PREPRINT SUBMITTED TO JOURNAL OF FIELD ROBOTICS

Complete Coverage Path Planning for wheeled agricultural robots

Danial Pour Arab[1,2*] | **Matthias Spisser**[2*] | **Caroline Essert**[1*]

[1]ICube, Université de Strasbourg, CNRS (UMR 7357), 300 Bd Sébastien Brant, 67400 Illkirch, France

[2]T&S - Technology and Strategy Strasbourg, 4 Rue de Dublin, 67300 Schiltigheim, France

Correspondence
Danial Pour Arab, ICube, Université de Strasbourg, CNRS (UMR 7357), 300 Bd Sébastien Brant, 67400 Illkirch, France
Email: pourarab@etu.unistra.fr

Funding information
This work was funded by research grant CIFRE 2019/1084 from Technology & Strategy Group (T&S) and the French National Association for Research and Technology (ANRT)

In the agricultural industry, an evolutionary effort has been made over the last two decades to achieve precise autonomous systems to perform typical in-field tasks including harvesting, mowing, and spraying. One of the main objectives of an autonomous system in agriculture is to improve the efficiency while reducing the environmental impact and cost. Due to the nature of these operations, complete coverage path planning approaches play an essential role to find an optimal path which covers the entire field while taking into account land topography, operation requirements and robot characteristics.

The aim of this paper is to propose a complete coverage path planning approach defining the optimal movements of mobile robots over an agricultural field. First, a method based on tree exploration is proposed to find all potential solutions satisfying some predefined constraints. Second, a Similarity check and selection of optimal solutions method is proposed to eliminate similar solutions and find the best solutions. The optimization goals are to maximize the coverage area and to minimize overlaps, non-working path length and overall travel time.

In order to explore a wide range of possible solutions, our approach is able to consider multiple entrances for the robot. For fields with a complex shape, different dividing lines to split it into simple polygons are also considered. Our approach also computes the headland zones and covers them automatically which leads to a high coverage rate of the field.

KEYWORDS
Complete Coverage Path Planning, Precision Agriculture, Autonomous Agriculture, Vehicle Routing Problem, Wheeled robots, Path Planning, Route Planning

[*]Equally contributing authors.

1 | INTRODUCTION

Robots are becoming more prevalent in our daily lives. Increasing the efficiency and reducing the costs makes them both popular and trendy in many industries. They can perform a wide range of specialized and often risky tasks with a remarkably high precision compared to humans. This transformation touches every domain, including agriculture.

On the one hand, a growing population necessitates more food and agricultural products (Marques et al. (2019)), yet agriculture is also known for polluting the air and increasing mortality rates (Giannakis et al. (2019)). An abuse of pesticides and herbicides also has a substantial impact on the environment, animals, plants, and humans, necessitating the development of some practical solutions (Meftaul et al. (2020)). In this situation, the role of a precise autonomous system capable of optimizing the cost and efficiency of field operations is more important than ever.

Finding a proper path that covers the entire field is a critical challenge for the majority of agricultural operations such as tillage, seeding, harvesting and pulverization. This challenge is commonly referred to as *Complete Coverage Path Planning* (CCPP). In general, the result of CCPP on a field is a set of back and forth trajectories connected by smooth and continuous half-turns at areas along the edges of the field called headlands. The spacing between these trajectories depend on the width of the implement connected to the robot. This spacing is generally referred to as *working width*.

In order to perform CCPP on a field and to generate the best possible path, it is important to take into account the topographical and geometrical characteristics of the field, the robot features and capabilities, as well as those of its implements. The quality of CCPP outcome is generally ensured by satisfying and optimizing a set of constraints. These constraints can either be hard constraints, where the conditions must be satisfied, or soft constraints, which have some variable values that are optimized via a cost or reward function. In general the main objectives of constraints for most of operations is to maximize the productivity, minimize the cost and environmental damages such as soil compaction and erosion.

However, constraints vary depending on the operation and the kind of machinery needed for that particular operation. For instance, in a seeding operation the implement is in contact with the ground when it is activated, therefore it can remain activated only in back and forth trajectories or slight turns. During half-turns it must be deactivated and lifted to prevent machinery damages. This constraint, however, is completely irrelevant for pulverization where the implement is not in contact with the ground. In both cases, the distance traveled while the implement is activated is referred to as *working path length*, otherwise it is referred to as *non-working path length*.

In addition to a wide range of constraints that may differ for each operation, the efficiency of some operations may depend on the result of other operations. For instance, the path for pulverization is predefined based on a tramline farming system which is usually done during tillage. Permanent parallel wheel tracks (tramlines) are created within the field area in order to eliminate soil compaction from the wheels within the cropped area Bochtis et al. (2010a). In this paper, we propose a generic solution based on a novel CCPP approach for operations in which the implement is in contact with the ground when it is activated. The result of our approach is a sequence of moves that covers the field and the headlands at best.

The remainder of this paper is organized as follows: related works are described in Section 2. The objectives and main contributions of this paper are presented in Section 3. In Section 4, our novel approach is described in detail. The results of an experimental study on real fields of various shapes and sizes, as well as a comparison to ground truth, are reported in Section 5. A discussion follows, underlining the importance and effectiveness of the proposed approach. Finally, Section 6 brings this study to a conclusion and provides new perspectives towards future works.

2 | RELATED WORKS

To address this difficult problem and its numerous constraints, most of the time the CCPP problem is addressed in the literature as two distinct tasks. The first task is to generate a set of parallel back and forth trajectories based on field data. The second task is to connect the parallel tracks to form an optimal sequence of trajectories connected by half-turns, and allocate them to a single or multiple robots. The first task is usually known as *Coverage Path Planning* (CPP), while the second part is mainly known as *Vehicle Routing Problem* (VRP), or in this particular application *Agricultural Routing Problem* (AVRP).

One common CPP approach is to generate trajectories parallel to the longest edge of the field, with a spacing equal to the working width, or to simply take a direction as input (Cariou et al. (2017); Hameed et al. (2011); Hameed (2017); Jeon et al. (2021); Nilsson and Zhou (2020); Zhou and Bochtis (2015); Zuo et al. (2010)). Extending this approach, Hameed et al. (2010); Jensen et al. (2012); Plessen (2019a); Zhou et al. (2020) proposed to generate parallel trajectories along a curved reference line as well. After this step, the parallel trajectories are usually simply connected sequentially by half-turns. Instead of choosing the longest boundary, Edwards et al. (2017) proposed to align the parallel trajectories to the boundary that minimized the number of necessary half-turns. Cao et al. (2019a,b) proposed an approach based on the *rotating calipers algorithm* (first proposed by O'Rourke et al. (1986)) to determine a reference line that minimizes the number of half-turns.

Most studies, however, entirely ignored the coverage of headlands. Only the approaches proposed by Edwards et al. (2017); Jeon et al. (2021); Nilsson and Zhou (2020) were able to cover the headlands automatically. To cover all corners of the field, Edwards et al. (2017) and Jeon et al. (2021) considered also reverse moves for trajectories inside the headlands. In case of a complex field, none of the approaches described in the literature can handle both headland coverage and field decomposition.

In the more complex case of concave fields, some authors have opted for a subdivision of the field into smaller convex sub-fields. Jin and Tang (2010) applied a decomposition method, identified a reference line for each resulting sub-field, and generated trajectories parallel to the reference line. Finally an optimal sequence of sub-fields was determined. Oksanen et al. (2007); Oksanen and Visala (2009) proposed a trapezoidal decomposition approach and a heuristic algorithm to select the best driving direction among 0, 30, 60, 90, 120, and 150 degree. They applied a weighted average cost function to optimize some metrics: 1) operated area divided by total time, including turning time, 2) area of the sub-field per remaining area and 3) distance operated in the sub-field.

Hameed (2014) applied a genetic algorithm to obtain the best possible reference line that minimizes the number of trajectories as well as the turning cost. Finally, They took the inclination of each trajectory into account to determine the optimal sequence of trajectories that minimizes fuel consumption. Dogru and Marques (2015a,b); Shen et al. (2020) applied a decomposition method considering inclination across the field, followed by a genetic algorithm to determine an optimal driving direction for each sub-field as well as an optimal sequence of sub-fields that minimizes energy consumption. Hameed et al. (2016) also considered inclination in their approach to find an optimal driving direction that simply minimizes the skips and/or overlaps. Jin and Tang (2011) proposed a decomposition method to classify the field into flat and slope areas, then a reference direction that leads to the minimum coverage cost (the weighted average of headland turning cost, soil erosion cost and curved trajectory cost) was chosen from field edge segments and the slope contours.

Numerous studies have exclusively investigated AVRP, considering the parallel lines as a given input. Some of them focused only on one single robot. Bochtis et al. (2013) presented an approach based on the Clarke-Wright algorithm. Plessen (2018, 2019b) proposed a pattern-based routing algorithm. Considering one stationary service unit Jensen et al. (2015a,b) proposed an approach based on the state-space search technique. Vahdanjoo et al. (2020)

proposed an approach based on simulated annealing while considering several stationary service units. Evans IV et al. (2020) applied genetic algorithms to their approach while considering a mobile service unit.

Some other studies addressed multi-robot AVRP. Burger et al. (2013) represented an approach based on mixed-integer linear programming. Seyyedhasani and Dvorak (2017, 2018a,b); Seyyedhasani et al. (2019) proposed an heuristic approach based on the Clarke-Wright algorithm and Tabu search. Cariou et al. (2020) proposed an approach to solve AVRP for a convoy of homogeneous robots. Conesa-Muñoz et al. (2015, 2016b,a) applied simulated annealing to their approach. Utamima et al. (2019a,b) presented an evolutionary approach enhanced by a neighborhood search. Khajepour et al. (2020) applied an adaptive large neighborhood search in their approach. Conesa-Muñoz et al. (2015, 2016b,a); Khajepour et al. (2020); Utamima et al. (2019a,b) also considered one stationary service unit in their approaches. Jensen et al. (2012) represented an approach based on Dijkstra algorithm while considering a mobile service unit cooperating with a primary service unit. Bochtis and Sørensen (2009); Bochtis et al. (2010b); Bochtis and Sørensen (2010) proposed an approach based on breadth-first search algorithm modified by additional heuristics while considering one or two stationary service units or mobile service unit was.

All of these methods considered the two tasks, CCP and AVRP, as two separate problems. This approach can however have some limitations and possibly miss some interesting solutions. We think that browsing the solution space in one single exploration/generation step has the potential to find more exhaustively the interesting solutions. The approach we propose in this paper is based on this assumption, and includes many benefits that are highlighted in the next section.

3 | MOTIVATIONS AND CONTRIBUTIONS

Performing a CCPP while respecting all constraints is a complex and difficult challenge. As previously mentioned, the novelty of the approach we propose is to generate the parallel tracks and the turns in one process, with the objective of allowing more possible alternatives.

Moreover, various simplifications of the problem and prior assumptions have been made in the literature. Most of these simplifications helped to find a good feasible solution in a reasonable time for a category of fields, even if it may not be the most optimal. In this work, we are aiming at considering a maximal range of possible field shapes and configurations.

Using simplifications could also lead to a risk of oversimplification and ultimately to unfeasible solutions. For instance, assuming that a field can be accessed from all of its edges, which is not the case for most of the fields where crossing some edges may damage the robot or even the neighboring field, could lead to a solution that the farmer can't apply in practice. A common solution to solve this problem was to consider an inner offset for a field polygon as headlands and doing the headland coverage manually. The automatic coverage of the headlands was only considered by Edwards et al. (2017); Jeon et al. (2021); Nilsson and Zhou (2020). However Edwards et al. (2017); Jeon et al. (2021) considered all field edges as accessible. Conversely, another simplification could be to consider as a possible entry or exit only one or two separate points, which would strongly limit the possible solutions. This is the approach followed by Nilsson and Zhou (2020), who considered only one point for both entering and exiting the field.

Other oversimplification for operations in which the implement is in contact with the ground while in use, include considering that tight turns can be made with the implement on, or considering that lowering and raising the implement could be done instantly. However in practice, it is not the case. Such simplifications would lead to an overestimation of the coverage rate.

The aim of our study is to provide a complete and realistic solution using a progressive CCPP approach for fields

of various shapes and sizes, while avoiding oversimplification as much as possible to prevent inappropriate solutions. We aim at considering several possible entrances and authorizing the robot to finish its task anywhere on an accessible edge of the field, to increase the possibility of finding the optimum solution. We also propose an approach that can be parameterized and customized for different type of machinery and operations. Our approach is capable of dealing with both headland coverage and field decomposition.

This paper describes our approach that includes the following contributions in one single integrated system:

- automatic selection of best entry and exit points among an accessible edge of the field
- one-step generation of tracks/turns optimizing coverage, overlaps, damage, and working time
- all types of field shapes considered: convex or non convex with various dividing lines
- output is several optimal alternative paths with a variety of properties
- intelligent coverage of the headlands
- geometry of the vehicle and the implement are taken into account:
  - offset between the vehicle and the implement
  - minimum turning radius of the vehicle when the implement is off
  - minimum turning radius of the vehicle when the implement is on and in touch with the ground
- distance needed for activating/deactivating or lowering/raising the implement is taken into account
- reverse moves are allowed for performing turns and half-turns
- curved edges are taken into account

4 | METHODOLOGY

Our approach consists of three steps: 1) preprocessing 2) exploration 3) Similarity check and selection of optimal solutions. The objective of the preprocessing step is to prepare the field. Its inputs are the field polygon (*i.e.* a sequence of counterclockwise points), one or several dividing lines to decompose the field polygon if needed, the access segments, the working width, and the minimum turning radius of the robot. The output of this step is a set of entrances, a set of zones that might be used as headland and a set of *turning spaces*. Turning spaces are useful to take a trajectory within a headland and/or perform a turn from one headland to another.

Finding every potential solution and creating a solution space are the goals of the second step which is referred to as *exploration algorithm*. A solution found by the exploration algorithm is a *path*, i.e. a sequence of trajectories that starts from an entrance, covers the field and the headlands at best, and ends on one of access segments.

These two steps could be repeated several times if several entrances and/or dividing lines are provided. For simple fields for which no decomposition is required the number of exploration is the same as the number of entrances. For a complex case, that d alternative dividing lines are provided and e entrances are detected, our approach performs $d * e$ explorations. The approach perform also e more explorations while considering no dividing line. Since for some concave fields the ideal solution could be identified without decomposing the field. For instance, providing three possible entrances and three alternative dividing lines requires running the preprocessing step four times (one time with no dividing line plus the number of dividing lines) and executing the exploration algorithm twelve times in total (the product of four different preprocessing results and three entrances). Consequently, the final solution space is the union of twelve solution spaces obtained from the exploration algorithm.

Finally during the similarity check and selection of optimal solutions, the cost of each solution is computed. Then, similar solutions are extracted using a similarity function, and grouped into families of solutions. Only the lowest-cost

solution is kept for each family. After similarity removal from the solution space, the best solutions that have the lowest cost are selected. In the next sections, after giving a few definitions, we describe each of these steps in more detail.

4.1 | Definitions

Let's start by defining a few notions and notations. We define w as the width of the implement attached to the robot. Our approach exclusively considers operations in which the implement is in contact with the ground while in use. As a result, the implement is regarded as having two possible states: *on* and *off*. When on, the implement is completely in contact with the ground, and when off it is completely raised. The surface covered when the implement is in contact with the ground is called *worked* area or surface, and conversely the surface never in contact with the implement is called *unworked* area or surface.

While the implement is in off state, it is possible to perform tight or half-turns respecting γ_{off} as the minimum turning radius of the robot. As the implement is elevated, the corresponding curved surface will not be worked. When the implement is on, to avoid damaging it during turns, only slight turns respecting a minimum turning radius of γ_{on} are authorized. In this case, the corresponding surface will be worked.

However, the robot cannot change the state abruptly from on to off or vice versa. It must be done gradually while the robot moves in a straight line. Therefore the *transition state* is defined as a third state when changing from on to off and conversely. The straight trajectories in transition state have a constant length of ℓ_t, and are referred to as *transition trajectories*. ℓ_t represents the distance required on the straight trajectory before or after performing a tight turn, where the surface will not be worked either. We refer to these unworked portions of straight trajectories as *gaps*. They are represented in Fig. 1 with crossed out red segments. Therefore, each tight turn causes two gaps in the trajectory.

4.2 | Preprocessing: definition of headlands and turning spaces

After acquiring the raw input data, and converting from the geographic coordinate system to the Cartesian coordinate system, three preprocessing operations are performed: the generation of entrances, the generation of headlands, and the generation of turning spaces. Both operations are detailed below.

4.2.1 | Entrances

Entrances are the locations where the robot can enter the field. Each entrance also contains the direction or heading of the robot at the corresponding location. Given the field polygon and the access segments, our approach determines the entrances on each corner of an access segment where its distance from the adjacent edge of the field polygon is $w/2$. Its direction is the same as the direction of the adjacent edge. Once all candidate entrances are determined, an expert must validate them or discard the entrances that seem irrelevant. Fig. 5 represents the selected entrances for a sub-set of the evaluation dataset.

4.2.2 | Headlands

A headland is a space adjacent to the boundary of the field that can be used to perform half-turns. When half-turns are performed in a headland, this space is not worked, and two gaps are also caused before and after it along the

trajectories. Then, this space must be worked separately after the rest of the field is finished. Our method starts by defining these headlands, and anticipating how they can be covered.

To be able to define these surfaces, for each boundary of the field a headland area is generated that contains:

- one *outer border* that corresponds to the field boundary
- one *inner border* that is generated parallel to the outer border inside the field at a distance of $p * w$, where p is either given as an input, or deduced from w and γ_{off}
- p *inner trajectories* that are generated between the outer and inner borders and parallel to them, at a distance of $w/2$ from the borders and w from each other
- one *gap covering trajectory* that is generated inside the field at a distance of $w/2$ from the inner border and parallel to it

These geometric elements are illustrated in Fig.1 for $p = 2$. If a dividing line is provided, another type of headland is also defined: it is fully included within the field polygon with two inner borders, p inner trajectories, and two gap covering trajectories centered around the dividing line, and it has no outer border. This is illustrated on Fig.1 where the dividing line is represented in brown.

Therefore the inner trajectories are mainly useful for covering the unworked surfaces caused by the half-turns and the gap covering trajectories are used to cover the unworked surfaces caused by gaps.

4.2.3 | Turning spaces

The third preprocessing operation is the computation of turning spaces. A turning space is defined at an angle between two adjacent headlands, and ensures a safe and feasible turn to or between inner or gap covering trajectories of a headland or switch between sub-fields, if a dividing line is provided. It is delimited by two *turning lines* parallel to the angle bisector.

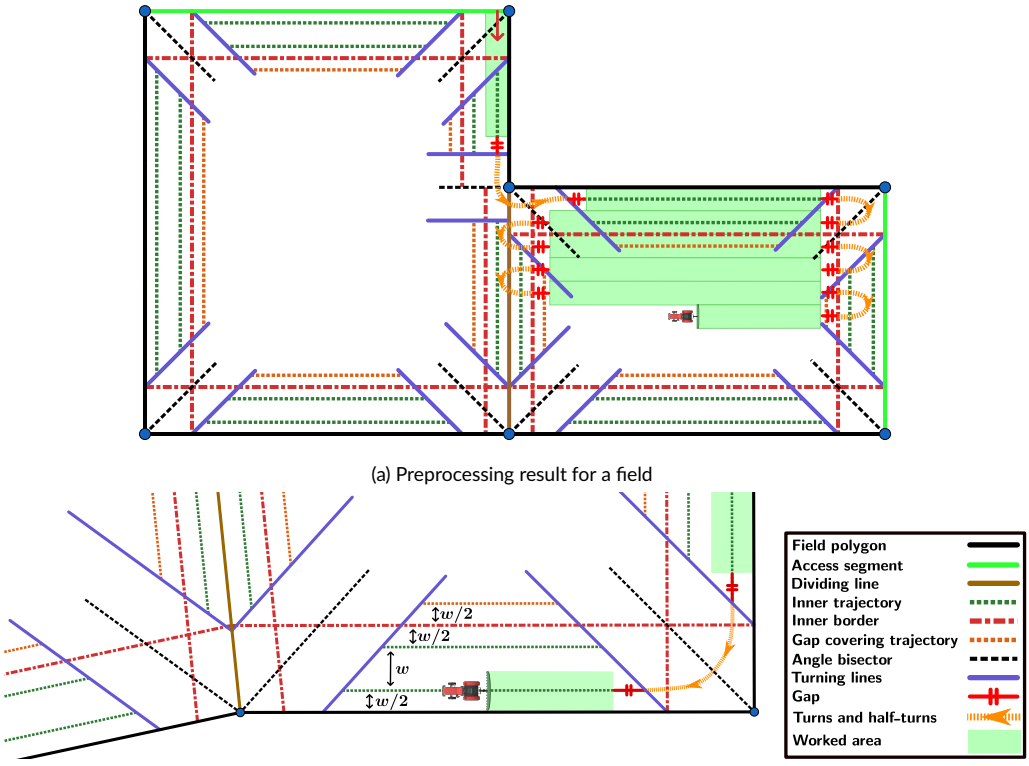
As illustrated in Figs. 2a 2c, for two adjacent headlands first their angle bisector is computed (black dashed line). Finally the two turning lines (blue solid lines) are computed parallel to the angle bisector with a spacing of $l/2$ where $l = \max\Big(\sqrt{2} * ((p-1) * w + \ell_o)\ ,\ 2 * (\ell_o + \gamma_{off})\Big)$, where ℓ_o is the length of the offset between the robot and the implement. Before a tight turn, there is a straight gap trajectory. Consequently the robot is ℓ_o meters ahead of the implement when the gap trajectory is performed. Therefore, ℓ_o must be taken into account while computing turning spaces to provide the robot enough space to turn.

The intersection between turning lines and the inner/gap covering trajectories will be used as candidate start/end point for turns from one headland to another. Therefore, regardless of the angle between two headlands, turning spaces can ensure a proper space for turns in order to travel to inner and or gap covering trajectories of adjacent headlands.

Let us note that not all of the headlands and turning spaces generated during the preprocessing step will necessarily be used. The exploration algorithm will decide which ones will be used. Section 4.4 provides more detail on how turning spaces are used to make turns and half-turns inside a headland.

4.3 | Trajectory types and metrics

To facilitate understanding of the proposed approach, it is essential to introduce some useful concepts and metrics. Dubins trajectories (Dubins (1957)) and Reeds–Shepp curves (Reeds and Shepp (1990)) were employed to generate



(a) Preprocessing result for a field

(b) A close-up on a preprocessing result

FIGURE 1 Preprocessing result for $p = 2$. The field polygon is represented by blue points and black and green solid segments

continuous and smooth turns. Taking as input a minimum turning radius, the starting and destination coordinates, and the direction of the robot at at these coordinates, both methods compute the shortest curve from the starting point to the destination point, with the difference that the Reeds-Shepp method also considers reverse moves while all turns generated by the Dubins method contain only forward moves. Therefore a turn generated by the Reeds-Shepp method is optimal in terms of trajectory length, but in terms of travel time a turn generated by the Dubins method may be better, as a reverse move requires the robot to stop, change the direction and accelerate to reach again its target speed. Moreover, the Dubins method may be used with the implement on during a slight turn when the radius is wide, whereas if a reverse move is needed then the implement must always be off. Consequently, in our approach turns with no reverse moves are preferred. However if performing a turn with no reverse moves is not possible, then a turn with reverse moves is also examined. Considering these two types of turns as well as the three different states of the implement (on, off, and transition), five different types of trajectories were defined as follows:

- *STRAIGHT_ON*: straight trajectory while the implement is on
- *DUBINS_ON*: turn computed via Dubins method in which the implement is on
- *DUBINS_OFF*: turn computed via Dubins method in which the implement is off
- *REEDS_OFF*: turn computed via Reeds-shepp method in which the implement is off

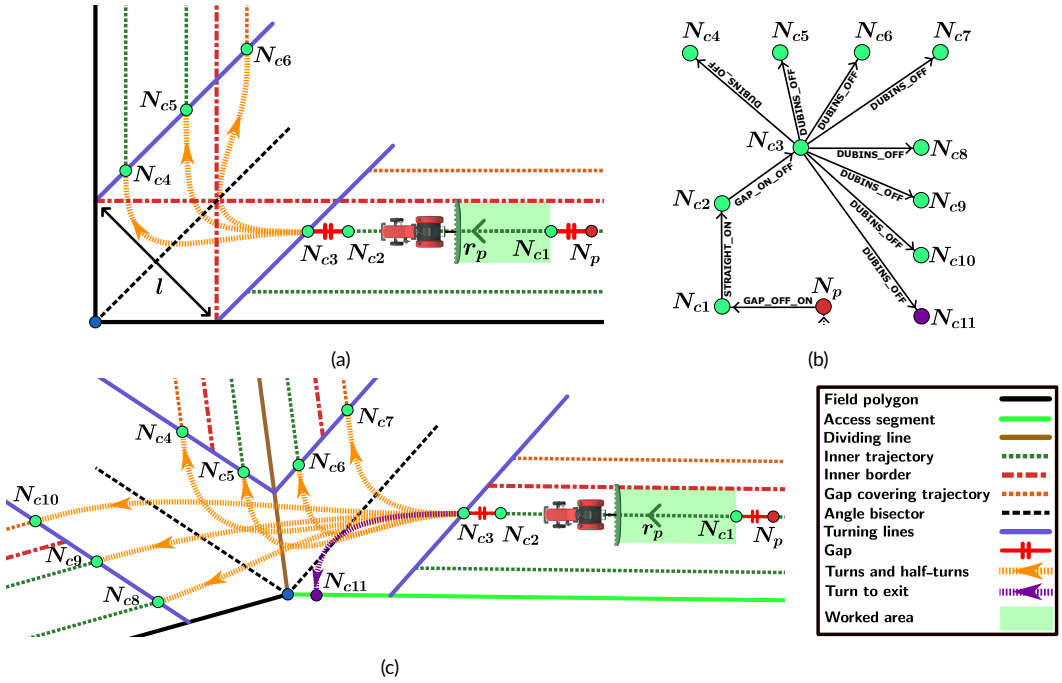


FIGURE 2 Trajectory (a,c) and tree (b) representations of turns inside the turning spaces. Sub-figure (*b*) is the tree representation of trajectories of (*c*)

- *GAP_OFF_ON*: transition trajectory of length ℓ_t during which the implement is transitioning from off to on
- *GAP_ON_OFF*: transition trajectory of length ℓ_t during which the implement is transitioning from on to off

A few rules apply when defining a sequence of trajectories of the above types. As a first rule, a *GAP_OFF_ON* trajectory must always be followed by a *STRAIGHT_ON* trajectory, where both trajectories have the same direction. The second rule is that a *STRAIGHT_ON* trajectory only can be followed by a *DUBINS_ON* or a *GAP_ON_OFF* trajectory. Consequently the third rule says that a *GAP_ON_OFF* trajectory can be either used for exiting the field or it must be followed by a *DUBINS_OFF* or a *REEDS_OFF* trajectory. According to the fourth rule, a *DUBINS_OFF* or a *REEDS_OFF* trajectory can be either used for exiting the field or it must be followed by a *GAP_OFF_ON* trajectory. We refer to these rules as the *trajectory sequence rules*. Let us note that a trajectory used for exiting the field must end up on an access segment.

A path is a sequence of k trajectories with a variety of these five types. Therefore, a trajectory Λ_i with a length of ℓ_i where $\{i \in \mathbb{N} | i \leqslant k\}$ is represented as $\Big((P_i, \vartheta_i), (P_i', \vartheta_i'), \Gamma_i\Big)$ where (P_i, ϑ_i) represents its start point and direction, (P_i', ϑ_i') represents its destination point and direction, and Γ_i precises its trajectory type. Assuming that the directions of the robot and its implement are the same, P_i and P_i' represent the location of the implement. It is relatively straightforward to compute the location of the robot on the path when the distance between the implement and the robot is known.

A *DUBINS_ON* trajectory can only be used to travel from one headland to another using a slight turn. The angle between these two headlands must be between $\pi - \alpha$ and $\pi + \alpha$ where $\alpha = \arcsin\left(\frac{l}{2 * r_{on}}\right)$, and l is the spacing between

the turning lines.

Finally, we define the four following metrics that are used to check the suitability of the candidate solutions in our approach.

Worked area: the worked area is the first of the key metrics in this approach. This metric is only computed for *STRAIGHT_ON* and *DUBINS_ON*. For all the other types of trajectories, the worked area is zero. For a trajectory Λ_i, if Γ_i is *STRAIGHT_ON*, then the worked area is given by $w * \ell_i$. If Γ_i is *DUBINS_ON*, the curve between P_i and P_i' is first sampled to a set of points with a spacing of $0.5m$ between two consecutive points. Then for each pair of consecutive samples and having the direction at each sample, the two lateral ends of the implement is computed and a trapezoid is constructed. Finally the worked area is computed as the sum of areas of all trapezoids.

Overlap area: this metric, that measures the overlap area of two trajectories, is only computed for two trajectories of types *STRAIGHT_ON*, two *DUBINS_ON*, or one *STRAIGHT_ON* and one *DUBINS_ON*. Otherwise it is equal to zero. Once the worked areas of the two trajectories are computed, the overlap area is deduced by calculating their intersection.

Damage: this Boolean metric verifies whether the robot is crossing a trajectory that was previously worked (types *STRAIGHT_ON* or *DUBINS_ON*), with a new trajectory while its implement is off (*DUBINS_OFF*, *REEDS_OFF*, *GAP_OFF_ON* or *GAP_ON_OFF*). This avoids to unnecessarily damage the previously worked zones with the wheels of the robot without working it again.

Inside: The last metric, named *inside*, is also Boolean. For a trajectory Λ_i if the robot and its implement remain inside the metric is true, otherwise it is false.

4.4 | Exploration algorithm

The overall concept of the exploration algorithm is to take as input the result of the preprocessing, the access segments, a set of hard constraints, γ_{on}, γ_{off}, ℓ_t, and the *coverage threshold*, to progressively build a tree of nodes representing possible sequences of trajectories (locations, directions, and trajectory types) satisfying the hard constraints. The following sections detail the notion of node, the hard constraints, and the method used to build the tree.

4.4.1 | Nodes

Each node of the tree represents a possible candidate for the next step of the trajectory. It contains a flag representing one of the five possible trajectory types, and a corresponding destination point and direction. A set of parent and child nodes N_p and N_c are represented respectively by $(P_p, \vartheta_p, \Gamma_p)$ and $(P_c, \vartheta_c, \Gamma_c)$. Consequently, a trajectory Λ_c from N_p to N_c is represented as $\Big((P_p, \vartheta_p), (P_c, \vartheta_c), \Gamma_c\Big)$.

The root of the tree is a specific node containing the entrance location and direction, and a default trajectory type with the implement off. A leaf node is a specific node containing an exit point located on an access segment, a direction, and a trajectory type with the implement off.

A solution is a path represented by a branch of the tree, *i.e.* a sequence of trajectories from the root to a leaf node.

4.4.2 | Hard constraints

The hard constraints are Boolean constraints that must be satisfied by a solution to be valid. We defined five hard constraints, most of them linked to one of the previously defined metrics: *inside constraint*, *limited overlap constraint*

global overlap constraint, *local loop constraint*, *damage constraint*, *switch constraint* and *minimum working distance constraint*. A node $N_c = (P_c, \vartheta_c, \Gamma_c)$ can be added to the tree as a child of $N_p = (P_p, \vartheta_p, \Gamma_p)$ only if the candidate trajectory $\Lambda_c \left((P_p, \vartheta_p), (P_c, \vartheta_c), \Gamma_c\right)$ satisfies all the hard constraints.

Inside constraint: according to the inside constraint, the inside metric of Λ_c must be true.

Damage constraint: the damage constraint ensures that the damage metric of Λ_c over trajectories constructed for all ancestor nodes of N_p remains false.

Limited overlap constraint: this constraint forbids overlaps in the center of the field, i.e. outside the headlands and the gap covering trajectories.

Global overlap constraint: the global overlap constraint limits the overall overlap, within authorized zones, to a certain threshold. When adding a new node, the overlap between Λ_c and all its ancestors is computed, and added to a cumulative sum. When the cumulative overlap exceeds a *global overlap threshold* Δ_{global} over the surface of the field, the branch is discarded. If a dividing line is provided, this metric is applied on each sub-field separately.

Local loop constraint: at a more local level, to avoid undesirable local loops, when adding a new node the overlap between Λ_c and the ancestors is computed. If an overlap is found that exceeds Δ_{local}, which is a percentage of the area of Λ_c named *local loop threshold*, then the branch is discarded. An exception is made to the local loop constraint when the objective in terms of coverage rate is satisfied, in order to let the robot reach an access segment to exit the field.

Switch constraint: in case a dividing line is provided, this constraint ensures that a switch between sub-fields is authorized: it is if the worked area since the last switch is more than a threshold Δ_{switch} called *switch threshold*, or if the objective in terms of coverage rate is already satisfied for the current sub-field.

The minimum working distance constraint: ensures that the sum of the lengths of consecutive trajectories of type *STRAIGHT_ON* or *DUBINS_ON* is not lower than a threshold Δ_{min_dist} named *minimum working distance threshold*. The main reason behind this constraint is that it is inconvenient and expensive to activate the implement over a short distance. Therefore, after a trajectory of type *STRAIGHT_ON* or *DUBINS_ON*, only another *STRAIGHT_ON* or *DUBINS_ON* trajectory is authorized if the minimum working distance constraint is not respected yet.

4.4.3 | Construction of the tree and its trajectories

The construction of the tree and its trajectories can be detailed at two different levels: the initialization, that defines how the initial trajectories are created, and the node generation and exploration, that describes how back and forth trajectories are created and added to the tree. In this section we also describe how and in which conditions a turning space is used for the generation of turns that lead the robot to an inner or gap covering trajectory of a headland.

Initialization: The first step consists of initializing the tree by adding an entrance $N_0 = (P_0, \vartheta_0, \Gamma_0)$ to the tree and generating the first branches of the tree. As illustrated in Fig. 3, three options can be followed to start working:

- go straight and start working immediately: in this case, the first trajectory will be of type *GAP_OFF_ON* to point N_1. After this, the next trajectory will have to be of type *STRAIGHT_ON*.
- cross the headland to point N_2 with implement off. After this, the next trajectory has to be of type *GAP_OFF_ON* then *STRAIGHT_ON*.
- turn immediately in the headland to points N_3 or N_4. After this, the next trajectory has to be of type *GAP_OFF_ON* then *STRAIGHT_ON*.

After being validated by the hard constraints, these nodes are added to the tree as children of N_0 and further

exploration is conducted on each leaf of the tree. In Fig. 3 gap covering trajectories are not illustrated because gap covering trajectories are mostly for covering the unworked areas caused by transition trajectories, therefore they are not used for initialization.

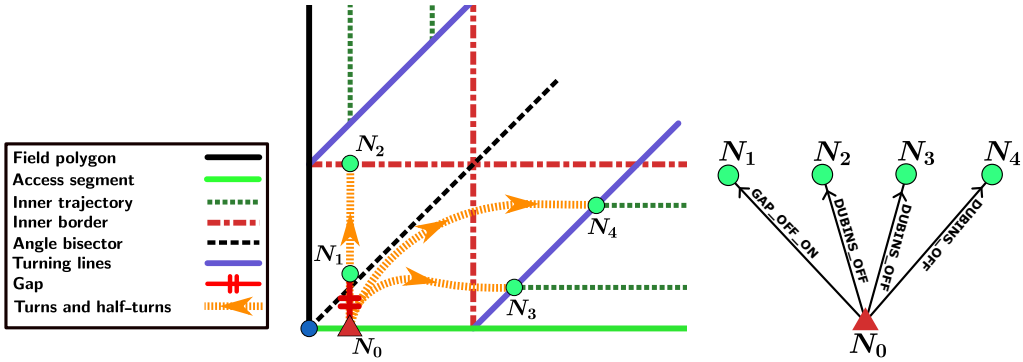


FIGURE 3 Trajectory and tree representation of the initialization. The entrance and root of the tree N_0 are represented by a red triangle. N_1, N_2, N_3 and N_4 are the first nodes that might be added to the tree as children of N_0. Orange dashed lines represent trajectories from N_0 toward its children

Node generation and exploration: after initializing the tree, a depth first exploration is performed. In general for each unvisited leaf of the tree, new nodes are generated respecting the trajectory sequence rules. The nodes that satisfy the hard constraints are added to the tree as the new children of the leaf. One of these children is then selected for further node generation and exploration. If all of the generated nodes for a leaf violate at least one hard constraint, the leaf is removed from the tree and the exploration continues on its siblings.

For each unvisited node $N_p = (P_p, \vartheta_p, \Gamma_p)$, a ray r_p that starts from P_p with the direction ϑ_p is generated. The intersection of r_p with inner borders of all headlands and all turning spaces is then calculated. The intersection of r_p with a headland inner border leads to the generation of a cycle of traversals and half-turns while an intersection with a turning space leads to a headland switch. In case that the field polygon is divided into sub-polygons by a dividing line, only the inner borders and turning spaces that are inside the same sub-polygon as P_p are considered for intersection computation with r_p. This is useful for limiting the unnecessary switches between sub-polygons.

Cycles of traversals and half-turns: a cycle of traversal and half-turn is a sequence of trajectories of types: *GAP_OFF_ON*, *STRAIGHT_ON*, *GAP_ON_OFF* and *DUBINS_OFF*. They can be made either in the main part of the field or within a headland along the boundary of the field.

As illustrated in Fig. 4, ray r_p hits the vertical headland's inner border on the right, called destination inner border, at the position of N_{c3}. As a result, trajectories from N_p to N_{c3} with two intermediate nodes N_{c1} and N_c3 are straight trajectories of types *GAP_OFF_ON*, *STRAIGHT_ON* and *GAP_ON_OFF* respectively. To complete the cycle the possible turns from N_{c3} are calculated. The positions of N_{c4} and N_{c5} are determined by computing the intersection of rays r_r and r_l with the destination inner border. These two rays are parallel to r_p with a spacing of w. The direction at N_{c4} and N_{c5} is the opposite direction of ϑ_p. In the particular case where we are close to a neighbor headland and the trajectory is oblique to it, point N_{c6} is also computed as the intersection of rays r_r or r_l and the inner border of the neighbor headland. This allows for more options for the algorithm in terms of satisfaction of the various overlap and damage constraints.

Note that in exceptional cases where the typical cycle does not satisfy the hard constraints, type *REEDS_OFF* is

also examined for the last trajectory.

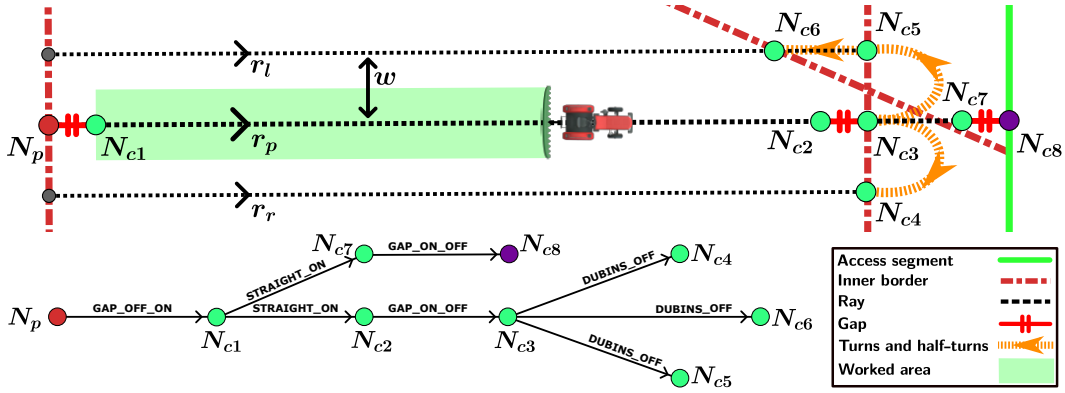


FIGURE 4 trajectory and tree representation of a cycle of traversal and half-turn. N_p is the selected leaf for which the node generation is conducted. The purple point represents an exit node. For readability purposes, turning spaces, inner and gap covering trajectories of headlands are not represented in this figure

headland switch: The process of traveling from one headland to an adjacent headland via a turning space is referred to as headland switch. It is used to cover the headlands that were not worked due to the half-turns. It is also useful for going from one sub-field to another, if a dividing line was provided to decompose the field polygon. Fig. 2a and 2c illustrate a trajectory representation of different possibilities of headland switch and Fig. 2b demonstrates a tree representation of Fig. 2c.

As illustrated in Figs. 2a and 2c, the position of N_{c3} is the intersection of r_p and a turning space referred to as selected turning space. Sub-figure *a* illustrates the intersection of r_p with a turning line. Sub-figure *c* represents a complex case in which all possible ways to switch from the right headland to two possible neighbor headlands are shown. In general complex cases happen only around a dividing line.

A headland switch is also a sequence of a *GAP_OFF_ON*, *STRAIGHT_ON*, *GAP_ON_OFF* and *DUBINS_OFF* trajectories. The first three trajectories (trajectories from N_p to N_{c3})), that are straight trajectories of types *GAP_OFF_ON*, *STRAIGHT_ON* and *GAP_ON_OFF*, are used to arrive at the selected turning space. Afterwards, all possible turns from N_{c3} to other headlands at destination nodes (N_{c4}, ..., N_{c10}) are determined by computing the intersection of the inner and gap covering trajectories of the target headlands with the corresponding turning spaces. The direction of a destination node corresponds to the direction of the target inner or gap covering trajectory. Their trajectory type is set to *DUBINS_OFF*, therefore turns are generated by the Dubins method. If they do not satisfy the hard constraints, their trajectory type is modified to *REEDS_OFF* and a turn generation with the Reeds-Shepp method is also examined.

The number of nodes generated to switch from one headland to another depends on the number of inner and gap covering trajectories of the target headland. It also depends on whether an inner or gap covering trajectory of the target headland intersects with an access segment or not.

Exiting the field: during the cycles of traversals and half-turns, if ray r_p intersects with an access segment (green solid line segment), a sequence of trajectories of type *GAP_OFF_ON*, *STRAIGHT_ON* and *GAP_ON_OFF* (illustrated by $N_p \rightarrow N_{c1} \rightarrow N_{c7} \rightarrow N_{c8}$ in Fig. 4) is also generated to examine the possibility of exiting the field.

Another way of exiting the field is when working the headlands. If the current direction crosses a turning space, and this turning space is adjacent to an access segment, then the possibility of turning to exit the field with a node of

type *DUBINS_OFF* is examined. This situation is illustrated in Fig. 2c with the sequence of nodes $N_p \rightarrow N_{c1} \rightarrow N_{c2} \rightarrow N_{c3} \rightarrow N_{c11}$.

In these two cases, a leaf node is generated to finish the branch. Then the sum of the worked areas of all the trajectories in the branch is computed. The branch is kept only if this sum is greater or equal to a predefined coverage threshold Δ_{cov}. We say that a branch satisfying this criterion and the hard constraints is *valid*. The output of the exploration algorithm is a set of valid solutions (tree of valid branches) that is referred to as the *solution space*.

4.5 | Similarity check and selection of optimal solutions

Once the solution space is generated, it is usually wide and browsing all the solutions would be a tedious task for the user. In order to present to the user only a few of the most relevant solutions, we first define a cost function based on the metrics, then we explain how we group the solutions into families based on a similarity criterion and select the optimal solution of each family.

In order to compute and compare the solutions found by the exploration method, we propose a cost function to minimize, built as a weighted average of four soft constraints: 1) S_{cov}: maximizing the coverage rate, 2) S_{ovl}: minimizing the overlaps, 3) S_{nwd}: minimizing the non-working distance and 4) S_{otm}: minimizing the operation time.

Maximizing the coverage rate: the first soft constraint S_{cov} is calculated as follows:

$$S_{cov} = 1 - \frac{C_i - C_{min}}{C_{max} - C_{min}} \tag{1}$$

where C_i is the area covered by solution i, and C_{min} and C_{max} are respectively the minimum and the maximum worked area over the entire solution space. Therefore minimizing S_{cov} leads to maximizing the coverage rate.

Minimizing the overlaps: the second soft constraint S_{ovl} is calculated as follows:

$$S_{ovl} = \frac{O_i - O_{min}}{O_{max} - O_{min}} \tag{2}$$

where O_i is the overlap area of solution i, and O_{min} and O_{max} are respectively the minimum and the maximum overlap area over the entire solution space.

Minimizing the operation time: the total time required to travel all trajectories of a solution i, denoted T_i, is computed as follows:

$$T_i = \frac{L_i^{on}}{V_{on}} + \frac{L_i^{off}}{V_{off}} + \frac{L_i^{gap}}{V_{gap}} \tag{3}$$

where L_i^{on} is the length of all *STRAIGHT_ON* and *DUBINS_ON* trajectories, L_i^{off} is the length of all *DUBINS_OFF* and *REEDS_OFF* trajectories, and L_i^{gap} is the length of all *GAP_OFF_ON* and *GAP_ON_OFF* trajectories and for solution i. V_{on}, V_{off} and V_{gap} are respectively the average speed of the robot when its implement is in on, off and transition modes. Therefore, the fourth constraint S_{otm} is defined as follows:

$$S_{otm} = \frac{T_i - T_{min}}{T_{max} - T_{min}} \tag{4}$$

where T_{min} and T_{max} are respectively the minimum and the maximum traveled time over solutions of the solution space.

Minimizing the non-working distance: the third constraint S_{nwd} is the sum of the lengths of all trajectories when the implement is not on, *i.e.* all *DUBINS_OFF*, *REEDS_OFF*, *GAP_OFF_ON* and *GAP_ON_OFF* trajectories. It is calculated as follows:

$$S_{nwd} = \frac{L_i^{nw} - L_{min}^{nw}}{L_{max}^{nw} - L_{min}^{nw}} \tag{5}$$

where $L_i^{nw} = L_i^{off} + L_i^{gap}$, and L_{min}^{nw} and L_{max}^{nw} are respectively defined in a similar fashion as the minimum and the maximum non-working distances over the entire solution space.

Final cost function: the final cost function f is defined as follows:

$$f = \frac{(S_{cov} * W_{cov}) + (S_{ovl} * W_{ovl}) + (S_{otm} * W_{otm}) + (S_{nwd} * W_{nwd})}{W_{cov} + W_{ovl} + W_{otm} + W_{nwd}} \tag{6}$$

where W_{cov}, W_{ovl}, W_{otm}, and W_{nwd} are weights given as input for the corresponding cost functions.

The generated solution space may contain several very similar solutions where their differences could be only one turn or two. To eliminate sub-optimal solutions while still preserving a variety of propositions, they are grouped into families of solutions based on a similarity criterion, and only the solution with the lowest cost within each family is kept. The similarity criterion is based on the *general direction* of the solutions, which corresponds to the main direction of the back and forth trajectories, *i.e.* the ones most frequently used inside the field polygon. If the field polygon is divided into sub-polygons, the general direction is computed for each sub-polygon. Therefore, two solutions are considered similar, and belonging to the same family, if they have the same general direction(s). After applying the similarity criterion and keeping only one optimal trajectory per family, the best solution of each family is presented to the user, and the *most optimal solution* among them is highlighted. The number of families is limited to the number of possible general directions, which corresponds to the number of borders of the field.

5 | RESULTS AND DISCUSSION

5.1 | Experiment

This approach was implemented in a program written in *C++*, that includes a GUI to set the input variables and visualize the generated solutions. To accelerate the computations, all the processes can run in parallel thanks to an implementation using OpenMP. The program was run on an Intel Xeon(R) W-2135 CPU @ 3.70GHz × 12 with 32GB RAM.

A collection of thirty fields from the real world were chosen to evaluate the proposed approach. These fields range in area from 1.83 to 13.20 hectares. Twenty fields have a simple shape for which no field decomposition was required, while ten fields have a complex shape for which at least two different dividing lines were provided to try

different field decompositions. To extract the data of these fields, the annotation tool of Géoportail was used. At least one access segment and two entrances were considered for each field.

The algorithm was run on each field with the variety of given dividing lines and entrances. For each field, we obtained several families of solutions, one best path for each family, one of them being the most optimal solution overall. All the best paths were compared to the ground truth, by visually comparing the results with the reference satellite image of the field, where the tracks are visible. Among the best solutions, the one found as having the best visual similarity with the reference satellite image is referred to as the *most similar solution*. The general directions of back and forth moves as well as how a field is divided into sub-fields were used as the similarity criterion. We assumed the path was chosen by the farmer who is an expert and is completely familiar with the field. Although it was relatively straightforward to delineate the headlands and the general directions of back and forth moves from the satellite images, determining the parameters of the vehicle and the implement, or the maximum coverage rate and overlap from these images only was almost impossible. These parameters were guessed at best, as average settings, and similar for all fields. The parameters of our approach are given in Table 1.

Parameter	Description	Value	Parameter	Description	Value
w	working width	$3m$	Δ_{cov}	coverage threshold	97%
γ_{on}	minimum turning radius - implement on	$15m$	Δ_{global}	global overlap threshold	5%
γ_{off}	minimum turning radius - implement off	$1.5m$	Δ_{local}	local loop threshold	95%
V_{on}	average speed - implement on	$3.5m/s$	Δ_{switch}	switch threshold	93%
V_{gap}	average speed - implement transition	$2.5m/s$	Δ_{min_dist}	minimum working distance threshold	$8m$
V_{off}	average speed - implement off	$1.5m/s$	W_{cov}	weight of S_{cov}	0.6
ℓ_t	transition trajectory length	$2m$	W_{ovl}	weight of S_{ovl}	0.1
ℓ_o	robot-implement offset	$2m$	W_{nwd}	weight of S_{nwd}	0.2
p	number of inner trajectories of headlands	2	W_{otm}	weight of S_{otm}	0.1

TABLE 1 The input and parameters of the proposed approach

To avoid weighing down the article, we give the information about the complete dataset and detailed results as Supplementary Materials. In this paper, we present the average results and illustrate them with a subset of fields to highlight some interesting properties. The Supplementary Materials contain for each field of the complete dataset a link to the field data in the web application of Géoportail, the coordinates of a point inside the field, figures illustrating the shape and the results, entrances and access segments, and the full results in a table.

5.2 | Analysis of the results

The average results of the evaluation are summarized in Table 2. According to these statistics, for simple fields our approach was able to achieve a coverage rate of 98.69% while limiting the overlap to 3.00% in average. For complex fields the average coverage rate was 98.23% while the average overlap was limited to 3.09%. This accomplishment is mainly due to the capability of our approach to cover the headlands and deal with curved edges.

For 85% of simple fields, the most optimal solutions and the solution most similar to the satellite images were the same. For the other 15% of cases, the most similar solution was found by our approach, but not considered as being the most optimal according to the predefined criteria.

For complex fields, 70% of the most optimal solutions were identical to the most similar to the satellite images. In 10% of the cases, the most similar solution was also found by our approach, but not considered as being the most

optimal. For the remaining 20%, no similar solution was found. We identified two possible reasons. First, as previously mentioned, we guessed the parameters at best. Perhaps the different values we imposed for the thresholds and the vehicle parameters were not the ones used by the farmer. Adjusting them, for instance decreasing the coverage threshold Δ_{cov} and/or increasing the global overlap threshold Δ_{global}, may lead to more solutions and consequently increase the possibility of finding the most similar solution. Second, some constraints or preferences may be considered by the farmer that were not applied in our approach. For instance, the farmer may have used visual clues to help working the land which may have had an influence on the choice of the trajectories, while we did not consider this as a useful factor for a robot.

As previously mentioned, the total number of explorations for a field depends on the number of selected entrances and the number of provided dividing lines. Therefor, for each field several explorations were performed. Consequently, in Table 2, single exploration time contains the average and standard deviation over all explorations performed on all simple and complex fields.

Area (ha)		Coverage		Overlap		Single exploration time (s)		Selection time (s)	
Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
4.87	2.82	98.69%	0.62%	3.00%	1.39%	64.70	81.21	1.71	2.71

(a) Simple fields

Area (ha)		Coverage		Overlap		Single exploration time (s)		Selection time (s)	
Mean	STD	Mean	STD	Mean	STD	Mean	STD	Mean	STD
4.69	2.41	98.23%	0.58%	3.09%	1.17%	617.20	610.02	10.60	15.73

(b) Complex fields

TABLE 2 Numerical results of the evaluation

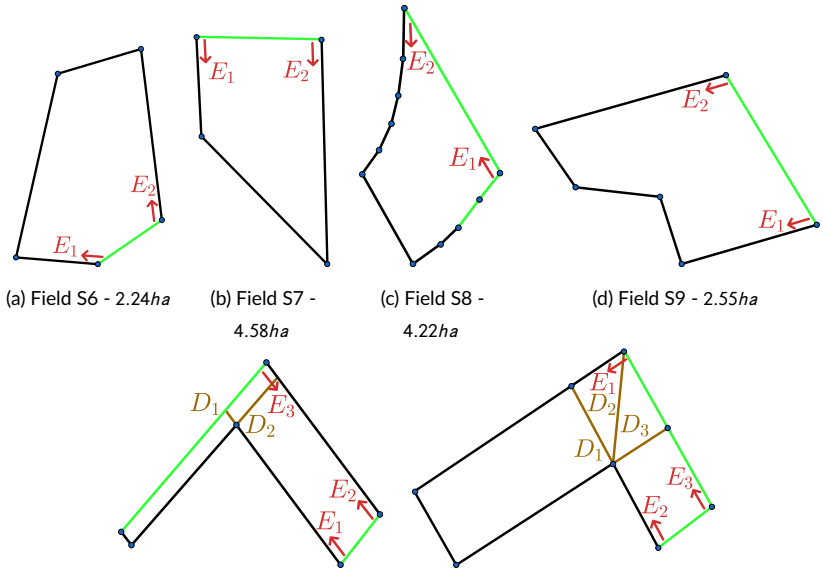
In the following, we illustrate the results on a sample of six fields extracted from the dataset: four simple and two complex fields. The name of each sample correspond to their original name in the dataset provided in Supplementary Materials. The shapes and input data of the six fields are shown in Fig. 5. Each field is represented by a set of counterclockwise ordered points. For the complex fields, respectively two and three dividing lines are provided and shown in brown. Table 3 contains a link to the data of these fields in the web application of Géoportail as well as the coordinates of a point inside the field.

Field	S6	S7	S8	S9	C4	C2
Link	bit.ly/3WGTfER	bit.ly/3DP8vqG	bit.ly/3NLmQJf	bit.ly/3EeTvUo	bit.ly/3E3l8OK	bit.ly/3zZK1dg
Lon / Lat	7.4311° / 48.8245°	2.4845° / 50.3106°	7.5924° / 48.831°	7.4641° / 48.8146°	3.1021° / 48.2449°	2.7067° / 50.3336°

TABLE 3 Links to the sample fields from the dataset in Géoportail and coordinates of a point inside the field

Fig. 6 illustrates the results obtained on the six fields, and Table 4 summarizes the numerical results. As illustrated in the figure, our approach not only successfully covered unworked area caused by half-turns and gaps, but also intelligently selected the headlands to perform half-turns. Fig. 6g illustrate the capacity of our approach to deal with curved trajectories within headlands.

For complex fields, our approach was not only capable of finding the most optimal solution but also determined



(a) Field S6 - 2.24*ha* (b) Field S7 - 4.58*ha* (c) Field S8 - 4.22*ha* (d) Field S9 - 2.55*ha*

(e) Field C4 - 6.25*ha* (f) Field C2 - 2.21*ha*

FIGURE 5 Sample of six fields from the dataset. A field polygon is represented by green and black line segments where green segments are accessible edges of the field. Dividing lines are represented in brown. Entrances are represented by red arrows

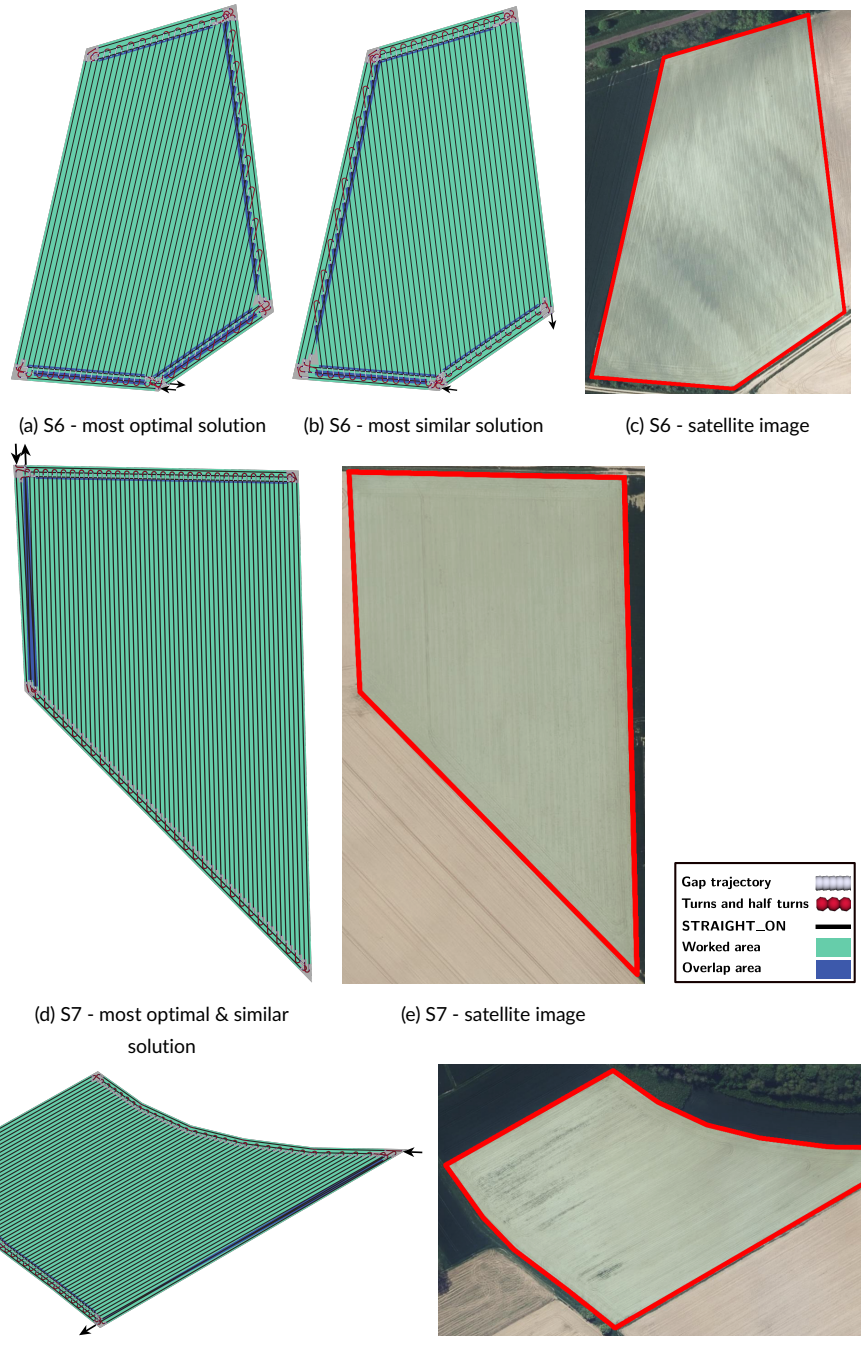
Field	Exploration			Similarity check and selection of optimal solutions							
	total	Solutions	time (s)	time (s)	$S_{cov} * W_{cov}$	$S_{ovl} * W_{ovl}$	$S_{nwd} * W_{nwd}$	$S_{otm} * W_{otm}$	f	Coverage	Overlap
S6	2	9459	147.30	0.24	0.000	0.081	0.038	0.049	0.168	98.43%	4.42%
S7	2	2511	22.32	0.07	0.000	0.036	0.022	0.044	0.101	98.58%	1.80%
S8	2	1258	42.16	0.06	0.000	0.048	0.012	0.023	0.083	98.35%	2.66%
S9	2	186	75.482	0.01	0.000	0.099	0.073	0.085	0.256	97.77%	4.87%
C4	9	280782	2848.74	12.32	0.056	0.036	0.011	0.017	0.121	98.91%	2.01%
C2	12	82	599.12	0.00	0.000	0.048	0.010	0.014	0.072	97.64%	3.61%

(a) Most optimal solution

Field	Exploration			Similarity check and selection of optimal solutions							
	total	Solutions	time (s)	time (s)	$S_{cov} * W_{cov}$	$S_{ovl} * W_{ovl}$	$S_{nwd} * W_{nwd}$	$S_{otm} * W_{otm}$	f	Coverage	Overlap
S6	2	9459	147.30	0.24	0.083	0.050	0.059	0.045	0.237	98.23%	3.48%
C2	12	82	599.12	0.00	0.398	0.038	0.013	0.012	0.461	97.22%	3.35%

(b) Most similar solution

TABLE 4 Numerical results of the illustration dataset. S_{cov}, S_{ovl}, S_{nwd}, S_{otm} and f are computed by equations 1, 2, 5, 4 and 6 respectively. The most optimal and most similar solutions are the same for Fields S7, S8, S9 and C4

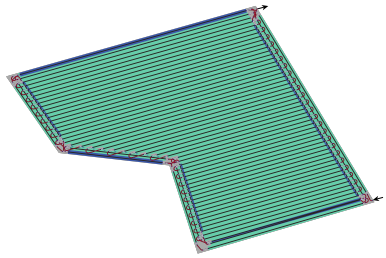


(a) S6 - most optimal solution (b) S6 - most similar solution (c) S6 - satellite image

(d) S7 - most optimal & similar solution (e) S7 - satellite image

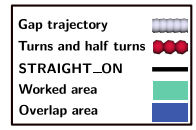
(g) S8 - most optimal & similar solution (h) S8 - satellite image

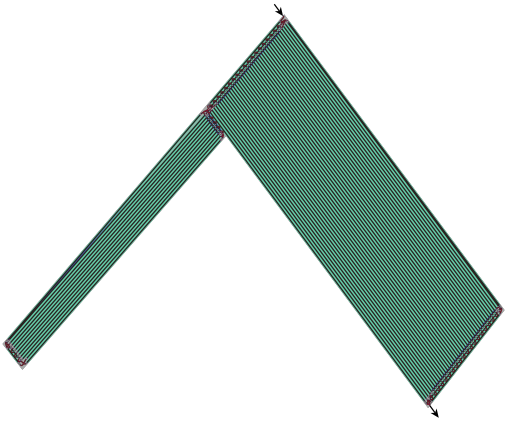
FIGURE 6 Obtained results on illustration dataset and reference satellite image. The black arrows indicate where the robot enters and exits



(i) S9 - most optimal & similar solution

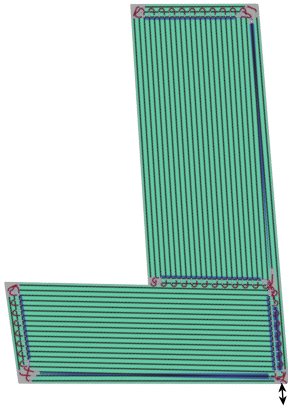
(j) S9 - satellite image



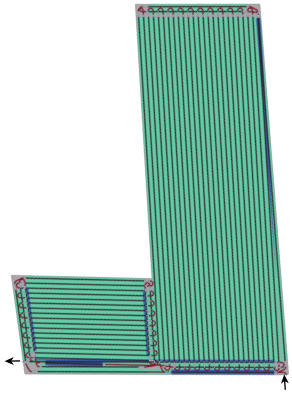


(l) C4 - most optimal solution

(m) C4 - satellite image



(n) C2 - most optimal solution



(o) C2 - most similar solution

(p) C2 - satellite image

FIGURE 6 Obtained results on illustration dataset and reference satellite image. The black arrows indicate where the robot enters and exits. (cont.)

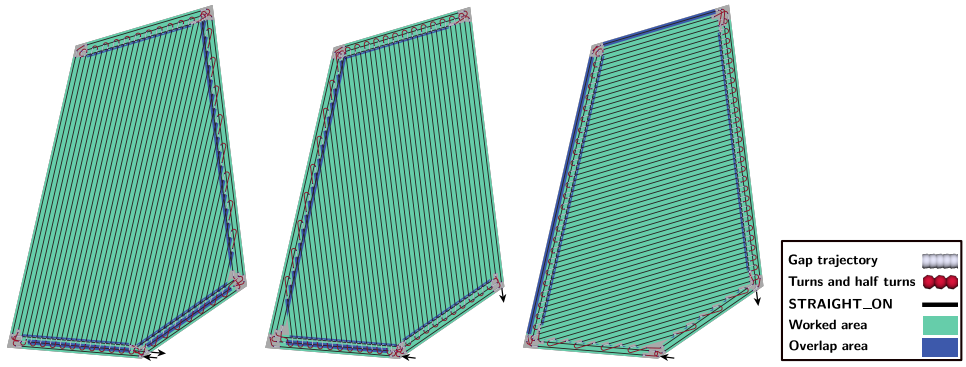
which dividing line leads to a better field decomposition. The results generated for the complex fields are represented in Figs. 6l, 6n and 6o.

The generation of the solution space for Fields C4 and C2 took almost 47 and 10 minutes respectively. Indeed for these fields, respectively nine and twelve successive explorations had to be performed to take into account all the combinations of entries and dividing lines. Computing in a preliminary step a smart polygon decomposition adapted to the agricultural use case, that considers not only the geometry of the field but also for instance the inclination, could avoid unnecessary explorations and speed up the process for complex fields.

As illustrated in Figs. 6l, 6n and 6m, after covering the main part and some parts of the headlands of the first sub-field, the path goes to the second sub-field and covers it entirely. Finally it comes again in the first sub-field and covers its remaining headlands and exits the field. This kind of solutions cannot be found in classic approaches that consist of two sequential steps (CPP and AVRP). This highlights the interest of our one-step approach. The following sections highlight some other interesting properties of the approach.

5.2.1 | Interest of grouping solutions into families

Fig. 7 illustrates the most optimal solution from three different families for Field S6. Table 5 summarizes the numerical results for each solution. As can be observed, the difference between the results for these families is rather small. However, as illustrated in Fig. 7 the solutions look completely different in terms of their general direction. Two of them have similar entry and exit, while the first one uses a different exit. This illustrates the interest of clustering the results into different families to identify and present a variety of good solutions to the farmer.



(a) S6 - first family (b) S6 - second family (c) S6 - third family

FIGURE 7 Most optimal solution for different families for Field S6. The black arrows indicate where the robot enters and exits

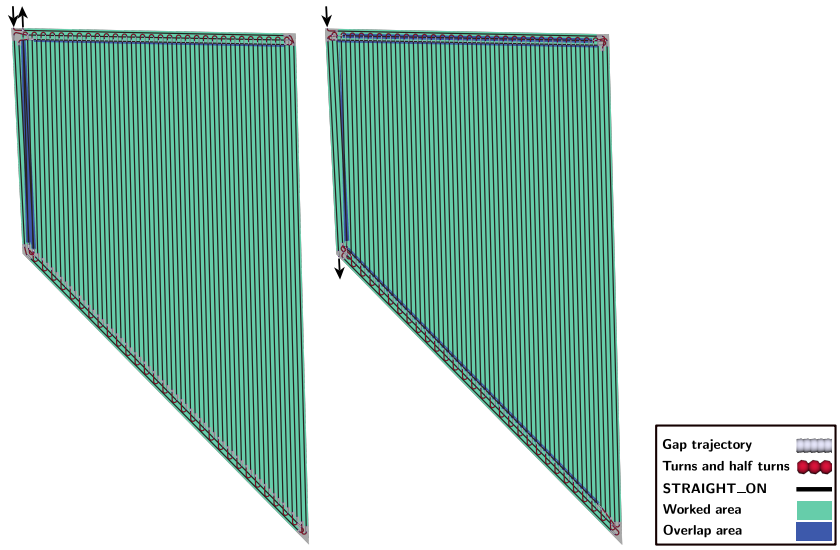
5.2.2 | Benefit of multiple entrances

Table 6 summarizes the numerical result for each entrance of Field S9 separately. The most optimal solution was through E_1. Considering E_2 as the entrance, our approach was also capable to find some acceptable results. However they were not as good as the one found while considering E_1. That means, taking into account more entrances enhances the probability of finding a better result. However it also increases the exploration time. In this specific

case, taking E_1 into account in addition to E_2 increased the coverage rate by 0.71% and nearly doubled the exploration time.

5.2.3 | Impact of field accessibility

Fig. 8 illustrates the most optimal solutions for two scenarios on Field S7: in the first one the field is only accessible from its upper edge, which corresponds to the real world scenario, and in the second one it is accessible from all its edges, which is an oversimplification. Table 7 summarizes the numerical results for each scenario. With the second scenario, we obtained a better solution according to the defined criteria. However, in the real world this solution would not be feasible, as the robot would exit to the neighbor's field at the bottom. Therefore, an inaccurate description of the accessibility of a field may lead to an unfeasible result. In some cases, it may damage the robot or the neighbor's field.



(a) S7 - one accessible edge (b) S7 - all edges are accessible

FIGURE 8 Most optimal solutions for Field S7 while considering it is only accessible from its upper edge (a) or from all its edges (b). The black arrows indicate where the robot enters and exits

5.2.4 | Forward and reverse half-turns

To demonstrate our method's ability to perform half-turns other than U-turns, one last test was performed on Field S7 where all parameters remained the same except γ_{off} that was set to $2m$. As illustrated in Fig. 9, our approach chose to perform half-turns using reverse moves on the left side of the field, while in the right side no reverse moves was required. Table 8 summarizes the numerical results for this test.

Field	Figure	$S_{cov} * W_{cov}$	$S_{ovl} * W_{ovl}$	$S_{nwd} * W_{nwd}$	$S_{otm} * W_{otm}$	f	Coverage	Overlap
S6	7a	0.000	0.081	0.038	0.049	0.168	98.43%	4.42%
S6	7b	0.083	0.050	0.059	0.045	0.237	98.23%	3.48%
S6	7c	0.312	0.099	0.181	0.094	0.685	97.68%	4.95%

TABLE 5 Numerical results for different families for Field S6. S_{cov}, S_{ovl}, S_{nwd}, S_{otm} and f are computed by equations 1, 2, 5, 4 and 6 respectively

Field	Entrance	Exploration		Similarity check and selection of optimal solutions						
		Solutions	time (s)	$S_{cov} * W_{cov}$	$S_{ovl} * W_{ovl}$	$S_{nwd} * W_{nwd}$	$S_{otm} * W_{otm}$	f	Coverage	Overlap
S9	E_1	168	39.79	0.000	0.099	0.073	0.085	0.256	97.77%	4.87%
S9	E_2	18	35.69	0.558	0.022	0.000	0.000	0.580	97.06%	3.09%

TABLE 6 Numerical results on Field S9 for each of its entrances

Field	accessible edges	Exploration		Similarity check and selection of optimal solutions		
		Solutions	time (s)	time (s)	$Coverage$	$Overlap$
S7	1 edge	2511	22.32	0.07	98.58%	1.80%
S7	all edges	10137	35.4729	0.30	98.61%	0.88%

TABLE 7 Comparison of results over Field S7 while considering it is accessible only by its upper edge (real world scenario) and all its edges

Field	Exploration			Similarity check and selection of optimal solutions							
	total	Solutions	time (s)	time (s)	$S_{cov} * W_{cov}$	$S_{ovl} * W_{ovl}$	$S_{nwd} * W_{nwd}$	$S_{otm} * W_{otm}$	f	Coverage	Overlap
S7	2	135	7.70	0.02	0.002	0.062	0.059	0.065	0.187	98.46%	2.96%

TABLE 8 Numerical results of Field S7 while γ_{off} was set to $2m$. S_{cov}, S_{ovl}, S_{nwd}, S_{otm} and f are computed by equations 1, 2, 5, 4 and 6 respectively

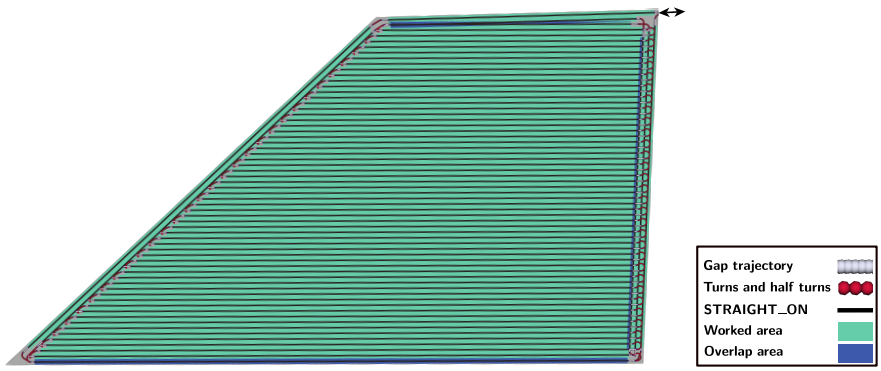


FIGURE 9 Most optimal solutions for Field S7 when γ_{off} was set to $2m$. The black arrows indicate where the robot enters and exits

5.2.5 | Ground truth and parameters

In this kind of study, it was difficult to compare our approach to previous approaches from the literature due to the very different nature of the previous approaches, the variety of constraints taken into account or not, and the lack of standardized dataset. We chose to visually compare our results to a ground truth made of satellite images. However, this choice also has some limitations, mostly due to the variety of the machinery, and the difference between the preferences and habits of a human operator and the constraints applicable to a robot. Nevertheless, we could demonstrate in this paper that our approach can provide complete optimal paths including headlands, comparable to the current practice, applicable to a large number of configurations and use cases, completely parameterizable, in a reasonable time.

6 | CONCLUSION AND PERSPECTIVES

In this paper, a novel CCPP approach based on tree exploration was proposed in order to generate an optimal path that starts from an entrance location, covers the field and the headlands and ends on one of accessible edges of the field. To accomplish this task, first, one or several explorations were conducted while considering multiple entrances and hard constraints. If some dividing lines were also provided, a combination of entrances and dividing lines were considered in the exploration. The result of this exploration was a solution space containing all possible solutions. Finally a similarity check and selection of optimal solutions was applied to extract a variety of most optimal paths without redundancies, by minimizing a weighted average cost function of the soft constraints. The goal of this approach was to maximize the worked area while minimizing the overlaps, the non-working path length and the traveled time.

This study revealed that considering multiple entrances could lead to better solutions. It was also shown that considering multiple dividing lines for fields with a complex shape increased the probability of finding a better solution. It also showed how critical it was to consider the actual accessibility of the field.

Currently, the proposed approach does not account for inclination and obstacles. Most often, modern robots include sensors to avoid moving obstacles, that could also be used to deal with static obstacles. However, this could cause some unwanted overlaps. A first perspective can be to consider the static obstacles inside a field beforehand, to avoid them with a more optimal route. The second perspective is to consider the inclination and the elevation across the field to correct overlaps and skips caused by the projection of a 3D surface to a 2D plane. This information could also be integrated as a new soft constraint to minimize the soil erosion and/or the energy consumption. Other perspectives include considering one or several mobile service units on the accessible edges of the field, and extension to a multi-robot approach.

references

Bochtis, D., Sørensen, C., Busato, P., Hameed, I., Rodias, E., Green, O. and Papadakis, G. (2010a) Tramline establishment in controlled traffic farming based on operational machinery cost. *Biosystems Engineering*, **107**, 221–231.

Bochtis, D., Sørensen, C. and Vougioukas, S. (2010b) Path planning for in-field navigation-aiding of service units. *Computers and Electronics in Agriculture*, **74**, 80–90.

Bochtis, D. and Sørensen, C. G. (2009) The vehicle routing problem in field logistics part i. *Biosystems engineering*, **104**, 447–457.

— (2010) The vehicle routing problem in field logistics: Part ii. *Biosystems engineering*, **105**, 180–188.

Bochtis, D. D., Sørensen, C. G., Busato, P. and Berruto, R. (2013) Benefits from optimal route planning based on b-patterns. *Biosystems Engineering*, **115**, 389–395.

Burger, M., Huiskamp, M. and Keviczky, T. (2013) Complete field coverage as a multi-vehicle routing problem. *IFAC Proceedings Volumes*, **46**, 97–102.

Cao, Y., Han, Y., Chen, J., Liu, X., Zhang, Z. and Zhang, K. (2019a) Optimal coverage path planning algorithm of the tractor-formation based on probabilistic roadmaps. In *2019 IEEE International Conference on Unmanned Systems and Artificial Intelligence (ICUSAI)*, 27–32. IEEE.

— (2019b) A tractor formation coverage path planning method based on rotating calipers and probabilistic roadmaps algorithm. In *2019 IEEE International Conference on Unmanned Systems and Artificial Intelligence (ICUSAI)*, 125–130. IEEE.

Cariou, C., Gobor, Z., Seiferth, B. and Berducat, M. (2017) Mobile robot trajectory planning under kinematic and dynamic constraints for partial and full field coverage. *Journal of Field Robotics*, **34**, 1297–1312.

Cariou, C., Laneurit, J., Roux, J.-C. and Lenain, R. (2020) Multi-robots trajectory planning for farm field coverage. In *2020 16th International Conference on Control, Automation, Robotics and Vision (ICARCV)*, 351–356. IEEE.

Conesa-Muñoz, J., Bengochea-Guevara, J. M., Andujar, D. and Ribeiro, A. (2015) Efficient distribution of a fleet of heterogeneous vehicles in agriculture: a practical approach to multi-path planning. In *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, 56–61. IEEE.

— (2016a) Route planning for agricultural tasks: A general approach for fleets of autonomous vehicles in site-specific herbicide applications. *Computers and Electronics in Agriculture*, **127**, 204–220.

Conesa-Muñoz, J., Pajares, G. and Ribeiro, A. (2016b) Mix-opt: A new route operator for optimal coverage path planning for a fleet in an agricultural environment. *Expert Systems with Applications*, **54**, 364–378.

Dogru, S. and Marques, L. (2015a) Energy efficient coverage path planning for autonomous mobile robots on 3d terrain. In *2015 IEEE International Conference on Autonomous Robot Systems and Competitions*, 118–123. IEEE.

— (2015b) Towards fully autonomous energy efficient coverage path planning for autonomous mobile robots on 3d terrain. In *2015 European Conference on Mobile Robots (ECMR)*, 1–6. IEEE.

Dubins, L. E. (1957) On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents. *American Journal of mathematics*, **79**, 497–516.

Edwards, G. T., Hinge, J., Skou-Nielsen, N., Villa-Henriksen, A., Sørensen, C. A. G. and Green, O. (2017) Route planning evaluation of a prototype optimised infield route planner for neutral material flow agricultural operations. *Biosystems Engineering*, **153**, 149–157.

Evans IV, J. T., Pitla, S. K., Luck, J. D. and Kocher, M. (2020) Row crop grain harvester path optimization in headland patterns. *Computers and Electronics in Agriculture*, **171**, 105295.

Giannakis, E., Kushta, J., Giannadaki, D., Georgiou, G. K., Bruggeman, A. and Lelieveld, J. (2019) Exploring the economy-wide effects of agriculture on air quality and health: evidence from europe. *Science of The Total Environment*, **663**, 889–900.

Géoportail () The national portal for territorial knowledge implemented by IGN. `https://www.geoportail.gouv.fr/`.

Hameed, I., Bochtis, D., Sørensen, C. and Nøremark, M. (2010) Automated generation of guidance lines for operational field planning. *Biosystems engineering*, **107**, 294–306.

Hameed, I. A. (2014) Intelligent coverage path planning for agricultural robots and autonomous machines on three-dimensional terrain. *Journal of Intelligent & Robotic Systems*, **74**, 965–983.

— (2017) Coverage path planning software for autonomous robotic lawn mower using dubins' curve. In *2017 IEEE International Conference on Real-time Computing and Robotics (RCAR)*, 517–522. IEEE.

Hameed, I. A., Bochtis, D. and Sorensen, C. (2011) Driving angle and track sequence optimization for operational path planning using genetic algorithms. *Applied Engineering in Agriculture*, **27**, 1077–1086.

Hameed, I. A., la Cour-Harbo, A. and Osen, O. L. (2016) Side-to-side 3D coverage path planning approach for agricultural robots to minimize skip/overlap areas between swaths. *Robotics and Autonomous Systems*, **76**, 36–45.

Jensen, M. A. F., Bochtis, D., Sørensen, C. G., Blas, M. R. and Lykkegaard, K. L. (2012) In-field and inter-field path planning for agricultural transport units. *Computers & Industrial Engineering*, **63**, 1054–1061.

Jensen, M. F., Bochtis, D. and Sørensen, C. G. (2015a) Coverage planning for capacitated field operations, part ii: Optimisation. *Biosystems Engineering*, **139**, 149–164.

Jensen, M. F., Nørremark, M., Busato, P., Sørensen, C. G. and Bochtis, D. (2015b) Coverage planning for capacitated field operations, part i: Task decomposition. *Biosystems Engineering*, **139**, 136–148.

Jeon, C.-W., Kim, H.-J., Yun, C., Han, X. and Kim, J. H. (2021) Design and validation testing of a complete paddy field-coverage path planner for a fully autonomous tillage tractor. *Biosystems Engineering*, **208**, 79–97.

Jin, J. and Tang, L. (2010) Optimal coverage path planning for arable farming on 2d surfaces. *Transactions of the ASABE*, **53**, 283–295.

— (2011) Coverage path planning on three-dimensional terrain for arable farming. *Journal of field robotics*, **28**, 424–440.

Khajepour, A., Sheikhmohammady, M. and Nikbakhsh, E. (2020) Field path planning using capacitated arc routing problem. *Computers and Electronics in Agriculture*, **173**, 105401.

Marques, A., Martins, I. S., Kastner, T., Plutzar, C., Theurl, M. C., Eisenmenger, N., Huijbregts, M. A., Wood, R., Stadler, K. and Bruckner, M. (2019) Increasing impacts of land use on biodiversity and carbon sequestration driven by population and economic growth. *Nature ecology & evolution*, **3**, 628–637.

Meftaul, I. M., Venkateswarlu, K., Dharmarajan, R., Annamalai, P. and Megharaj, M. (2020) Pesticides in the urban environment: A potential threat that knocks at the door. *Science of The Total Environment*, **711**, 134612.

Nilsson, R. S. and Zhou, K. (2020) Method and bench-marking framework for coverage path planning in arable farming. *Biosystems Engineering*, **198**, 248–265.

Oksanen, T. and Visala, A. (2009) Coverage path planning algorithms for agricultural field machines. *Journal of field robotics*, **26**, 651–668.

Oksanen, T. et al. (2007) *Path planning algorithms for agricultural field machines*. Helsinki University of Technology.

OpenMP () The OpenMP API supports multi-platform shared-memory parallel programming in C/C++ and Fortran. `https://www.openmp.org/`.

O'Rourke, J., Aggarwal, A., Maddila, S. and Baldwin, M. (1986) An optimal algorithm for finding minimal enclosing triangles. *Journal of Algorithms*, **7**, 258–269.

Plessen, M. G. (2019a) Freeform path fitting for the minimisation of the number of transitions between headland path and interior lanes within agricultural fields. *arXiv preprint arXiv:1910.12034*.

— (2019b) Optimal in-field routing for full and partial field coverage with arbitrary non-convex fields and multiple obstacle areas. *Biosystems Engineering*, **186**, 234–245.

Plessen, M. M. G. (2018) Partial field coverage based on two path planning patterns. *Biosystems engineering*, **171**, 16–29.

Reeds, J. and Shepp, L. (1990) Optimal paths for a car that goes both forwards and backwards. *Pacific journal of mathematics*, **145**, 367–393.

Seyyedhasani, H. and Dvorak, J. S. (2017) Using the vehicle routing problem to reduce field completion times with multiple machines. *Computers and Electronics in Agriculture*, **134**, 142–150.

— (2018a) Dynamic rerouting of a fleet of vehicles in agricultural operations through a dynamic multiple depot vehicle routing problem representation. *biosystems engineering*, **171**, 63–77.

— (2018b) Reducing field work time using fleet routing optimization. *Biosystems Engineering*, **169**, 1–10.

Seyyedhasani, H., Dvorak, J. S. and Roemmele, E. (2019) Routing algorithm selection for field coverage planning based on field shape and fleet size. *Computers and electronics in agriculture*, **156**, 523–529.

Shen, M., Wang, S., Wang, S. and Su, Y. (2020) Simulation study on coverage path planning of autonomous tasks in hilly farmland based on energy consumption model. *Mathematical Problems in Engineering*, **2020**.

Utamima, A., Reiners, T. and Ansaripoor, A. H. (2019a) Evolutionary estimation of distribution algorithm for agricultural routing planning in field logistics. *Procedia Computer Science*, **161**, 560–567.

— (2019b) Optimisation of agricultural routing planning in field logistics with evolutionary hybrid neighbourhood search. *Biosystems Engineering*, **184**, 166–180.

Vahdanjoo, M., Zhou, K. and Sørensen, C. A. G. (2020) Route planning for agricultural machines with multiple depots: Manure application case study. *Agronomy*, **10**, 1608.

Zhou, K. and Bochtis, D. (2015) Route planning for capacitated agricultural machines based on ant colony algorithms. In *HAICTA*, 163–173.

Zhou, K., Jensen, A. L., Bochtis, D., Nørremark, M., Kateris, D. and Sørensen, C. G. (2020) Metric map generation for autonomous field operations. *Agronomy*, **10**, 83.

Zuo, G., Zhang, P. and Qiao, J. (2010) Path planning algorithm based on sub-region for agricultural robot. In *2010 2nd International Asia Conference on Informatics in Control, Automation and Robotics (CAR 2010)*, vol. 2, 197–200. IEEE.