A Travelling Salesman Problem Approach to Efficiently Navigate Crop Row Fields with a Car-Like Robot

Ismael Ait, Ernesto Kofman, and Taihú Pire

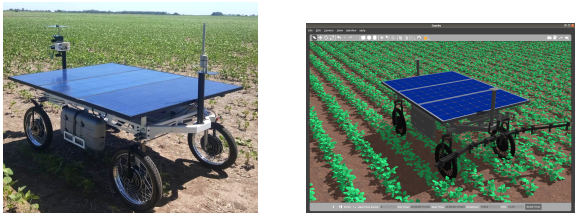
***Abstract*—In recent years, interest in the use of mobile robots in the agricultural industry has increased, both to address labor shortages in rural areas and to increase food production in a more sustainable way. In order to have an efficient navigation system to cover long crop row fields, a path planner algorithm must consider maneuvering restrictions of the targeted robot. Most state-of-the-art works in agricultural navigation systems are intended for robots with a high degree of maneuverability that can typically make in-place turnings. This work aims to fill the gap in terms of the development of an efficient navigation system for car-like robots with limited turning radius in crop row fields. For this, we combine the global path planner A* and the local trajectory planner Timed Elastic Band (TEB). Additionally, we state the problem of finding an optimal path that covers the entire field as a Travelling Salesman Problem (TSP) that is based on the different turning maneuvers the robot can perform at field headlands. The solution of the TSP results in a time efficient coverage strategy that aligns with the robot's kinematics. Experiments performed in the Gazebo simulation environment show a reduction in field completion times of up to $20\,\%$, compared to trivial coverage paths. On the other hand, deviation of the robot with respect to the center of the field furrows was in all cases less than $10\,\mathrm{cm}$, which proves that the entire system operates with sufficient accuracy to avoid damaging the crops.**

***Index Terms*—Autonomous Navigation, Robot Simulation, Precision Agriculture, Agricultural Robotics, Travelling Salesman Problem.**

I. Introduction

Precision agriculture is constantly changing the way farmers all over the world operate fields. It promotes the use of new technologies to combat labor shortages in rural areas and increase food production in a more sustainable way, a major concern as the world population continues to grow [1]–[3]. The applications are very diverse and involve multiple areas like artificial intelligence [4], deep learning [5], [6], IoT [7], Unmanned Aerial Vehicles (UAV) [8], and Unmanned Ground Vehicles (UGV) [9], [10].

Agrochemicals have been used for years by farmers as weed and insect regulators to increase yields. However, chemical residues negatively impact human health through environmental and food contamination [11]–[13]. It is in this context that the soybean weeding robot shown in Figure 1 has been developed by CIFASIS (French Argentine International Center for Information and Systems Sciences, CONICET-UNR, Argentina), conceived as an environmentally friendly and low-cost alternative for weed control. The robot is designed to move along the crop rows in a completely autonomous way, while detecting through computer vision, the presence of weeds and applying herbicides in a localized way with precision sprayers, thus preventing damage to the environment and to people [14].



(a) Actual prototype (b) Gazebo simulated world

Fig. 1. Soybean weeding robot developed by CIFASIS.

Most of today's applications of mobile robots in agriculture are intended for vehicles with a high degree of maneuverability that can typically make in-place turnings. This work aims to fill the gap in terms of the development of an efficient navigation system for car-like robots with limited turning radius in crop row fields. The path planning problem consists of determining an obstacle-free geometric path from an initial to a goal point, while trajectory planning algorithms take a given geometric path and endow it with time information, involving not only the robot's kinematics but also its dynamics. Much work can be found in the robotic literature dealing with these problems. Path planning algorithms are usually divided according to the methodologies used to generate the geometric path, namely: *roadmap* techniques based on the reduction of the configuration space to a set of one-dimensional path to search, cell decomposition algorithms [15], and artificial potential methods [16]. Roadmap techniques include works using search algorithm like Dijkstra [17] or A* [18], rapidly exploring random trees (RRT) [19], probabilistic roadmap methods (PRM) [20], and sampling-based methods [21]. On the other hand, the trajectory planning deals with the obstacle avoidance in dynamic environments based on the feedback information obtained from the robot's sensors. Those algorithms modify the trajectory of the robot in real time, and include methods like Virtual Force Field (VFF) [22], Vector Field Histogram (VFH) [23], Dynamic Window Approach [24], and Elastic Band concept [25]. Our work uses the traditional global path planner A* in combination with the local trajectory

Ismael Ait is with FCEIA, National University of Rosario, Argentina e-mail:ismaelaitd@gmail.com.

Ernesto Kofman and Taihú Pire are with CIFASIS, CONICET-UNR, Argentina.

planner Timed Elastic Band (TEB) [26], [27].

Most of the agricultural robots found in the literature consist of vehicles with the ability to make zero turning radius rotations. Some of them are made up of four steerable wheels [28]–[30], and others use a skid-steer configuration [31]–[34]. The path planning methods used include state machine-based algorithms, pure pursuit algorithms or just PID controllers to keep the robot centered between crops, but all are custom implementations and none take into account unexpected obstacle avoidance. The few works found that use car-like robots have an application to orchards rather than crop fields, and they also use custom-developed local planner methods [35], [36].

In this work we are looking to take advantage of the ROS *navigation stack* and its standard libraries. There are some relevant papers studying planners available in ROS but in other non-agricultural domains. In [37], a great comparison of the most commonly used local planners in ROS is provided. The TEB planner stands out as the one that generates smoother trajectories during obstacle avoidance, resulting in shorter execution times. Likewise, [38] performs an exhaustive analysis of both the global and local planners available in ROS. Regarding global planners, the results indicate that A* and Dijkstra generate the shortest but not the smoothest paths. However, as these paths are adjusted later by the local planner in a two-planner model, it is concluded that smoothness should not be a problem. For local planners, the effectiveness and robustness of the TEB is again highlighted. In addition, [39] explores the use of Dijkstra's global planner together with the local planner TEB with a car-like vehicle autonomously driving around a university campus.

Most of the agricultural robot prototypes use custom-built planners that divide the problem into two stages: one that keeps the robot in the middle of the crop rows as it drives in a straight line, and another stage that handles the turns at the headlands. However, these methods usually do not take into account the possibility of avoiding unexpected obstacles and are not suitable for non-holonomic robots with limited turning radius.

We will use the Gazebo simulator [40] to develop a virtual model of the weeding robot and different layouts of agricultural fields. This will allows us to safely experiment with multiple approaches, which in real environments would involve high costs in terms of logistics and implementation. For the navigation system, we will use the ROS framework [41] and the approach of two (global and local) planners [42]. All source code for the simulation and navigation system, along with the experiments and data analysis performed, is publicly available for the benefit of the Agricultural Robotics community [43].

The contribution of this work can be summarized as follow:

- Creation of a virtual model of the weeding robot and multiple soybean fields in the Gazebo simulator.
- Development of a fully functional navigation system, properly designed to work with the Ackermann steering mechanism present in the robot, which uses the ROS navigation stack, the A* global path planner and the TEB local trajectory planner.
- Analysis of the possible turning maneuvers of the robot, and the subsequent generation of an efficient coverage route for soybean fields arranged in rows. For this, we define the time-optimal coverage problem in terms of the Travelling Salesman Problem.

II. Materials and Methods

This work makes extensive use of the ROS platform and the Gazebo simulator. The targeted robot consists of a four-wheeled car-like vehicle specially designed for performing autonomous weed control in soybean fields. It has multiple sensors available for perception and navigation, such as the motor encoders, a stereo camera, an IMU, and a GPS-RTK. Power is provided by four batteries that are charged with solar panels located at the top of the vehicle.

The field consists of soybean crops arranged along multiple parallel rows, which have to be covered by the robot's sprayers. We aim to prevent the robot from crushing the plants, so its navigation will consist of driving in a straight line through the crop rows with its wheels in the furrows that have no plants. On each run along the field, the robot will cover a certain number of crop rows with its sprayers, which is known as a swath. When it reaches the end of a swath, it must perform a turning maneuver at the field headland to move to the next swath to be covered.

We divide the problem into two main parts. On one hand, we propose a method to determine an efficient way to cover the entire field by formulating the problem in terms of TSP. For this, we first collect the execution times of the different possible turning maneuvers for the targeted robot, then solve the resulting TSP in an offline manner, to finally use the obtained swath ordering as input to the navigation system. These steps are depicted in Figure 2. On the other hand, we develop a navigation system that allows the robot to move autonomously between two waypoints in the field, avoiding the driving over the crops. This last part of the system will run on an on-board computer in real-time while the robot navigates through the field.

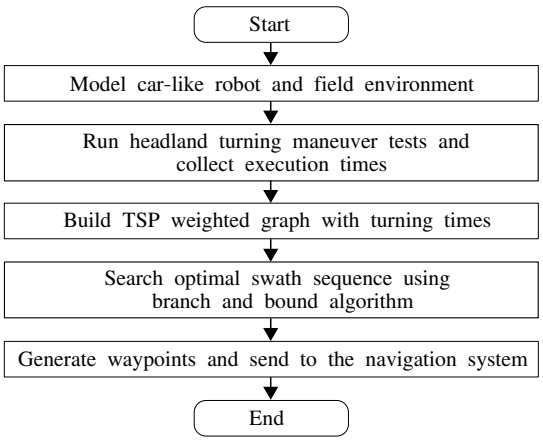
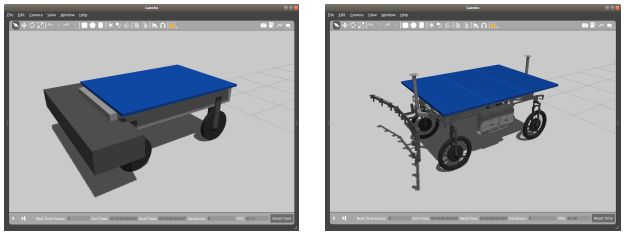


Fig. 2. Flow chart of the different steps involved.

A. Simulation Setup

An XML file in URDF (Unified Robot Description Format) is used to create the virtual model of the robot. This format allows to express the robot structure as a tree of links and joints: where links represent rigid parts of the robot, and joints represent connections between them. For each part, physical properties (like mass, center of mass, moment of inertia and friction coefficients) and visual properties are described. As shown in Figure 3, two virtual models of the weeding robot are developed: a visual model with 3D meshes exported from SolidWorks and a simplified collision model. The collision model is used by the physics engine to compute the interactions between the robot and the environment, while the visual model just provides the looks to the simulation. The use of a simplified collision model reduces the computational complexity of running the simulation.



(a) Collision model (b) Visual model

Fig. 3. Collision model is used by the physics engine, while visual model just provides a realistic rendering of the scene.

These car-like vehicles have an Ackermann steering mechanism that prevents the wheels from sliding sideways when turning. The mechanism uses an arrangement of four-bar linkage to ensure the correct steering angle for the front wheels, making the perpendicular lines to both steered wheels intersect at one point known as its center of rotation (CoR). Turning radius of the vehicle ρ will be determined by the distance between CoR and the center of the rear axle. To simplify this model, an imaginary wheel at the center of the front axle with steering angle φ is envisioned. Then, using simple trigonometry we can obtain the turning radius with equation (1):

$$\rho = \frac{L}{\tan(\varphi)}, \tag{1}$$

where L denotes the wheelbase of the robot. The actual angles of the front left and right wheels, φ_L and φ_R, are determined using the Ackermann equations shown in (2):

$$\varphi_L = \arctan\left(\frac{L}{\rho - \frac{T}{2}}\right), \qquad \varphi_R = \arctan\left(\frac{L}{\rho + \frac{T}{2}}\right), \tag{2}$$

where T denotes the track width of the robot.

The minimum and maximum values of the steering angle limit the vehicle turning radius. For the targeted weeding robot, the minimum and maximum steering angles are by design $\pm 33°$. Replacing this value in equation (1) gives a minimum turning radius of approximately $2.4\,\mathrm{m}$.

Since it is not possible to define a closed loop of four links with URDF, we emulate the Ackermann steering geometry with a driver as shown in the diagram of Figure 4. The navigation component provides the linear and angular velocity controls (v, w). Firstly, these commands are converted to an angular velocity ω for the rear wheel and an angular position φ for the direction of the front imaginary wheel. After that, another conversion is performed using the Ackermann geometry equations (2) to obtain the left and right wheel steering angles.

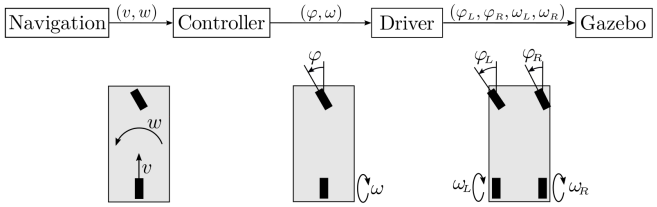


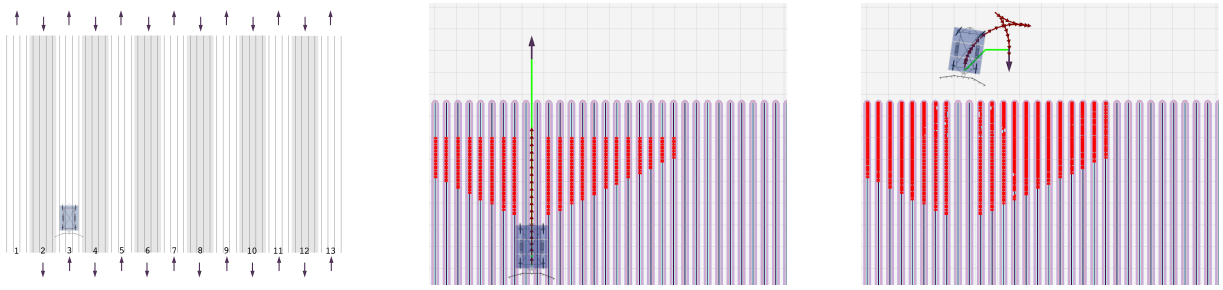
Fig. 4. Control commands received from the navigation system, adapted for a car-like vehicle in the Gazebo simulator.

B. Navigation Between Waypoints

A waypoint is generated at the beginning and end of each swath in the order they should be visited, as shown in Figure 5a. The robot must be able to plan a path to move from one waypoint to another while avoiding driving over the crops. The component responsible for doing this, is divided into a global planner that creates a path with the prior knowledge of the environment, and a local planner that fine-tunes the path taking into account the robot's maneuverability constraints and unexpected obstacles.

For the global planner, we use the well-known A* algorithm. As input for this planner we provide a static map depicting the information known in advance about the field, consisting of the dimensions and layout of each crop row. For the local planner we use the TEB (Timed Elastic Band) method, described in [27]. This planner makes adjustments to the global path in order to adapt it to the dynamic and kinematic constraints of the robot. It creates a series of intermediate poses of the form $p_i = (x_i, y_i, \theta_i)$ that the robot must follow and then generates the commands of linear velocity v and angular velocity w that will be sent to the robot's controller. This planner uses a map of the robot's surroundings that is built and updated online with the information obtained from the sensors of the robot. From the stereo camera data, a point cloud is generated and used to detect the crop rows, and other obstacles that may exist in the field. Since there will be two crop rows crossing the robot footprint when it drives along the field, we apply a filter to the point cloud to clear the points that can pass under the robot. More precisely, the points removed are those located below the robot's chassis, up to a height of $50\,\mathrm{cm}$, and within the robot's wheel width, approximately $1\,\mathrm{m}$. The local map is then built from this filtered point cloud and the two crop rows under the robot are not considered as obstacles.

The Figures 5b and 5c show the paths generated by the global and local planners when the robot is driving between crops and when it is maneuvering to the next swath, respectively. The region of the environment that represents the local



(a) (b) (c)

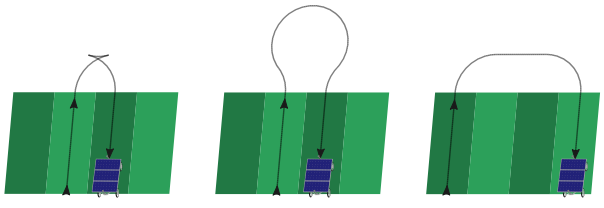
Fig. 5. Navigation system steps captured from RViz. (a) Field showing crop rows, swaths made up of 4 rows and waypoints with purple arrows. (b) Shows navigation between crop rows along the field and (c) shows a turning maneuver at headland. Global and local paths are depicted by a green line and by red arrows, respectively.

map is constantly updating as the robot moves through the environment, a technique known as rolling window.

C. Field Coverage

For field weeding to be successful, by the time the robot completes its route throughout the entire field, the sprayers should have passed over each of the crop rows at least once. To achieve this, we first divide the field into disjunct swaths (not sharing rows) and number them from left to right, in order to be able to identify them easily, as it is shown in Figure 5a. Then we focus on the problem of determining the most convenient order in which to travel these swaths.

The trivial swath sequence consist of covering the swaths in order, starting from the leftmost and continuing to the immediately next to the right until the rightmost swath. Since the robot has a minimum turning radius greater than the distance between two consecutive swaths, a turn to an adjacent swath is not particularly time efficient. In order to get better sequences, we need to consider turns to farther swaths and analyze the time the robot will need to perform each of them. The works [44], [45] identify three main different turning maneuvers, as shown in Figure 6.



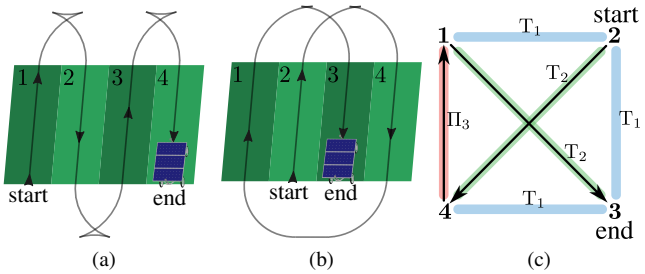
(a) T-turn (b) Ω-turn (c) Π-turn

Fig. 6. Turning types. For T-turns the robot must have reverse. For Ω and Π-turns the robot always drives forward. To make a Π-turn the distance between the centers of initial and target swaths must be at least twice the minimum turning radius.

Every crop row, comprising the sowing area and half a furrow on each side, is $0.52\,\mathrm{m}$ wide. This means that a swath of 4 crop rows is $2.08\,\mathrm{m}$ wide. Considering that the robot's minimum turning radius is $2.4\,\mathrm{m}$, a Π-turn can only be made by jumping at least 3 or more swaths. For maneuvers of 1 or 2 swath jumps, a T or Ω-turn should be used. From now on, we will represent a turning maneuver with the letter corresponding to its type (i.e. T, Ω or Π) followed by a subscript indicating the number of swath jumps. We measure the time the robot takes to perform all possible turns, in order to determine how efficient each one is. To find the optimal swath sequence that minimizes the time the robot spends at field headlands, we build a graph $G = (V, E)$ where the set of vertices V represents the swaths and the set of edges E represents the different turning maneuvers between swaths. This graph has the following characteristics:

- Complete, since the robot can go from one swath to any other with a turning maneuver at the headland.
- Weighted, with a weight or cost proportional to the time the robot needs to go from one swath to another.
- Undirected, since the time needed to jump from one swath to another depends only on the distance between them. It does not matter whether the target swath is at right or left of the initial one.

The objective now is to find the minimum cost path that traverses all the nodes in the graph, without visiting a node more than once. This problem is equivalent to the classic *Travelling Salesman Problem* [46]. TSP is an NP-hard problem, which implies that the computational cost required to solve it will increase dramatically with the size of the problem. In Figure 7 we can see, as an example, the resulting graph for a 4-swath field. One optimal solution is the sequence $\langle 2, 4, 1, 3 \rangle$, which consists of two T_2 turns and one Π_3 turn. This sequence is more efficient than the trivial sequence $\langle 1, 2, 3, 4 \rangle$, that has all T_1 turns.



(a) (b) (c)

Fig. 7. Coverage of a 4-swath field. Figure (a) shows the trivial sequence, while (b) shows an optimized one obtained after solving the TSP represented by graph (c).

D. Final Navigation Framework

The diagram in Figure 8 summarizes the entire navigation and simulation system for the weeding robot. Starting from the bottom left, the *waypoint_server* node is in charge of generating a series of ordered waypoints that split the field into disjoint swaths. These waypoints are fetched in order by the *goal_publisher* node which publishes them one at a time using the ROS action protocol. Each time the robot reaches a waypoint within a configurable proximity, the goal is replaced by the next waypoint.

Then, the *global_planner* node generates a collision-free path to the next goal using the A* algorithm, but without taking into account the robot's dimensions and kinematic constraints. To perform this, it receives through the *global_costmap* and *map_server* nodes a static global map constructed from a PGM image with the previously known information about the field (i.e. its dimensions and layout of the crop rows). On the other hand, it receives the current location of the robot through the *localization* node, which in turn gets the information directly from the simulator.

Next, the path generated by the global planner is taken by the *local_planner* node, which makes the necessary adjustments to adapt it to the robot's maneuverability capabilities. This local planner also receives the robot's location through the *localization* node, but unlike the global one, it uses a map reduced to the robot's surroundings and updated online by the robot's sensors. The trajectory generated by this node is converted to linear and angular velocity commands for the robot chassis, and the commands are sent to the Ackermann controller component.

Subsequently, the *ackermann_steering_controller* converts these commands to rear wheel rotation speeds and front wheel steering positions, to be sent to the Gazebo simulator through the *Transmission* and the *Hardware Interface* components. Meanwhile, the simulator provides the state of each part of the robot through the *State Transmission* component to the *joint_state_publisher* node, that publishes the data in a topic to be used by the ROS visualization tool *RViz*.

III. Results and Discussion

Different experiments were designed to evaluate the accuracy, robustness and efficiency of the developed navigation system using the simulation environment. All trajectories generated were evaluated offline with data recorded into bag files during the execution of the experiments. The hardware used for running the experiments corresponds to an Intel® Core™ i7-1065G7 1.30 GHz quadcore computer, 8 GB RAM and Nvidia GeForce MX230 graphics card.

A. Turn type Evaluation

The robot will not be able to perform a Π-turn towards a swath that is less than 3 swaths away. For jumps of 1 and 2 swaths it should perform a turn of type T or Ω. To evaluate its performance, 8 different turns are taken into account. For each turn, 30 trials were run with the simulated weeding robot and path length and duration data were collected. Figure 9a displays one of the trajectories performed by the robot for each of the turns analyzed and the boxplots in Figure 9b reveal the obtained results in terms of the time required to perform each maneuver. Time variations within the same maneuver come from noise in sensors and actuators incorporated in the Gazebo simulation, that mimics the real situation. These experiments must be corroborated in future work with field tests.

TABLE I
Median length, duration and speed by turn type.

Turn type	Length (m)	Duration (s)	Speed (m/s)
T_1	**8,941**	20,730	0,437
T_2	9,119	19,475	0,468
Ω_1	16,303	25,108	0,648
Ω_2	13,367	20,733	0,642
Π_3	9,667	**14,093**	0,687
Π_4	11,716	16,307	0,718
Π_5	14,010	19,425	0,725
Π_6	16,149	22,036	**0,735**

If only the length of each turning maneuver is considered, we might think that T-turns are the most efficient, but if we analyze the duration data we can see that turns Π_3 and Π_4 perform considerably better than the previous ones. It is only after turn Π_4 that we obtain a time duration similar to turn T_2. This implies that sequences with more 3 and 4 swath jumps will be more efficient than the trivial sequence with only 1-swath jumps. Table I summarizes the median values for the different turns in terms of length, duration and speed. The duration column values will be used as the costs of the weighted TSP graph presented in the previous section. The cost assigned to the edge (v, w) will correspond to the time the robot needs to make a jump of $|v - w|$ swaths. For the case of 1 and 2 swath jumps, T-turns are chosen, since they are more efficient than Ω-turns and the targeted robot can perform them due to its reverse driving capability. For jumps of 3, 4, 5 and 6 swaths, the corresponding Π-turns are used. And for jumps of more than 6 swaths, the time of the Π_6 turn is used, as a lower bound to the real cost. Then, the problem of finding the sequence that minimizes the time at the headlands is reduced to solving the TSP problem for that graph.

B. Optimal Swath Sequence

In this work, we tested three exact algorithms to solve the TSP problem. The most direct solution for the TSP problem is to try all possible sequences and see which one has the minimum cost (i.e. brute-force search). Let n be the number of swaths in the field, the time complexity for this approach is $\mathcal{O}(n!)$. As a result, it was only possible to compute the solution for a field of up to 12 swaths using the brute-force algorithm with the available hardware.

A better approach to find an exact solution to this problem is by using the dynamic programming technique, which eliminates the recalculation time for optimal solutions of subproblems. The algorithm for solving the TSP problem using this approach is named Bellman-Held-Karp after its authors and has a computational complexity of $\mathcal{O}\left(n^2 2^n\right)$. Using this algorithm, it was possible to find the optimal solution for fields of up to 23 swaths. In this case the limitation was the memory utilization of the algorithm rather than the execution time.

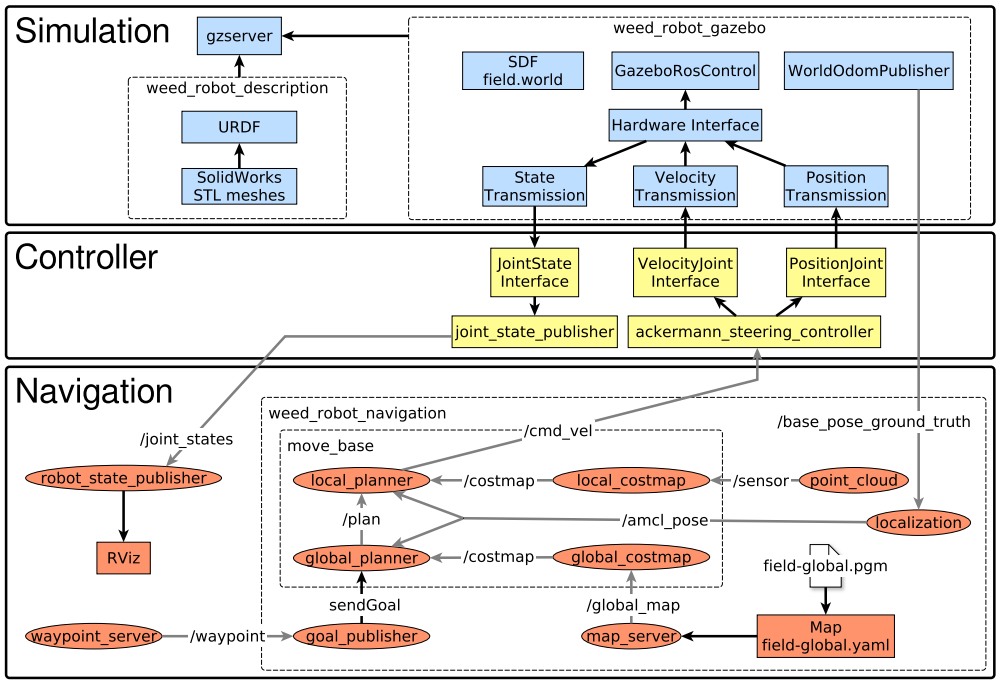
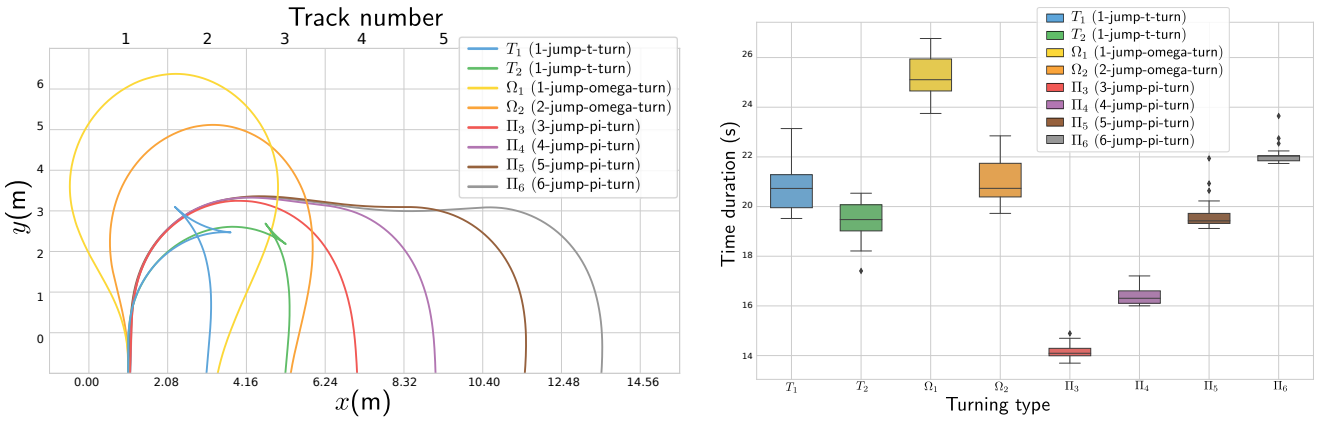


Fig. 8. Summarizing diagram showing the interaction between ROS nodes and Gazebo components.



(a) (b)

Fig. 9. Turn type evaluation. (a) Shows trajectories made for the different turning maneuvers. The swath number is displayed at the top. Each swath has a width of 2.08 m. Boxplots (b) compare the execution times after 30 trials.

Finally, we tested the branch and bound approach, which performs an exploration of the search space represented by a tree. This technique systematically explores each branch of the tree and saves time by not continuing the search along paths that are detected as impossible to lead to an optimal solution. The theoretical worst-case complexity for this algorithm is the same as that of brute force. However, for this particular use case it turns out to be the most efficient of the three algorithms, being able to solve fields of up to 35 swaths. Basically, because branches with partial solutions that already have several low-performing turns can be pruned quite early.

The Table II shows the results obtained with the branch and bound method for fields of up to 35 swaths. For a 15-swath field we see that there is an optimal sequence that uses only turns of type Π_3 and Π_4, so the entire path can be performed without using reverse driving. For fields with a number of swaths greater than 15, we see that the optimal sequence obtained solving the TSP follows a pattern that uses almost all Π_3-turns except for only two turns that are of type T, one at the right end and one at the left end of the field. An example of this pattern can be seen in Figure 10a. We find that the pattern can be extended to fields of any number of swaths as follows. The robot starts at a swath near the left end of the field and makes Π_3-turns until the right end. Then, it makes a T-turn and returns to the left end again making Π_3-turns. When it reaches this other end, it makes a second T-turn and returns to the right end making Π_3-turns once again. In each travel between the left and right ends of the field, the robot

TABLE II
RESULTS OF THE OPTIMAL SWATH SEQUENCE SEARCH.

# Swaths	# Turns				Total	Average Time
	1	2	3	4	Time (s)	per Turn (s)
4		2	1		53,043	17,681
5	1		2	1	65,223	16,305
6			3	2	74,893	14,978
7			4	2	88,986	14,831
8			5	2	103,079	14,725
9	1		6	1	121,595	15,199
10		2	7		137,601	15,289
11		2	8		151,694	15,169
12	1		8	2	166,088	15,098
13			8	4	177,972	14,831
14			9	4	192,065	14,774
15			10	4	206,158	14,725
16		2	13		222,159	14,810
17		2	14		236,252	14,765
18	1	1	15		251,600	14,800
19		2	16		264,438	14,691
20		2	17		278,531	14,659
21	1	1	18		293,879	14,693
22		2	19		306,717	14,605
23		2	20		320,810	14,582
24	1	1	21		336,158	14,615
25		2	22		348,996	14,541
26		2	23		363,089	14,523
27	1	1	24		378,437	14,555
28		2	25		391,275	14,491
29		2	26		405,368	14,477
30	1	1	27		420,716	14,507
31		2	28		433,554	14,451
32		2	29		447,647	14,440
33	1	1	30		462,995	14,468
34		2	31		475,833	14,419
35		2	32		489,926	14,409

will be covering all the swaths which dividing by 3 result in the same reminder (modulus operation). Given n the total number of swaths in the field:

- If $n \bmod 3 = 0$. Start at swath 1. Visit all swaths s such that $s \bmod 3 = 1$. Make a T_2-turn from $n-2$ to n. Visit all swaths s such that $s \bmod 3 = 0$. Make a T_1-turn from 3 to 2. And visit all swaths s such that $s \bmod 3 = 2$.
- If $n \bmod 3 = 1$. Start at swath 2. Visit all swaths s such that $s \bmod 3 = 2$. Make a T_2-turn from $n-2$ to n. Visit all swaths s such that $s \bmod 3 = 1$. Make a T_2-turn from 1 to 3. And visit all swaths s such that $s \bmod 3 = 0$.
- If $n \bmod 3 = 2$. Start at swath 2. Visit all swaths s such that $s \bmod 3 = 2$. Make a T_2-turn from n to $n-2$. Visit all swaths s such that $s \bmod 3 = 0$. Make a T_2-turn from 3 to 1. And visit all swaths s such that $s \bmod 3 = 1$.

C. Case Study: 15-Swath Field

We chosen a field composed of 15 swaths and tested the coverage with the simulator using 3 different sequences. First of all, we tried the trivial sequence which the robot starting from the leftmost swath and visiting each swath to the right in order using turns of type Ω_1 and T_1. Figure 10a shows the sequence following the pattern that uses almost all turns of type Π_3 and only two of type T. And finally, Figure 10b shows the optimal sequence for 15 swaths, which does not use reverse driving. We can see that in the headland surroundings, the robot moves slightly away from the center of the swath, but in a few meters it is re-aligned correctly

TABLE III
LENGTH AND DURATION FOR DIFFERENT SEQUENCES.

Sequence	Length (m)	Duration (s)
Trivial path using Ω_1	532,02	741,36
Trivial path using T_1	**428,51**	680,17
Pattern using almost all Π_3	441,87	611,81
Optimal path not using reverse	447,41	**598,50**

The Table III shows the full length and duration of each pattern. There is a clear time savings with the optimal pattern. However, the impact on the overall time will depend on the length of the crop rows. Avoiding the use of Ω turns also removes the need for large headlands, which results in more space being used for planting.

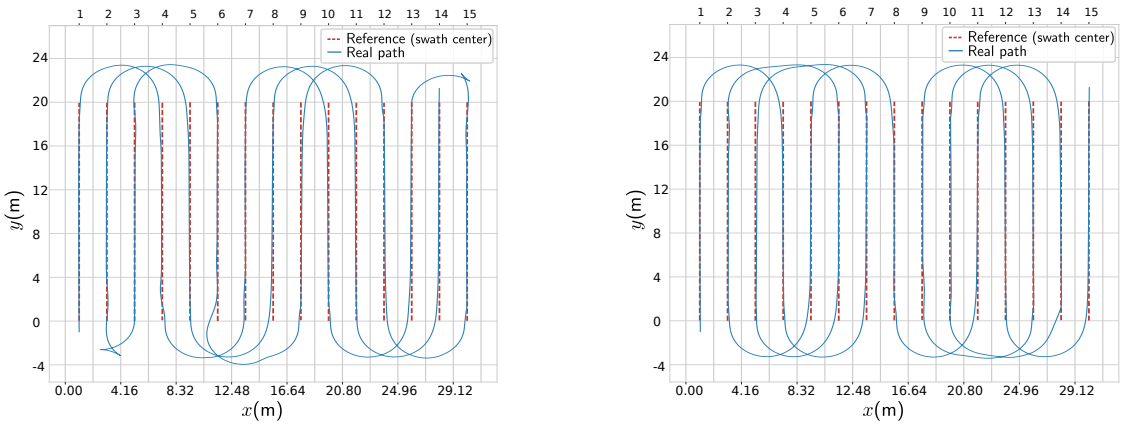
In order to test the effectiveness of the planner in terms of not damaging the crops, we measured how far the robot drives from the center of the swaths. The graphs in Figure 11 plot the translational and rotational ATE errors for the optimal 15-swath path. The peaks shown in the graphs correspond to moments when the robot performs turning maneuvers. In general, the errors obtained were relatively small, yielding in all cases an average distance to swath center of less than $10\,\mathrm{cm}$ and an average deviation in angle of about $0.035\,\mathrm{rad}$ ($2°$).

D. Future Work

Despite obtaining satisfactory results with the TEB planner in terms of not driving over the crops, it was necessary to create several intermediate waypoints at the field headlands for the robot to perform the desired turning maneuvers. The TEB planner supports avoidance of previously unknown dynamic obstacles, but further testing is needed to verify it works as expected in our system. Another feature that would improve robustness of the system is the incorporation of a recovery strategy in case the robot gets stuck on obstacles. As future work, it is also of great interest the development and testing of other planners to compare with the results obtained in this work. Finally, it will be necessary to move the tests performed on the simulator to the real prototype, in order to check aspects such as correct real-time operation, sensor responses and battery charge duration.

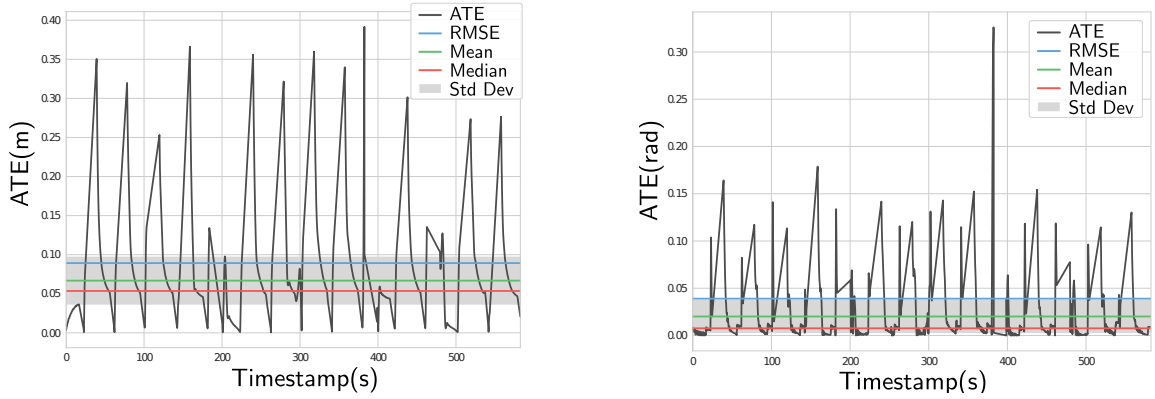
IV. CONCLUSIONS

The main contribution of this work consisted of the development of a fully functional and efficient navigation system for a car-like robot operating crop row fields. To this end, we performed a deep analysis of the different turning maneuvers that a robot with Ackermann steering mechanism and limited turning radius can perform at field headlands, and formulated the problem of finding the least time-consuming coverage path as the Travelling Salesman Problem. Experiments showed that the optimized path reduces the total field coverage time by up to $20\,\%$ compared to trivial coverage paths. We brought the use of the ROS navigation stack and the TEB trajectory planner to an agricultural environment, and proved that the entire system



(a) Pattern with almost all Π_3 (b) Optimal without reverse

Fig. 10. Different coverage patterns for a 15-swath field. The solid blue line corresponds to the path made by the weeding robot and the red dashed line shows the center of each swath.



(a) (b)

Fig. 11. Absolute Trajectory Errors (ATE) for optimal sequence in a 15-swath field, comparing real path performed against ideal path that passes through the center of each swath. The translational error in (a) shows the distance to the swath center in meters and the rotational error in (b) shows the angle difference with respect to $\pm$ 90°, depending on driving direction.

can operate satisfactorily under the kinematic constraints of the robot and in a field layout of long parallel crop rows. The resulting navigation system allows the robot to navigate the field safely without driving over the crops, giving a deviation of the robot to the swath center of less than 10 cm in all of the performed tests.

ACKNOWLEDGMENTS

This work was supported by the CIFASIS, French Argentine International Center for Information and Systems Sciences (CONICET-UNR), under Grant PUE 0015-2016 and by the Santa Fe province (Argentina) Government under Grant PEICID-2021-170.

REFERENCES

[1] S. Vougioukas, "Agricultural Robotics," *Annual Review of Control, Robotics, and Autonomous Systems*, vol. 2, no. 1, pp. 365–392, 2019.

[2] J. Lowenberg-DeBoer, I. Huang, V. Grigoriadis, and S. Blackmore, "Economics of robots and automation in field crop production," *Precision Agriculture*, pp. 278–299, 5 2020.

[3] Y. Lu and S. Young, "A survey of public datasets for computer vision tasks in precision agriculture," *Computers and Electronics in Agriculture*, vol. 178, p. 105760, 2020.

[4] K. Jha, A. Doshi, P. Patel, and M. Shah, "A comprehensive review on automation in agriculture using artificial intelligence," *Artificial Intelligence in Agriculture*, vol. 2, 6 2019.

[5] A. Kamilaris and F. X. Prenafeta-Boldú, "Deep learning in agriculture: A survey," *Computers and Electronics in Agriculture*, vol. 147, pp. 70–90, 2018.

[6] M. Saleem, J. Potgieter, and K. Arif, "Automation in Agriculture by Machine and Deep Learning Techniques: A Review of Recent Developments," *Precision Agriculture*, vol. 22, 4 2021.

[7] G. Zhang, X. Chen, L. Zhang, B. Feng, X. Guo, J. Liang, and Y. Zhang, "STAIBT: Blockchain and CP-ABE Empowered Secure and Trusted Agricultural IoT Blockchain Terminal," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 7, 9 2022.

[8] Q. Ding and X. Xu, "Improved GWO Algorithm for UAV Path Planning on Crop Pest Monitoring," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 7, pp. 30–39, 2022.

[9] S. Bonadies and S. A. Gadsden, "An overview of autonomous crop row navigation strategies for unmanned ground vehicles," *Engineering in Agriculture, Environment and Food*, vol. 12, no. 1, pp. 24–31, 2019.

[10] P. Maini, B. M. Gonultas, and V. Isler, "Online Coverage Planning for an Autonomous Weed Mowing Robot With Curvature Constraints," *(IEEE) Robotics and Automation Letters*, vol. 7, no. 2, pp. 5445–5452, 2022.

[11] D. Ecobichon, "Pesticide use in developing countries," *Toxicology*, vol. 160, no. 1, pp. 27–33, 2001.

[12] K. Lewis, J. Tzilivakis, D. Warner, and A. Green, "An international database for pesticide risk assessments and management," *Human and Ecological Risk Assessment: An International Journal*, vol. 22, no. 4, pp. 1050–1064, 2016.

[13] M. Tudi, H. Daniel Ruan, L. Wang, J. Lyu, R. Sadler, D. Connell, C. Chu, and D. T. Phung, "Agriculture development, pesticide application and its impact on the environment," *International Journal of Environmental Research and Public Health*, vol. 18, no. 3, 2021.
[14] T. Pire, M. Mujica, J. Civera, and E. Kofman, "The Rosario dataset: Multisensor data for localization and mapping in agricultural environments," *Intl. Journal of Robotics Research*, vol. 38, pp. 633–641, 2019.
[15] M. Kloetzer, C. Mahulea, and R. Gonzalez, "Optimizing cell decomposition path planning for mobile robots using different metrics," in *2015 19th International Conference on System Theory, Control and Computing (ICSTCC)*, pp. 565–570, 2015.
[16] J. Sun, J. Tang, and S. Lao, "Collision Avoidance for Cooperative UAVs With Optimized Artificial Potential Field Algorithm," *IEEE Access*, vol. 5, pp. 18382–18390, 2017.
[17] M. Luo, X. Hou, and J. Yang, "Surface Optimal Path Planning Using An Extended Dijkstra Algorithm," *IEEE Access*, vol. 8, pp. 147827–147838, 2020.
[18] F. Duchoň, A. Babinec, M. Kajan, P. Beňo, M. Florek, T. Fico, and L. Jurišica, "Path Planning with Modified a Star Algorithm for a Mobile Robot," *Procedia Engineering*, vol. 96, 12 2014.
[19] X. Wang, X. Luo, B. Han, Y. Chen, G. Liang, and K. Zheng, "Collision-Free Path Planning Method for Robots Based on an Improved Rapidly-Exploring Random Tree Algorithm," *Applied Sciences*, vol. 10, p. 1381, 2 2020.
[20] L. Qiao, X. Luo, and Q. Luo, "An Optimized Probabilistic Roadmap Algorithm for Path Planning of Mobile Robots in Complex Environments with Narrow Channels," *Sensors*, vol. 22, no. 22, 2022.
[21] L. Kenye and R. Kala, "Optimistic Motion Planning Using Recursive Sub-Sampling: A New Approach to Sampling-Based Motion Planning," *International Journal of Interactive Multimedia and Artificial Intelligence*, vol. 7, pp. 87–99, 2022.
[22] J. Han, M. Cui, Y. Lv, K. Liu, Q. Dai, and H. Guo, "Moving Horizon path planning for intelligent vehicle oriented to dynamic obstacle avoidance," in *2022 6th CAA International Conference on Vehicular Control and Intelligence (CVCI)*, 2022.
[23] R. Bijay, M. Amarendra, and D. Asim, "Steer Guidance Of Autonomous Agricultural Robot Based On Pure Pursuit Algorithm And LiDAR Based Vector Field Histogram," *Journal of Applied Science and Engineering*, vol. 26, pp. 1363–1372, 1 2023.
[24] D. Fox, W. Burgard, and S. Thrun, "The dynamic window approach to collision avoidance," *IEEE Robotics & Automation Magazine*, vol. 4, no. 1, pp. 23–33, 1997.
[25] S. Quinlan and O. Khatib, "Towards real-time execution of motion tasks," in *Experimental Robotics II*, pp. 239–254, Springer Berlin Heidelberg, 1993.
[26] C. Rösmann, W. Feiten, T. Wosch, F. Hoffmann, and T. Bertram, "Efficient trajectory optimization using a sparse model," in *European Conf. on Mobile Robots (ECMR)*, pp. 138–143, 9 2013.
[27] C. Rösmann, F. Hoffmann, and T. Bertram, "Online Trajectory Planning in ROS Under Kinodynamic Constraints with Timed-Elastic-Bands," in *Robot Operating System (ROS): The Complete Reference (Volume 2)*, (Cham), pp. 231–261, Springer Verlag, 5 2017.
[28] T. Bakker, K. van Asselt, J. Bontsema, J. Müller, and G. van Straten, "Autonomous navigation using a robot platform in a sugar beet field," *Biosystems Engineering*, vol. 109, no. 4, pp. 357–368, 2011.
[29] M. Post, A. Bianco, and X.-T. Yan, "Autonomous Navigation with Open Software Platform for Field Robots," in *Informatics in Control, Automation and Robotics*, (Cham), pp. 425–450, Springer Verlag, 1 2020.
[30] J. Chen, H. Qiang, J. Wu, G. Xu, Z. Wang, and X. Liu, "Extracting the navigation path of a tomato-cucumber greenhouse robot based on a median point Hough transform," *Computers and Electronics in Agriculture*, vol. 174, p. 105472, 2020.
[31] F. B. Malavazi, R. Guyonneau, J.-B. Fasquel, S. Lagrange, and F. Mercier, "LiDAR-only based navigation algorithm for an autonomous agricultural robot," *Computers and Electronics in Agriculture*, vol. 154, pp. 71–79, 2018.
[32] J. Iqbal, R. Xu, S. Sun, and C. Li, "Simulation of an Autonomous Mobile Robot for LiDAR-Based In-Field Phenotyping and Navigation," *MDPI Robotics*, vol. 9, no. 2, 2020.
[33] D. Aghi, V. Mazzia, and M. Chiaberge, "Local Motion Planner for Autonomous Navigation in Vineyards with a RGB-D Camera-Based Algorithm and Deep Learning Synergy," *Machines*, 5 2020.
[34] P. Ruangurai, M. N. Dailey, M. Ekpanyapong, and P. Soni, "Optimal vision-based guidance row locating for autonomous agricultural machines," *Precision Agriculture*, 2 2022.
[35] R. Linker and T. Blass, "Path-planning algorithm for vehicles operating in orchards," *Biosystems Engineering*, vol. 101, no. 2, pp. 152–160, 2008.
[36] R. F. Carpio, C. Potena, J. Maiolini, G. Ulivi, N. B. Rosselló, E. Garone, and A. Gasparri, "A Navigation Architecture for Ackermann Vehicles in Precision Farming," *(IEEE) Robotics and Automation Letters*, vol. 5, no. 2, pp. 1103–1110, 2020.
[37] B. Cybulski, A. Wegierska, and G. Granosik, "Accuracy comparison of navigation local planners on ROS-based mobile robot," in *12th International Workshop on Robot Motion and Control (RoMoCo)*, pp. 104–111, 7 2019.
[38] A. Filotheou, E. Tsardoulias, A. Dimitriou, A. Symeonidis, and L. Petrou, "Quantitative and Qualitative Evaluation of ROS-Enabled Local and Global Planners in 2D Static Environments," *Journal of Intelligent & Robotic Systems*, vol. 98, 6 2020.
[39] P. Marín, A. Hussein, D. Martín Gómez, and A. de la Escalera, "Global and Local Path Planning Study in a ROS-Based Research Platform for Autonomous Vehicles," *Journal of Advanced Transportation*, vol. 2018, 2 2018.
[40] N. Koenig and A. Howard, "Design and use paradigms for Gazebo, an open-source multi-robot simulator," in *IEEE/RSJ Intl. Conf. on Intelligent Robots and Systems (IROS)*, vol. March, pp. 2149–2154, 2004.
[41] M. Quigley, K. Conley, B. Gerkey, J. Faust, T. Foote, J. Leibs, R. Wheeler, and A. Ng, "ROS: an open-source Robot Operating System," *ICRA Workshop on Open Source Software*, vol. 3, 1 2009.
[42] J. Crowley, "Navigation of an Intelligent Mobile Robot," *IEEE Journal of Robotics and Automation*, vol. RA-1, pp. 31–41, 4 1985.
[43] https://github.com/CIFASIS/weed_robot_simulation
[44] D. Bochtis and S. Vougioukas, "Minimising the non-working distance travelled by machines operating in a headland field pattern," *Biosystems Engineering*, vol. 101, pp. 1–12, 9 2008.
[45] D. Bochtis, S. Vougioukas, and H. W. Griepentrog, "A Mission Planner for an Autonomous Tractor," *Transactions of the ASABE*, vol. 52, 9 2009.
[46] G. Laporte, "The traveling salesman problem: An overview of exact and approximate algorithms," *European Journal of Operational Research*, vol. 59, no. 2, pp. 231–247, 1992.

Ismael Ait received the licentiate degree in Computer Science from National University of Rosario, Argentina in 2021. His research interests are Robotics, Motion Planning, Robot Navigation and Autonomous Ground Vehicles.

Ernesto Kofman received his Electronic Engineer degree and his PhD in 1999 and 2003, respectively, both from Universidad Nacional de Rosario (UNR, Argentina). He holds a Full Professor position at the UNR and a Principal Researcher position at CONICET (National Research Council of Argentina). His main research interests include continuous and hybrid system simulation algorithms and set-theoretic control methods.

Taihú Pire received the licentiate degree in Computer Science (2010) at the National University of Rosario and the PhD in Computer Science (2017) at the University of Buenos Aires. He is a Research Scientist at the National Science and Technology Council of Argentina and Adjunct Professor at National University of Rosario. Currently, his research interests are in developing new SLAM algorithms and autonomous navigation systems.