A Coverage Planning Algorithm for Agricultural Robots

Sanjiban Choudhury\textsuperscript{1} \hspace{1cm} Sanjiv Singh\textsuperscript{2}

\textsuperscript{1}Department of Electrical Engineering, Indian Institute of Technology Kharagpur, India.
\textsuperscript{2}Department of Computer Science, Carnegie Mellon University, USA

August 2009
## Contents

<table>
<thead>
<tr>
<th>S.No.</th>
<th>Section</th>
<th>Page No.</th>
</tr>
</thead>
<tbody>
<tr>
<td>1.</td>
<td>Introduction</td>
<td>5</td>
</tr>
<tr>
<td>2.</td>
<td>Background work</td>
<td>7</td>
</tr>
<tr>
<td>3.</td>
<td>Coverage by cellular decomposition</td>
<td>8</td>
</tr>
<tr>
<td>4.</td>
<td>Coverage by trajectory bending</td>
<td>25</td>
</tr>
<tr>
<td>5.</td>
<td>Headland turning</td>
<td>37</td>
</tr>
<tr>
<td>6.</td>
<td>Optimisation of cellular traversal</td>
<td>43</td>
</tr>
<tr>
<td>7.</td>
<td>Conclusion</td>
<td>47</td>
</tr>
<tr>
<td>8.</td>
<td>References</td>
<td>48</td>
</tr>
</tbody>
</table>
Abstract

We present a new scheme for an offline route planner for coverage of an agricultural area, which is capable of local modifications in reaction to obstacles. For a given vehicle dynamics and specifications, the route planner generates optimal routes which not only maximise the area covered in a given time but also respect all constraints. The paper uses the help of spline functions to model curved trajectories as well as provide a bound on the curvature of the route. The issue of turning at headlands is also taken up and a framework is created to enable turning at any angle avoiding “no go” areas. Finally the paper considers newer tactics of cellular traversal, providing guaranteed better results in bounded time than current methods.
1. Introduction

The focus of this work is in the area of agricultural vehicles operating on agricultural areas, even though the issue of coverage planning is equally significant in mining and land moving operations. When it comes to crop farming, several operations are carried out throughout the year. These operations are mainly – cultivation, seeding, fertilizing, spreading chemicals and harvesting. For each of these operations, different tools exist which is attached to the tractor and the field is covered. These tools are known as implements.

With the advent of automation in the agriculture industry, automated coverage algorithms must be developed. Till now agriculture operations are usually manually planned by contractors who have some sort of experience in this area. However, a change is required in this area. Firstly, there is no way of determining whether a given route is optimal or suboptimal, since a human would not follow any definite heuristic. Secondly, a naive method like “following the longest edge” or “spiralling in” does not always generate the perfect route for complicated fields. Lastly, the self driven agricultural machines need route inputs which cater to the specifications of the vehicle, which is not possible to be ensured by a human being who manually draws routes.

Besides the traditional problems of designing routes for complicated fields, there are many restrictions on where the vehicle can and cannot go. An agricultural plot contains several ridges, furrows, basins and wetlands. Irrigation can be solid set sprinkler systems, which when in operation have to be avoided with minimum overlap possible. Permanent sprinklers are placed in buried laterals or between alternate trees in orchards. Portable sprinklers like drip/micro systems are also in use. These result in patches of land which are sometimes avoided, and sometimes are treated indifferently. So a framework which analysed the size and the geometry of such areas and makes decisions to alter plans based on them is high in demand.

There are areas in the farm called “no-go” areas which even though they lie outside the area to be operated, still affect the outcome of the plan. Since areas outside the field is used as buffer areas of turning, putting a restriction on turning in turn changes how the route is planned. “No go” areas are usually lake embankments or fences lining the field.

![Figure 1. Schematic of the problem at hand](image)

The issue of turning is also not trivial. For a human operating a tractor, to turn from one row to another requires and intuitive driving sense which machines are not expected to have. So a framework
which considers the steering angle constraints while trying to optimise the turn as much as possible is required. The trajectories generated by this framework also have to comply with all constraints mentioned before. An agricultural robot usually will have an online planner with reactive behaviours which avoid dynamic obstacles and follow the route as smoothly as possible. To aid this online planner, the offline planner has to optimise the route as much as possible before hand where time is not a very big constraint. Thus, this paper is motivated by the need for such a robust agricultural coverage planning framework.

Figure 2. (a) Solid set sprinkler systems (b) Finite radius turning by lawn mowers
2. Background work

Coverage path planning research, unlike conventional start-goal problems, has been motivated by floor-cleaning, demining, lawn mowing and harvesting problems [7]. The target region becomes an integration of the robots footprint (sensor/instrument) over the coverage path. The problem is similar to a covering salesman problem, a variant of travelling salesman where the agent has to visit a neighbourhood of each city thus minimising his travel time. However, a coverage planning robot must pass over all points in the target environment as opposed to just passing through neighbourhoods.

Earlier works used heuristics which were simple yet they didn’t provide any provable guarantees on coverage of the whole area. In attempts to provide definitive guarantees, algorithms employing cellular decomposition have been invented. In cellular decomposition, the free space is broken into several regions which guarantee coverage. The robot has to visit each cell in the minimum possible time. The types of cellular decomposition are approximate, semi-approximate and exact [7]. Randomized approaches work well for floor cleaning problems. The advantage of this method is no sensors are required for localization. The method works well for systems which benefit financially by removing localisation systems [2]. Another reason is that if the probability of the detector or the footprint succeeding is less, then randomized approaches succeed over systematic coverage approaches. But with respect to agricultural operations, vehicles have high fuel consumption, thus the determining criteria is usually the travel time along with the fuel expenditure rather than construction cost.

Approximate cellular decomposition divides the target region into grids of unit size. Zellinsky [27] uses a wave-front distance transform algorithm which assigns 0 to goal cell and 1 to all surrounding and gradually iteratively builds up. Then the robot uses gradient descent to visit each cell. Spanning Tree Covering is also a method under approximate cellular decomposition. However using such methods does not ensure kinematic constraints will be satisfied.

Semi-approximate coverage is where the sides of the cell are fixed but top and bottom are open and union of the cells represent the area. Exact cellular decomposition uses defined cells and will be discussed in section 3. [26] Yang and Luo also presented a Neural Network approach for the coverage path planning problem.

Research into driving directions have also been done - Witney (1996), Hunt (2001), Bochtis Vougioukas, Tsatsarelis, and Ampatzidis (2006), and Hansen.Zhang, and Wilcox (2007). The field is assumed to be rectangular and driving directions are chosen to minimise coverage time. [18] Pandey proved the relationship between rounding a field and going back and forth. Stoll worked on a scheme of subdividing fields based on the longest side of the field. [22] Ryerson et.al worked on a method of employing genetic algorithm of finding the sequence of cells which are the fittest, i.e. cover the maximum area with minimum traversal distance.

[16] Oksanen et.al. proposed two methods of coverage planning – the “Split and Merge: method and Model Predictive Control. The former method is discussed in detail in section 3, while the latter is used as a starting point in section 6. The drawback of MPC is that it works in a receding horizon and cannot account for obstacles beforehand.

In conclusion, we see that not much stress has gone into incorporating vehicle constraints or considering arbitrary curves to bypass obstacles. This paper will try to look into the needs and possible advantage of finding effective solutions apart from using straight line motion to cover areas.
3. Coverage by Cellular Decomposition

The focus of this work is on coverage of areas by the means of exact cellular decomposition. An exact cellular decomposition is dividing a region into a set of non-intersecting group of regions, referred to as a cell, whose union fills the target environment. With the assumption that the robot can cover each cell with simple back and forth motions, the problem is reduced to planning motions from one cell to another. Each cell is represented as a node in the graph, and adjacent cells have a common edge connecting their nodes. This graph is known as the adjacency graph. A walk through this adjacency graph visiting each node at least once solves the coverage problem. Since this model translates into the “Travelling Salesman Problem”, it is guaranteed that a solution, possibly sub-optimal, exists. In the following section we examine the various methods which implement this technique and their relative improvements.

3.1 Coverage by Cellular Decomposition: Preliminary Techniques

3.1.1 The Trapezoidal decomposition

The trapezoidal decomposition technique (Latombe, 1991) [7] deals with the free space being divided into trapezoidal cells. These cells, having two parallel sides, can be covered by simple back and forth motions parallel to either side. Therefore coverage is ensured by visiting each cell in the adjacency graph.

A vertical line, known as a slice, sweeps from left to right through the environment. The environment is a bounded polygon filled with polygonal obstacles. Whenever a slice encounters a vertex of a
polygon, an event is said to occur. An event leads to a set of open and close operations on the cells leading to formation of new cells. The 3 types of events are – IN, OUT, MIDDLE. In an IN event, the current cell is closed and two new cells are opened (See Figure 3.a.), while in an OUT event, the current two cells are closed and a new cell is initiated. The third type of event, MIDDLE, results in a current cell being closed and a single new cell being opened. So by these 3 processes, cells are finished in construction and at the same time initialised in construction. As a result after the sweep is completed, it results in decomposition into trapezoidal cells.

Figure 4. (a) Trapezoidal decomposition of a field (b) Adjacency Graph of the decomposition
Figure 4.a shows the trapezoidal decomposition of a field into 11 cells, while Figure 4.b shows the adjacency graph which is translated into a TSP (Travelling Salesman Problem) and solved. The shortcoming of this method is that it requires far too much redundant back and forth motions to guarantee complete coverage. Since each vertex results in formation of new cells, the MIDDLE category of events results in redundant back and forth motions as shown in Figure 5.a.

Figure 5. (a) Trapezoidal decomposition – redundant back and forth motions at cell boundaries. (b) 3 cells merged as one – redundant movements removed.

On merging the cells into one as shown in Figure 5.b, the redundant movements are removed. In the following Figure 6, the field serves as a benchmark for comparison. Since the polygonal field has many vertices, the number of redundant movements increases linearly.
Figure 6. (a) Trapezoidal decomposition of test field (b) Distance traversed and traversal time for the field.

Figure 6.b shows the total distance and total time (which includes estimated turning time). The redundant swaths can be seen as two lines placed much closer than the track width.

3.1.2 The Boustrophedon Decomposition

The Boustrophedon decomposition [6] addresses the issue of redundant movements by merging together cells between successive IN and OUT movements. A slice sweeps over a bounded environment populated by polygonal obstacles. In the case of an IN event, the connectivity of the slice increases and the current cell is closed and two new cells are opened. In the case of an OUT event, connectivity decreases, two current cells are closed and a new one is opened.

The major difference is in the case of a MIDDLE event, no new cell is formed, but the current cell is updated in width. So cells are only created on change of connectivity, as a result no of cells are less than in the case of trapezoid decomposition.

The point which changes the connectivity of the cell is termed as a critical point. In implementation of this algorithm, the MIDDLE event is replaced by two new events called – FLOOR and CEILING. The FLOOR event corresponds to vertices on top of obstacles, while CEILING event corresponds to vertices that are at the bottom of obstacles. Both an IN event as well as an OUT event are associated with FLOOR and CEILING pointers. So for a given cell, CEILING and FLOOR pointers points to the top and bottom of a cell.

**Boustrophedon Algorithm:**

**Step 1:**

The leftmost cell is artificially opened and is assumed to be bounded by closing edges. The CEILING and FLOOR pointers point to these edges.
Step 2:
When an IN event is encountered, the slice intersects the CEILING and FLOOR, and these points are the last points of the set of CEILING and FLOOR edges respectively. Thus a cell is considered closed.
Two new cells are opened – a bottom one and a top one. FLOOR pointer for bottom cell is set as the FLOOR pointer to previously closed cell while the CEILING pointer of bottom cell is set as the CEILING pointer of the IN event.
Similarly, for the top cell, the FLOOR pointer is set to the FLOOR pointer for the IN event, while the CEILING pointer is set to the CEILING pointer of the previously closed cell.

Step 3:
When a FLOOR event occurs, the FLOOR pointer of the cell is updated. The set of FLOOR edges is updated to include the FLOOR edge for the event.

Step 4:
When a CEILING event occurs, the CEILING pointer of the cell is updated. The set of CEILING edges is updated to include the CEILING edge for the event.

Step 5:
When an OUT event occurs, the top and the bottom cells are to be closed. The CEILING list of edges for the top cell terminates with the intersection of the slice with the current CEILING edge. The FLOOR list of edges for the top cell ends with the event location.
The CEILING list of edges for bottom cell ends with the even location. The FLOOR list of edges for the bottom cell terminates with the intersection of the slice with the current FLOOR edge.
For the new cell, the FLOOR pointer is set to the floor pointer of the closed bottom cell. The CEILING pointer is set to the CEILING pointer of the closed top cell.

![Figure 8](image)
Figure 9. (a) The benchmark field with boustrophedon decomposition (b) Total distance and total traversal time for this decomposition.

As our benchmark field Figure 9.a shows, we can see the effect of merging where the colour of a region changes only when connectivity changes. Figure 9.b shows the results of this new decomposition. Figure 10. below shows head to head comparison between Trapezoidal decomposition and Boustrophedon decomposition. Results indicate due to drastic reduction in number of cells, distance traversed also decreases and hence total time for coverage improves.
<table>
<thead>
<tr>
<th>Cellular Decomposition</th>
<th>No of cells</th>
<th>Distance Traversed (units)</th>
<th>Percentage improvement in distance traversed</th>
<th>Total traversal time (seconds)</th>
<th>Percentage improvement in traversal time</th>
</tr>
</thead>
<tbody>
<tr>
<td>Trapezoidal Decomposition</td>
<td>29</td>
<td>10788</td>
<td>---</td>
<td>5755.2</td>
<td>---</td>
</tr>
<tr>
<td>Boustrophedon Decomposition</td>
<td>8</td>
<td>7591</td>
<td>29.63%</td>
<td>3932.4</td>
<td>31.67%</td>
</tr>
</tbody>
</table>

Figure 10. Head to head comparison between Trapezoidal and Boustrophedon decomposition

### 3.2 Coverage by Cellular Decomposition: Improved Version

The Split and Merge technique [16] was developed because the previous methods specified that there be a single driving direction. If a field is non-convex and has obstacles, these result in “bays”. As a result, a field can be split into sub-fields which are convex or near convex. To drive one such subfield, an optimal solution can be found. Even though the solution is not globally optimal, but suboptimal, it still provides a feasible concept of splitting a complex field into simple subfields which are then solved by back and forth swathing (boustrophedon).

#### Split and Merge Algorithm

**Step 1:**

For a given driving direction, the field is subjected to trapezoidal decomposition, with the slice direction equal to driving direction.

![Diagram](image)

Theta = 80 degrees

Figure 11. (a)

**Step 2:**

As many trapezoids as possible are merged, following two rules. Firstly, the trapezoids should have one common parallel edge. Secondly, the angle of the parallel line to the ending line should be near 90 degrees (a threshold value can be used). The second rule prevents steep turning angles at the edge of the field.
Step 3:

Once the merging process is complete for a particular direction, the best possible block for that direction must be chosen. This is done by selecting a cost function. This is chosen as follows:

\[
\text{Cost} = \alpha \cdot \frac{\text{Area of the block}}{\text{Total area remaining in field}} + \beta \cdot \frac{\text{Effective distance covered}}{\text{Moving time + Turning time}} + \frac{\text{Average Straightline Speed}}{}
\]

The effective distance per total time term ensures that the more efficient blocks are chosen to reduce the total time it takes to cover the field, while the normalized area term prevents long thin rectangular strips from being selected.

<table>
<thead>
<tr>
<th>Cell</th>
<th>Normalized area*bias</th>
<th>Normalized efficiency*bias</th>
<th>Cost Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>0.248</td>
<td>0.742</td>
<td>0.991</td>
</tr>
<tr>
<td>2</td>
<td>0.024</td>
<td>0.451</td>
<td>0.475</td>
</tr>
<tr>
<td>3</td>
<td>0.034</td>
<td>0.533</td>
<td>0.567</td>
</tr>
<tr>
<td>4</td>
<td>0.176</td>
<td>0.640</td>
<td>0.816</td>
</tr>
<tr>
<td><strong>Best Cell - 1</strong></td>
<td><strong>0.248</strong></td>
<td><strong>0.742</strong></td>
<td><strong>0.991</strong></td>
</tr>
</tbody>
</table>

Figure 11. (c)
Step 4:
Once the best cell for a particular direction are known, the best cell overall must be established. However, the change in cost function for the cells is not smooth. So the gradient method cannot be used for optimization.
A new heuristic is used as follows:

- Cost is first calculated in six directions: 0, 30, 60, 90, 120, and 150 deg.
- The three best directions are selected; others are dropped.
- The step size in the direction angle of search is halved.
- New search directions are added to both sides of the three best directions.
- Cost is calculated for directions that are not yet calculated.
- If the goal resolution is reached, exit, otherwise continue.

![Figure 11. (d)](image-url)
Step 5:
The selected block is removed from the rest of the field and the process is continued till the whole field is covered.

Figure 11. (e)

The following figure shows the Split and Merge method as applied to the benchmark field.
From the comparison we see that improvement isn’t very significant, however for larger areas the improvement becomes more apparent. Also to be noted that there is a better improvement over time as compared to improvement over distance traversed. This is because effectively area covered by both methods is same, but, split and merge method results in selection of an optimal driving direction which reduces the number of turns as compared to boustrophedon method. Visually looking at the two decompositions we can see while boustrophedon had slice directions of 0 degree, split and merge method has slice directions varying around 30 degree or 60 degree as it reduces the number of turns. For greater variation in slices, the difference becomes more apparent.

In the case shown below, in all 3 cases we see that distance covered is approximately same as redundant movement and overlap is very less in these cases. The striking difference is in the time taken, as the difference lies primarily in the turning time. When slice direction is 90 degrees, the number of turns is less as compared to the turning in the case of 0 degree slices. But the split and merge method has a time significantly less than both as it alternates between slice directions.
Figure 13. Demonstration of utility of Split and Merge method.
3.3 Coverage by Cellular Decomposition: Existing Problems

In the context of agricultural applications, the coverage problem still faces several real problems. Certain areas in the field appear, which have to be avoided, and preferably a route without re-planning is preferred. The headland areas are regions at the edges of the field which are used for turning. The headland turns are usually calculated before hand with some safety headland width assumption. The shape and boundaries of “no-go” areas often determine shape of turning at the end and hence determine the minimum possible headland width. The field boundaries also influence following the curved paths instead of being restricted to simple straight line motions.

3.3.1 Re-planning due to obstacles

Often during agricultural operations with tractors, people come across plots of land which have to be avoided or left out. This problem is present in cases of wetland, under drainage or appearances of embankments.

A wetland is an area of land whose soil is saturated with moisture either permanently or seasonally. Such areas may also be covered partially or completely by shallow pools of water. Wetlands include swamps, marshes, and bogs, among others. Wetlands can be categorized to 3 cases – moist land, waterlogged lands, and immerged lands.

In the case of moist lands, the water table fluctuates from the surface down to 30 to 40 cm in the ground. The ground may occasionally be flooded during heavy rains. So during drier parts of the month, we may want routes to go over such areas, while during moist conditions, such areas are avoided. In case of permanently waterlogged lands, the area may be completely avoided.

In other cases, prohibited regions may appear due to height variations or under drainage. In such cases either a certain driving direction is prohibited or entirely an area must be missed.

Since we are primarily focussing on offline computations, we analyse with less emphasis on time of computation and more emphasis on the output quality and nature. Appearance of a prohibited area results in re-planning with the current schematic. Re-planning works in certain cases, while fails in other cases.

The major problem of re-planning is that since new obstacles change the connectivity of the free-space, the sub-optimal nature of the solution is lost. The Split and Merge method deals with segments, which are no longer connected with the introduction of obstacles. New sub-optimal blocks emerge, which in the long run affect the globally optimal solution because the number of cells drastically increases and keeps adding to the traversal time. Figure 14. shows the complete change in route in case of large number of obstacles.

Figure 14. A simple field with introduction of small obstacles.
Figure 15. Effect on output of route planning with increasing number of small obstacles:

- **(a)**
  - No of obstacles: 0
  - No of cells: 4
  - Original area: 70435.5
  - Distance traversed: 14727
  - Coverage time: 6930.8

- **(b)**
  - No of obstacles: 1
  - No of cells: 9
  - Original area: 70327
  - Distance traversed: 15841
  - Coverage time: 7552.4

- **(c)**
  - No of obstacles: 2
  - No of cells: 11
  - Original area: 70142
  - Distance traversed: 16069
  - Coverage time: 7771.6

- **(d)**
  - No of obstacles: 3
  - No of cells: 12
  - Original area: 70004.5
  - Distance traversed: 16267
  - Coverage time: 7882.8

- **(e)**
  - No of obstacles: 4
  - No of cells: 16
  - Original area: 69849.5
  - Distance traversed: 15543
  - Coverage time: 7753.2

Note: The figures show the changes in the route planning with increasing numbers of small obstacles.
Figure 16. (a) Increase in no of cells in decomposition with no of obstacles

Figure 16. (b) Effect of increase in obstacles on area of field, distance traversed for coverage, time taken for coverage.
Figure 16.(a) shows that the no of cells increase dramatically on introduction of the first obstacle, and
then continue increasing on further addition in a non-smooth manner. Figure x shows that as obstacles are
being added; the change in cellular decomposition is certainly not local. A previously optimal cell, whose
area is infected with a relatively small prohibited zone, no longer is a cell. The slice function now splits the
region into several sub-cells which may not be as “good” as other cells which were previously ignored. This is the significant change that we see in the transition of no obstacle to one obstacle.
The first obstacle destroys the connectivity of the green cell shown in Figure 15.(a). New suboptimal
cells are spawned in Figure 15.(b) which change the whole global distribution of cells. Transit from 1
obstacle to 2 obstacles changes the major cell driving direction by over 45 degrees. Figure 15.(d) is
not so different from Figure 15.(c) since the obstacle doesn’t happen to create or ruin any optimal
cells. Figure 15.(d) to Figure 15.(e) again emphasises the “chain” effect of change in the cellular
distributions. Since the obstacle hinders one sub-optimal region, that in turn isn’t able to produce a
better cell later on, and so on and so forth.
When we shift out focus to the statistics shown in Figure 16.(b), it’s surprising to see how the change
affects the performance values. Keeping in mind that area to be covered decreases almost uniformly
as obstacles of almost equal size appear, it is expected that ideally distance traversed and time taken
also should decrease. However we have to keep some margin for the fact that an obstacle should
create some hindrance to the flow of movement and so coverage time may increase to some extent.
The effective distance covered takes a leap with the introduction of an obstacle because the no of cells
increase and many more swaths are required to fill up the area around the obstacle. The distance
continues increasing with increasing no of cells, but after a point of time, since effectively obstacles
are taking up space, the effective distance moved starts to decrease as expected.
The fluctuation of time is different although it is very similar in nature to distance. Since the time is a
summation of traversal time (proportional to distance) and turning time, it follows the drastic increase
in distance. As obstacles increases, turning however keeps on increasing while distance naturally falls.
So the total time saturates for a longer time before starting to dip down.
As a result of these characteristics, there is a need for several performance enhancements. The
changes in route have to be made localised for “small” obstacles; the total time curve has to drop at a
faster rate. The motive behind this is that for objects of size smaller than a given threshold, the
globally planned route should still hold as the optimal route, and new routes can be derived by local
modifications. These family of solutions will not increase the effective distance, but rather try to
minimise the avoidance time incurred by the obstacle.
3.3.2 Limitations due to straight line motions

Till now, decompositions are done keeping in mind that cells are to be covered by simple straight line back and forth motion. This is done because modelling curves to find more optimal solutions exponentially increases the complexity of the solution space. However, there are primarily two cases that require some implementation of bent trajectories. The first case is when obstacles of a fixed size appear in the free space. For obstacles less than a threshold size, the local online planner can avoid an area. Also for obstacles above a threshold size, the free space is significantly divided to several free spaces which tend to yield desires results using the given algorithm. Obstacles between these two threshold sizes result in straight lines trying to find a best fit solution about the obstacle.

![Figure 17. Local view of routes around obstacles](image)

As we can see above, the intuitive solution of going around the area isn’t arrived at, but a rather a complicated series of cells are generated.

The second case is that of covering a field effectively. Farmers intuitively follow the edge of their field instead of forcing straight-line motion because of better results. This could result in rounding the field as a spiral.

3.3.3 Pre-determining headlands

Headland turning has been solved [17] by Optimal Control Theory. Solutions require headland width and headland angle as input and calculates the optimal trajectory to turn from one path to another. Headlands are areas of the field which are buffer spaces for turning. The current structure requires the headland width to be specified by the farmer. Usually fields are accompanied by “no-go” areas, like lake embankments, wetlands, which have to be taken into consideration. For the farmer to take these factors, as well as terrain features and layout a headland area is difficult. An automated process of defining a variable headland area must be determined.
4. Coverage by trajectory bending

In the previous section, we presented problems that were occurring due to rigid straight line representation, be it inability to treat obstacles as a local problem, or inability to represent “driving around” or “driving along” prohibited areas. Figure x. shows the modified solution presented in the paper. The obstacles are driven around, rather than complete re-planning. This ensures retention of the original route with local modifications. In the following sections we will discuss the methods used, the algorithm and the improved results.

Figure 18. Solution of route planning of a farm with appearance of new obstacles and no-go areas.
4.1 Representation of tracks: B-splines

For a collection of \( n+1 \) control points \( \{Q_i\}_{i=0}^{n} \), a B-spline curve is given by,

\[
X(t) = \sum_{i=0}^{n} N_{i,d}(t)Q_i
\]

(1)

The dimension of all the control points must be equal. The degree of a B-spline curve, \( d \), must satisfy \( 1 \leq d \leq n \). The functions \( N_{i,d}(t) \) are the B-spline basis functions, which are defined recursively and require a set of non-decreasing scalars \( t_i \) for \( 0 \leq i \leq n + d + 1 \). Here, \( t_i \) is known as a knot, belonging to a set of knot vectors. The set of basis functions that start the definition is:

\[
N_{i,0}(t) = \begin{cases} 
1, & t_i \leq t \leq t_{i+1} \\
0, & \text{otherwise}
\end{cases}
\]

(2)

for \( 0 \leq i \leq n + d \). The recursion itself is

\[
N_{i,j}(t) = \frac{t - t_i}{t_{i+j} - t_i} N_{i,j-1}(t) + \frac{t_{i+j+1} - t}{t_{i+j+1} - t_{i+1}} N_{i+1,j-1}(t)
\]

(3)

for \( 1 \leq j \leq d \) and \( 0 \leq i \leq n + d - j \).

The important property of B-splines is that \( N_{i,j}(t) \) is nonzero only over \( [t_i, t_{i+j+1}] \). Thus locally the curve is influenced by only a small number of control points, a property called local control. Local control allows modification only under “critical areas” and thus saves on a lot of computation time.

4.2 Deformation of tracks: Energy of tracks

Each track is represented as a B-spline. Since a track is represented as a straight line segment in this case, the control points lie directly on the line spread uniformly. The splines in question are clamped; the first and last knots are of \( d+1 \) multiplicity.

As was previously discussed, our solution is based on deforming these tracks to drive around obstacles like a human driver would do. Deformation is brought about by forces. The forces acting on these tracks are the following:

1. Forces due to obstacles.
2. Forces due to curvature.
3. Forces due to interaction between tracks.

These forces result in each track having energy. These energies later on determine deformation traits.

4.2.1 Energies due to obstacles

Each obstacle generates a vector field. These vector fields interact with the tracks and create energy. The vector fields are two fold in motive – to repulse parts of track lying inside an obstacle outwards and to force track lying outside it to follow the boundary edges. This results in tracks trying come as close to the obstacle as possible.

\[
\overrightarrow{F_{\text{inside}} = K_{in} * e^{-w_{obs} * d} \hat{n}}
\]

(4)

where \( F_{\text{inside}} \) is the force inside an obstacle, \( d \) is the distance from the projected centre of the obstacle, \( \hat{n} \) is the direction perpendicular to the track direction.
\[
\overline{F}_{\text{outside}} = \sum K_{\text{out}} * l_{\text{side}} * e^{-w_{ob} * d} \hat{n}
\]  
(5)

where \( F_{\text{outside}} \) is the force outside the obstacle, \( l_{\text{side}} \) is the length of side of obstacle, \( d \) is distance from the edge of obstacle, \( \hat{n} \) is along the obstacle edge either clockwise or anticlockwise depending on which direction points away from centre.

Figure 19. Potential field generated by an obstacle.

Once the forces are generated, the energy is to be calculated. Keeping in mind that we want the tracks to align according to the deforming force, the perpendicular component of the force is responsible for the energy content of the track.

\[
E_{\text{obstacle}} = \int K * F_{\text{obstacle}} (t) \sin(\theta_{\text{diff}} (t)) \, dt
\]  
(6)

where \( \theta_{\text{diff}} (t) \) is the difference in angle between the force and the track.

4.2.2 Energy due to curvature

Each track must respect the curvature constraint of the vehicle, i.e., the minimum radius of curvature, at all points along it. As a result, a cost function for a particular value of curvature is to be set. Since the curvature is never to be allowed to reach maximum value, cost function profile can be chosen as shown.
So the energy due to the curvature is as follows

\[ E_{\text{curve}} = \int K \cdot F_{\text{curve}}(t) dt \]  

(7)

where \( F_{\text{curve}}(t) \) is the cost function due to curvature.

4.2.3 Energy due to track interaction

A model must be so built so that the repulsion due to obstacles takes place quickly. This can be done by introducing a spring model connecting adjacent tracks. By introducing springs, we allow tracks to push each other out of the obstacles way. Also to restrict overlap of tracks, springs become useful.

Figure 21. Spring model of tracks

The cost function due to perpendicular distances between tracks is shown
Figure 22. Cost function profile due to perpendicular distance between tracks.

The energy due to the track spring connection is as follows

\[ E_{spring} = \sum \int K \cdot F_{spring}(d_i(t)) \, dt \]

where \( F_{spring}(d_i(t)) \) is the cost function due to spring tension between track ‘i’ and the current spring, \( (d_i(t)) \) is the perpendicular distance between the two springs at point \( t \).

4.3 Algorithm for deformation

Step 1:
A search is done among all the tracks to locate the concerning cells and tracks which pass through the obstacle.

Step 2:
Once a track is selected, knots are added at intervals.
- The first and last knot is added at locations where the obstacle potential field rises above a threshold and falls below a threshold respectively. This is done to subject deformation to only the concerned part of the track.
- \( d \) knots, where \( d \) is the order of the spline, are added at a total distance span of \( \pi \cdot R_{\text{min}} \). This is provided to allow for the curve to change its initial heading from any angle to any other feasible angle within a span of the minimum turning radius.
- \( m \) knots are added at equal intervals till the end
- \( d \) knots are added after the end at a distance span of \( \pi \cdot R_{\text{min}} \) to restore the original heading of the curve and to prevent knots beyond this point having a nonzero basis function value in the operated area.

Step 3:
Energy associated with each track is calculated using the equations as mentioned above.
Step 4:
The force on the control points is found by gradient method. Only the control points whose associated basis functions are non-zero over the region is to be moved.

\[ F_{x,P_i} = - \frac{\partial}{\partial x_{P_i}} (E_{total}) = - \frac{\partial}{\partial x_{P_i}} (E_{total,P_i}) \]

(9)

\[ F_{y,P_i} = - \frac{\partial}{\partial y_{P_i}} (E_{total}) = - \frac{\partial}{\partial y_{P_i}} (E_{total,P_i}) \]

(10)

Figure 23. Schematic of deformation

Figure 24. Instance of local modification property.
Figure 25. Effect of no of obstacles on modified route plan
From Figure 26, we see that the characteristics of curves differ from that of Figure 16.(b). The area to be covered being the standard measure, we see the distance travelled to complete the field increases slightly with increasing obstacles, as well as time to cover continues increasing. As obstacle numbers increase, unlike previously instead of terminating paths existing paths are bent around obstacles and hence increased in length. This factor is responsible for the increase in distance. As far as travel time goes, since no new cells are being formed, no extra turns at the end of the field are required. As a result, the time which is a sum of travel time as well as turning time increases only due to increase in travel time which is relatively less. The advantage of this method is irrespective of the number of small obstacles, the traversal time doesn't increase significantly. As seen from Figure 25.(a-e), the overall form of the route remains unaffected while the local area around the obstacle changes.

### 4.4 Head to head comparison

<table>
<thead>
<tr>
<th>Area</th>
<th>Distance travelled</th>
<th>Total Time</th>
</tr>
</thead>
<tbody>
<tr>
<td>7202.2</td>
<td>7205.2</td>
<td>7209.9</td>
</tr>
<tr>
<td>7214.3</td>
<td>7218.8</td>
<td></td>
</tr>
<tr>
<td>14136</td>
<td>14144</td>
<td>14155</td>
</tr>
<tr>
<td>14167</td>
<td>14177</td>
<td>14185</td>
</tr>
<tr>
<td>70435.5</td>
<td>70327</td>
<td>70045.5</td>
</tr>
<tr>
<td>70142</td>
<td>70004.5</td>
<td>69849.5</td>
</tr>
</tbody>
</table>

![Figure 26](image_url)
Figure 27. (a) Two different route plans, one with modification of route, other with re-planning

<table>
<thead>
<tr>
<th>Method</th>
<th>Re-Planning</th>
<th>Bending Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance Travelled</td>
<td>4689.49 m</td>
<td>5014.65 m</td>
</tr>
<tr>
<td>Moving Time</td>
<td>4689.49 s</td>
<td>5014.65 s</td>
</tr>
<tr>
<td>Turning Time</td>
<td>2594.73 s</td>
<td>1569.66 s</td>
</tr>
<tr>
<td>Total Time</td>
<td>7284.22 s</td>
<td>6584.31 s</td>
</tr>
</tbody>
</table>

Moving speed = 1m/s, Turning speed = 0.5 m/s, Improvement = 9.6%

Figure 27.(b) Statistics between the two methods.

Figure 27.(b) clearly concludes that the merits of route-modification as it results in almost a 10% improvement in coverage time. The statistics show that this method is clearly a trade-off between covering more distance to compensate for having to turn about its radius at the edge of the obstacle.

4.5 Failures of the method

From the statistics of figure 26, the question that comes to mind is that as the area to be covered continues to decrease, the distance covered will continue to increase as well as the time. Referring to the re-planning statistics in figure 26, we see that the curve for time starts to dip-down eventually because actual distance traversed is less. So one of the obvious flaws that appear is that as the total area covered by all obstacles goes above a certain percentage, the tractor would spend more time overlapping previous tracks just to avoid the obstacle. In such cases, complete re-planning will give a drastically modified, yet a more optimal route.

Another failure is if the size of the obstacle is above a threshold. In such cases, the overlap for the single obstacle becomes more time consuming than turning at the edge of the obstacle. So there is an
The optimum size of the obstacle for which route modification holds. From the statistics below, we will attempt to frame an estimate in terms of (width of obstacle/track width).

<table>
<thead>
<tr>
<th>Method</th>
<th>Re-Planning</th>
<th>Bending Paths</th>
</tr>
</thead>
<tbody>
<tr>
<td>Distance Travelled</td>
<td>4511.78 m</td>
<td>5001.15 m</td>
</tr>
<tr>
<td>Moving Time</td>
<td>9023.56 s</td>
<td>10002.3 s</td>
</tr>
<tr>
<td>Turning Time</td>
<td>2152.42 s</td>
<td>1806.05 s</td>
</tr>
<tr>
<td>Total Time</td>
<td>11175.98 s</td>
<td>11808.43 s</td>
</tr>
</tbody>
</table>

Result is **5.659%** worse (Average moving speed = 0.5 m/s)

Figure 28. Effect of a large obstacle on both methods
Re-planning for a new route

<table>
<thead>
<tr>
<th>Size of obstacles</th>
<th>1 obstacle</th>
<th>2 obstacles</th>
<th>3 obstacles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small ~ 2-3 w</td>
<td>7552.4 s</td>
<td>7771.6 s</td>
<td>7882.8 s</td>
</tr>
<tr>
<td>Medium ~ 8-9 w</td>
<td>8007 s</td>
<td>8106.4 s</td>
<td>8302.1 s</td>
</tr>
<tr>
<td>Large ~16-17w</td>
<td>8090 s</td>
<td>7816 s</td>
<td>7623 s</td>
</tr>
</tbody>
</table>

where w is (width/track_width).

Figure 29.(a) Traversal times for completely re-planned routes

Bending old route to form new route

<table>
<thead>
<tr>
<th>Size of obstacles</th>
<th>1 obstacle</th>
<th>2 obstacles</th>
<th>3 obstacles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small ~ 2-3 w</td>
<td>7205.2 s</td>
<td>7209.9 s</td>
<td>7214.3 s</td>
</tr>
<tr>
<td>Medium ~ 8-9 w</td>
<td>7215.6 s</td>
<td>7230.5 s</td>
<td>7257.7 s</td>
</tr>
<tr>
<td>Large ~16-17w</td>
<td>8543.04 s</td>
<td>9775.9 s</td>
<td>11208.6 s</td>
</tr>
</tbody>
</table>

where w is (width/track_width)

Figure 29.(b) Traversal times for modification of route

Improvement in route modification

<table>
<thead>
<tr>
<th>Size of obstacles</th>
<th>1 obstacle</th>
<th>2 obstacles</th>
<th>3 obstacles</th>
</tr>
</thead>
<tbody>
<tr>
<td>Small ~ 2-3 w</td>
<td>4.6 %</td>
<td>7.2%</td>
<td>8.4%</td>
</tr>
<tr>
<td>Medium ~ 8-9 w</td>
<td>9.9%</td>
<td>10.8%</td>
<td>12.58%</td>
</tr>
<tr>
<td>Large ~16-17w</td>
<td>-5.6%</td>
<td>-25%</td>
<td>-47%</td>
</tr>
</tbody>
</table>

Figure 29(c) Percentage improvement by route modification
On analysing the statistics we see that the most decisive factor in the failure of the method is the area covered by obstacles. Once this exceeds a threshold, the amount of redundancy in We try to approximate an optimal scenario in which the route modification will work best by decoupling the dual effects of increase in obstacle width and the number of discrete obstacles.

The approximate optimal width is determined from the graph shown below in Figure 30.

![Figure 31. Plot of percentage improvement v/s obstacle width along with quadratic fit](image)

Since a quadratic estimation is assumed for simplicity, the peak appears at 7.84, i.e., the medium obstacle is 7.84 in width as compared to track_width.

![Figure 32. Plot of improvement with no of obstacles.](image)

For medium sized obstacles the peak occurs around 3-4 obstacles.
5. Headland turning

The part of the field allocated for turning the vehicle is referred to as headlands. These areas are usually covered before hand by driving along the headland area. Usually if the vehicle is covering an area using boustrophedon method of back and forth swaths, it needs to turn from one track to another. Hence turning is at the following areas –
1. Field edges which are unrestricted
2. Field edges which are restricted, i.e, no-go areas.
3. At the edge of an obstacle

5.1 Solution framework: Optimal Control Problem

The problem of a vehicle with finite radius of turn, turning from one track to another, with a change of heading of one angle to another, in as short a time as possible is not trivial. This problem has to be framed as an Optimal Control Problem (OCP) [17]

5.1.1 The Vehicle Model

\[
\begin{bmatrix}
\dot{x} \\
\dot{y} \\
\dot{\theta} \\
\dot{\alpha}
\end{bmatrix} =
\begin{bmatrix}
v \cos \theta \\
v \sin \theta \\
v L \tan \alpha \\
0
\end{bmatrix} + \begin{bmatrix}
0 \\
0 \\
0 \\
1
\end{bmatrix} u
\]

Where,
\(x\) and \(y\) are the co-ordinates of the rear-axle centre
\(\theta\) is the heading angle
\(\alpha\) is the steering angle
\(u\) is the steering rate
\(v\) is the velocity
\(L\) is the wheel base

5.2 The Optimal Control Problem

The optimal control problem generates a set of states which satisfy a set of constraints while minimising the cost function.
The cost function is as follows:
\[
\min J(u) = \int_{t_0}^{t_f} F(x(t), u(t), t) dt
\]
The differential equation is as follows:
\[
\dot{x}(t) = f(x(t), u(t), t), t_0 \leq t \leq t_f
\]
The path constraint function is as follows:
\[
C(x(t), u(t), t) \leq 0, \quad t_0 \leq t \leq t_f
\]
The boundary constraint function is as follows:
\[
E(x(t_0), x(t_f)) \leq 0
\]
Where
\( x \) is the state vector
\( t_0 \) is the initial time
\( t_f \) is the final time

5.2 Solution for different cases of headlands

5.2.1 Case I: No restriction on headlands

The above trajectory generated to join the two lines respecting the bounds of curvature as well as trying to minimise overall time, is what we require from the OCP.

We frame a general case of this problem under no restrictions.

The boundary conditions are as follows:

\[
\begin{align*}
x(t_i) &= 0 \\
x(t_f) &= d \cot \theta \\
y(t_i) &= 0 \\
y(t_f) &= d \\
\theta(t_i) &= 0 \\
\theta(t_f) &= \pi \\
\alpha(t_i) &= 0 \\
\alpha(t_f) &= 0
\end{align*}
\]
There are no path constraints for this case. The cost function is as follows:

\[ J = Kt_f + a^2(t_f) + \int_{t_i}^{t_f} \frac{r}{\alpha_{max}^2 - \alpha^2} dt \]

(16)

Figure 35. The output trajectory under given conditions

5.2.2 Case II: Restricted turning

The case can be represented by the figure below

Figure 36. Schematic for no-go area case
In such a case, solutions have been derived for a given headland width. However, we consider cases where we do not expect the farmer to provide a headland width from the start. Headland width is a function of curvature constraints and headland angle, and thus we design the OCP in such a fashion that the solution appears. We add a path constraint so that no part of the trajectory spills over to the “no-go” area.

The boundary conditions are as follows:

\[
\begin{align*}
    x(t_i) &\leq 0 \\
    x(t_f) &\leq d \cot \theta \\
    y(t_i) &= 0 \\
    y(t_f) &= d \\
    \theta(t_i) &= 0 \\
    \theta(t_f) &= \pi \\
    \alpha(t_i) &= 0 \\
    \alpha(t_f) &= 0
\end{align*}
\]

Path constraints are as follows:

\[
y(t) - x(t) \cdot \tan \varphi \leq 0
\]

The cost function is as follows:

\[
J = x(t_i)^2 + x(t_f)^2 + K t_f + \alpha^2(t_f) + \int_{t_i}^{t_f} \frac{r}{\alpha_{max}^2 - \alpha^2} \, dt
\]

(17)

Figure 37. The generated trajectory for the case of “no go” area.
5.2.3 Case III: Obstacle turning

Obstacles are treated as no go areas and the situation is framed such that turns do not go into the restricted space.

5.3 Fitting turns to specific boundaries

For a given headland angle, a typical OCP computation by GPOPS software takes 5-10 minutes on Matlab 7.1. Thus calculation of trajectories for every track by OCP will take excessive amounts of time. As a result, we use the method of look up tables. For a given machine, the track width will be constant. We pre-compute all sets of left and right turns with headland angle varying from 10 to 170 degrees with 10 degree intervals. During route planning, the angle between tracks is found out and then it’s rounded off to the closest interval. Another issue is that sometimes the “no go” areas may not match with our specifications, as shown in the examples below

![Diagram](image1)

Figure 38. Violation of the “no go” solution framework; the lines in blue is the trajectory to be followed, the yellow area is restricted and the red area is the overlapping area.
As a result, the trajectory is modified to fit with the specific shape of the no go areas. The set of the points generated by OCP is converted to a B-spline with much fewer control points. This spline is now subjected to the “bending” as mentioned in section 4. The no go areas act as repulsive units this time and push the curve outward.

As a result of the forces the control points are modified. Only the first and last point is slid along the curve so that it remains on track.

Figure 39.(a) Before and after effects of trajectory modification

Figure 39.(b) Example of final turn plan
6. Optimisation of Cellular Traversal

6.1 Need for optimisation

Till now the focus of this work has been that the cellular traversal are boustrophedon, i.e. back and forth. This was done because it was a norm that farmers follow and because of the straight line approximation, the dimension of the problem does not explode. However, there is another norm that we consider in this work. Farmers also tend to cover simple convex fields by spiralling in from the edges. This is done because of primarily two reasons. Firstly, turning for the vehicle 180 degrees is a time consuming process. Also while spiralling in the farmers tend to collect the produce by throwing it inside the spiral and this helps the harvesting process. The figure below shows an example.

![A farmed field which appears to have been spirally traversed.](image)

Of course the method refers to a simplistic field. When non-convex fields with bays and holes appear, simply following the edge will not do.

Solutions using Model Predictive Control [16] have been used. The shortcoming of this method is that it uses a receding horizon planning, and hence does not take the features of the whole field into consideration. Moreover, the method is very much computationally expensive.

6.2 Optimisation at the cellular level

We consider the case of a circular shaped cell. Mathematically for a perfect circle, we can prove that spiralling in to the centre is faster than covering it in any direction by back and forth traversals. In such a case we have to make some naive relations between steering angle and velocity, as well as negligible overlaps incurred during the spiralling process.

Similarly for a rectangular shaped cell, using similar naive assumptions, we can prove that spiralling in will take more time than back and forth motions along longer edge.

For back and forth motions for a rectangular cell of length \( l \) and breadth \( b \), and track width \( d \), we assume that turn time for any turns with angle greater than 90 degree is \( t_{\text{turn}} \) and velocity is \( v \). Then,

\[
t_{\text{move\_bf}} = (b/d) \times (l/v)
\]
t_turn_bf = (b/d)*t_turn;

For spiralling in motion,

t_move_sp = t_move_bf = (b*l)/(d*v)

t_turn_sp = no_turns*t_turn
       = (b/2d)*(4)*t_turn
       = (2*b/d)*t_turn

Thus we see that spiralling in time is twice as much as back and forth motion. In this case, back and forth motion is faster.

Consider the case of a circular cell of radius r. For back and forth motion,

\[ t_{move\_bf} = \frac{\text{area}}{\text{track\_width}} \]
\[ t_{turn\_bf} = \text{no\_of\_turns} \times t_{turn} \]
       = (2*r/d)*t_turn

For spiralling in motion

\[ t_{move\_sp} = t_{move\_bf} \]
\[ t_{turn\_sp} \sim 0 \]

The turning time is assumed to be near zero by excluding shifting of tracks and decreasing radius of curvature.

6.3 Algorithm

The algorithm applied tries to modify the split and merge method to also include the possibility of spiralling.

Step 1:
The best cell is chosen by the cost metric of “split and merge” method.

Step 2:
The normal traversal time is calculated. Then the polygon is offset by one track width and the remaining cell is checked by normal back and forth traversal.

Step 3:
Step 2 is repeated till offsetting not possible. Then the best intermediate time is taken and the cell is traversed in such fashion.

Step 4:
“Split and merge” is applied to remaining parts of the field.
Even though there is no concrete mathematical conclusion that links the shape of the cells to the characteristic of the output, some intuitive conclusions can be drawn from it. Asides from the fact that cells with smoother boundaries, “circular”, are preferably spiralled in because of the near zero turning time. On the other hand more acute angled shapes prefer back and forth motion rather than the difficulty of following an edge. Also, the spiralling stops after sometime and the remainder is covered by back and forth movement. This is because, after offsetting enough, even smooth edges become acute and turning time rises so much so that back and forth motion is faster in finish off the cell.

6.4 Results

The final result of the method guarantees a better or equal approach than “split and merge” method. This is because at the optimisation level, the worst case result is the output of the “split and merge” decomposition. Since the time consuming process in this is cellular decomposition, the spiral method does not add any extra computation there. Once a cell is determined, then the check is repeated for all offsetting. Hence the order of complexity isn’t increased much.

Figure 41. Some examples of the optimal traversal method of cells.
Figure 42. Results of applying the spiral algorithm
7. Conclusion

In this paper we created a scheme for defining the scenario of an agricultural plot and solved it keeping three important considerations in mind –

i. Temporary obstacles

ii. “No go” areas

iii. Curvature constraints of the robot

Considering the case of obstacles, a dual scheme was presented. For a certain threshold size and no of obstacles as determined in the paper, we presented a method of modifying existing routes to go around obstacles. Exceeding that threshold, the standard norm of re-planning is to be applied. This scheme revolved around the concept of local modification of curves retaining the global structure of the route.

The “no go” areas were handled as areas lying outside the field but not accessible. These areas demanded that the robot come as close as it can before it is forced to turn. The OCP was modified to solve this issue. Then the curves were expressed as splines and further bent to fine tune to fit perfectly against the edge.

The curvature constraints had to be taken into account both in the case of bending a route as well as turning in the headlands. This paper explicitly considers the case of a tractor arrangement where radius of turn is finite and large.

Besides the framework, the paper also suggests a grass-root change. A trade-off between partial spiral motion and partial back and forth motion is done to guarantee better or equal coverage time than provided by conventional “split and merge” method.

The future aspects of the work are:

1. No go areas should add a bias for spiral motion in those boundaries, as compared to approaching as close as possible. Boundaries which demand motion along the edges will in turn bias the cellular decomposition process.

2. Currently cellular decomposition is based on “split and merge” method. However, these may not be the most optimal cellular decomposition for spiral motion.

3. The spline modification can be treated as a snake spline interpolation. This will however push the track energies to very large values since the entire curve approaches the curvature limit. As a result by relaxing the track energy function, more optimal curves can be calculated.

4. This also provides a very good framework for linking with online planners and integrating dynamic characteristic.

5. The concept of splitting of splines hasn’t been considered. Although it is intuitive, the mathematical framework of such an operation hasn’t been thought of.
8. References