Chapter 3

An Evaluation of Three Different Infield Navigation Algorithms

Peter Bernad, Peter Lepej, Črtomir Rozman, Karmen Pažek and Jurij Rakun

Additional information is available at the end of the chapter

http://dx.doi.org/10.5772/intechopen.79942

Abstract

In this chapter, we present and evaluate three different infield navigation algorithms, based on the readings from a LIDAR sensor. All three algorithms are tested on a small field robot and used to autonomously drive the robot between the two adjacent rows of maze plants. The first algorithm is the simplest one and just takes distance readings from the left and right side. If robot is not in the center of the mid-row space, it adjusts its course by turning the robot in the right direction accordingly. The second approach groups the left and right readings into two vertical lines by using least-square fit approach. According to the calculated distance and orientation to both lines, it adjusts the course of the robot. The third approach tries to fit an optimal triangle between the robot and the plants, revealing the most optimal one. Based on its shape, the course of the robot is adjusted. All three algorithms are tested in a simulated (ROS stage) and then in an outdoor (maze test field) environment comparing the optimal line with the actual calculated position of the robot. The tests prove that all three approaches work with an error of 0.041 ± 0.034 m for the first algorithm, 0.07 ± 0.059 m for the second, and 0.078 ± 0.055 m error for the third.

Keywords: infield algorithms, navigation algorithms, LIDAR, field robot, ROS

1. Introduction

The usual approach of autonomously driving agriculture machinery trough the field is by using a precise differential [1] or RTK global positioning systems [2]. In order to work, these systems require a previously known path of movement that is repeated with each iteration

when the crop needs to be treated [3]. This is not always possible, if no prior GPS information exists, or simply because an accurate GPS system is not always available. So, different systems should be used in these situations.

A possible solution to solve this is to use cameras to detect plant lines [4] or even stereo cameras to build a 3D cloud of points [5], describing the plants, and drive the agricultural machinery to drive between them. Another possible approach is the use of LIDAR systems [6] that take accurate measurements of the scene, by reading distances from the sensor to the first obstacle and repeating this for the whole range (usually 270° or 360° in 1°, 0.5° or even 2.25° steps).

With the help of these systems, the machines [7–9] can drive even in an unfamiliar field where they have not been applied before. They rely on the property of the field and the crops planted. The plants are in parallel to each other, and the machines can drive between the crop lines, in mid-row spaces, in order to not damage the plants. This can be done by using different approaches/algorithms to guide the machines.

More advance systems using different SLAM methods [10, 11] even build a map of the environment and localize the machines in a new environment that is only being discovered. Based on the map, it then constructs a path using path planner [12] and path follower [13] to follow the path. These approaches come in use in case of unfamiliar scenes but are not really necessary in infield situations where prior information regarding the pattern of the plants is known.

So, the purpose of this work is to first present three different algorithms that could be used to autonomously drive the machines and to evaluate their accuracy for infield navigation. All three were implemented as part of robotic operating system (ROS) [14] and applied on a small field robot, making it autonomous when driving through the field in mid-row spaces. Their purpose is not to plan and follow the path but to adjust the heading of the machines/robot at every measured location. The field robot that was used to test the algorithms is presented in Section 2, along with all three algorithms used, which are then evaluated in Section 3.

2. Materials and methods

The algorithms are tested on a small field robot [16] depicted in **Figure 1**. It is an electric driven robot with 50 × 50 cm in size, small enough to fit in 75 cm wide space between two parallel lines of maze plants. It has four in-wheel BLDC motors capable of delivering 200 W max peak power, equipped with four additional motors to individually turn each of the wheels, making possible to drive in different steering modes like skid steer or Ackermann. Besides the odometry from the wheels, the robot is equipped with three different additional sensors: two digital cameras, an IMU unit, and a LIDAR sensor. The data from the sensors are processed by two onboard computers, a low-level computer build around Raspberry PI 3B and Intel NUC i7 (gen. 7) computer, used as high-level processing unit. The two units run a Linux-based distribution with robotic (meta) operating system (ROS) on top, configured in multimachine mode to split the essential processing from time consuming, advance algorithms, making the robot as responsive as possible.

Figure 1. A small field robot FarmBeast while performing Task 1—Basic navigation at Field Robot Event 2018.

If the robot, as the one in **Figure 1**, has to drive autonomously through the crop fields, it must have a navigation algorithm. The navigation algorithm relies on the distance measurements from the LIDAR sensor. The experimental robot is equipped with the SICK TIM310 LIDAR, with a 270° area at a 1° angular resolution. It detects obstacles up to 4 m away and then chooses the optimal path based on the readings. Once the robot reaches the end of the crop line, it uses data from the on-board compass (IMU unit) and turns in the next row.

In this chapter, we investigate the accuracy of three different navigation algorithms and compare them, with left and right row distance as a reference for optimal path that should be in the middle of the rows and with as little oscillations as possible.

2.1. Minimal row offset-based algorithm

The first algorithm is the simplest one. It takes 30 readings from left and 30 from right side, as shown in **Figure 2**.

From these two sets, it first eliminates ones that are too far away, that is more than 0.75 cm and belong to the other crop lines, and then calculates an average distance value for each side. These average distances can be written as d_r and d_l for right and left side, respectively. These two are then used to calculate an offset, as shown in Eq. 1:

$$Offset = (d_r - d_l) \tag{1}$$

The value of the *offset* is then used to adjust the course of the robot, if necessary, as shown in Eq. 2.

$$Orientation = Offset/(d_r + d_l) \tag{2}$$

So, the current course of the robot is adjusted by the new value *orientation* which also corresponds directly how much the wheels should turn in radians.

2.2. Least-square fit approach algorithm

This second algorithm was designed to navigate the robot between two walls that are parallel to each other, this being either an artificial barrier, e.g., walls, or real crop lines, such as maze plants. The overview of the approach is depicted in **Figure 3**, explaining each sensing—adjustment cycle.

(A) The sensor reads the data—the distances between the sensor and the obstacles for each degree—and triggers a callback function each time it completes the measuring sequence.

(B) The callback function first filters the data depending on the distance readings. The points that are too far away and points that are too near to count are discarded. The algorithm makes possible to set how many points should be included for each count for each side that corresponds to how many degrees will use in the subsequent steps of the algorithm. The useful readings are stored in two data sets, one for left and one for right side.

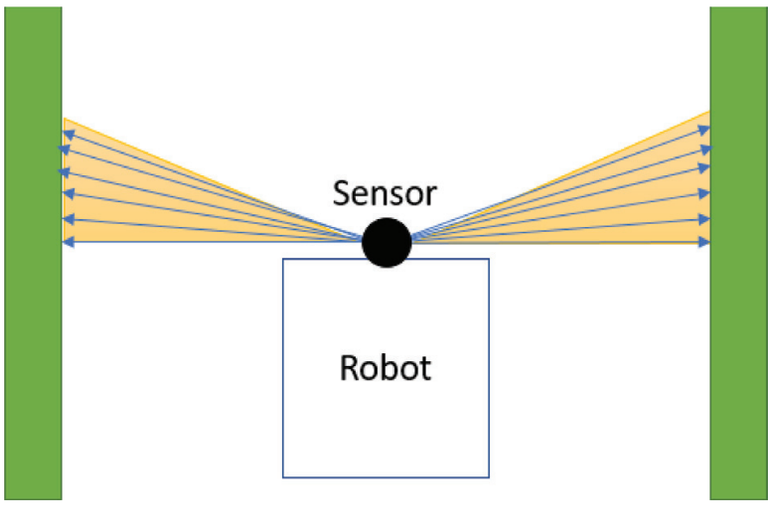

Figure 2. Robot standing in the middle of the row. Each double-sided arrow represents a measurement of the sensor, where the actual number of reading is 30 on each side. The sensor returns the distance for each degree it measures.

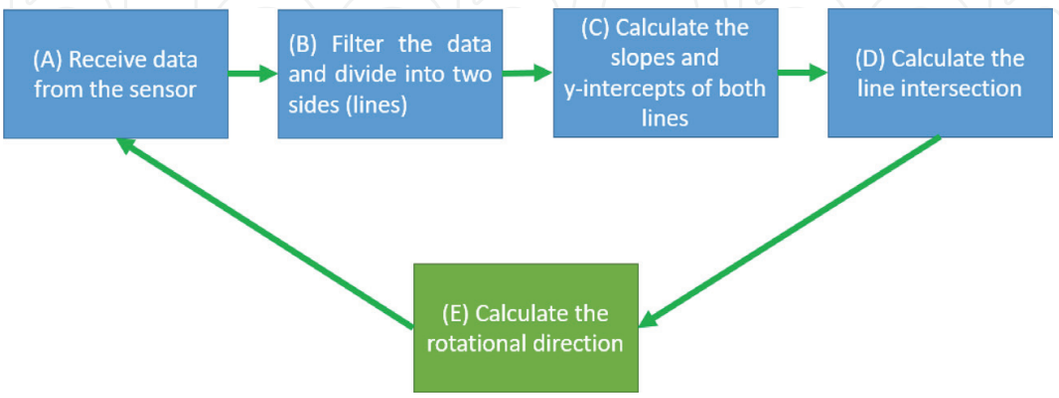

Figure 3. Performed steps in each cycle.

(C) In the third step, a linear fit is used, for which a least squares method [17] was chosen. This way the slope and y-intercept of a linear equation describing each set for each side is calculated.

The least squares method allows us to linearly fit the measurements with a smaller number of heavy duty mathematical operations. For this, we need to define some additional parameters with which we then calculate the slope and y-intercept of each line:

x_{sum} – *sum of all the distances taken for the line*,

$x_{square\ sum}$ – *sum of all the squared distances taken for the line*,

y_{sum} – *sum of all the angles taken for the line* (*in degrees*),

$y_{square\ sum}$ – *sum of all the squared angles taken for the line* (*in degrees*),

yx_{sum} – *sum of the products of the angle and distance of each point*.

Once these parameters are known, the slope (k) and y-intercept (n) can easily be calculated as shown in Eqs. 3 and 4:

$$k = \frac{number\ of\ points * yx_{sum} - x_{sum} * y_{sum}}{number\ of\ elements * x_{square\ sum} - x_{sum}^2} \tag{3}$$

$$n = \frac{x_{square\ sum} * y_{sum} - x_{sum} * yx_{sum}}{number\ of\ elements * x_{square\ sum} - x_{sum}^2} \tag{4}$$

(D) With the calculated slopes and y-intercepts, a crossing point is calculated where those two lines cross each other. This is the point, which depends on the rotation of the robot and its position between the two walls.

Based on step (C), with calculated slope and intercept for each side, the following parameters are defined:

k_L—slope of the left side,

n_L—y intercept of the left side,

k_D—slope of the right side,

n_D—y intercept of the right side.

and distances x_L and y_L can be calculated using Eqs. (5) and (6):

$$x_L = \frac{n_2 - n_1}{k_1 - k_2} \tag{5}$$

$$y_L = k_1 * x_L + n_1 \tag{6}$$

(E) Once the information about the intersection from the two lines is known, the position of the robot is calculated. The distance y_L stays constant, if the robot is aligned up with the row no matter which wall it is closer. The x_L describes how far away it is from both of walls/lines. With just looking at the two distances, the problem is simplified and can be solved with Eqs. (1) and (2).

Based on the position of the robot, the described approach can describe two different situations. The first situation is when the robot is in the right location, as shown in **Figure 4**, and the second if the robot is not in the right location and needs adjustment, as shown in **Figure 5**. In both cases, the parameters of the two linear equations and the distance to the path of the robot are used.

Since the walls are always apart from each other with a constant width, the triangle covers the same surface. What changes is the orientation and position changes with the robot that produces different triangles. If the robot is not aligned in parallel to the walls or crop lines, an asymmetric triangle is constructed, as the one in **Figure 5**.

2.3. Triangle-based navigation algorithm

The third approach [15] of finding an optimal path for the robot consists out of multiple steps. As shown in **Figure 6**, the algorithm uses trigonometric functions to calculate the distance of a segment between every two sequential points from LIDAR sensor. The segment distance must be wide enough to drive the robot trough, and if they do not meet the criteria, they are disposed. This is depicted in **Figure 6** where red colored segment distances written as (d_1, d_5, d_6, d_8, d_9, and d_{13}) are disposed and green colored distances (d_2, d_3, d_4, d_7, d_{10}, d_{11}, and d_{12}) are retained for further procedure.

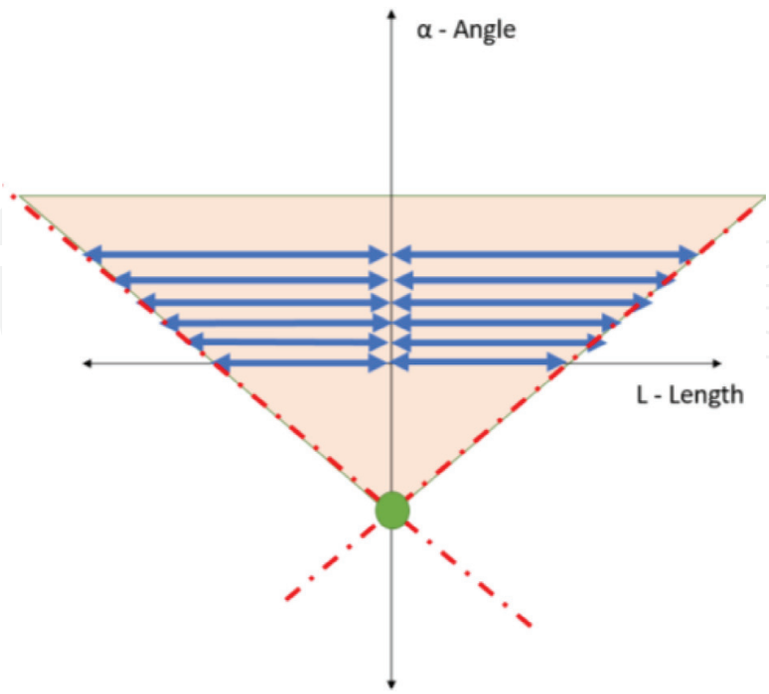

Figure 4. Arranged distances (blue) depicted on a graph that correspond to the length and the degree, it is located on. The red dotted line shows the two lines that are created out of these measurements. The green circle represents where the lines meet. This way a triangle is calculate and its shape depends on the position of the robot.

http://dx.doi.org/10.5772/intechopen.79942

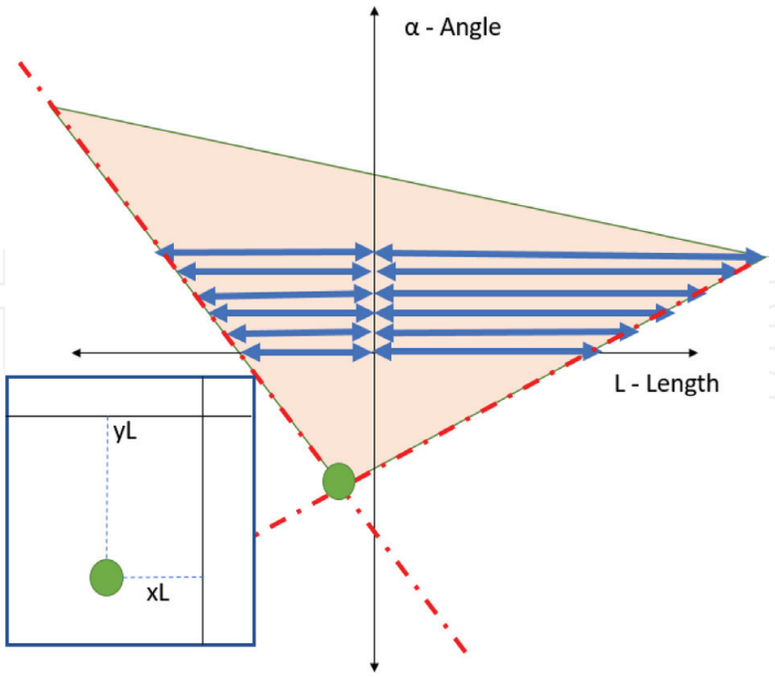

Figure 5. This picture shows us what we can get from the position of the intersection of the lines (distance y_L and x_L). We can figure out the position and the rotation of the robot with the two lengths.

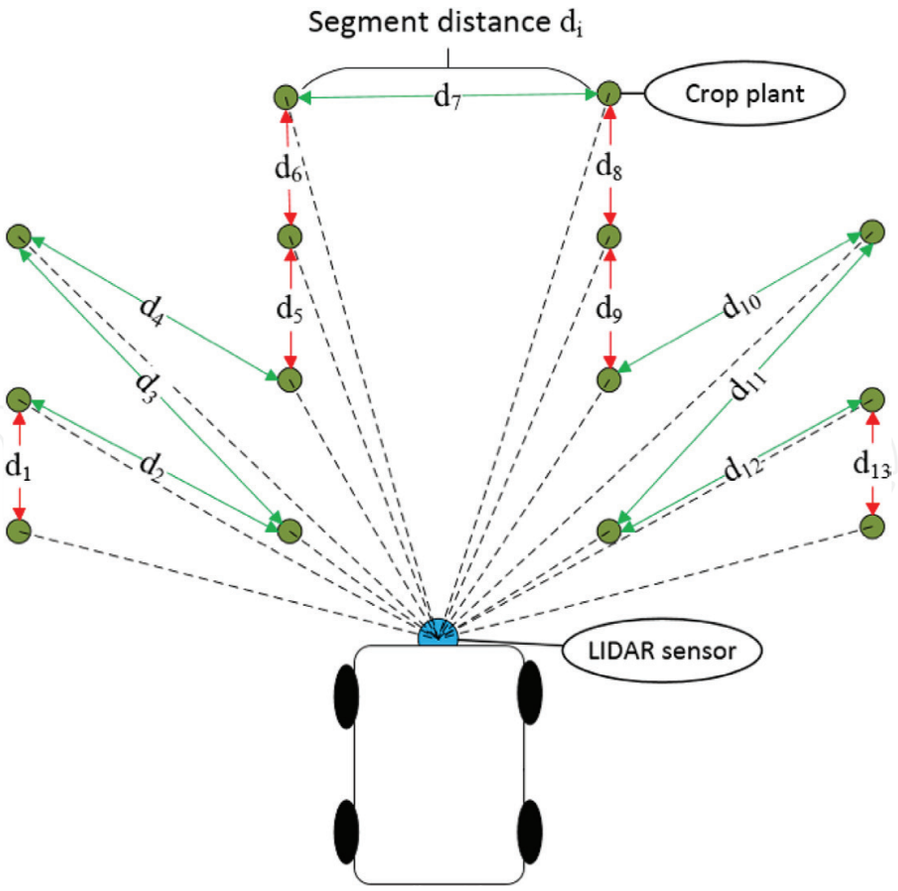

Figure 6. The way the robot calculates each segment of distances between every two sequential points from LIDAR sensor.

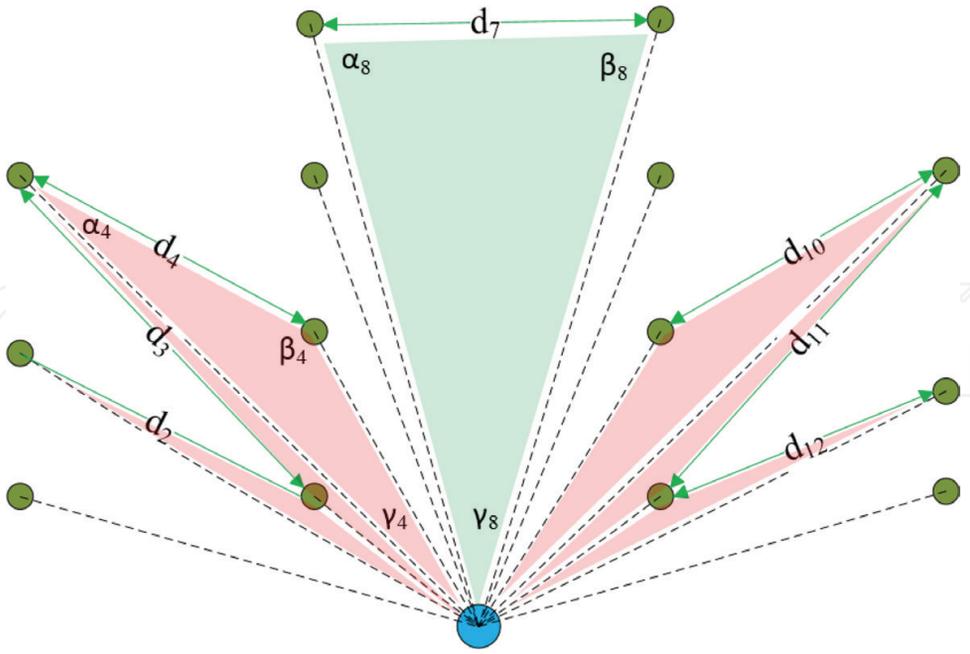

Figure 7. The procedure eliminates inappropriate solutions.

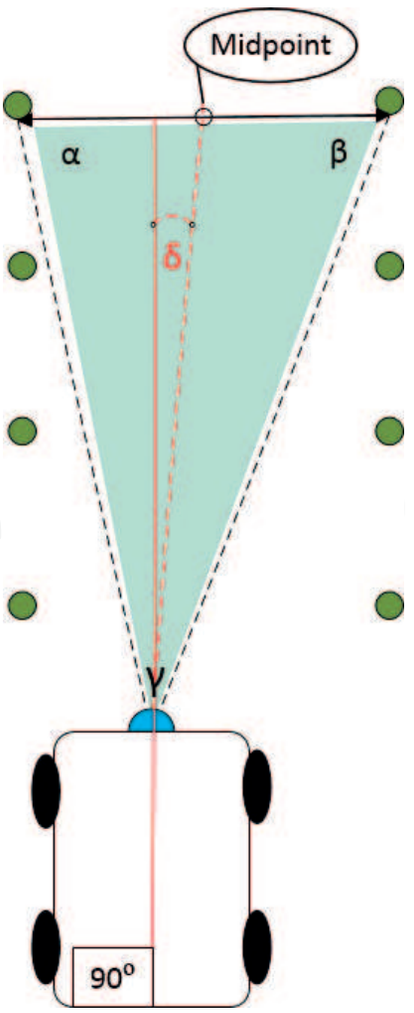

Figure 8. Calculating front wheels turn angle δ.

Figure 6 shows that only segment d_7 is appropriate for robot to drive trough. The procedure eliminates inappropriate segments as shown in **Figure 7**. Algorithm calculates the angles: α, β, and γ in triangle limited between two sequential points and LIDAR sensor. If any of the angles α or β is bigger than predefined threshold, set to 100° or more, which would produce a triangle that would not fit in field situations, the segment is disposed. Disposed segments are marked with red triangle in **Figure 7**, and the optimal segment d_7 which pass the criteria is marked as green triangle.

When the optimal segment is determined, the algorithm calculates the angle for the front wheels to turn. The midpoint of optimal segment is calculated for robot to drive to. Angle δ in **Figure 8** represents the angle for robot to turn front wheels to follow the midpoint of optimal segment.

3. Results

In order to test and compare all three algorithms from Section 2, two separate approaches are used. In the first, a simulated environment is built to test all three using precisely the same data sets in order to verify if they work. In the second approach, a real testing environment is used in order to evaluate the approaches in uncontrolled environment.

In contrast to the first experiment, where the algorithms were tested one time, the robot in the second approach is tested using one of three algorithms and repeated five times, for example, it drives between the two rows five times, in order to calculate an average absolute value with its standard deviation. The average value is computed for all three algorithms and evaluated as the average displacement from the center point and its standard deviation. The algorithm that preforms best should have the average closest to the real middle point, with as little deviation as possible, corresponding to a minimal oscillation pattern. **Figure 9** depicts an exaugurated principle of how the robot moves. The oscillation pattern is of course small and neglectable with some algorithms.

3.1. Simulated environment

A simulated environment was build using ROS stage simulator [18] in order to test all three algorithms before the application in real environment. In this experiment, all tests start at the same starting point, and the robot moves in semi-parallel direction to two artificial walls, mimicking two plant rows (**Figure 10**).

3.2. Real environment

In the second experiment, the robot moves between two plant rows. In this experiment, the data sets are not precisely the same as the plants can move due to wind. Even in no wind situation, the data sets might differ due to quantization steps of the LIDAR that might not be at the same location each time. **Figure 11** depicts the environment used in this approach.

Figure 12 depicts an example of how the both corn rows are sensed using a LIDAR sensor with readings depicted as colored dots.

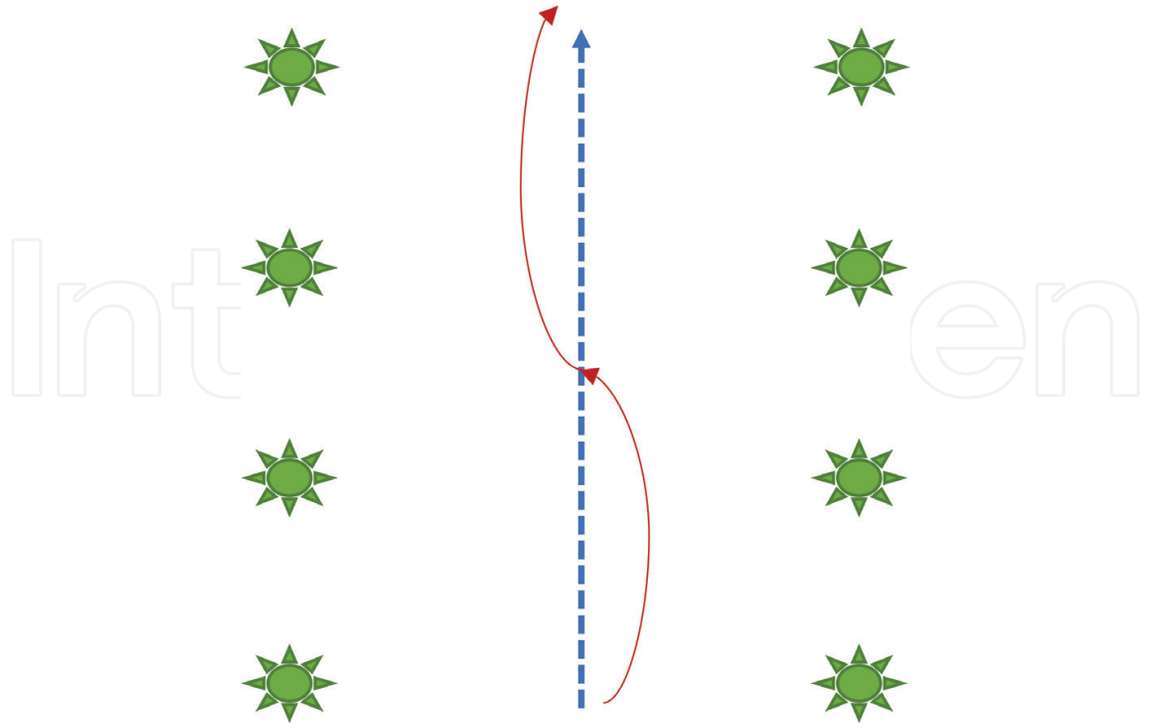

Figure 9. An example how the robot moves with the simplest algorithm. The movement is depicted with red line, the optimal mid-row path with blue, and the plants with green markers.

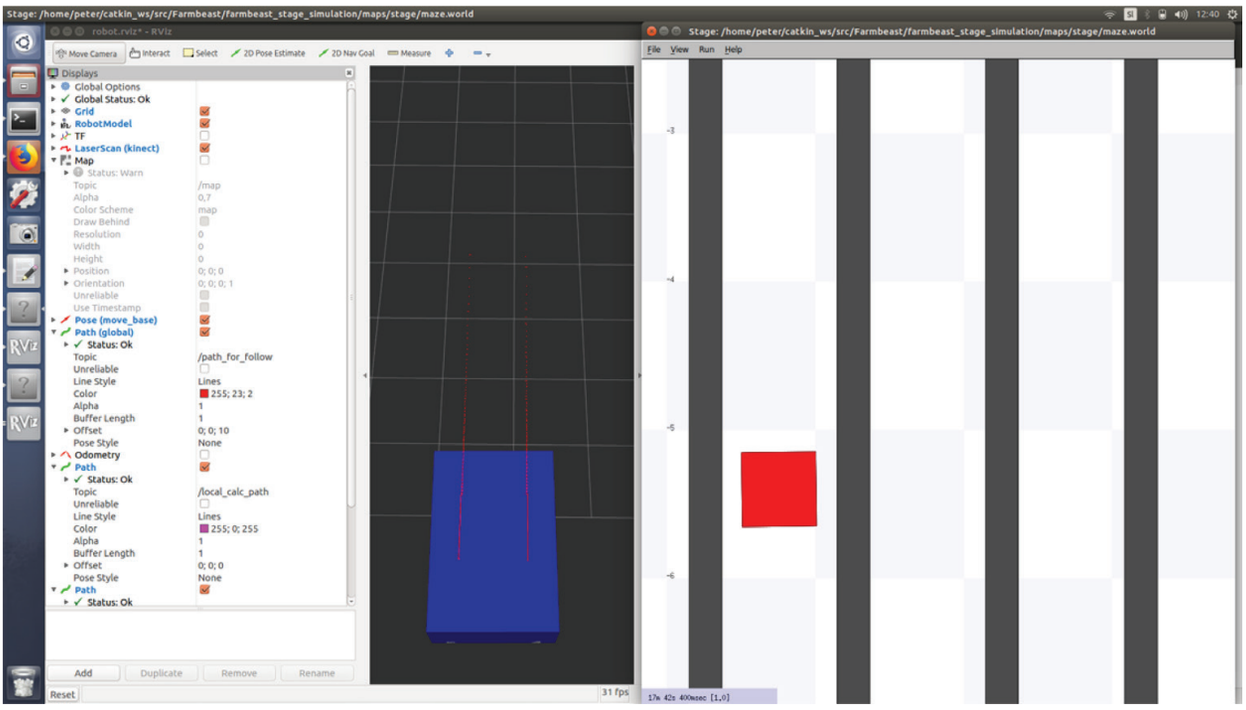

Figure 10. A screenshot of the simulation environment STAGE used in this test.

The results of the second test are presented in **Table 1**, where in each iteration, the average distance from the mid-row path was calculated on 8 m long test runs, but this time using real plants in real environment.

http://dx.doi.org/10.5772/intechopen.79942

Figure 11. An environment with real plants with 8 m in length and 0.75 m in width.

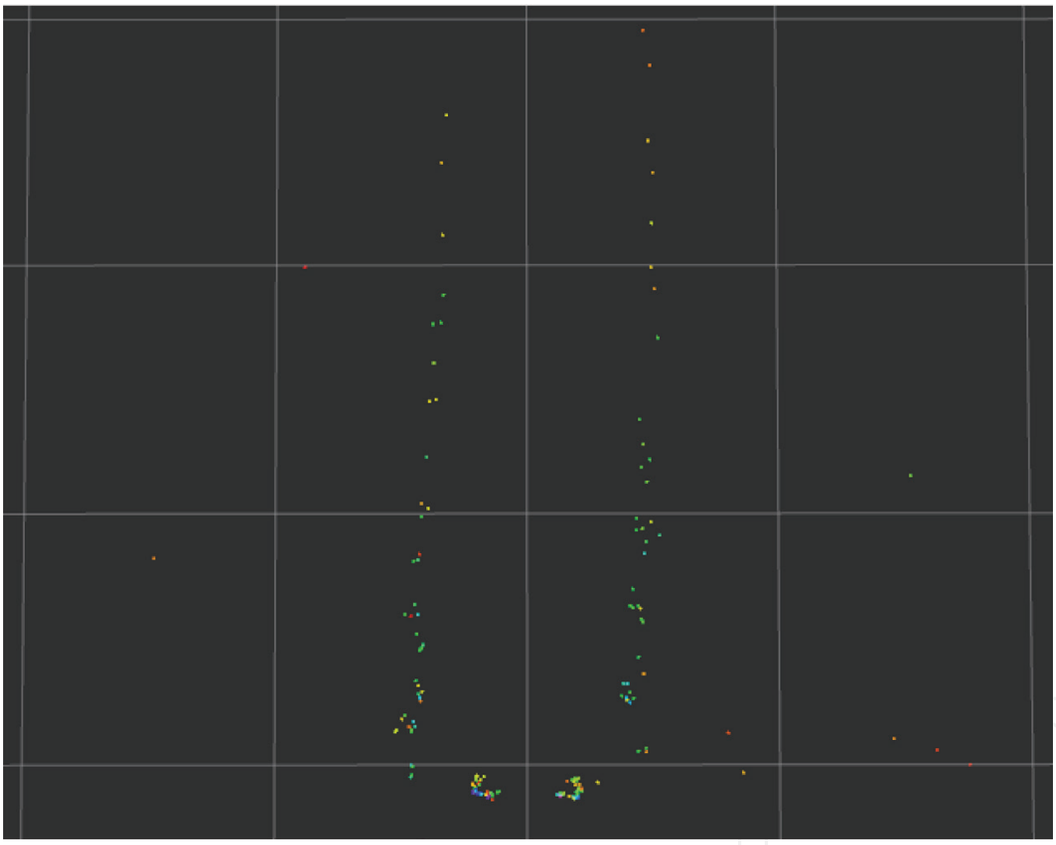

Figure 12. The environment from **Figure 12** seen by LIDAR sensor mounted on the front of the robot. The dots at the bottom center of the robot represent the wheels of the robot and are filtered out.

The results in **Table 1** show that the best performing algorithm is the algorithm from Section 2.1 with an error of 0.041 ± 0.034 m, and the second and third are very close with an average error of 0.07 ± 0.059 m for the second from Section 2.2 and 0.078 ± 0.055 m error for the third from Section 2.3. It should be noted that all three performed well reaching the end of each row without any problems. The difference between the first and other two is that the first uses the values in small proximity to the sensor on either side, but the second and third use a bigger range on both sides making them more useful in situations when rows are not straight as in

Iteration	Algorithm 2.1	Algorithm 2.2	Algorithm 2.3
First	0.04 ± 0.035 m	0.082 ± 0.06 m	0.069 ± 0.049 m
Second	0.038 ± 0.03 m	0.063 ± 0.049 m	0.071 ± 0.045 m
Third	0.045 ± 0.037 m	0.077 ± 0.073 m	0.073 ± 0.053 m
Fourth	0.039 ± 0.03 m	0.063 ± 0.044 m	0.07 ± 0.052 m
Fifth	0.042 ± 0.036 m	0.065 ± 0.056 m	0.099 ± 0.065 m
AVERAGE:	0.041 ± 0.034 m	0.07 ± 0.059 m	0.078 ± 0.055 m

Table 1. A performance comparison of three different algorithms from Section 2 using a real environment.

this experiment. The scenario in which the first algorithm might fail is in situation where the corn plants are bigger and the leaves from the corn over leap the mid-row space, obstructing a clear view for the sensor. In this case, the algorithms from Section 2.2 and 2.3 would be more efficient due to higher robustness in comparison to the first algorithm.

4. Conclusion

In this chapter, we presented three different algorithms for infield navigation and then tested them using first a simulated environment and, second, evaluated them in a real environment. The results from Section 3 show that all three algorithms perform good, with the best one in terms of optimal mid-row driving and minimal oscillation, the algorithm from Section 2.1 with an error of 0.041 ± 0.034 m, the second being algorithm from Section 2.2 with an error of 0.07 ± 0.059 m, and third, the algorithm from Section 2.3 with an error of 0.078 ± 0.055 m.

The algorithms presented in this chapter adjust the movement of the robot according to each iteration of measured distance sets recorded by LIDAR sensor. As a logical improvement to the best performing algorithms from Section 2, the accuracy could be improved by taking into account the measurements further away from the robot, measurements of the parallel lines, that are currently filtered out, as well as measurements of the previously driven row(s), as the rows are always in parallel to each other in which we get a prior information for the current rows. All this would further improve the performance of the algorithms.

Author details

Peter Bernad[1], Peter Lepej[3], Črtomir Rozman[2], Karmen Pažek[2] and Jurij Rakun[2]*

*Address all correspondence to: jurij.rakun@um.si

1 Faculty of Natural Sciences and Mathematics, University of Maribor, Hoče, Slovenia

2 Faculty of Agriculture and Life Sciences, University of Maribor, Hoče, Slovenia

3 Vistion d.o.o., Slovenska Bistrica, Slovenia

References

[1] Vazquez J, Lacarra E, Sanchez MA, Rioja J, Bruzual J. EDAS (EGNOS Data Access Service): Differential GPS corrections performance test with state-of-the-art precision agriculture system. In: 30th International Technical Meeting of The Satellite-Division-of-the-Institute-of-Navigation (ION GNSS+); Oregon; 2017. pp. 1988-1998

[2] Yang LL, Gao DH, Hoshino Y, Suzuki S, Cao Y, Yang SM. Evaluation of the accuracy of an auto-navigation system for a tractor in mountain areas. In: IEEE/SICE International Symposium on System Integration (SII). Taiwan. New York: IEEE; 2017. pp. 133-138

[3] Esau TJ, Zaman QU, Chang YK, Schumann AW, Percival DV, Farooque AA. Spot-application of fungicide for wild blueberry using an automated prototype variable rate sprayer. Precision Agriculture. 2014;**15**(2):147-161. DOI: 10.1007/s11119-013-9319-4

[4] Jiang GQ, Wang ZH, Liu HM. Automatic detection of crop rows based on multi-ROIs. Expert systems with applications. 2015;**42**(5):2429-2441. DOI: 10.1016/j.eswa.2014.10.033

[5] Ball D, Upcroft B, Wyeth G, Corke P, English A, Ross P, Patten T, Fitch R, Sikkareih S, Bate A. Vision-based obstacle detection and navigation for an agricultural robot. Journal of Field Robotics. 2016;**33**(8):1107-1130. DOI: 10.1002/rob.21644

[6] Kragh M, Jorgensen RN, Pedersen H. Object detection and terrain classification in agricultural fields using 3D Lidar data. In: 10th International Conference on Computer Vision Systems (ICVS). Denmark; 2015. pp. 188-197

[7] Reiser D, Ulrich G, Hübner R, Reihle D, Griepentrog HW. Hefty. In: Proceedings of the 15th Field Robot Event 2017; United Kingdom; 2017. pp. 15-24

[8] Kemmerling M, Sontag C, Schaub C, Wulferding H, Höverling S, Schröder T, Schneider L. Helios. In: Proceedings of the 14th Field Robot Event 2016. Germany; 2016. pp. 120-125

[9] Nachtigall L. Beteigeuze. In: Proceedings of the 13th Field Robot Event 2015. Slovenia; 2015. pp. 7-11

[10] Kohlbrecher S, Meyer J, Graber T, Petersen K, Klingauf U, Stryk O. Hector Open Source Modules for Autonomous Mapping and Navigation with Rescue Robots. Germany: Department of Computer Science, TU Darmstadt; 2013

[11] Lepej P, Rakun J. Simultaneous localisation and mapping in a complex field environment. Biosystems Engineering. 2016;**150**:160-169. ISSN 1537-5110

[12] Lepej P, Maurer J, Uran S, Steinbauer G. Dynamic arc fitting path follower for skid-steered mobile robots. International Journal of Advanced Robotic Systems. 2015;**12**:139. DOI: 10.5772/61199

[13] Shen TT, Radmard S, Chan A, Croft EA, Chesi G. Optimized vision-based robot motion planning from multiple demonstrations. Autonomous Robots. 2018;**42**(6):1117-1132. DOI: 10.1007/s10514-017-9667-4

[14] Quigley M, Gerkey B, Smart WD. Programming Robots with ROS: A Practical Introduction to the Robot Operating System. 1st ed. USA: O'Reilly Media Inc; 2015 448 p. ISBN-13: 978-1449323899

[15] Lakota M, Berk P, Rakun J, Kraner J. Cornstar. In: Proceedings of the 13th Field Robot Event. University of Maribor. Vol. 12; 2015. p. 20. ISBN 978-961-6317-48-1

[16] Bernad P, Zajc A, Rasl M, Lakota M, Rakun J. Farmbeast robot. In: Proceedings of the Field Robot Event. University of Hohenheim; 2018. (in press)

[17] Zhao L, Ding J. Least squares approximations to lognormal sum distributions. IEEE Transactions on Vehicular Technology. 2007;**56**(2):991-997. DOI: 10.1109/TVT.2007.891467

[18] Gerkey BP, Vaughan RT, Howard A. The player/stage project: Tools for multi-robot and distributed sensor systems. In: Proceedings of the International Conference on Advanced Robotics (ICAR 2003); 2003. pp. 317-323