I solve the CPP problem partially, and time are the only open...


time are the only open-

INTRODUCTION

I solve the CPP problem partially, and time are the only open-

INTRODUCTION

I develop countries, there is a shortage of skilled workers
to operate agricultural machinery [1]. This shortage can be al-
leviated with the development of autonomous machinery. Unlike
manually operated machinery, autonomous vehicle operations
need meticulous planning beforehand. The problem of deter-
mining a path to cover a field is known as coverage path planning
(CPP). CPP is of high importance for cleaning [2], surveillance
robots [3], lawn mowers [4], and agricultural vehicles [5], where
it has been addressed in several works.

Whilst there have been many efforts, most of the (partial)
CPP solutions have not been released as open-source software
thus hindering more rapid advances in CPP by the scientific
community. The packages shown in Table I are the only open-
source software to the best of our knowledge. Note that the soft-
ware packages listed in Table I solve the CPP problem partially,
but require several modifications in order to be customized to
different unmanned vehicles and applications.

This letter aims to fill the above mentioned gap by proposing
and releasing to the community an open-source CPP library
for field coverage. The library was designed focusing in four
modules that are the core of CPP solutions: a headland generator,
a swath generator, a route planner and a path planner. Each
module includes at least one state-of-the-art method and one
objective function. The library currently only supports convex
fields on arable farmland. Regardless, there is an urgent need for
an open source software solution to fill the existing gap in the
CPP problem in agriculture. The ultimate goal of the library is to
ease the state-of-the-art algorithm benchmark and to accelerate
CPP research and application.

A. Related Work

Owing to the non-holonomous nature of agricultural vehicles,
a region of the field known as headlands must be reserved for
turning the vehicle. The most basic approach is to allocate a
constant width area around the field. This strategy allocates a
large amount of space to a poor yield area. Depending on how
the swaths are arranged, some headland areas are parallel to
the swaths and hence they are not needed for turning. By only
constructing headlands along the field edges where turns are
made, the area reserved for them can be minimized [5], [17].

Swaths are generated in the inner field, which is the remain-
ing region after subtracting the headlands. In two-dimensional
planar fields, a reference line can be applied as a guide for the
generation of swaths, where each parallel creates a swath [5],
[17], [18]. This line can be chosen for convenience or by an
algorithm such as brute force or a meta-heuristic. Oksanen [5]
describes a driving angle search strategy that requires fewer
iterations than brute force search but it does not guarantee finding
the global minimum. Objective functions such as the number
of turns or the sum of swath lengths are used to determine
optimality in swath generation [17].

The distance [17] and time [19] required to cover the field
are affected by the order of the swaths. A route is the sequence
of the swaths to cover. The Boustrophedon order, which trav-
els the swaths sequentially from one side of the field to the
other, and the snake order, which skips one swath at each turn
and returns through the uncovered swaths, are popular preset
or presents an approach for calculating or clothoids which, although the are employed to smooth the turn only compute the route to cover a are minimized by finding the optimal route through or Reeds-Shepp’s can be adopted. 



\[ \text{Table I} \]

| Package name                  | Docs | No grid used | Non-holonomous | Headlands support | Customizable objective function | Terrestrial agricultural vehicles |
|-------------------------------|------|--------------|----------------|-------------------|---------------------------------|----------------------------------|
| RTJwp/coveragePlanning [6]    | ✓    | ✓            | ✗              | ✗                | ✗                              | ✗                                |
| Nobleo/full_coverage_path_planner [7] | ✓    | ✗            | ✓              | ✗                | ✗                              | ✗                                |
| Ipas200/ips_coverage_planning [5], [8], [9] | ✓    | ✓            | ✗              | ✗                | ✗                              | ✗                                |
| Efiz-ssl/polygon_coverage_planning [10], [11] | ✓    | ✓            | ✓              | ✗                | ✗                              | ✗                                |
| Ivingyvasquez/ocpp [12], [13] | ✓    | ✓            | ✓              | ✗                | ✗                              | ✗                                |
| Greenzie/boustrophedon_planner [14] | ✓    | ✓            | ✓              | ✗                | ✗                              | ✗                                |
| Ipiano/coverage_planning [15], [16] | ✓    | ✓            | ✓              | ✗                | ✗                              | ✗                                |
| **Fields2Cover**             | ✓    | ✓            | ✓              | ✗                | ✗                              | ✗                                |

routing patterns [20]. Objective functions such as distance, number of rotations, or time necessary to traverse the field [17], [19] are minimized by finding the optimal route through meta-heuristics [21].

A path is composed of the swaths of a route connected by turns, forming a continuous line along which the vehicle will drive. Dubins’ [22] or Reeds-Shepp’s [23] curves are turns that minimize the path length of the turns. These curves are made by either curve segments or straight lines. The main problem is that there is an instantaneous change of curvature at the transition point between two segments. Techniques such as numerical integrators [24] or clothoids [25] are employed to smooth the turn to avoid the curvature discontinuity. Furthermore, to navigate from a swath to the headlands, turns such as non-uniform rational B-spline (NURBS) curves [26] can be adopted.

CPP problems are composed of numerous sub-problems, several of which have received special attention in literature. For example, Spekken [21] presents an approach for calculating the coverage path in undulating terrain that however does not consider turns between rows or headland creation. Nilsson [27] and Nørremark [28] divide the CPP problem into two major modules: Field Partitioning/Representation, where the distribution of headlands and swaths in the field is set up, and Route Planning, which determines the optimal order of travelling the swaths within sub-fields. In the latter framework, each module has more than one function, increasing the complexity of comparing multiple variations of the module.

B. Existing Open-Source Software

There have been web applications, such as GAOS [18], that allowed farmers to design or adapt coverage paths with a user-friendly interface. Many of such web applications, despite being a great help to the farming community, have been developed in collaboration with companies, restricting the possibility to release the code to the public domain.

The currently existing open source CPP repositories are listed in Table I. Although seven other projects were found, none of them can be adopted for farming purposes with ground robots. As mentioned above, ground robots in agriculture are generally non-holonomous, so turning maneuvers must be planned to move from one swath to another. Unfortunately, some packages [6], [10], [12], [15] only compute the route to cover a region. These packages are designed for quadrotors [10], [12] or for indoor robots [6]. However, the code needs to be modified to support path generation for non-holonomous robots. A special case of CPP is the Nobleo package [7] which, although the vehicle used is non-holonomous, uses a grid to define the nodes that should be covered at least once. In agriculture it is important to reduce the damage caused by the wheels of the vehicle, so it is not recommended to cover the same swath several times [7] or to cross through the main field [6], [8], [10]. On the other hand, Greenzie [14], which was developed for lawn mowers, is the only package that supports headlands, along with Fields2Cover. Unlike arable farming, mowers are constrained to avoid repeated tracks for field traffic, thus the coverage path is created with random sweep angles. For this reason, Greenzie does neither provide an optimizer nor an objective function for planning the swaths. In contrast, Ipiano [15] provides an interface to change the objective function used by its optimizer, but here no headland support is offered. Fields2Cover is the only software solution that provides algorithms to create a coverage path for terrestrial agricultural robots, including optimizers and objective functions to generate the best path, headland support and turn planning.

C. Contributions

The main contributions of this letter and the Fields2Cover library are:

1. A publicly-available library (Fields2cover) providing connectable modules to address CPP problems with unmanned agricultural vehicles. Those modules can be effortlessly customized for other CPP problems.
2. Benchmark tools for quantitative comparison between the CPP algorithms and approaches.
3. A quantitative comparison using 38 convex fields between eight state-of-the-art CPP approaches/methods and seven objective functions.
4. Experiments with a commercial unmanned agricultural vehicle demonstrating Fields2Cover’s capability to provide real-world solutions.
Fig. 1. Diagram of the Fields2Cover library. The library contains four core modules: 1) Headland Generator, 2) Swath Generator, 3) Route Planner and 4) Path Planner. Each of the modules is represented as a box with 3 slots: the name of the module, the methods implemented, and the objective functions available. The output of a module is the input of the next module.

5) Contribute to building a research community by providing an accessible platform for discussing novel ideas, solving issues and sharing new variants of the problem.

II. FIELDS2COVER

Fields2Cover is designed in four modules (Fig. 1): 1) Headland Generator, 2) Swath Generator, 3) Route Planner and 4) Path Planner. Each of the modules is represented as a box with 3 slots: the name of the module, the methods implemented, and the objective functions available. The output of a module is the input of the next module.

A. Headland Generator Module

The Headland Generator module currently implements a single method that buffers the border of the field in inward direction by a custom constant width (see Module 1 in Fig. 1). The objective function of this module is the area of the remaining field after removing the headlands.

\[ A_r = \frac{A_{hl}}{A_f} \]  

where \( A_{hl} \) is the area of the field without headlands (mainland), \( A_f \) the area of the original field, and \( A_r \) is the ratio of mainland to the original field.

B. Swath Generator Module

The inner field (i.e., excluding the headlands) is the input of the Swath Generator module (see Module 2 in Fig. 1). This region is divided into parallel swaths matching the operating width. In the current version, the library only supports parallel non-overlapping swaths. Fields2Cover has a brute force algorithm to find the optimal sweep angle by trying discretized angles using a given step size. If the computer running the library supports multiple threads, several sweep angles are tried in parallel [29].

This module currently implements 3 objective functions:
- Minimize the Number of Swaths. This objective function depends on the shape and the area of the field, and the width of the robot. The number of swaths is limited by the equation:

\[ 0 \leq \#S_\alpha \leq \frac{A_{hl}}{R_w}, \]  

where \( \#S_\alpha \) is the number of swaths for a given sweep angle \( \alpha \), \( A_{hl} \) is the area of the field without headland, and \( R_w \) is the operational width of the robot. The shape of the field that maximizes the minimum number of swaths is the square field, which results in:

\[ \min_\alpha \#S_\alpha^0 \simeq \sqrt{\frac{A_{hl}}{R_w}}, \]  

where \( \#S_\alpha^0 \) is number of swaths in a square field with a given sweep angle \( \alpha \). Therefore, the optimal value of this objective function is less than the square root of the area of the field.

- Maximize the Field Coverage:

\[ A_{cov} = \frac{A_{hl} \cap \bigcup S_i}{A_{hl}}, \]  

where \( A_{cov} \) is the fraction of area covered, \( A_{hl} \) is the field without headlands, \( S_i \) is the ith swath, \( \cap \) is the intersection operator, and \( \bigcup S_i \) is the union of all the swaths.

- Minimize the Swaths Length:

\[ \sum_i \text{length}(S_i) = \sum_i \sum_j ||S_{ip+j+1}^i - S_{ip+j}^i||_2, \]  

where \( \sum_i \text{length}(S_i) \) is the sum of the length of the swaths, \( N \) is the number of swaths, \( S_p^i \) is the number of points that the ith swath has, \( i_{p+j}^i \) is the jth point of the ith swath, and \( ||x||_2 \) is the Euclidian norm.

C. Route Planner Module

The Route Planner module uses the swaths created earlier to produce the route (see Module 3 in Fig. 1). Fields2Cover contains several predefined route patterns, which include the boustrophedon pattern, the snake pattern, the spiral pattern and...
a custom pattern. The Boustrophedon pattern covers the swaths sequentially, and the Snake pattern skips one swath each time to traverse the field in one direction and returns through covering the uncovered swaths. The Spiral pattern is a variation of the Snake pattern, that sorts the swaths in clusters of a fixed size with the snake pattern. The custom pattern requires specification of the swath order by the user. To compare different routes, the library provides as objective function the length of the path generated by the Path Planning module. It also computes the path length with in-place turns, which correspond to zero turning radius. The path length of in-place turns is computed as:

$$L_0 = \sum_{i=1}^{N} length(S^i) + \sum_{i=2}^{N} ||S^i_p - S^{i-1}_p||_2$$ (6)

Where $L_0$ is the path length with in-place turns, $\sum_{i=1}^{N} length(S^i)$ is the sum of the lengths of the swaths, $N$ is the number of swaths, $S^i_1$ is the first point of the $i$th swath, $S^i_p$ is the last point of the $i$th swath, and $||x||_2$ is the Euclidian norm.

D. Path Planner Module

The inputs of the Path Planner module (see Module 4 in Fig. 1) are the route (sorted swaths) and the vehicle parameters. Once the route is known, the turns to complete the path are computed. In the current version of the library, the path planner applies the same type of curves for all the headland turns. Fields2Cover currently supports straight curves, the Dubins’ curves [22] and the Reeds-Shepp’s [23] curves, using the path length as the single objective function.

E. ROS Wrapper

Although the Fields2Cover library does not depend on ROS, an interface with ROS1 and ROS2 is provided as an add-on. The fields2cover_ros package provides functions that convert Fields2Cover data types into ROS messages. Services are created to execute modules directly from ROS topics. Launch files are used to script examples of the package. RVIZ-support is also provided to visualize the results of the modules. Methods, objective functions and parameters can be modified on real time thanks to rqt_reconfigure.²

F. Design & Implementation

Fields2Cover is implemented using C++17, with a Python interface using Swig [30], and released under BSD-3 license. The design of Fields2Cover aims to serve both scientists and service providers and is intended to be easily used.

The reason for making Fields2Cover an open-source library is that doing so encourages the development of additional functionality by providing the code to the community. Likewise, Fields2Cover widely employs open-source libraries from third parties to streamline the development process of state-of-the-art algorithms. For scientists, priority is given to a flexible design, which allows to extend or modify existing algorithms. Additionally, a benchmark against which to compare new solutions is added. For service providers, utility concerns the ability to plan the best coverage path for a given objective function in a straightforward manner. The modularity of Fields2Cover is key to ensure its usefulness for both cases. In addition, the library provides tests, tutorials, and extended documentation³ to reduce the learning curve.

III. RESULTS

Several experiments were conducted to demonstrate the functionalities of Fields2Cover. Firstly, coverage paths were created for convex fields from the Nilsson’s benchmark [27]. In these simulations, the experiments focus on the optimization of the objective functions and the computation time of those methods. Secondly, real field experiments were conducted in an agricultural field with a commercial robot (Fig. 2) of the company AgXeed B.V (The Netherlands). The aim of the experiment was to program the coverage trajectory of the robot using the Fields2Cover library and assess whether a designed coverage path is efficiently traversed by the robot. The planned path is previously transferred to the robot with Protobuf [31]. The protobuf message defines the path as timestamps, positions, velocities and orientations. It also contains the geometry of the field boundary to prevent the vehicle from leaving the field. The sensor data collected during the coverage path, such as the GNSS position and the velocity, is returned from the AgBot as a rosbag [32].

Experiments were done with a laptop MSI GF627RE with Intel(R) Core(TM) i7-7700HQ CPU @ 2.80 GHz (4 cores, 8 threads) with Ubuntu 20.04.5.

1[Online]. Available: https://github.com/Fields2Cover/fields2cover_ros
2[Online]. Available: http://wiki.ros.org/rqt_reconfigure
3[Online]. Available: https://fields2cover.github.io/
A. Simulation Results

Three simulation experiments were performed. Firstly, the optimal route was computed for three different fields to visually inspect the effects of the objective function. Secondly, the coverage path was computed for 38 convex fields with every possible combination of the algorithms provided by the library. The combination of algorithms for creating a coverage path were compared using the path length as the objective function. Thirdly, the time for computing coverage paths was recorded using several objective functions of the Swath Generator module. The relationship between the area of the field and the computation time was found.

The first decision for coverage path planning of a field is the objective function to be optimised by the swath generator (Brute force algorithm). The optimal pitch angle of the swaths may vary with the chosen objective function. Therefore, the first experiment provides examples of optimal swaths for the fields REC_A, CIR_B and SAL_B from the Nilsson’s benchmark [27], which are shown in Table II. The fields were re-scaled to an area of 100 m². If the number of swaths is minimized, the number of turns is also reduced. For instance, fields CIR_B and SAL_B are covered using a single turn. If maximum field coverage is to be achieved, CIR_B needs seven turns while SAL_B needs five. Field coverage is typically achieved when swaths are parallel or perpendicular to one of the edges. In contrast, the swath-length objective function may produce many short swaths (bottom-left of CIR_B with swath length), that reduce the total length of the swaths.

The second experiment was conducted using 38 convex fields of the Nilsson’s benchmark [27], re-scaled to an area of 1 ha (Fig. 3). For each field, a headland of 7.5 m (three times the operational width of the robot) was generated with the constant width generator. Next, the brute force algorithm was used to generate the optimal swaths for each objective function criterion shown in Table II. The route planners sorted the swaths with the boustrophedon, snake or spiral (bulk of 6 swaths) pattern. Lastly, the computed path length \( L_i \) was used for comparing the coverage paths computed with Dubins’ and Reeds-Shepp’s curves against the length of paths with in-place turns \( L_0 \), which have the least possible path length for a holonomic vehicle. Each column of Fig. 3 refers to the objective function criterion that was optimized by the swath generator and each row denotes a particular route planner pattern. A subplot represents the optimized path length (y-axis) with respect to the optimised path length for in-place turns (x-axis). A single dot represents a computed coverage path, with position \( (L_0, L_i) \) for the ith coverage path. The color of the dots denotes the type of curve (Dubin or Reed-Shepp), while black lines represent the 1:1 relation. Greater values for \( L_0 \) imply that the route generated is longer and distance between swaths is larger. The difference between \( L_1 \) and \( L_0 \) manifests the length of the turns. This difference relates to the time that the machinery is non-productive. Therefore, a substantial difference between the black line and the coloured dots denotes a path for which turning takes more time. As shown in Fig. 3, a percentage between 0.5% and 50% of the coverage path was spent on turns. When the number of turns is reduced, the distance traveled is reduced accordingly. The distance used for turning increases when the boustrophedon pattern is applied since a shorter width between swaths requires a larger turn to comply with the minimum turning radius requirement. For instance, in the first column of the Fig. 3, the difference between the path length using Dubins’ curves and in-place turns is smaller than in the other columns. Field coverage and swath length behaved equally in terms of coverage path length. With any of the objective functions presented, the boustrophedon pattern produced the shortest pattern with in-place turns, the snake pattern was the second and the spiral pattern the longest. The length of the boustrophedon pattern increases when the minimum turning radius is required.

In the last simulated experiment, the computation time of planning a coverage path was measured in relation to the area of the field and the objective function of the swath generator (Fig. 4). The constant headland width was set to three times the width of the robot. Next, the parallel brute force algorithm optimized the pitch angle of the swaths, which were sorted using a boustrophedon pattern. Finally, the path planner used Dubins’ curves to create the coverage path. This experiment measures the computation time for coverage paths according to the three swath generator’s objective functions in relation to the area of the field. Fields2Cover computed a coverage path for a field of 1 ha in less than 3.5 seconds using Field coverage as the objective function, while only 0.5 seconds were needed using the number of swaths or the swath length as the objective functions. Since the computation of the number of swaths and swath length is proportional to the number of swaths and the number of swaths is proportional to the width of the field perpendicular to the driving direction, the computational time grows proportional to the square root of the area of the field. The computation time using the latter two objective functions can be approximated by:

\[
T_c = C_0 \times \sqrt{\frac{A_{lt}}{R_w}} + C_1
\]  

(7)

where \( T_c \) is the computation time, \( C_0 \) and \( C_1 \) are constants, \( A_{lt} \) is the area of the field, and \( R_w \) is the operational width of the robot. This relationship is only true when the field is convex so it can be covered with a single pattern.
Fig. 3. Coverage path length comparison. Columns refer to the objective function that was optimized using the brute force algorithm, while each row refers to a particular route planner pattern. Each subplot represents the optimized path length (y-axis) with respect to the optimised path length for zero-radius turns (x-axis). A single dot represents a coverage path length for a field with a chosen size from the Nilsson’s benchmark [27]. Red dots correspond to paths using Dubins’ curves, blue dots are for Reeds-Shepp’s curves. The black lines show the 1:1 relations.

Fig. 4. Time required to compute a path according to the objective function used. Algorithms used are constant width headland generator, parallel brute force for swath generation, Boustrophedon route order and Dubins’ curves.

The field coverage is computationally the most demanding objective function because it computes the difference between the field and the union of the areas of each swath. The computation time of this objective function grows linearly with the area of the field. Geometrical operations such as ‘difference’ and ‘union’ are more expensive than returning the number of swaths, which is the size of the vector of swaths.

Computational time analysis focused on the objective function of the brute force algorithm which consumes more than 80% of the total time of the coverage path planning.

B. Field Experiment

A field experiment was conducted using the AgBot shown in (Fig. 2). In the extreme case shown in Fig. 5, the AgBot covered an elongated narrow area. Objective functions like the minimum swath length or the number of turns would produce swaths parallel to the longest edge of the field. However, here we show a coverage pattern given a custom angle that allows observing the turns in the field. The produced swaths were sorted using the Snake pattern and connected by Dubins’ curves. The difference between the planned path and the recorded track in Fig. 5 can be attributed to the planned minimum turning radius being shorter than permissible for the AgBot. Therefore, the recorded GNSS data show slightly wider turns than the planned path. Turns made with the snake pattern always skip one swath, except for the turn at the rightmost part of the field where the coverage direction changed. This turn is sharper, causing wider tracks on the ground, greater soil slippage, and thus more soil damage [33]. Despite the slippage, the AgBot was capable of covering the field with the path designed by the library routines.

IV. Conclusion

In this work, we introduced Fields2Cover, a Coverage Path Planning open-source library for agricultural vehicles. Fields2Cover was implemented to bundle the research knowledge on this topic and to help other developers to accelerate their
projects. Currently, it supports the creation of coverage paths for convex fields, with a flexible and simple structure thanks to its modular design. The library has four modules, which are: the headlands generator, with a constant width headlands generator; the swath generator, with a brute force optimizer; the route planner, with three types of patterns; and the path planner, with Dubins’ and Reed-Shepp’s curves. The last three modules have their own objective functions specific to their domains. Field2Cover was tested using simulation with a public benchmark and in a real field.

Field2Cover is an on-going project, which means the functionality of the library will be expanded in the coming years. Future developments are supported and maintained by the first author of this letter, with the collaboration of the open-source community. Assumptions, such as flat topography, convex fields, enough maneuvering space in the headlands, absence of capacity limits and planning for arable crops, were made to reduce complexity for the first release. Non-convex fields with obstacles, sloping land, capacitated vehicles and permanent crops such as orchards provide challenges for further research and development, part of which are considered within the Field2Cover project.

Since the release of this software library, a community has formed around the coverage path planning problem. In less than 6 months since its release, many developers have shown their support for this project through assigning 150 github stars, code contributions and by providing suggestions for future versions. This community support shows the value of the project, which emphasizes the relevance of addressing the coverage path planning problem.

REFERENCES

[1] L. Christiaensen, Z. Rutledge, and J. E. Taylor, “The future of work in agriculture: Some reflections,” World Bank Policy Res. Working Paper. 9193, 2020.
[2] R. Bormann, F. Jordan, J. Hampp, and M. Hägelse, “Indoor coverage path planning: Survey, implementation, analysis,” in Proc. IEEE Int. Conf. Robot. Automat., 2018, pp. 1718–1725, doi: 10.1109/ICRA.2018.8460566.
[3] K. R. Jensen-Nau, T. Hermans, and K. K. Leang, “Near-optimal area-coverage path planning of energy-constrained aerial robots with application in autonomous environmental monitoring,” IEEE Trans. Automat. Sci. Eng., vol. 18, no. 3, pp. 1453–1468, Jul. 2021, doi: 10.1109/TASE.2020.3016276.
[4] I. A. Hameed, “Coverage path planning software for autonomous robotic lawn mower using Dubins’ curve,” in Proc. IEEE Int. Conf. Real-Time Comput. Robot., 2017, pp. 517–522, doi: 10.1109/RCTR.2017.8311915.
[5] T. Oksanen and A. Visala, “Coverage path planning algorithms for agricultural field machines,” J. Field Robot., vol. 26, no. 6, pp. 651–668, 2009, doi: 10.1002/rob.20300.
[6] R. Jiapin, “Rjxjz/coverageplanning github,” Accessed: Feb. 17, 2023. [Online]. Available: https://github.com/Rjxjz/CoveragePlanning.
[7] T. Clephas, C. López, and Nobeelo, “Nobeelo/full_coverage_path_planner github,” Accessed: Feb. 17, 2023. [Online]. Available: https://github.com/nobeelo/full_coverage_path_planner.
[8] F. IPA, “Ipa320/ipv_coverage_planning github,” Accessed: Feb. 17, 2023. [Online]. Available: https://github.com/ipa320/ipv_coverage_planning.
[9] R. Bormann, F. Jordan, W. Li, J. Hampp, and M. Hägelse, “Room segmentation: Survey, implementation, and analysis,” in Proc. IEEE Int. Conf. Robot. Automat., 2016, pp. 1019–1026, doi: 10.1109/ICRA.2016.7487234.
[10] R. Bachneman, L. Liu, and D. Kleiser, “Ezh-asl/polygon_coverage_planning github,” Accessed: Feb. 17, 2023. [Online]. Available: https://github.com/ezh-asl/polygon_coverage_planning.

[11] R. Bahnemann, N. Lawrence, J. J. Chung, M. Pantic, R. Siegwart, and J. Nieto, “Revisiting boustrophedon coverage path planning as a generalized traveling salesman problem,” in Field and Service Robot., Berlin, Germany: Springer, 2021, pp. 277–290, doi: 10.1007/978-981-15-9460-1_20.
[12] J. L. Vasquez, “Irvingvasquez/ocpp github,” Accessed: Feb. 17, 2023. [Online]. Available: https://github.com/Irvingvasquez/ocpp.
[13] J. I. V. Gomez, M. M. Melchor, and J. C. H. Lozada, “Optimal coverage path planning based on the rotating calipers algorithm,” in Proc. IEEE Int. Conf. Electron. Automat. Eng., 2017, pp. 140–144, doi: 10.1016/j.icemat.2017.11.
[14] T. M. Driscoll, “Complete coverage path planning in an agricultural environment,” Ph.D. dissertation, Dept. Comput. Sci., Iowa State Univ., Ames, IA, USA, 2011.
[15] J. Jin, “Optimal field coverage path planning on 2D and 3D surfaces,” Ph.D. dissertation, Dept. Agricultural Biosyst. Eng., Iowa State Univ., Ames, IA, 2009, doi: 10.31274/etd-180810-3122.
[16] S. de Bruin, P. Lerink, J. J. La Riviere, and B. Vanmeulebroek, “Systematic planning and cultivation of agricultural fields using a geo-spatial arable field optimization service: Opportunities and obstacles,” Biosyst. Eng., vol. 120, pp. 15–24, 2014, doi: 10.1016/jbiosystems.2013.07.009.
[17] R. J. Meuth and D. C. Wunsch, “Divide and conquer evolutionary TSP solution for vehicle path planning,” in Proc. IEEE Congr. Evol. Comput. IEEE World Congr. Comput. Intell., 2006, pp. 676–681, doi: 10.1109/CEC.2008.4630068.
[18] K. Zhou, A. L. Jensen, D. D. Bochiis, and C. G. Sørensen, “Quantifying the benefits of alternative fieldwork patterns in a potato cultivation system,” Comput. Electron. Agriculture, vol. 119, pp. 228–240, 2015, doi: 10.1016/j.compag.2015.10.012.
[19] M. Speksen, S. De Bruin, J. P. Molin, and G. Sparovek, “Planning machine paths and row crop patterns on steep surfaces to minimize soil erosion,” Comput. Electron. Agriculture, vol. 124, pp. 194–210, 2016, doi: 10.1016/j.compag.2016.03.013.
[20] L. E. Dubins, “On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents,” Am. J. Math., vol. 79, no. 3, pp. 497–516, 1957, doi: 10.2307/2372560.
[21] J. Reeds and L. Shepp, “Optimal paths for a car that goes both forwards and backwards,” Pacific J. Math., vol. 145, no. 2, pp. 367–393, 1990, doi: 10.2140/pjm.1990.145.367.
[22] J. Backman, P. Piirainen, and T. Oksanen, “Smooth turning path generation for coverage path planning in arable farming,” Biosyst. Eng., vol. 139, pp. 76–86, 2015, doi: 10.1016/jbiosystems.2015.08.005.
[23] D. Sabelhaus, F. Röben, L. W. M. zu Hellenig, and P. S. Lamers, “Using continuous-curvature paths to generate feasible headland turn manoeuvres,” Biosyst. Eng., vol. 116, no. 4, pp. 399–409, 2013, doi: 10.1016/jbiosystems.2013.08.012.
[24] M. Hoffmann, S. Patel, and C. Büskens, “Weight-optimized NURBS curves: Headland paths for nonholonomic field robots,” in Proc. 8th Int. Conf. Automat. Robot. Appl., 2022, pp. 81–85, doi: 10.1109/ICAARS50944.2022.9738525.
[25] R. S. Nilsson and K. Zhou, “Method and benchmarking framework for coverage path planning in arable farming,” Biosyst. Eng., vol. 198, pp. 248–265, 2020, doi: 10.1016/jbiosystems.2020.08.007.
[26] M. Hetzmark, R. S. Nilsson, and C. A. G. Sørensen, “In-field route planning optimisation and performance indicators of grain harvest operations,” Agronomy, vol. 12, no. 5, 2022, Art. no. 1151, doi: 10.3390/agronomy12051151.
[27] Intel, “oneapi threading building blocks,” Accessed: Feb. 17, 2023. [Online]. Available: https://github.com/oneapi-src/oneTBB.
[28] D. M. Beazley et al., “SWIG: An easy to use tool for integrating scripting languages with C and C++,” in Proc. Tcl/Tk Workshop, 1996, p. 74.
[29] Google, “Protobuf on c,” Accessed: Feb. 17, 2023. [Online]. Available: https://developers.google.com/protocol-buffers.
[30] T. Field, J. Leibs, J. Bowman, and D. Thomas, “Rosbag package,” Accessed: Feb. 17, 2023. [Online]. Available: https://github.com/irvingvasquez/ocpp.
[31] A. Janulevičius and K. Giedra, “The slippage of the driving wheels of a lawn mower using Dubins’ curve,” in Proc. Int. Conf. Robot. Automat., 2017, pp. 517–522, doi: 10.1109/RCTR.2017.8311915.