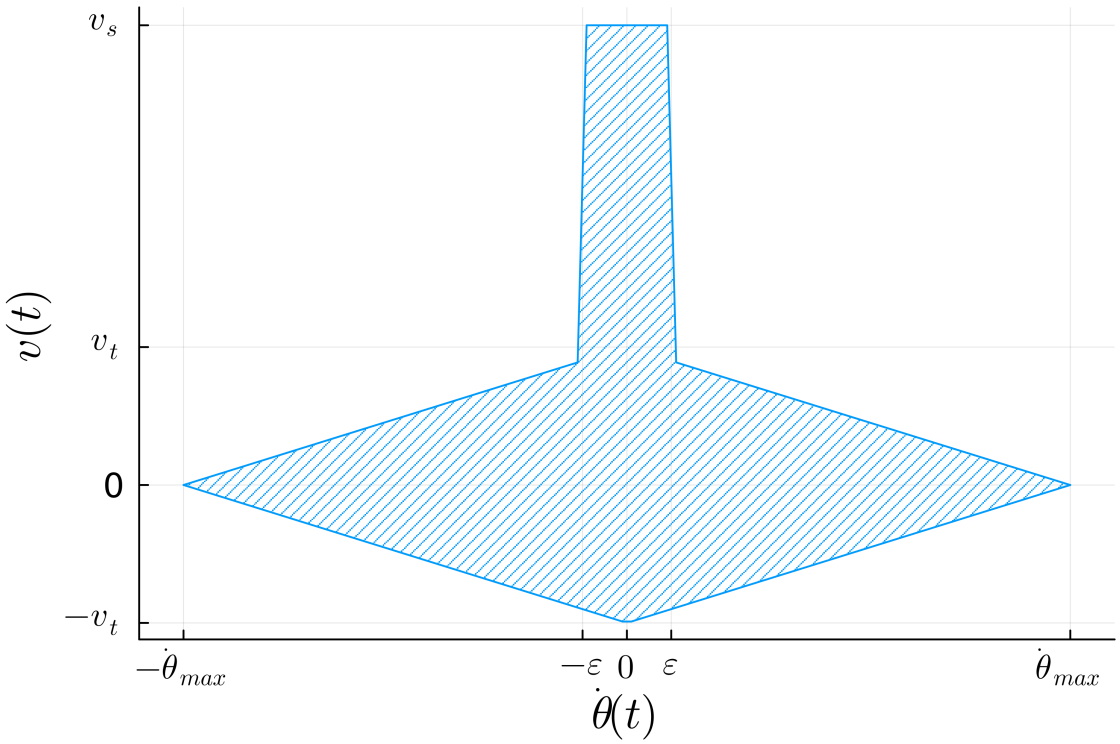


Figure 2.2: Plot showing the allowed area for the combination of the speed and angular velocity

2.1.3 Steady-state turning cutting width of a three disc mower

When conducting a turn with the three disc mower, the cutting width is reduced as the triangular pattern is projected onto the heading direction. Since the cutting blades are displaced in the direction of motion, the definition of the cutting width is not clear during transient changes in heading angle. It is however possible to give a simple definition of cutting width during steady state cornering.

By defining a virtual steering angle β measured in the center of the middle cutting disc as shown in Figure 2.3, the length r_{COR}, which is given from the center of rotation (COR) to the (virtual) center of the driving axle, is given by

$$r_{cor} = l_d \cot \beta. \tag{2.4}$$

The radius of the path for each cutting disc can then be expressed as

$$r_l = \sqrt{(r_{cor} - l_b)^2 + l_s^2}, \qquad r_m = \sqrt{r_{cor}^2 + l_d^2}, \qquad r_r = \sqrt{(r_{cor} + l_b)^2 + l_s^2}. \tag{2.5}$$

where l_b is the lateral distance between the center of an outer disc to the center of the middle disc, and l_s is the longitudinal distance from the mower drive axle to the center of the outer cutting discs. The cutting width l_w as a function of the steering angle β as

$$l_w(\beta) = \begin{cases} 2l_b + 2r_d & \text{if} \quad \beta = 0 \\ \max(r_r, r_m) - r_l + 2r_d & \text{if} \quad \beta \in (0, \frac{\pi}{2}], \end{cases} \tag{2.6}$$

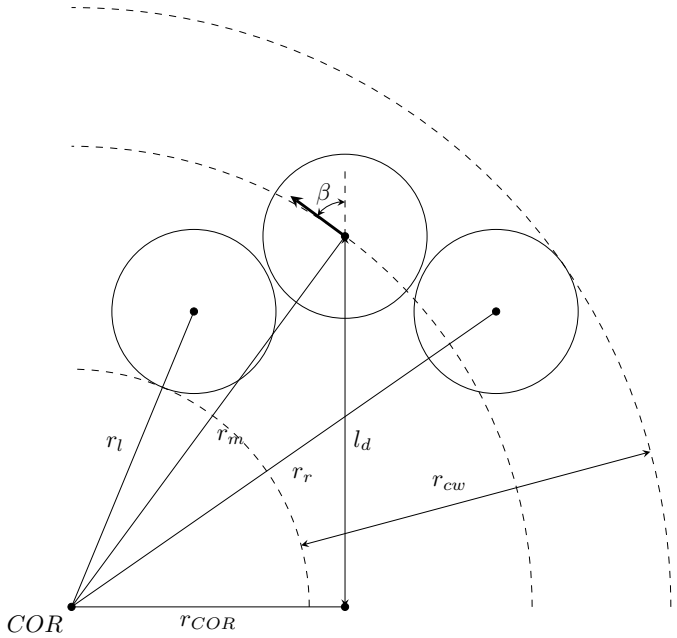


Figure 2.3: A schematic of the three disc mower turning, given a virtual steering angle β, the dashed lines illustrate the coverage.

where r_d is the radius of each cutting disc. It is possible to define a function of the loss of the cutting width with respect to the steering angle

$$f_w(\beta) = 1 - \frac{l_w(\beta)}{l_w(0)}, \tag{2.7}$$

which is plotted in Figure 2.4. When following a path with a reference point it can be important how much coverage is provided on each side of the path. If following an arc shaped path with a radius r_m with the tracking point located at the center of the middle cutting disc, the cutting on the left side l_{wl} and right side l_{wr} can be expressed as

$$l_{wr}(\beta) = r_r - r_m + r_d \tag{2.8}$$
$$l_{wl}(\beta) = r_m - r_l + r_d, \tag{2.9}$$

which are depicted in relation to the full cutting width in Figure 2.5.

These calculations indicate the if a coverage path includes corners, adaptions have to be made to account for the lost cutting width. This can be simplified by using cell decomposition algorithms producing cells that can be covered with boustrophedon paths where turning is performed outside of the planned cells.

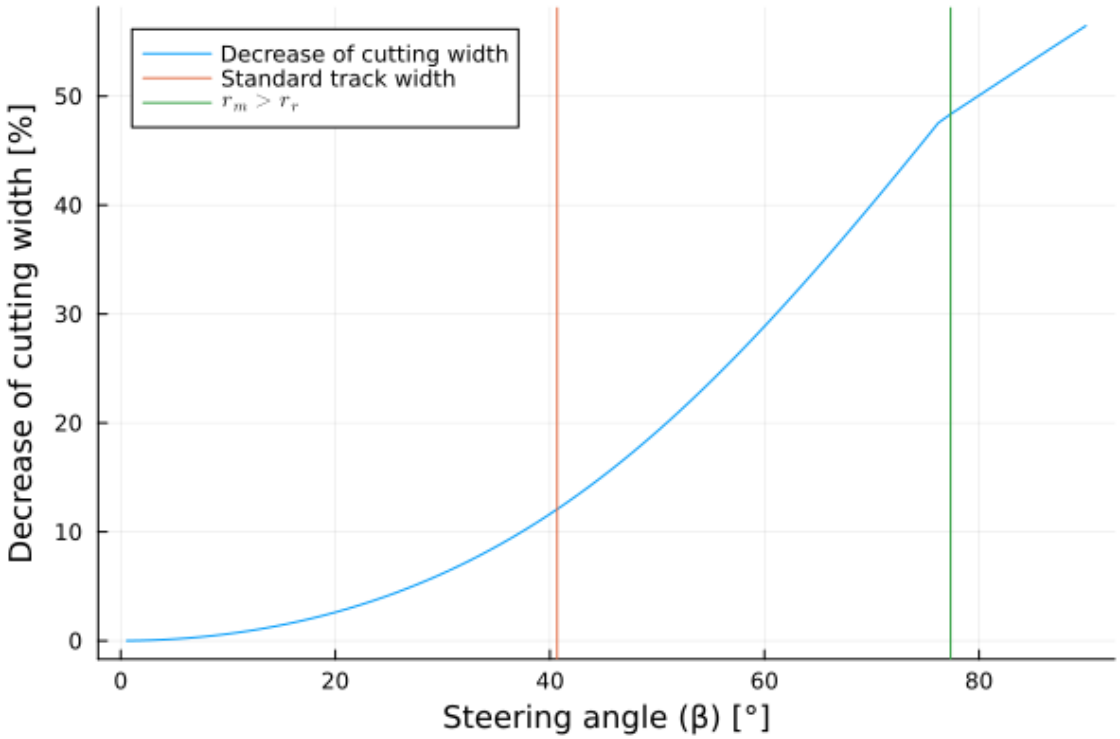


Figure 2.4: Plot showing the decrease of cutting width as a function of steering angle β, where the steering angle is based on the point in the center of the middle disc

2.2 Map geometry

For the purposes of planning the coverage path it is important to define the work area. The work area is the area where the mower should mow the grass while keeping inside the bounds of the area. The work area is defined by an outer boundary together with the boundaries of the inner obstacles such as trees, houses, etc. An example of a work area with three obstacles can be seen in Figure 2.6. In this section a definition of a polygon is given together with the relevant properties that polygons can have when planning the coverage. A short description of lines and line segments is also provided.

2.2.1 Geometric entities

A point is a single coordinate in euclidean space and can represent a vector going from the origin to the coordinate. A line segment is the finite straight line connecting two points (also called vertices). A line string is a collection of connected line segments, and when the start and end point of a line string are connected, and the line string is simple (non-intersecting) it can be called a ring and corresponds to the perimeter of a simple (non-intersecting) polygon.

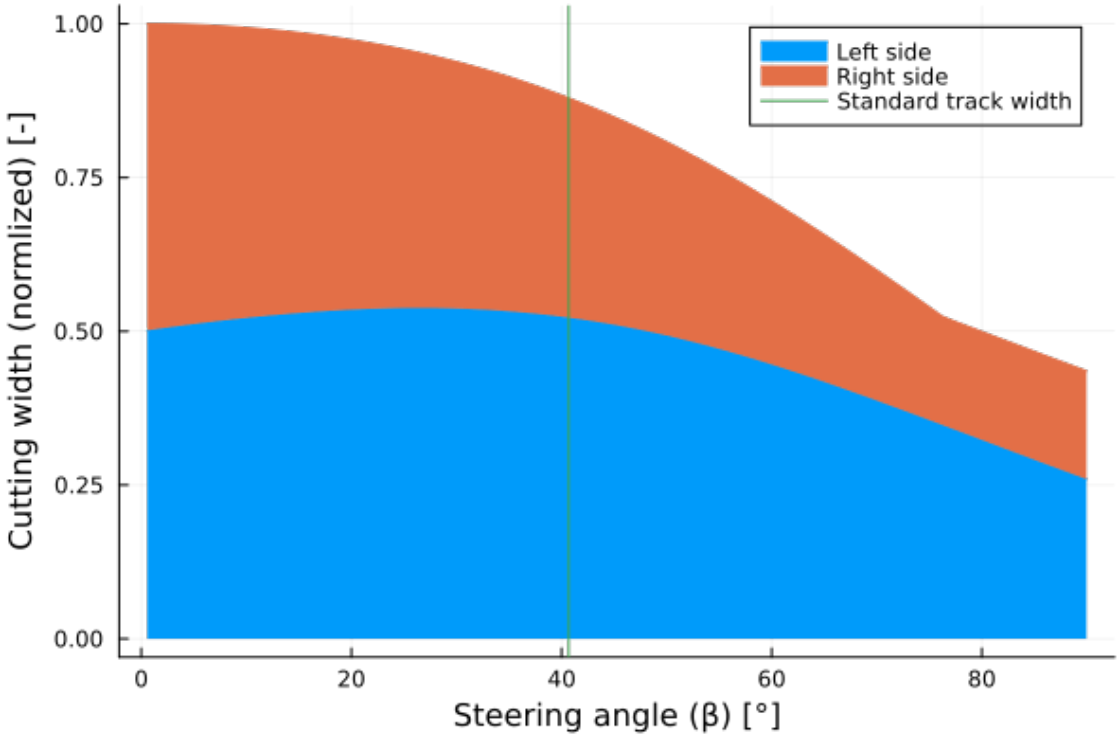


Figure 2.5: A plot visualizing the mower cutting width in relation to the left and the right side of the reference point when performing a steady state turn

2.2.2 Polygon

A polygon is a 2D geometry that is defined by an ordered set of points in 2D-space, connecting each point is a straight line segment representing the boundary of the polygon, examples of polygons are rectangles, triangles, and octagons, which all have the properties simple, monotonic, and convex. Also, polygons can contain holes, to simplify the descriptions a distinction is made between a polygon with or without holes such that when a polygon contains holes it is called a *polygon with holes* (PWH), while a polygon without holes is simply called a *polygon*. A polygon $\mathcal{P}$ is defined by its vertices v as

$$\mathcal{P} = \langle v_0, \ldots, v_n \rangle = \langle \langle x_0, y_0 \rangle, \ldots, \langle x_n, y_n \rangle \rangle \tag{2.10}$$

A PWH is a polygon defined by multiple simple polygons, a simple polygon that defines the outer perimeter and multiple polygons defining the holes. Thus, a PWH is defined as

$$\mathcal{P}_{WH} = \{P_O, P_H^1, \ldots, P_H^N\}, \tag{2.11}$$

where subscript O indicates the outer perimeter polygon and the subscript H indicates a hole.

If a straight line perpendicular to an arbitrary axis direction can be moved along the axis and at most intersect the polygon twice, the polygon is *monotone* with respect to the axis direction. A special case of monotone polygons are convex polygons where any straight line intersects the polygon at most twice.

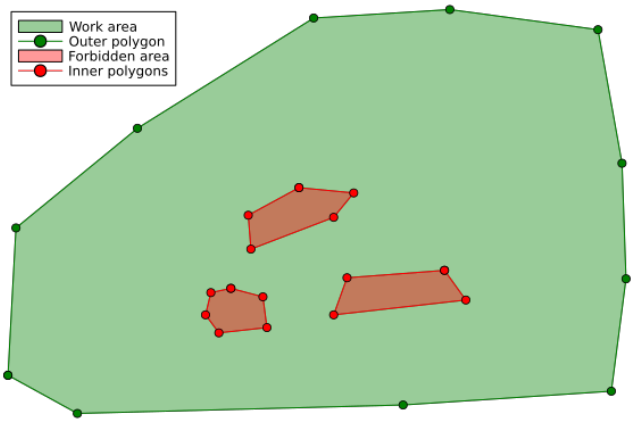


Figure 2.6: An example map showing a work area, forbidden areas, the outer polygon (perimeter) and the inner polygons (holes)

.

For a simple polygon the vertices can be oriented clockwise (CW) or counter-clockwise (CCW). A CW orientation follows the edges CW around the interior of the polygon, and CCW follows the edges in the opposite direction. The orientation can be computed with the signed sum of the polygon

$$A = \frac{1}{2}\sum_{i=0}^{n-1}(x_i y_{i+1} - x_{i+1} y_i), \tag{2.12}$$

where a negative area A indicates a CW orientation and a positive A indicates a CCW direction. A convention is to orient the polygons of a PWH as CW for the outer boundary and CCW for the holes, which can simplify algorithms in terms of handling special cases between the perimeter polygon and the holes.

In this work all polygons are simple, all PWH are made up of simple polygons, and no holes intersect with the boundary of any other polygon.

2.2.3 Padded polygon

When performing path planning in general it is common practice to perform padding of a polygon to avoid planning paths that are too close to the boundary. This can also be referred to as inflating/deflating, offsetting or buffering the polygon. Two applicable padding algorithms to polygons with holes are the use of the Minkowski sum/difference using a circle as implemented in the GEOS computational geometry library [9] or using the offset property from the straight skeleton algorithm [10]. In this thesis the GEOS Minkowski sum/difference implementation is used.

3

Coverage path planning

In this chapter the coverage path planning algorithms to be evaluated and implemented are described. The Boustrophedon cellular decomposition (BCD) algorithm is here described using ordered sets and boolean logic for locating events, together with an algorithm for event pruning. A description of the constriction decomposition method is given, followed by preliminary results. Lastly, a simple greedy route planner is introduced along with a description of alternative routing algorithms.

3.1 Boustrophedon cellular decomposition

The BCD, described in [7] and in more detail in [11], is an exact cell decomposition algorithm generating monotone polygon cells that can be covered by parallel back and forward passes. BCD can be explained by visualizing the map with obstacles in the x-y plane together with the slicing line, which is a straight line parallel to the y-axis and placed to the left of the polygon. At the start of the algorithm the slicing line starts moving in the positive x direction, and builds up cells by following a set of predefined events. When the slicing line encounters an event vertex it either opens new cells or closes cells. An illustration of the movement of the slicing line along with the encountered events can be seen in Figure 3.1.

The events are located on the vertices of the polygons that define the PWH. The event vertices are located by evaluating the corners of the polygons, a corner can be left or a right facing, as well as pointing inwards or outwards in respect to the polygon interior. A left corner is a corner defined as having the x values of both line segments greater than the x value of the corner vertex, while right corners will have smaller x values. These four combinations of left or right and inwards or outwards each map to an event. If a vertex does not satisfy the left or right criteria it does not define an event.

To know which combination of left or right and inwards or outwards corresponds to an event, it is also necessary to know if it is the outer or an inner polygon of PWH as they have different meaning, i.e a right and outwards facing vertex defines a *merge* event for an inner polygon while it defines a *close* event for the outer polygon as seen in Figure 3.1.

However, it is possible to make use of the polygon direction (CW for the outer polygon and CCW for the inner polygons) when locating the events by iterating over the vertices. This makes it possible to define the event localizing algorithm for

an inner or an outer polygon and be able to use it for both without changes, the following section will present the locating of the events.

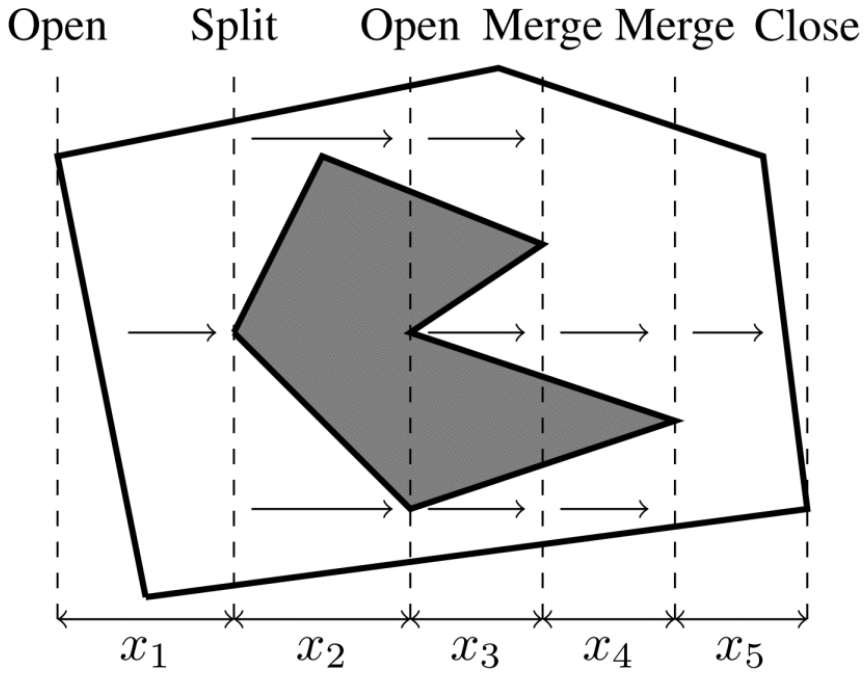


Figure 3.1: Figure showing the events for the boustrophedon cell decomposition. Note that the right *open* event only acts on the upper vertex that coincides with the slicing line. From [12] © 2016 IEEE

3.1.1 A short description of vertices and coordinates

As previously stated, a PWH consist of multiple polygons, each polygon consists of multiple vertices $v_i \in \mathcal{P}$ (with $i \in [1, N]$) where each vertex v_i corresponds to a coordinate vector $[x_i, y_i]^T$. The use of v_i, x_i, y_i will here be used loosely to simplify the description.

Since a polygon is a circular chain of vertices, it is favorable to easily be able to refer to the last and first vertices as the vertex before the first and the vertex after the last, respectively. Thus wrap around indexing is used such that index 0 corresponds to index N and index $N + 1$ corresponds to index 1, i.e. v_{i-1} for $i = 1$ corresponds to the vertex before v_1 which is the last vertex v_N.

3.1.2 Locating events

To identify the events we find the left and right sided event vertices, with the following

$$\mathcal{R} = \langle r_1, \ldots, r_i, \ldots, r_N \rangle \quad \text{where} \quad r_i = (x_{i-1} < x_i) \wedge (x_{i+1} < x_i) \tag{3.1}$$

$$\mathcal{L} = \langle l_1, \ldots, r_i, \ldots, l_N \rangle \quad \text{where} \quad l_i = (x_{i-1} > x_i) \wedge (x_{i+1} > x_i) \tag{3.2}$$

where $\mathcal{R}$ and $\mathcal{L}$ are lists of booleans indicating if a vertex is a left or right facing corner respectively. The list of events is the given by $\mathcal{E} = \langle e_0, \ldots, e_N \rangle$ where

$$e_i = \begin{cases} open & \text{if} \quad l_i \wedge \det\Big((v_i - v_{i-1}), \quad (v_{i+1} - v_i)\Big) < 0 \\ close & \text{if} \quad r_i \wedge \det\Big((v_i - v_{i-1}), \quad (v_{i+1} - v_i)\Big) < 0 \\ merge & \text{if} \quad r_i \wedge \det\Big((v_i - v_{i-1}), \quad (v_{i+1} - v_i)\Big) > 0 \\ split & \text{if} \quad l_i \wedge \det\Big((v_i - v_{i-1}), \quad (v_{i+1} - v_i)\Big) > 0 \\ none & \text{otherwise,} \end{cases} \tag{3.3}$$

where the determinant expression defines a reflex vertex[1] if it is less than zero or a convex vertex if it is above zero. The above expressions can be compared to the events of the inner polygon in Figure 3.1, where the open event has a left facing reflex vertex and the split event has a left facing convex vertex.

3.1.3 Corner cases of locating events

For ease of understanding, a corner case has until now been left out, this is in the case that one or multiple vertical line segments are the source of an event. To handle these events the left and right events can be defined as

$$r_i = ((x_{i-1} < x_i) \wedge (x_{i+1} = x_i)) \vee ((x_{i-1} = x_i) \wedge (x_{i+1} > x_i)) \tag{3.4}$$

$$l_i = ((x_{i-1} = x_i) \wedge (x_{i+1} < x_i)) \vee ((x_{i-1} > x_i) \wedge (x_{i+1} = x_i)) \tag{3.5}$$

$$i \in [1, N]. \tag{3.6}$$

Though this will lead to duplicates of events as both vertices of a vertical line segment will register an event, this can be handled by the event pruning presented in Section 3.1.5.

3.1.4 Converting event types

Instead of using the original event descriptions for BCD, they are converted from *open*, *close*, *split*, and *merge* into the events *open*, *close*, *close below*, and *close above*. In this way a *merge* event becomes both a *close below* and *close above* event, a *split* event becomes a *close* event and the *open* and *close* events stay the same. This makes it easier to handle the edge case where a *merge* event consists of a vertical line segment such that the upper vertex of the line closes the cell above and the lower vertex closes the cell below. The conversion of events can be summarized by the following relations

$$\begin{aligned} open &\rightarrow open \\ close &\rightarrow close \\ split &\rightarrow close \\ merge &\rightarrow close\ above \text{ and } close\ below. \end{aligned}$$

[1] A vertex which internal angle is above 180 degrees, i.e. points inwards towards the inner of the polygon.

3.1.5 Event pruning

A part of the algorithm that is not explained in [7] and [11] is how to handle duplicate events. These duplicates have in this thesis been found to originate from three sources: the event localization defined in Section 3.1.2, the shape of the polygon, and the combinations of the polygons defining the PWH.

The event pruning algorithm is shown in Algorithm 1. For all closing events $e_i \in \mathcal{E}$ gather each unique x-value, then for each x-value sort the list in descending order with respect to the y-value. For each closing event except the last one, check if the line connecting the closing event to the next event in the list intersects with the work area boundary. If it does not intersect, the closing event is a duplicate and should be removed. End the sequence by adding the last event to the list. Here *close above* and *close below* are special cases where only intersections above or below a *close above* and *close below* event, respectively, count as an intersection. Then proceed to add the last closing event to get the list of closing events to be kept.

Algorithm 1 Pruning algorithm

```
E' ← ∅
P_H ← polygon_holes(P_WH)                          ▷ Only polygon holes
X ← unique_x_values(E)
for x in X do
    E_x ← sort_y(E(x))
    for i in 1 to size(E_x) − 1 do
        e ← E_x[i]
        e_+ ← E_x[i + 1]
        if e.event ≠ close upper then              ▷ close above always added
            l ← line_segment(e.v, e_+.v)
            P_I ← P_H \ {e.P, e_+.P}               ▷ Ignore the event polygons
            if not intersects(l, P_I) then         ▷ Prune event
                continue
            end if
        end if
        E'.push(e)
    end for                                        ▷ Always add last event
    E'.push(last(E_x))
end for
return E'
```

3.1.6 Ceilings and floors

As shown in the second paper on BCD [13], the edges for each polygon in the PWH can be split up into floors and ceilings. A floor is a multi-segment line which is directly above the work area while the ceiling is a multi-segment line which has the work area below it. If the polygon is convex, i.e. only contains two events, it can only have one ceiling and a floor, while a non-convex polygon can have multiple floors and ceilings depending on where the reflex vertices are located. The events

list can be used to define the ceilings and floors, as each event marks the change from a ceiling to a floor, or a floor to a ceiling. To attain the floors and ceilings, the polygon ring is cut at each event vertex keeping the orientation of the polygon such that the points in the line follow the same order around the polygon. As previously, the opposite orientation of the inner and outer polygons simplify the implementation as the same criteria work for both types. Given the nature of the event vertices, all multi-segment lines are either strictly increasing or strictly decreasing in the x-direction. For a hole polygon, a ceiling is a multi-segment line where all points are strictly decreasing in x-direction and a floor is is a multi-segment line where all the points are strictly increasing in x-direction. When using the ceilings and floors in the cell decomposition a ceiling or a floor can be fully included in one cell or partially included in multiple, but a cell can only consist of one (partial) ceiling and one (partial) floor.

3.1.7 The cell decomposition

With the lists of events, ceilings, and floors for each polygon it is possible to generate the cells. First all events for each polygon are merged into one list sorted in ascending order with respect to the x-coordinate. In the same way the ceilings, and floors from each polygon are merged into their respective list. If an *open* and *close* event are located at the same x-value the *close* event is prioritized, such that the ceilings and floors added by the *open* event do not interfere with the closing of the cells when finding the closest ceiling and floor.

Further two empty lists that will contain the active floors and ceilings are initialized. An active floor or ceiling are the lines which are relevant for the current placement of the slicing line, as the slicing line progresses in the x direction. Adding to the active lists is the *open* events that add the floors and ceilings that start at the x-coordinate for the *open* event. When the slicing line encounters a *close* event a cell i closed, first the closest ceiling above the vertex and closest floor below the vertex are found and removed from the active lists. The removed ceiling and floor lines are then cut by the vertical slicing line intersecting the closing vertex, the part of the lines that are to the left of the slicing line are connected at the ends with vertical lines and form the new cell. The lines to the right of the slicing line are added back to the lists of active floors and ceilings.

The *close above* and *close below* events are handled in a similar way to the *close* event. Such that the *close above* is constrained to use the floor line that coincides with the *close above* vertex and the *close below* is constrained to use the ceiling line that coincides with the *close below* vertex.

3.2 Optimized BCD

In the following section a variant of the split and merge algorithm of [14] is proposed. The algorithm described here differs in that the cell decomposition described in Section 3.1 above is used instead of a trapezoidal cell decomposition [15] where the cells are merged based on a criteria.

The algorithm starts by rotating the map with evenly spaced angles from 0 to 180 degrees, and for each rotation the cell decomposition algorithm is run, followed by calculating the objective function J for each cell. The angles for the three highest scoring cells (per angle) are then chosen for the next iteration. For the forthcoming iterations the angle spacing is halved and used to generate six new angles by adding and subtracting the spacing for the three best angles from the previous iteration. The algorithm continues for a given number of iterations, and lastly picks the highest scoring cell. The picked cell is then removed from the map and saved in the list of picked cells. The procedure is then repeated until the map is empty, and we are left with the exact cellular decomposition of the map (PWH).

In the original thesis it was found that starting out with 6 rotations with an even spacing of 30 degrees from 0 to 180 degrees, and re-running the algorithm for 5 iterations until the angle resolution was below 1 degree was sufficient to match 97% of the brute force solutions.

As the split and merge algorithm is designed for big agricultural machines it mentions the use of headland i.e. a turning zone before proceeding with the next straight segment. As mowers are driven by differential drive, the non-holonomic constraints are more relaxed and the need for a headland might not be of as big importance, especially for single blade mowers. In this implementation the map is first padded to account for boundary passes around the perimeter and all obstacles of the map, allowing turning outside the planned cell.

As the algorithm sequentially chooses a cell and then removes it from the PWH it is evident that the PWH will be split up into separate PWH which here are called islands. Each island can then be handled separately since each island is not affected by cell generation in other islands.

3.2.1 Objective functions

Given that the optimized BCD algorithm sequentially chooses the best cell given an objective, it is important to firstly choose a criterion and tune the parameters before evaluating the performance.

For the evaluation two different criteria are chosen, one simple and one complex. The simple criterion takes the weighted sum of the normalized area with respect to the remaining area, and a measure of tallness of the cell.

$$J^s = \rho_1 \frac{\text{area}(cell)}{\text{area}(island)} + \rho_2 \frac{\text{area}(cell)}{\text{width}_x(cell)} \tag{3.7}$$

The remaining area is based on the remaining area of the island where the cell resides.

A more advanced and computationally heavy criterion is based on the estimated time used to cover the cell. The cell lines are planned and time estimated in accordance to section 3.3.2. This estimation is then normalized using a simplified measure which is the time it would to take for the mower to cover the same area when going straight at a given average speed. This is weighted and summed with the normalized area and

an added benefit $f(\cdot)$ for a cell that is angled in the same direction as a neighboring previously selected cell.

$$J^c = \rho_1 \frac{\text{area}(cell)}{\text{area}(island)} + \rho_2 \frac{\text{t}_{est}(cell)}{\text{t}_{norm}(cell)} + \rho_3 f(\cdot) \tag{3.8}$$

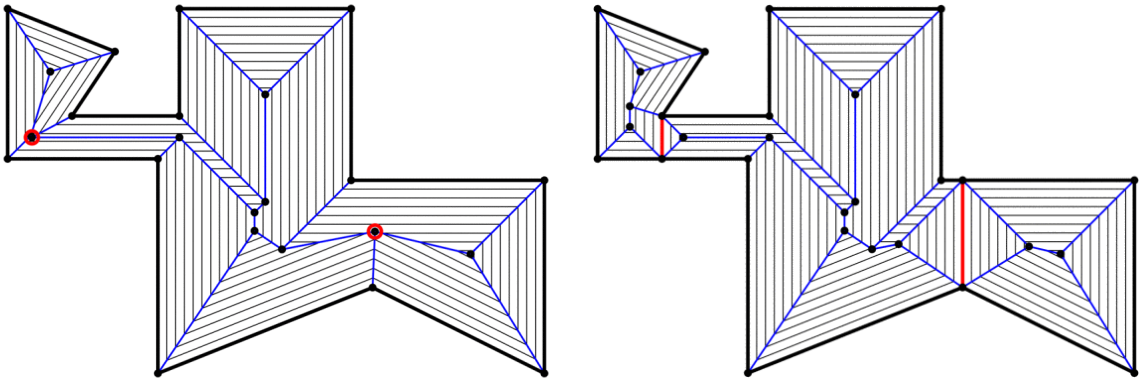
The parameters are tuned manually for each map and objective function.

3.2.2 Baseline BCD

When comparing the result from the optimized BCD algorithm the regular BCD algorithm is used, to attain good results from the regular BCD algorithm the map is rotated as in the first step of the optimized BCD algorithm. Instead of removing the best cell and continue, the map rotation with the maximum sum of all cells given the objective functions stated in (3.7) and (3.8). This can be seen as generating an optimal rotation of the map for regular BCD, given an optimization criteria.

3.3 The constriction decomposition method

Another Exact cellular decomposition method is the constriction decomposition method (CDM) [16], [17]. The method builds on the straight skeleton [10] of a PWH, using the straight skeleton split points to identify constriction points in the map. The vertex connected to the split point is connected to the nearest boundary edge or vertex of the face opposite of the split vertex. Figure 3.2 shows an example of the straight skeleton of a map, together with the decomposed and planned map.



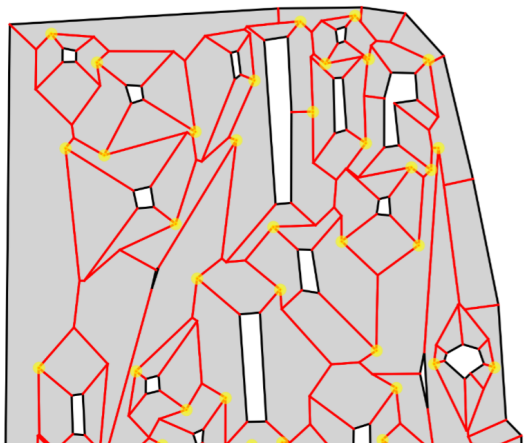
(a) The straight skeleton with the offsets lines, The straight skeleton is depicted with blue lines, and the split points are marked in red.

(b) The decomposed map along with three cells split by the red lines, along with a planned route computed with the straight skeleton.

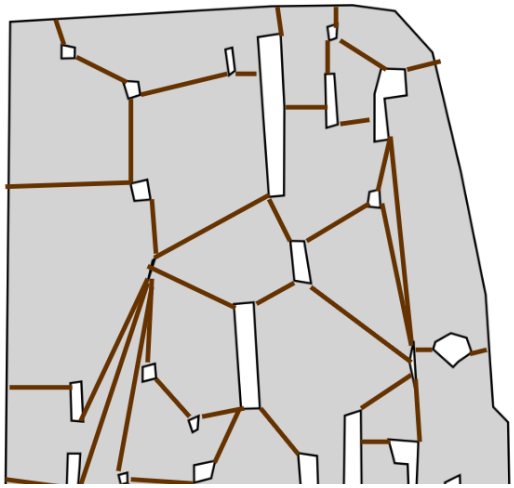
Figure 3.2: An example map showing the straight skeleton (a) together with the decomposed and planned map (b) © 2016 IEEE

In the original paper and thesis the method showed promising results being able to decompose office environments. Although, preliminary results on complex gardens have here shown to give undesirable results such as the example in Figure 3.3,

where the cells are highly irregular. Edge cases include wedge-shaped obstacles such as the two wedges in Figure 3.3b which result in pointy cells that are hard to cover efficiently. The original paper does not mention or show examples of similar type, and the algorithm could possibly be adapted to handle such scenarios better, but is not covered in this thesis.



(a) The straight skeleton (red) of a complex map with split vertices marked (yellow)



(b) The CDM cell decomposition of a complex map, generated manually

Figure 3.3: An illustration of the straight skeleton along with the CDM cell decomposition of a complex map

3.3.1 Individual cell planning

For the monotone polygon cells generated by the BCD, the cells are filled by laying straight lines with an even spacing. The spacing has to be smaller or equal to the mowers track width [2] to guarantee coverage. The number of cells and spacing for a given cell is

$$N_l = \text{floor}\left(\frac{x_{max} - x_{min}}{t_w}\right) + 1. \tag{3.9}$$

The lines are subsequently planned in a back and forth pattern depending on the starting point which is decided by the route planner.

3.3.2 Time estimation for monotone cell planning

To simplify the calculation of the time taken for the mower to cover a monotone cell containing only parallel straight lines, 180-degree turns, and transport segments, the time to cover the cell is estimated based on the parameters of the mower, straight line distances, and a fixed turning time. This is similar to the estimations done in [18].

[2] The predetermined spacing between the lines that account for cutting overlap

For the straight line segments a constant acceleration a is assumed until the desired velocity v_s is reached, the time t_a and distance s_a it takes to fully accelerate is expressed as

$$t_a = \frac{v_s - v_0}{a} \qquad s_a = t_a \frac{v_s + v_0}{2}, \tag{3.10}$$

where v_0 is the velocity coming in or out of a turn. The time t_s it takes to travel the distance of of segment i of length $s_l(i)$ is calculated as

$$t_s(i) = \begin{cases} 2\left(-\frac{v_0}{a} + \sqrt{\frac{v_0^2}{a^2} + \frac{s_l(i)}{a}}\right) & \text{if} \quad s_l(i) < 2s_a \\ \frac{s_l(i) - 2s_a}{2v_s} + 2t_a & \text{otherwise.} \end{cases}. \tag{3.11}$$

The total number of turns for a map is

$$N_t = \sum_{i=0}^{N_c} N_l(i) - 1, \tag{3.12}$$

where N_c is the number of cells and $N_l(i)$ is the number of lines for a given cell i.

The total accumulated time for all cells is then given by

$$T = \sum_{i=0}^{N_t} t_{turn}(i) + \sum_{j=0}^{N_s} t_s(j). \tag{3.13}$$

3.4 Inter-cellular path planning

For the path planning to be complete the order of the cells and the path between has to be decided. The algorithm that decides the order in which the cells are connected in conjunction with the path is here denoted as a "route planner". In the following section an optimal route planner and a proposed greedy route planner are described along with the visibility graph which they both depend on.

3.4.1 Visibility graph

Given that every cell has a starting and ending point, the transport between cells can be computed using a path planning algorithm based on the maps visibility graph [15]. The visibility graph is computed by finding all vertices in the map's polygons that are visible to each other. This can then be used to find the shortest path from a given point to another using a search algorithm such as A* [15]. Before generating the visibility graph, the polygons of the map should be inflated or deflated to account for the size of the robot to avoid collisions with the obstacles.

3.4.2 Route planning as a general traveling salesman problem

In [18], the route planning is formulated as an Equality Generalized Traveling Salesman Problem (E-GTSP), where each cell has multiple possible coverage patterns each with an estimated coverage time together with the start and end point of the pattern. In the case of the BCD algorithm each cell has four patterns where each pattern corresponds to the mower starting at each of the end points for the outermost lines. Given all possible cell patterns the estimated traveling time to all other patterns in the map is calculated with the visibility graph, and added to a graph together with the weighted edges representing the transport time. The graph is then fed to an optimizer with the objective to minimize the time, given that only one pattern for each cell is chosen. A similar approach is used in [12].

3.4.3 A greedy route planner

In this thesis a simple greedy route planner is used, performing the following steps.

1. Choose a starting cell that touches the map's outer boundary;
2. Plan the lines of the cell from the outer boundary in a back and forth pattern until the end of the last line is reached;
3. Find the closest cell and plan from the closest point of an outermost line in a back and forth pattern until the end of the last line is reached;
4. Repeat step 3. until no cells are left to visit.

When all cells have been planned individually, the start and end points are connected using a visibility graph to avoid possible obstacles. This approach avoids having to evaluate the visibility graph for all possible cell pattern combinations with the cost of attaining a sub-optimal path.

3.5 Implementation of the BCD and optimized BCD

The BCD and optimized BCD are implemented in the Julia programming language [19] using the LibGEOS.jl package, a Julia wrapper for the computational geometry package GEOS [9]). LibGEOS.jl is used for handling most geometric objects, predicates and operations. The implementation follows the description in Section 3.1 and only deviates on how the close events are handled and certain edge cases. The close above and close under events are handled as two regular close events that are vertically offset from the event vertex, in this way the implementation is simplified as only the open and close events need handling. The offset is set at $\pm 10^{-8}$ meters, which was assumed a negligible offset in the scale of a whole map.

The implemented BCD algorithm has proven to be robust, although is has been discovered to be problematic when used in the optimized BCD, especially for larger complex maps for certain parameter ranges. The problems arise especially for parameters that prefer long and thin cells, where the BCD algorithm looses track of

the ceiling and floors when generating the cells to evaluate. The problem is believed to be combination of less ideal geometries that arise from removing the cells one by one and floating point errors when slicing the ceilings and floors of the map.

The optimized BCD sequentially finds and removes part of the map for each cell that it chooses, this removal of cells can generate less ideal geometries for the remaining map. The current implementation removes the cell by rotating the cell such that it aligns correctly with the original orientation and removing it with the *difference* function from the LibGEOS.jl package. To account for the slight anomalies that arise from using floating point numbers, the cell is first inflated by 10^{-8} meters before removal. The method of inflating the cell has been found to generate its own set of problems although giving a more consistent result than simply removing it.

The problems that can arise in the optimized BCD is mainly believed to be a problem of this specific implementation and have not been investigated thoroughly, as the results attained are believed to be adequate for the evaluations performed in this thesis.

4

Simulation and control

In this chapter, possible path following controllers for differential drive robots are described, followed by a description of the simulation environment developed and used to verify the CPP algorithms.

4.1 Path following controllers

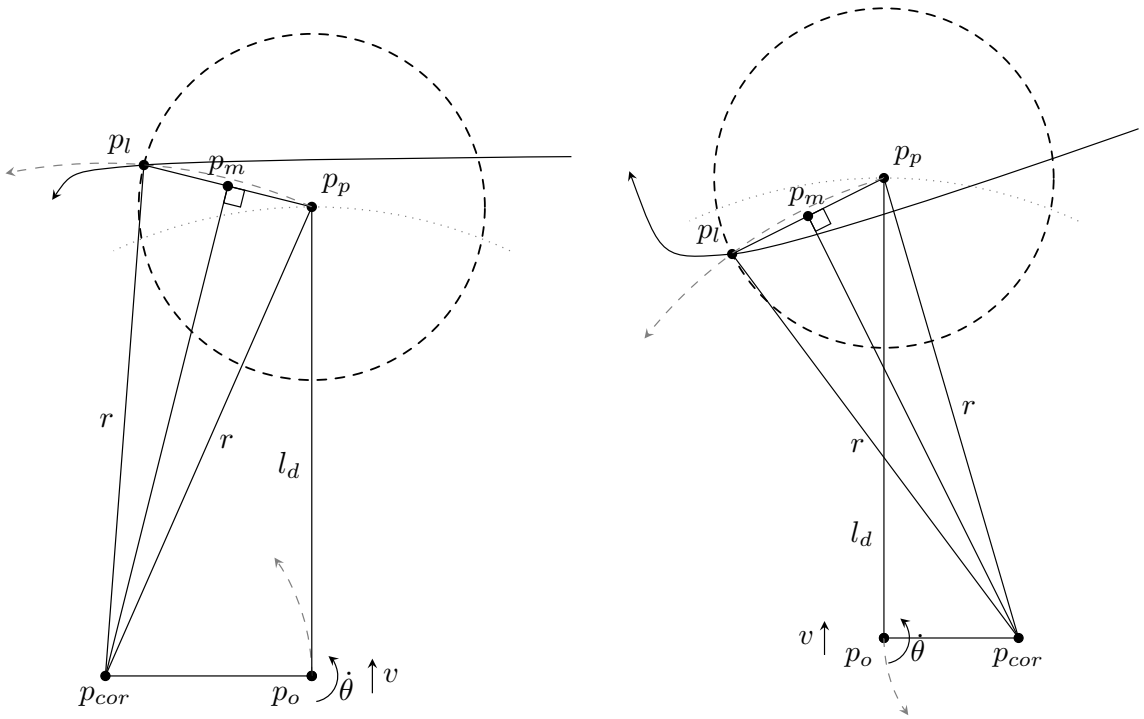
To evaluate the performance of the CPP algorithms in simulation, a path following controller for the mower is needed. Common controllers for differential drive robots are the pure pursuit controller [20], Stanley controller [21], linear quadratic regulators (LQR) [22], and model predictive controllers (MPC) [23]. The pure pursuit and Stanley controllers are relatively simple controllers both based on geometric relations between the vehicle and the path guaranteeing convergence to the path, while the LQR and MPC controllers act on a reference error with the possibility of including more advanced vehicle dynamics. The MPC controller can additionally plan ahead while taking the constraints of the mower into account but at a computational cost.

One possible disadvantage of the controllers in their standard form is that a point in between the two wheels, not the cutting deck, follow the path. This behavior leads to the cutting deck diverging from the path when following sharp corners. There exist examples of controllers that follow with an external reference such as the Fuzzy pure pursuit with a front axle reference (FPPC-FAR) [24] and a proportional controller for a drawing robot [25]. Here a pure pursuit controller inspired by the FPPC-FAR controller with the possibility of reversing is presented and used in simulation.

4.1.1 Pure pursuit controller with an external reference

In this section a version of the pure pursuit controller acting on an external reference such as a cutting deck is described. It is inspired by the FPPC-FAR with handling of reversing along with speed limiting according to Section 2.1.1.

The controller builds on the same geometric principles as the regular pure pursuit controller. While the regular pure pursuit controller finds a curve going through the look-ahead point and mower center point with a common center of rotation (COR), the extended pure pursuit controller finds the common COR for the look-ahead point, the external reference point. When the COR point is found, the curvature can be converted into a velocity and angular velocity for the robot based on a reference velocity.



(a) Positive forward velocity

(b) Negative forward velocity

Figure 4.1: Examples of the extended pure pursuit controller. The dashed gray line shows the trajectory of the points p_o and p_p, the dotted line shows the border between forward and reverse

The curvature needed to calculate the forward velocity along with the angular velocity can be expressed as

$$\mathbf{p}_{dir} = \mathbf{p}_l - \mathbf{p}_d \tag{4.1}$$

$$\mathbf{p}_m = \mathbf{p}_d + \frac{\mathbf{p}_{dir}}{2} \tag{4.2}$$

$$\mathbf{p}_r = \begin{bmatrix} 0 & -1 \\ 1 & 0 \end{bmatrix} \frac{\mathbf{p}_{dir}}{||\mathbf{p}_{dir}||_2} \tag{4.3}$$

$$R = p_m^y - \frac{p_m^x p_r^y}{p_r^x} \tag{4.4}$$

$$\kappa = \begin{cases} \frac{1}{R} & \text{if} \quad R \neq 0 \\ 0 & \text{otherwise.} \end{cases} \tag{4.5}$$

where $\mathbf{p}_l$, $\mathbf{p}_d$, and $\mathbf{p}_m$ are points represented as vectors spanning from the origin p_o located in the middle of the (virtual) drive axle. Examples of the forward and reversing scenarios are depicted in Figure 4.1. As the curvature is not defined when having a look-ahead point straight in front of the reference point ($R = 0$), the curvature κ is set to zero. For positive reference velocities, velocity and angular velocity are given by

$$v = v_{ref}, \tag{4.6}$$

$$\dot{\theta} = \kappa\, v_{ref}, \tag{4.7}$$

where v_{ref} is the desired reference velocity of the mower. A disadvantage of only having forward motion emerges when following tight corners where reversing is needed to follow a path correctly. The need for reversing arises when the look-ahead point is inside the disc defined by radius l_d with the center point in the middle of the (virtual) drive axle (p_o). In these scenarios the controller attempts to follow the point by going forwards and doing a full rotation to reach the point diverging from the path. To avoid this behavior the direction of the velocity can be set by

$$v_{dir} = \text{sign}(\|p_l\|_2 - l_d) = \text{sign}(p_l^x R) \tag{4.8}$$

and (4.6), (4.7) are then rewritten as

$$v = v_{dir} v_{ref} \tag{4.9}$$

$$\dot{\theta} = \kappa\, v_{ref}. \tag{4.10}$$

To account for the mower constraints on angular velocity in (2.1), the angular velocity defined in (4.7) is first computed using $v_{ref} = v_s$ (maximum velocity when going straight) and checked if it exceeds the limit $|\dot{\theta}| > \epsilon$. If the limit is exceeded, v_{ref} is recalculated according to (2.1) as

$$v_{ref} = \frac{\dot{\theta}_{max}}{|\kappa| + \frac{\dot{\theta}_{max}}{v_t}}, \tag{4.11}$$

$\dot{\theta}$ and v in (4.9) and (4.10) are then recalculated with the new v_{ref}.

4.2 Simulation environment

The simulation is a hybrid system simulation able to handle continuous dynamics coupled with discrete events. The continuous mower dynamics are based on a variable time-step 4th order Runge-kutta scheme with fixed maximum time steps for updating the discrete-time control inputs. Inputs to the simulation environment are the CPP path along with the map and a controller callback function. The simulation is used to verify that the CPP algorithms produce feasible fully covering paths. Post-processing of the simulation data includes the calculation of coverage percentage, total time duration, and the distance traveled by the robot.

5

Optimized BCD results

In the following chapter results from the optimized BCD are presented. The results are shown for four maps and each presented individually with both the estimated time from Section 3.3.2 and simulated time. All simulation results show a coverage of above 99 % and the coverage percentage has therefore been omitted from the comparisons. To simplify the comparison all results shown are for the smaller single cutting disc mower.

The maps are all roughly $1000\,m^2$ each. An important note is that all maps shown here are deflated from the original maps by two track widths, such that the mower is allowed to turn outside of the cells, the times reported are without the inclusion of boundary cutting. To distinguish between the optimized BCD, Baseline BCD and and the objective functions, the notations $J_s^O, J_c^O, J_s^B, J_c^B$ are used. The superscript denotes the algorithm O for optimized BCD and B for the baseline BCD, while the subscripts denote the objective functions c for complex and s for simple as in Section 3.2.1. The following sections present the result for each map.

5.1 Map 1: Convex boundary and convex obstacles

The cell decomposition for the different algorithms and objective functions can be seen in Figure 5.1. For the J_s^O algorithm the first cell chosen is a long and thin cell followed by two big cells (2 and 3), in comparison to the J_C^O which chooses the two big cells (1 and 2) before proceeding with the smaller cells in between cell 1 and 2. A peculiarity is cell number 5 in the J_C^O algorithm, where the BCD composition manages to extend the cell between cell 1 and 4 and avoid having to generate additional cells to fill the small void.

For the numeric results found in Table 5.1 and Table 5.2, all algorithms are of similar total estimated time, differing at most by 3.5 % where the fastest path is generated by the J_S^B objective. All objectives generate 7 cells, except J_C^O which only creates 5 cells and also has a shorter transport length compared to the other cells. Note that the estimated time, simulated time and number of lines follow the same ranking of the algorithms. When comparing the time deviation between the simulated and estimated time, the deviations are all close to 2 %.

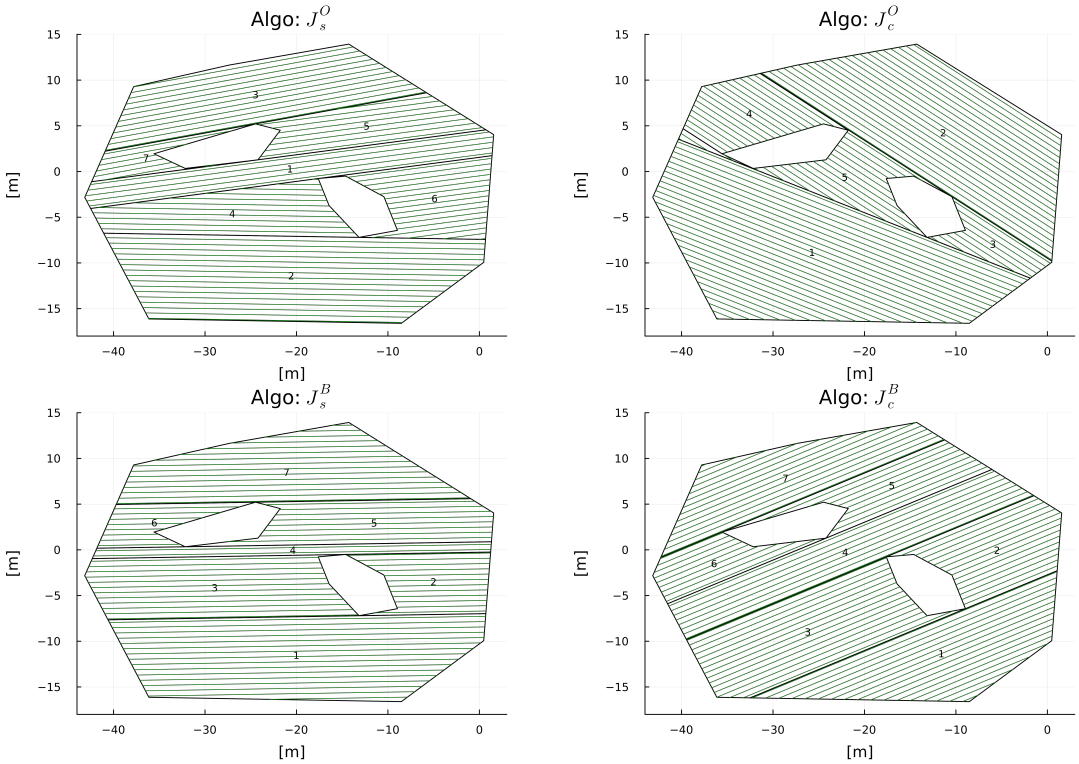


Figure 5.1: The results of the cell decompositions for map 1, the numbers indicate the order the cells are generated. For presentation purposes every third line is depicted in the plots

Table 5.1: Planning results for map 1

Algo.	No. cells [-]	No. lines [m]	Transport len. [m]
J_s^O	7	225	37.90
J_c^O	5	282	13.53
J_s^B	7	219	42.56
J_c^B	7	253	43.74

Table 5.2: Simulation results for Map 1. Note the distance is measured as the distance driven by the robot and not that of the cutting blade

Algo.	Sim. time [min]	Est. time [min]	Time Dev. [%]	Distance [m]
J_s^O	161.79	158.31	2.20	5259.93
J_c^O	166.05	162.98	1.88	5162.20
J_s^B	160.74	157.62	1.98	5243.46
J_c^B	164.54	161.41	1.94	5236.90

5.2 Map 2: V-shaped boundary with obstacles

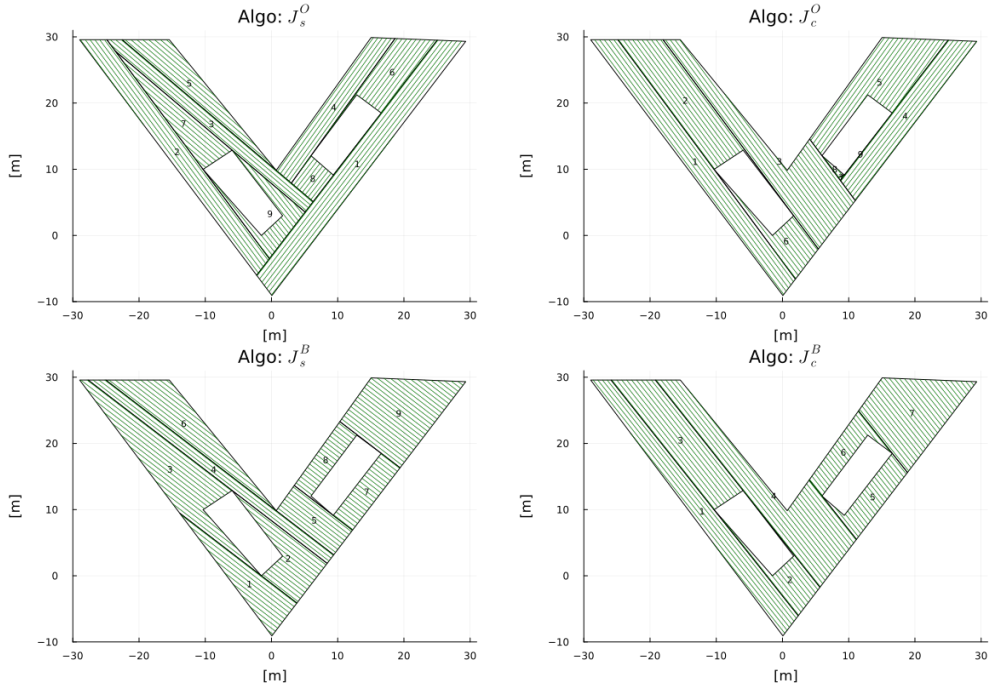


Figure 5.2: The results of the cell decompositions for map 2, the numbers indicate the order the cells are generated. For presentation purposes every third line is depicted in the plots

The cell decompositions for the v-shaped map is shown in Figure 5.2. This map was chosen as it highlights the benefit of the optimized BCD that is able to take advantage of the map topography following the narrow passages as opposed to the

baseline algorithms which can only choose one direction. A thing to note in the J_c^O are the small cells 7, 8 and 9, which are artifacts of the algorithm leaving small irregularly shaped islands to be planned.

The numeric results in Table 5.3 and Table 5.4 show that J_c^O has the shortest estimated time which is 11 % faster compared the fastest baseline algorithm, along with least number of lines. Here the time deviation between the simulation time and estimated time has a bigger discrepancy compared to Map 1, although the ranking of the algorithms stays the same for both times.

Table 5.3: The planning results for map 2

Algo.	**No. cells [-]**	**No. lines [m]**	**Transport len. [m]**
J_s^O	9	209	102.18
J_c^O	9	194	40.96
J_s^B	9	380	34.26
J_c^B	7	334	74.74

Table 5.4: Simulation results for Map 2. Note the distance is measured as the distance driven by the robot and not that of the cutting blade

Algo.	**Sim. time [min]**	**Est. time [min]**	**Time Dev. [%]**	**Distance [m]**
J_s^O	123.86	118.52	4.51	3853.68
J_c^O	120.17	116.69	2.98	3801.15
J_s^B	139.38	136.05	2.44	3682.48
J_c^B	136.14	131.57	3.47	3762.61

5.3 Map 3: complex boundary and obstacles

The cell decompositions for the complex map are shown in Figure 5.3. This map might resemble a garden with flower beds or a golf course with bunkers. In this example the baseline solutions converge to the exact same solution. It can further be seen that the first cell of J_s^O resembles that of cell 13 in J_s^B. The cells 11, 14, 15 of J_s^O show the result of the optimized BCD having to fill the void generated by other cells. Although the J_s^O algorithm generates less than half the number of cells of the baseline solutions together with a shorter total travel distance, the number of lines exceed that of the baseline and results in worse cutting performance. Again, as in maps 1 and 2, the simulation and estimated times are close with a consistent bias in the deviation.

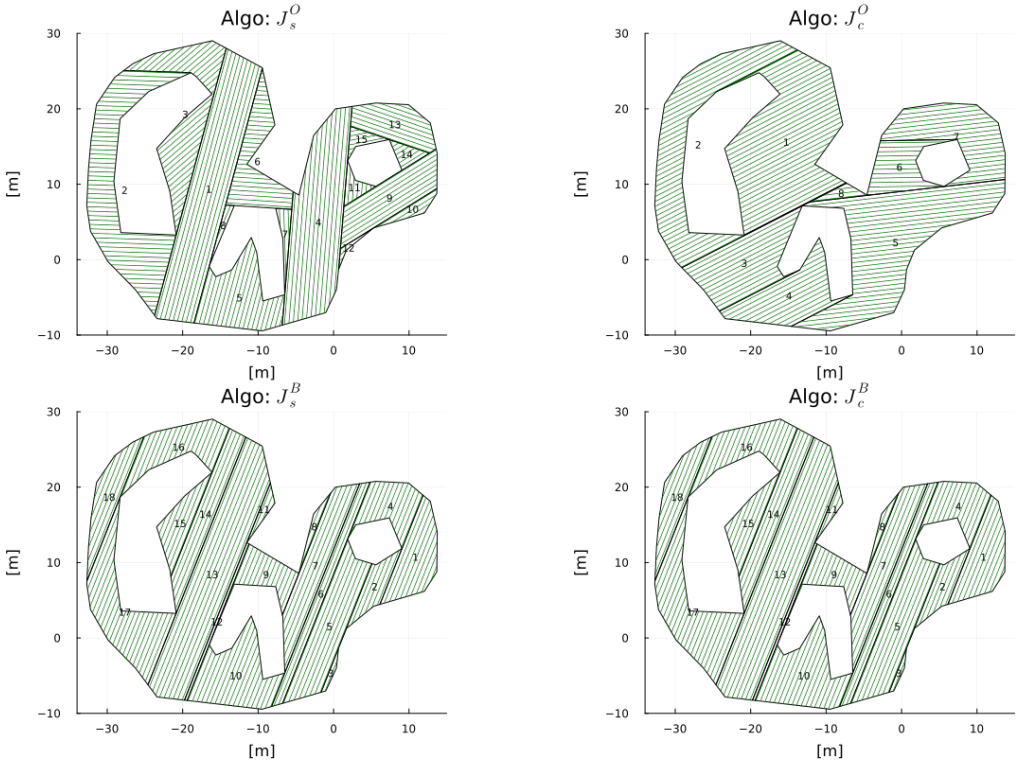


Figure 5.3: The results of the cell decompositions for map 3, the number indicate the order the cells are generated. For presentation purposes every third line is depicted in the plots

Table 5.5: The planning results for map 3

Algo.	No. cells [-]	No. lines [m]	Transport len. [m]
J_s^O	15	649	87.71
J_c^O	8	491	96.97
J_s^B	18	438	156.31
J_c^B	18	438	156.31

Table 5.6: Simulation results for Map 3. Note the distance is measured as the distance driven by the robot and not that of the cutting blade

Algo.	Sim. time [min]	Est. time [min]	Time Dev. [%]	Distance [m]
J_s^O	195.64	191.42	2.20	4730.24
J_c^O	180.02	173.55	3.73	4734.75
J_s^B	176.77	169.31	4.40	4838.68
J_c^B	176.77	169.31	4.40	4838.68

5.4 Map 4: Triangle shaped boundary and 3 obstacles

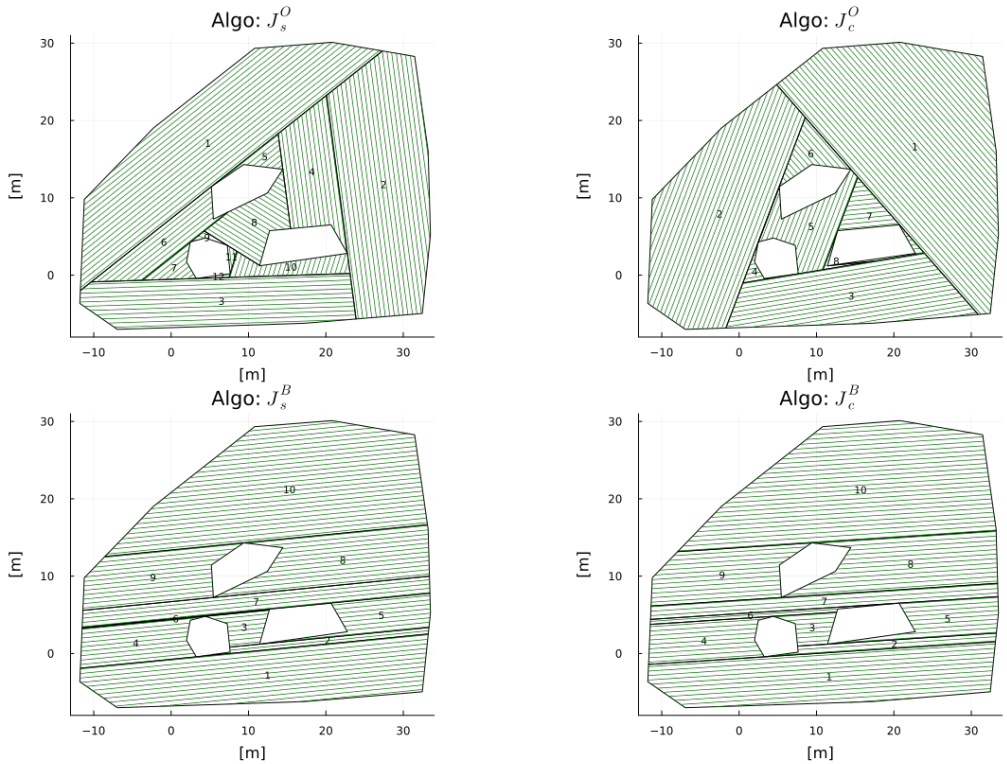


Figure 5.4: The results of the cell decompositions for map 4, the numbers indicate the order the cells are generated. For presentation purposes every third line is depicted in the plots

The cell decompositions for map 4 are shown in Figure 5.4 along with the numeric results in Table 5.7. Here both cases of the optimized BCD have a longer cutting time than the baseline algorithms. In addition both optimized BCD algorithms generate small cells around obstacles.

The simulation results for map 4 are presented in Table 5.8, the results diverge with the estimated time by a margin of 2.5 %. For the J_s^O and J_c^O the simulation times are almost equal, while the estimated time differs by more than 3 minutes.

Table 5.7: The planning results for map 4

Algo.	No. cells [-]	No. lines [m]	Transport len. [m]
J_s^O	12	409	52.79
J_c^O	8	471	52.53
J_s^B	9	273	104.24
J_c^B	10	275	83.37

Table 5.8: Simulation results for Map 4. Note the distance is measured as the distance driven by the robot and not that of the cutting blade

Algo.	Sim. time [min]	Est. time [min]	Time Dev. [%]	Distance [m]
J_s^O	215.81	212.84	1.40	6566.99
J_c^O	215.29	216.79	-0.69	6548.40
J_s^B	203.49	198.55	2.49	6669.65
J_c^B	203.88	199.26	2.32	6660.56

6
Discussion and future work

In this chapter the results and implementation of the BCD and optimized BCD are discussed along with results from the route planner and simulation. In the last section suggestions for future work are presented.

6.1 Discussion

For the BCD results two different types of the algorithm with two different optimization criteria were used. One simpler baseline algorithm only executing the first step of the optimized BCD effectively generating only parallel lines, and secondly the optimized BCD generating monotone cells of different orientations each filled with parallel lines.

The optimized BCD algorithm has shown to be able to generate faster cutting patterns when compared to the baseline BCD, while also performing worse or delivering negligible improvements over the baseline algorithm.

A drawback of the optimized BCD algorithm is that it is a sub-optimal greedy algorithm, it chooses the best cell given the objective and then removes it before proceeding with the next. This behavior can lead to small and odd cells which can lead to ineffective coverage. An example of the algorithm ending with multiple odd shaped cells is map 4 in Section 5.4 where the optimized BCD generates a much larger number of lines to be covered leading to more turns and accelerations.

One important aspect of the optimized BCD is the objective function and its tuning parameters. The results show that the parameters, algorithm, and objective functions can generate a wide variety of compositions. The process of finding a good objective function with good general parameters has proven to be hard task and still leaves room for improvement. In [14], the parameters and objective functions was found not to be an issue, and so might only be a problem for the specific implementation in this thesis. Furthermore, [14] does not state how different aspects of the objective function are normalized and this might be a cause for the issue. In the current implementation the area normalization is performed with the area left on the map, while the normalization for the time or width aspects are done in regard to the geometry of each cell. A possible improvement could be to normalize the estimated time with regard to all cells being evaluated for the given iteration.

Another important aspect to take into account when comparing to the split and merge algorithm is the difference in the underlying cell decomposition algorithm.

The split and merge algorithm merges trapezoidal cells and has a limit on the change in angle for the merged cells. This avoids L-shaped cells and other irregular shapes, and was here assumed to have less of an importance for differential drive mowers. It could be that this has a bigger effect on the end result than originally anticipated for the optimized BCD and possibly leading to sub-optimal results and a more fragile algorithm. It should be possible to imitate the trapezoidal cell merge limitation in the BCD algorithm by adding close events to the polygon vertices for which the corner exceeds a certain angle, however this has yet to be tested.

That the deviation between the simulated and estimated times show a clear bias towards the estimated time to be shorter than the simulated time, while having a reasonable variance, might indicate that the time estimation of the maps can be improved using system identification. Further possible improvement to estimating the time is to use the lengthwise offset of the lines. The estimated time for a given turn could be pre-computed based on simulations and added to a lookup table. However, in general, the results indicate that the estimation gives a reasonable estimate in both time and especially when comparing solutions.

When looking at the results for the greedy route planner, the proportion of the length of the transport compared to the total path length varies greatly from 0.2 % to above 3 %. Since the greedy algorithm is a less sophisticated algorithm compared to the optimization based algorithms described in Section 3.4.2 it might be viable alternative if computational expense is to be minimized. As the optimization based algorithms are not implemented here it is hard to draw conclusions from this result. A given fact is that the proportion of the transport length cannot go below 0 % which might indicate that implementing a more advanced algorithm might give negligible improvements.

6.2 Future work

In this section two suggestions for future work are described, firstly a way to do better map dependent parameter estimation and secondly a suggestion to take advantage of the user running the algorithm.

6.2.1 Parameter estimation

When evaluating the optimized BCD algorithm is has been seen that to attain good results, the parameters have been specific for each map. During this work a good choice of general parameter selection has not been found, although it might be possible to estimate good parameters based on the mower and map such as the track width, area, number of obstacles, size and shape of obstacles and the outer perimeter.

6.2.2 User assisted cell decomposition

When planning a map the user might be of assistance for the planning. If the user is an experienced landscaper and/or gardener he or she might have prior knowledge

and expectation of the mowing pattern. Furthermore, the user could be able to select cells that are unwanted and ask for these cells to be re-planned given some input.

7
Conclusion

This thesis set out to find a suitable coverage path planning algorithm useful for autonomous lawnmowers with and without multiple cutting discs. After an initial literature study, the work focused on exact cellular decomposition methods. For evaluation, two exact cellular decomposition coverage path planning algorithms where further studied, the split and merge algorithm, and the constriction decomposition method. Preliminary results suggest that the constriction decomposition method can generate highly irregular cells when applied to complex areas with many obstacles and was deemed less ideal for the case studied.

An adaption of the split and merge algorithm using the boustrophedon cellular decomposition instead of conditionally merged trapezoidal cells was implemented. Results for both estimated and simulated times show that the adapted algorithm performed irregularly, exhibiting slower or faster times depending on the evaluated map and objective function. A set of general parameters for the optimization functions were not found and calls for more research. An implication of the algorithm is the removal of cells that for the implementation has proved to be problematic for certain parameters and maps returning no results. The underlying problem to this has not been fully investigated in this work.

Given both simulation and estimated times, the results indicate that using time estimation can be of great value when comparing exact cell decomposition algorithms using boustrophedon paths, providing a computationally inexpensive way of comparing results.

In the process of evaluating coverage path planning algorithms, a pure pursuit controller using an external reference with the possibility of reversing was developed and implemented. The controller has shown to work well in ideal conditions following the generated paths. In addition, a simple greedy algorithm for choosing the visitation order of the pre-planned cells was presented, delivering transport paths with a length of under 3 % of the total path length.

Bibliography

[1] R. Moeller, T. Deemyad, and A. Sebastian, "Autonomous navigation of an agricultural robot using RTK GPS and pixhawk", in *2020 Intermountain Engineering, Technology and Computing (IETC)*, 2020, pp. 1–6. DOI: 10.1109/IETC47856.2020.9249176.

[2] *New Husqvarna autonomous operation - brings self-operated mowing to large areas | Husqvarna Group*. [Online]. Available: https://www.husqvarnagroup.com/en/press/new-husqvarna-autonomous-operation-brings-self-operated-mowing-large-areas-1728683 (visited on 12/06/2022).

[3] *Husqvarna automower® 550 epos™ | Husqvarna Group*. [Online]. Available: https://www.husqvarna.com/se/robotgrasklippare/automower-550epos/ (visited on 05/25/2023).

[4] E. Galceran and M. Carreras, "A survey on coverage path planning for robotics", *Robotics and Autonomous Systems*, vol. 61, no. 12, pp. 1258–1276, Dec. 2013. DOI: 10.1016/j.robot.2013.09.004.

[5] E. M. Arkin, S. P. Fekete, and J. S. Mitchell, "Approximation algorithms for lawn mowing and milling", *Computational Geometry*, vol. 17, no. 1, pp. 25–50, 2000. DOI: https://doi.org/10.1016/S0925-7721(00)00015-8.

[6] C. S. Tan, R. Mohd-Mokhtar, and M. R. Arshad, "A comprehensive review of coverage path planning in robotics using classical and heuristic algorithms", *IEEE Access*, vol. 9, pp. 119 310–119 342, 2021. DOI: 10.1109/access.2021.3108177.

[7] H. Choset, "Coverage of known spaces: The boustrophedon cellular decomposition", *Autonomous Robots*, vol. 9, no. 3, pp. 247–253, Dec. 2000.

[8] J. Latombe, *Robot Motion Planning* (The Springer International Series in Engineering and Computer Science). Springer US, 2012, ISBN: 9781461540229. [Online]. Available: https://books.google.se/books?id=nQ7aBwAAQBAJ.

[9] GEOS contributors, "GEOS coordinate transformation software library", 2021. [Online]. Available: https://libgeos.org/.

[10] P. Felkel and S. Obdrzalek, "Straight skeleton implementation", in *Proceedings of spring conference on computer graphics*, 1998, pp. 210–218.

[11] H. Choset and P. Pignon, "Coverage path planning: The boustrophedon decomposition", in *Proceedings of 1st International Conference on Field and Service Robotics (FSR '97)*, Dec. 1997, pp. 216–222.

[12] S. Bochkarev and S. L. Smith, "On minimizing turns in robot coverage path planning", in *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, IEEE, Aug. 2016. DOI: 10.1109/coase.2016.7743548.

[13] H. Choset, "Coverage for robotics - a survey of recent results", *Annals of Mathematics and Artificial Intelligence*, vol. 31, no. 1, pp. 113–126, 2001, ISSN: 1573-7470. DOI: 10.1023/A:1016639210559. [Online]. Available: https://doi.org/10.1023/A:1016639210559.

[14] T. Oksanen, "Path planning algorithms for agricultural field machines", Ph.D. dissertation, Helsinki University of Technology, 2007.

[15] H. Choset, K. M. Lynch, S. Hutchinson, *et al.*, *Principles of Robot Motion : Theory, Algorithms, and Implementations*. (Intelligent Robotics and Autonomous Agents Ser). MIT Press, 2005, ISBN: 9780262255912.

[16] S. Brown and S. L. Waslander, "The constriction decomposition method for coverage path planning", in *2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, Oct. 2016. DOI: 10.1109/iros.2016.7759499.

[17] S. Brown, "Coverage path planning and room segmentation in indoor environments using the constriction decomposition method", M.S. thesis, Aug. 2017. [Online]. Available: https://uwspace.uwaterloo.ca/handle/10012/12240.

[18] R. Bähnemann, N. Lawrance, J. J. Chung, M. Pantic, R. Siegwart, and J. Nieto, "Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem", in *Field and Service Robotics*, Singapore: Springer Singapore, 2021, pp. 277–290, ISBN: 978-981-15-9460-1.

[19] J. Bezanson, A. Edelman, S. Karpinski, and V. B. Shah, "Julia: A fresh approach to numerical computing", *SIAM Review*, vol. 59, no. 1, pp. 65–98, Jan. 2017. DOI: 10.1137/141000671.

[20] R. C. Coulter, "Implementation of the pure pursuit path tracking algorithm", Carnegie Mellon University, Pittsburgh, PA, Tech. Rep. CMU-RI-TR-92-01, Jan. 1992.

[21] G. M. Hoffmann, C. J. Tomlin, M. Montemerlo, and S. Thrun, "Autonomous automobile trajectory tracking for off-road driving: Controller design, experimental validation and racing", in *2007 American Control Conference*, 2007, pp. 2296–2301. DOI: 10.1109/ACC.2007.4282788.

[22] J. Ni, Y. Wang, H. Li, and H. Du, "Path tracking motion control method of tracked robot based on improved LQR control", in *2022 41st Chinese Control Conference (CCC)*, 2022, pp. 2888–2893. DOI: 10.23919/CCC55666.2022.9902113.

[23] S. Bouzoualegh, E.-H. Guechi, and R. Kelaiaia, "Model predictive control of a differential-drive mobile robot", *Acta Universitatis Sapientiae, Electrical and Mechanical Engineering*, vol. 10, no. 1, pp. 20–41, 2018. DOI: doi:10.2478/auseme-2018-0002. [Online]. Available: https://doi.org/10.2478/auseme-2018-0002.

[24] L. Yu, X. Yan, Z. Kuang, B. Chen, and Y. Zhao, "Driverless bus path tracking based on fuzzy pure pursuit control with a front axle reference", *Applied Sciences*, vol. 10, no. 1, 2020, ISSN: 2076-3417. DOI: 10.3390/app10010230.

[25] C.-L. Shih and L.-C. Lin, "Trajectory planning and tracking control of a differential-drive mobile robot in a picture drawing application", *Robotics*, vol. 6, no. 3, p. 17, Aug. 2017. DOI: 10.3390/robotics6030017.

DEPARTMENT OF SOME SUBJECT OR TECHNOLOGY
CHALMERS UNIVERSITY OF TECHNOLOGY
Gothenburg, Sweden
www.chalmers.se

CHALMERS
UNIVERSITY OF TECHNOLOGY