Received: 19 October 2018 | Revised: 23 April 2019 | Accepted: 8 May 2019

DOI: 10.1002/rob.21878

SYSTEMS ARTICLE

WILEY

A low-cost and efficient autonomous row-following robot for food production in polytunnels

Tuan D. Le* | **Vignesh R. Ponnambalam*** | **Jon G. O. Gjevestad** | **Pål J. From**

Department of Science and Technology, Norwegian University of Life Sciences, Ås, Norway

Correspondence
Tuan D. Le and Vignesh R. Ponnambalam, Department of Science and Technology, Norwegian University of Life Sciences, Ås, Norway, 1433.
Email: tuan.dung.le@nmbu.no (T. D. L.) and vignesh.raja.ponnambalam@nmbu.no (V. R. P.)

Abstract

In this paper, we present an automatic motion planner for agricultural robots that allows us to set up a robot to follow rows without having to explicitly enter waypoints. In most cases, when robots are used to cover large agricultural areas, they will need waypoints as inputs, either as premeasured coordinates in an outdoor environment, or as positions in a map in an indoor environment. This can be a tedious process as several hundreds or even thousands of waypoints will be needed for large farms. In particular, we find that in unstructured environments such as the ones found on farms, the need for waypoints increases. In this paper, we present an approach that enables robots to safely traverse not only between straight rows but also through curved rows without the need for any predetermined waypoints. Most types of infrastructure found in agriculture, such as polytunnels, are built on uneven terrain, thus containing a mix of straight and curved plant rows, for which traditional methods of row following will fail. Different from traditional approaches of row following that only consider straight-line-of-sight rows, our approach identifies the rows on each side with the goal of staying in the middle of the rows, even if the rows are not straight. Waypoints are only needed on the very extreme of the rows, and these will be automatically generated by the system. With our approach, the robot can just be placed in the corner of the field and will then determine the trajectory without further input from the user. We thus obtain an approach that can reduce the installation time from potentially hours to just a matter of minutes. The final autonomous system is low cost and efficient for various tasks that requires moving between plant rows inside a polytunnel. Several experiments were performed and the robot demonstrates 1.4% position drift over 21 m of navigation path.

KEYWORDS
agriculture, planning, sensors, wheeled robots

1 | INTRODUCTION

In this paper, we address the problem of autonomous row following for an agricultural robot in a tightly constrained space such as polytunnels.

*Tuan D. Le and Vignesh R. Ponnambalam equally contributed to this work.

FIGURE 1 A design model of the Thorvald robot carrying ultarviolet light bulbs [Color figure can be viewed at wileyonlinelibrary.com]

This study is part of a larger project,[1] in which we develop agricultural robots to automate food production (Grimstad & From, 2017a, 2017b; Grimstad, Pham, Phan, & From, 2015). The Thorvald II robot has been used for different purposes in food production such as phenotyping (Grimstad, Skattum, Solberg, Loureiro, & From, 2017) and strawberry picking (Xiong, From, & Isler, 2018). The Thorvald II robot is a highly versatile robot due to its unique modular design.[2] The robot for example can be retrofitted to carry UV light bulbs for UV light treatment tasks, as shown in Figure 1. Currently, the model robot has been actively used at a cucumber greenhouse to provide UV light treatment (Grimstad, Zakaria, Le, & From, 2018), in addition to strawberry polytunnels.

In this paper, we address the problem of autonomous navigation in commonly found agricultural domains such as polytunnels or greenhouses. A polytunnel/greenhouse is a structured agricultural environment, where plants are grown in trays, which are organized as rows on top of several poles along the polytunnel. The rows are evenly spaced and spanned across the polytunnel and create a tightly constrained environment. For polytunnel-related tasks, the robot is usually required to navigate between plant rows. In a tightly constrained space such as polytunnels, curved rows make navigation more challenging.

We specifically aim to develop a low-cost and efficient autonomous system that is able to traverse through a polytunnel while performing assigned tasks without human intervention. The robot is equipped only with a planar laser scanner. The 2D laser scanner exploits the structured environment to provide navigation cues for the robot. To move along a row, a carefully designed RANSAC algorithm (Fischler & Bolles, 1987) is used to filter laser scans and reliably detect two parallel straight lines, which represent a part of the plant row on both sides of the robot. Note that a row comprises of several straight lines locally, which together form a curved row. A pure pursuit controller is implemented to make the robot follow the row. When the laser scanner cannot detect any parallel lines, the robot assumes it has reached the end of a row. It then switches to row transition mode to turn and enter the next row. The proposed navigation method has been tested in both simulations and in a mock-up polytunnel.

The main contribution of this paper is a novel autonomous navigation system that allows the robot to operate freely in a polytunnel. It is a low-cost and efficient system using only one type of sensor, a planar laser scanner. Even though row-following methods have been proposed in earlier work (Åstrand & Baerveldt, 2005; Bakker et al., 2008; Bergerman et al., 2015; Biber, Weiss, Dorna, & Albert, 2012; Hiremath, van Evert, van der Heijden, ter Braak, & Stein, 2012; Moorehead, Wellington, Gilmore, & Vallespi, 2012; Subramanian, Burks, & Arroyo, 2006; Zhang, Chambers, Maeta, Bergerman, & Singh, 2013), they might not be suitable for challenging constrained environments such as polytunnels.

This paper is organized as follows. In Section 2, related works are discussed. Section 3 provides details about the system including line detection and navigation. Simulated and experimental results are presented in Section 4. Conclusions are discussed in Section 5.

2 | RELATED WORK

Autonomous navigation systems are popular research areas, not limited to any particular fields or type of robots. Most systems, for example, Stoll and Kutzbach (2012), Cariou, Lenain, Thuilot, and Berducat (2009), and Biber et al. (2000), depend on several types of sensors such as inertial measurement units (IMU), high-precision RTK GNSS, 3D lidar, and so forth. Systems with high-precision RTK GNSS sensor navigate well only in open environments. Its performance will suffer inside a polytunnel because GNSS signals may be blocked. Inclusion of IMU will help with localization. Admittedly, fusion of multiple sensor types might yield better results in navigation; however, it also incurs a higher budget to the end users. Hence, in this study, we aim to develop a low-cost and efficient system.

There has been a lot of research on autonomous systems in agricultural applications, such as Bergerman, Singh, and Hamner (2012), Reid (2011), and Ting, Abdelzaher, Alleyne, and Rodriguez (2011) to name a few. Among those, autonomous row following has attracted interest (Åstrand & Baerveldt, 2005; Bakker et al., 2008; Bergerman et al., 2015; Biber et al., 2012; Hiremath et al., 2012; Moorehead et al., 2012; Subramanian et al., 2006). In Bakker et al. (2008) and Åstrand and Baerveldt (2008), even though the authors also develop autonomous systems for navigating between rows, they rely on cameras to perform a Hough transform for row detection. In Hiremath et al. (2012), a different method based on a particle filter to extract lines from images is proposed to detect row lines. The usage of computer vision for robotic applications has a long history. The main drawback for camera-based navigation systems is that they are totally dependent on lighting conditions. For example, UV light treatment needs to be carried out in a dark environment so that the effect of UV radiation is not nullified by sunlight or any other white light sources. In that situation, camera-based navigation fails. Hence,

[1] https://rasberryproject.com/

[2] https://sagarobotics.com/

FIGURE 2 A polytunnel for growing strawberry in Norway [Color figure can be viewed at wileyonlinelibrary.com]

a laser-based sensor is the most suitable candidate for navigation because it is independent of lighting conditions.

Navigation with 2D planar scanners has been a research topic for the last decades. One of the most extensively used solutions for autonomous navigation for ground mobile robots is *move_base*, a package that is implemented in robot operating system (ROS).[3] In order for the robot to move, one must provide a goal for the robot to reach. Topological navigation (Lázaro, Grisetti, Iocchi, Fentanes, & Hanheide, 2017) is one way to automatically generate goal points for the robot. However, the process to produce a topological map, which contains all the necessary goal points, is tedious and time-consuming because one must manually add all the goal points. Given the fact that a typical polytunnel is 60–120 by 9 m, the total number of goal points can be easily in the hundreds, which makes topological navigation unsuitable for the task (Figure 2).

Our proposed solution, on the other hand, does not rely on any a priori goal points. By detecting the two parallel lines in front of the laser scan, the robot follows the path between those lines. When it reaches the end of that path, it will continue to detect another set of parallel lines in front of it to follow. In case it cannot detect any more lines, the robot will try to determine whether it is possible to transit to the next row. First, the robot detects the number of poles currently in the field of view of the scanner. If the number of poles are more than two, the robot will go into transition mode, which makes it enter the next row. If the number of detected poles are less than two, the robot will stop moving because it has already reached the end of the polytunnel. With this solution, the robot can automatically traverse between all the rows inside a polytunnels, for example, to deliver a UV light treatment. The desired number of rows to traverse can also be predefined for the robot, so that the robot will cover only a specific area of a polytunnel.

We found that the work in Subramanian et al. (2006), Bergerman et al. (2015), and Riggio, Fantuzzi, and Secchi (2018) is similar to ours. Subramanian et al. (2006) also use a 2D laser scanner in combination with a camera for row following in a citrus grove. However, a challenging tightly space constrained condition like a polytunnel does not apply to their environment. Bergerman et al. (2015) also use 2D laser scanner to navigate in rows in tree fruit orchards but required reflective landmarks for row transition, which we do not. Similar to Subramanian et al. (2006), the robot in Bergerman et al. (2015) does not have to deal with tightly space constrained environment. In Zhang et al. (2013), the authors employ a spinning 2D laser scanner to detect 3D positions of tree rows and tree trunks in orchards for row following. The spinning 2D laser scanner generates 3D point cloud for registration. In comparison, a tree trunk is much bigger than a steel pole used in a polytunnel. Hence, 3D detection might not detect poles. Furthermore, like previously mentioned methods, orchards environment is not tightly space constraint as polytunnels. In Riggio et al. (2018), the authors developed a similar low-cost system of row following using only a 2D laser scanner but did not explicitly address the problem of following curved rows.

3 | NAVIGATION INSIDE A POLYTUNNEL

To navigate inside a polytunnel, the robot must be able to localize itself inside a given environment. We used adaptive Monte-Carlo localization (AMCL)[4] (Thrun, 2002), the de facto SLAM method for 2D laser scanner without further development. The navigation strategy for the robot inside a polytunnel is as follows. The robot is positioned in front of a row. The robot can only see the poles, and not the tables placed on the top of the poles or the plants. By detecting virtual lines between poles, the robot traverses plant rows by following the central path between them. When the robot reaches the end of a row, for example, it cannot detect any more lines, the transition row module is activated to get the robot to the next row. The navigation system as in Figure 3 comprises of row following and row transition module that can operate seamlessly in and out of the polytunnel rows. The robot localizes itself relative to poles using a prebuilt map. The laser scanner will monitor consistently for the static or the dynamic obstacle in front of the robot and make an emergency stop if an obstacle is detected within the boundary region. The robot will remain stopped until the detected obstacle is moved by itself or by the nearby worker since there is not an adequate area for avoiding them.

[3]http://wiki.ros.org/move/_base

[4]http://wiki.ros.org/amcl

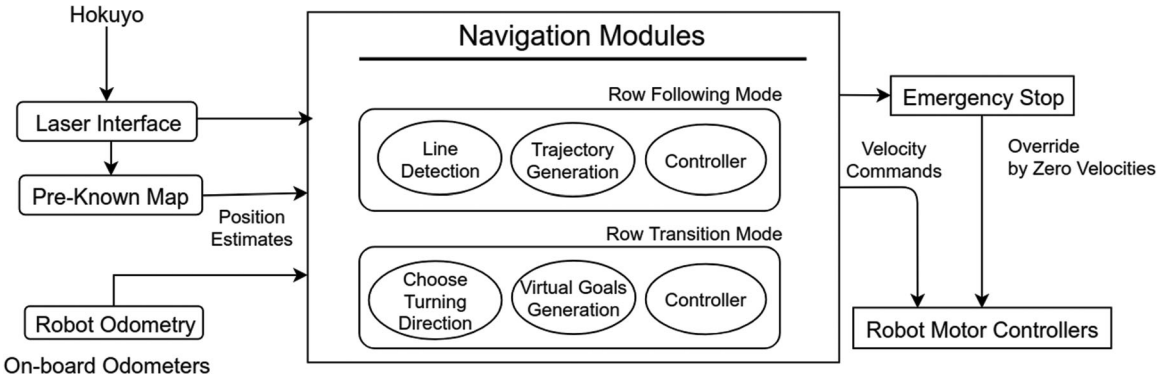

FIGURE 3 Modules of navigation

3.1 | Line detection and following

The laser scanner can detect the poles that are aligned along every polytunnel's rows. This section of poles can be coupled together as virtual lines so the robot can navigate by following the generated trajectory between them as illustrated in Figure 4. A RANSAC algorithm (Tarsha-Kurdi, Landes, & Grussenmeyer, 2007) is implemented to fit a pair of poles as individual line features l_n. Unlike the solid walls, the laser scanner observes the poles in the polytunnels as a cluster of points at equal distant from each other. This scenario makes it challenging for the line detection algorithm and consequently, a bounding box is established with a designated boundary region R of length 6 m and width 2 m in size (Figure 5b). The designated search boundary region R is constructed for the sake of eliminating the scan points from another rows as best line fits. From the laser scanner data, a data set $a_t^n = [a_{x_t}^1, a_{y_t}^1, a_{x_t}^2, a_{y_t}^2, ..., a_{x_t}^n, a_{y_t}^n]$ is generated which contains the x and y axis positions of n number of scan points that are extracted from the laser scan range and bearing values within the boundary region R.

To execute the line detection algorithm, let us suppose that the line model L_t can be expressed as a function $f(S_t)$ which depends on randomly generated subset of points in S_t taken from the set a_t^n as

$$L_t = f(S_t), \tag{1}$$

where $S_t = [p_{x_t}^1, p_{y_t}^1, p_{x_t}^2, p_{y_t}^2] \subseteq a_t^n$ comprises the position of the two randomly generated points p^1 and p^2 from the set a_t^n at time t, respectively. The function $f(S_t)$ computes the model line parameters such as slope m_t and y-intercept b_t based on the two randomly selected points $p_{(x_t,y_t)}^1$ and $p_{(x_t,y_t)}^2$ is given by

$$\begin{aligned} m_t &= \frac{p_{y_t}^2 - p_{y_t}^1}{p_{x_t}^2 - p_{x_t}^1}, \\ b_t &= p_{y_t}^1 - m_t p_{x_t}^1. \end{aligned} \tag{2}$$

When the line model parameters are computed, let $E(L_t, a_t^n)$ be the objective function constructed using the least squares method for line fitting (York, 1966). The objective function $E(L_t, a_t^n)$ proclaims the sum of all the residual values for each point belonging to the set a_t^n with respect to the estimated line model L_t. Therefore minimizing the objective function $E(L_t, a_t^n)$ will eventually minimize the residual values so that the estimated line will be close enough to most of the points from set a_t^n. The optimal set of line model parameters m_t and b_t are need to be found as per least squares method for minimizing the residuals. Hence the objective function minimizing the sum of the the squared normal distances from each point takes on the form:

$$E(L_t, a_t^n) = \sum_{p=1}^{n} \| (a_{y_t}^p - m_t a_{x_t}^p - b_t) \|^2. \tag{3}$$

The standard RANSAC algorithm has few parameters defined beforehand as preconditions that are suitable for the polytunnel environment. For minimizing the objective function $E(L_t, a_t^n)$, a

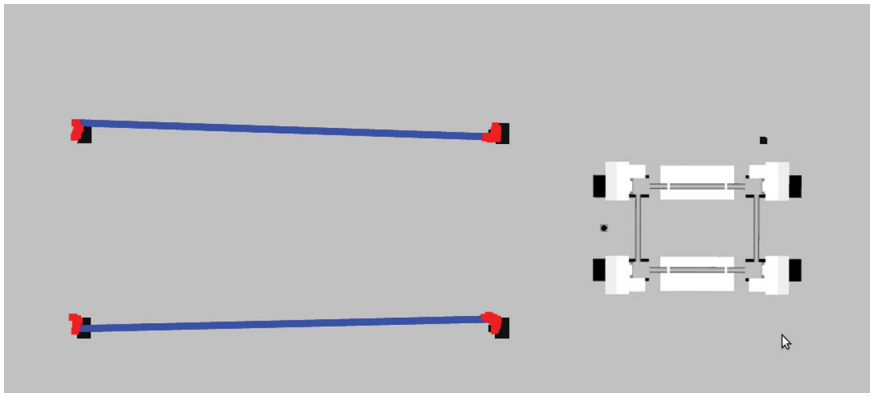

FIGURE 4 The proposed line detection algorithm identifies two lines (blue) from the laser scanner points using a Hokuyo laser which is mounted on the robot. The standard RANSAC algorithm is used for line extractions. The robot detects only two poles on each side [Color figure can be viewed at wileyonlinelibrary.com]

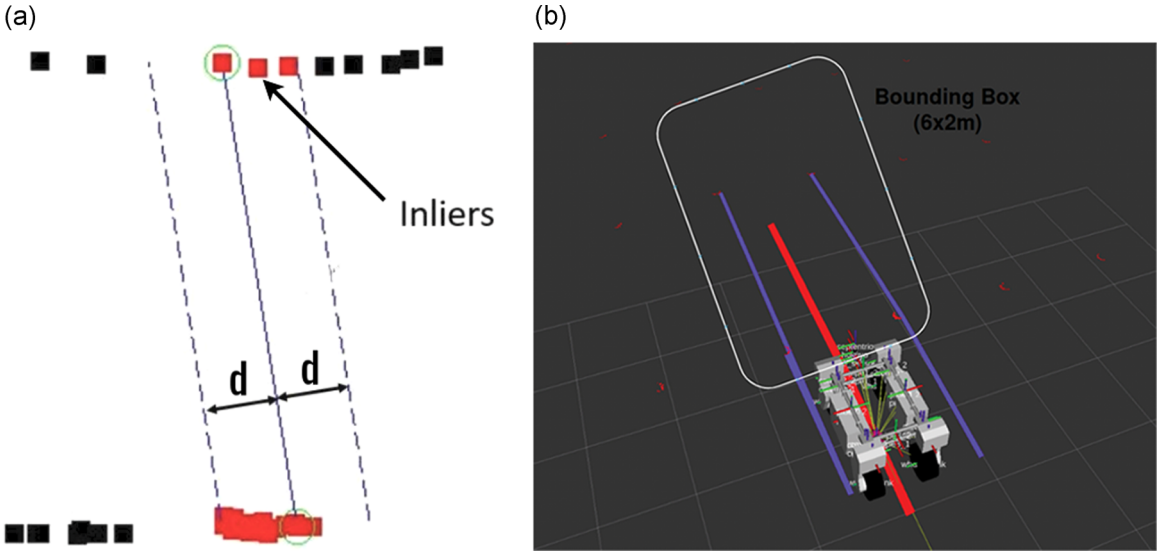

FIGURE 5 Row-following module: Line detection methodology. (a) Two randomly selected laser points (green) classifies the remaining laser points within d limits as inliers (red). (b) Detected pair of lines (blue) and desired path (red) within the bounding box region (white) [Color figure can be viewed at wileyonlinelibrary.com]

threshold parameter d is introduced which represents the threshold distance from the two chosen random points for fitting the remaining scan points as inliers (see Figure 5a). The parameter k describes the total iterations required to determine the best line fit and therefore it will keep updating the best line fit if the better line feature with more inliers are found for the entire kth number of iterations. The parameter inliers_{min} represents the minimum number of inliers to be necessitated for finalizing the estimated line as a best line model. The RANSAC estimates the line after the predefined conditions are satisfied. Thus the parameters for the preconditions are tuned in such a way that it fits the parametric line model given as

$$\begin{aligned} &E(L_t, a_t^n) \le d, \\ &\text{where} \quad d = 0.05(\text{m}), \quad k = 100, \quad \text{inliers}_{min} = 10. \end{aligned} \tag{4}$$

The RANSAC will run twice such that after satisfying the parameters in the preconditions, two best line fits will be estimated. Thus the points from the set S_t are incorporated as line features l_1 and l_2. For the sake of simplicity, we do not include time t in the line feature equations. As soon as the RANSAC algorithm identifies the lines $l_1 = [l_1^{x1}, l_1^{y1}, l_1^{x2}, l_1^{y2}]$ and $l_2 = [l_2^{x1}, l_2^{y1}, l_2^{x2}, l_2^{y2}]$ by satisfying the predefined conditions, some additional constraints are considered to avoid multiple detections, overlaps, and other false positives (see Figure 6). The false detections are ignored. If no pair of lines is detected, the algorithm uses the previous detections until the new set of lines appears.

These constraints aid in fitting the best line features for the entire navigation system. The first constraint is the minimum distance between the end-points $(p^1_{(x_t,y_t)}, p^2_{(x_t,y_t)})$ for each of the two detected lines (l_1^t, l_2^t) that has to be always more than the threshold value τ as in Equation (5). This particular constraint avoids the possibility of detecting the incorrect best line fit when more inliers are stacked up together at one place (the black circled area in Figure 6a) and it is expressed as

$$\begin{aligned} &\sqrt{(l_1^{x1} - l_1^{x2})^2 + (l_1^{y1} - l_1^{y2})^2} > \tau, \\ &\sqrt{(l_2^{x1} - l_2^{x2})^2 + (l_2^{y1} - l_2^{y2})^2} > \tau. \end{aligned} \tag{5}$$

While traversing through the inclined shaped rows, the robot can also find the line features diagonally between two parallel rows as the best line fit at the same time (Figure 6b). For avoiding this situation, the constraint based on the angle between two end-points of the detected line is introduced. This angle is presumed to be less than Φ which is assigned as 15° at maximum so that it can still detect the curved shaped poles but it can also avoid finding cross line detections at the same time. The second constraint can be written as

$$\begin{aligned} &\arctan \frac{l_1^{y2} - l_1^{y1}}{l_1^{x2} - l_1^{x1}} < \Phi, \\ &\arctan \frac{l_2^{y2} - l_2^{y1}}{l_2^{x2} - l_2^{x1}} < \Phi. \end{aligned} \tag{6}$$

There is another possibility of false detection in which the RANSAC could detect the already chosen best line fit as second best line fit again (see Figure 6c) because the line detection will keep finding the two best line fit l_1 and l_2 at time t and this case will also satisfy the first and second constraints as well. Therefore the third constraint is proposed as

$$l_1^{x1} \neq l_2^{x1} \quad l_1^{y1} \neq l_2^{y1} \quad l_1^{x2} \neq l_2^{x2} \quad l_1^{y2} \neq l_2^{y2}. \tag{7}$$

This added constraint as given in Equation (7) will avoid the situation where both the detected lines do not overlap each other. If the overlapping is detected using this constraint, then this pair of lines from the concerned iteration in RANSAC is rejected. After fulfilling all the three proposed constraints, the two lines l_1^t and l_2^t will be extracted on both sides of the robot to navigate between them. The desired trajectory has been derived as an average of the two estimated lines as in Figure 5b. Once the desired trajectory has been estimated, a low-level controller is

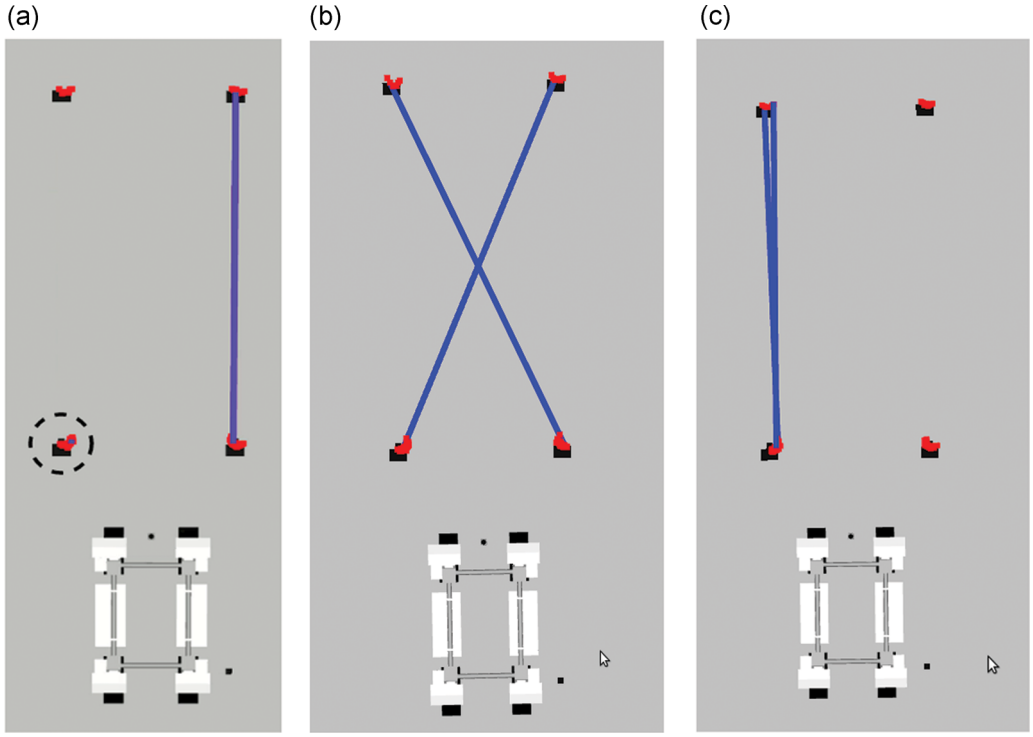

FIGURE 6 False cases of line detection. (a) False line detections covering not more than one pole (black circled); (b) cross line detections; (c) overlaps in line detections [Color figure can be viewed at wileyonlinelibrary.com]

used for sending the necessary velocities as joint commands. The linear velocity V_t is constant and it moves at 0.3 m/s for safety reasons. To steer the robot, a low-level pure pursuit controller (Coulter, 1992) is used to calculate the respective steering velocity ω_t for following the center line based on two estimated lines l_1^t and l_2^t by the line detection algorithm as in Figure 5. The steering velocity ω_t equation is written as

$$\omega_t = \arctan\left(2R_l \frac{\sin e_{\theta_t}}{e_{(x_t,y_t)}}\right) \tag{8}$$

where R_l, e_{θ_t}, $e_{(x_t,y_t)}$ are the total length of the robot, errors along its rotation angle and its position with respect to the current goal at the time t respectively.

Algorithm 1: Navigation algorithm

input : 2D laser scan measurements, number of rows to traverse

```
while the end of the polytunnel is not reached do
    for each laser scan measurement do
        Perform RANSAC fitting for 2 lines l1 and l2;
        if no lines are detected then
            Check whether the robot has reached the end of the last row;
            if true then
                Stop;
            else
                Initialize row transition;
                Transits to next row;
            end
        else
            Compute the middle trajectory between l1 and l2;
            Follow the middle trajectory;
        end
    end
end
```

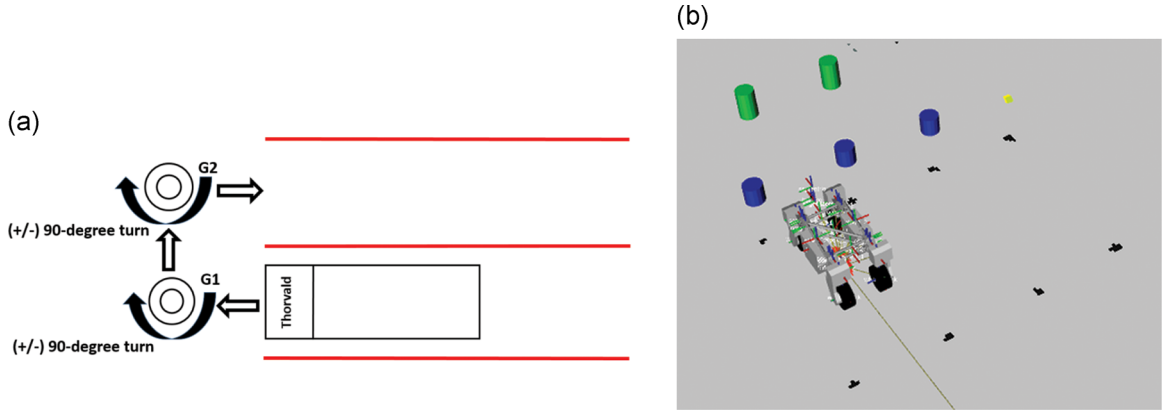

FIGURE 7 Row-transition module: Row-transition methodology. (a) Robot assigned to reach goal points (G1, G2) and make (+/–) 90° turn around the goal points for transiting into new row. (b) Poles at end of the row (blue) are detected and generate goal points (green) by implying an offset to it [Color figure can be viewed at wileyonlinelibrary.com]

3.2 | Row transition

Once the row-following module could not detect any more new lines, the robot will navigate till the end of the current desired trajectory using the last pair of detected lines. When it reaches the end of the current desired trajectory, it shifts autonomously to the row transition module. The operation of the row transition comes to an end when the robot progressed to the beginning of the next row and switches back to the row-following module.

In this module, the pole detection algorithm identifies the closest three poles which comprises of two poles on one side and another one pole on the other side of the robot based on its next course of direction. For instance, if the robot needs to transit to the new row on the right-hand side then the pole detection algorithm will give the pair of closest poles on right-hand side (Figure 7b) and one pole on the left-hand side or vice versa for the turning to the next row on the left-hand side condition. Thus the virtual goal points are generated by taking the average between the three detected poles and adding a constant offset to it as seen in Figure 7. Then the pure pursuit controller is designed in such a way that it will navigate the robot to the first virtual goal point from the current row and makes a 90° turn around that first goal point. Furthermore, it repeats the same process for the second virtual goal point (Figure 7a) to shift into the new row. Therefore the course of the turning direction should be given beforehand such that the robot can navigate any polytunnels which has a larger number of rows. Moreover, the row transition module brings the integration with the row-following module and makes a complete autonomous navigation system exclusively for polytunnels like environments. The pseudocode (Algorithm 1) exhibits the integration of both the navigation modules for both the straight and curved shaped polytunnels.

4 | EXPERIMENTAL RESULTS

4.1 | Simulations

The proposed method is verified in simulation and field trials. We show that our system can move along rows efficiently. We also discuss how our system can be extended to different environments, such as polytunnels that hang plant trays instead of using poles.

The simulated environment (Figure 8) is created using Gazebo to mimic the real polytunnel environment. It consists of a plane ground and several sets of cylinders with plant trays on top. The spacing between each set of cylinders can be modified to match reality. In this environment, the robot is tasked to traverse all the rows, while in reality, it might not have to do so due to the requirement of UV light treatment, for example, not every row requires UV light treatment. In the simulated environment, the robot is fixed at an initial known position in front of one of the rows. The row-following module in the navigation system begins traversing through the initial row that it perceives first in the environment. The robot can find the curved shaped rows and extract the trajectory lines by the line detection technique at every time step t. Then, the robot can steer in both clockwise and counter-clockwise directions and can revert back to straight row following with the lesser amount of steering as shown in Figures 910.

The pure pursuit controller in both the row-following module and the row-transition module assist the robot to steer between and outside the rows of the polytunnels. The error in the graph indicates the angular difference between the current robot position and current dynamic goal position that the controller should correct at each time step t. The controller maintains the required steering angle error to be less than 0.15 radians in row-following and 0.01 radians in row-transition modules. They can maintain the error close to zero as shown in Figure 9 throughout the entire trajectory. The steering error increases whenever the robot needs to traverse through curved areas in the rows but it reduces again over time. In the row transition module, the controller makes the robot move along the two virtual goal points with the given steering commands and transit to the beginning of the next row.

4.2 | Navigating in a mock-up polytunnel

The robot used in the field test is shown in Figure 13. We constructed a mock-up polytunnel, which is 24 m long by 9 m

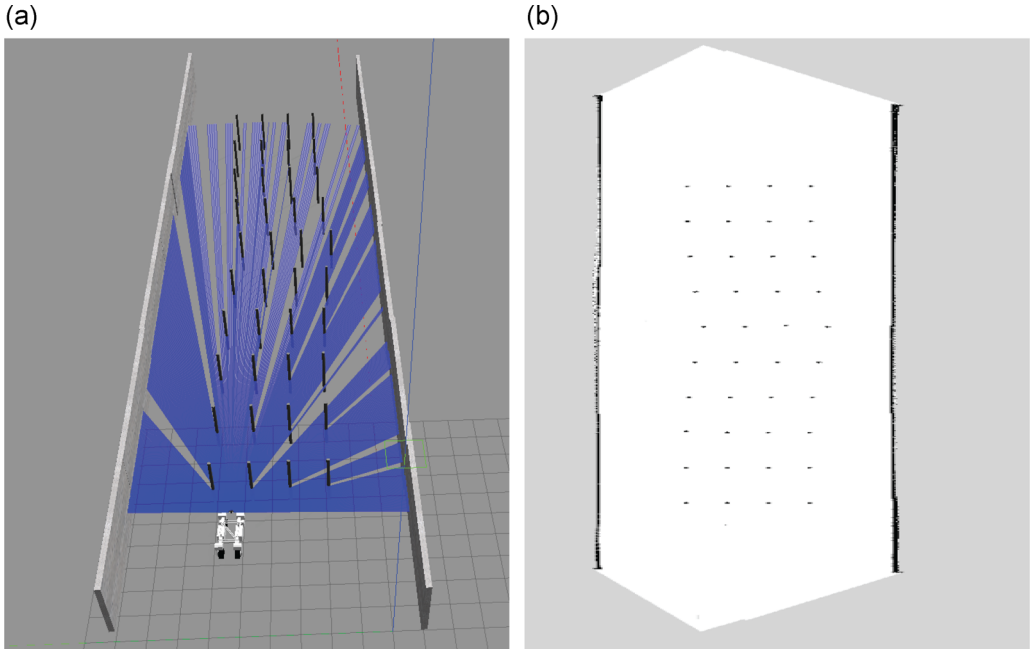

FIGURE 8 Experiments in simulated environments. (a) Curved poles environment in gazebo; (b) two-dimensional map of polytunnel environment [Color figure can be viewed at wileyonlinelibrary.com]

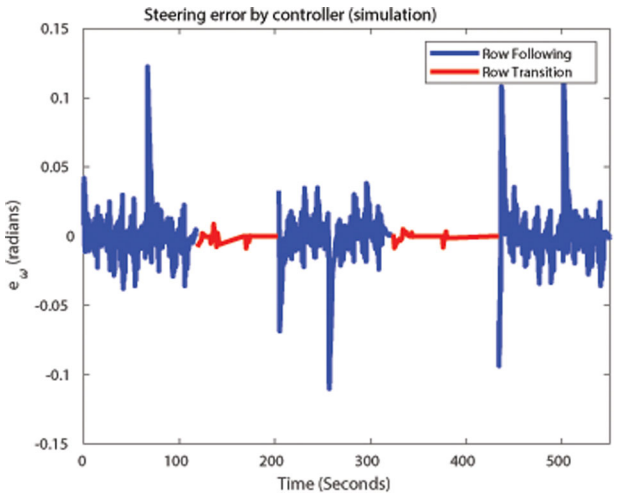

FIGURE 9 Steering controller error in simulated polytunnel. Steering errors when the robot follows rows are shown in blue. Steering errors when the robot changes rows are shown in red. Best view in color [Color figure can be viewed at wileyonlinelibrary.com]

wide. The mock-up polytunnel has 32 poles, which create three long rows. We deliberately added constant displacements to 12 middle poles (inside the square) as shown in Figure 12a. In Figure 12, we show how the robot detects parallel lines and moves in the center of a row. The curved lines are detected and shown on Figure 12d–g. Transition points (two green circles) are shown in Figure 12i. They are automatically computed when the robot reaches the end of the row and detects three poles (blue circles) in front of it. The whole trajectory is shown in Figure 12j.

The robot is tasked to traverse through every row. We evaluate the robot navigation quality by two metrics, including displacement to centering lines and distances to poles on both sides of the robot. We explain these metrics in details next.

(a) *Translation error*: All the poles' positions are carefully measured using a Leica Total Station TCA1100 with distance measurement accuracy ±1 mm. These measurements are for computing virtual centering line segments on each row, which are considered ground truth. We evaluate the quality of navigation by calculating the deviation of the actual trajectory from the ground truth. For each segment of a row, we first compute the Euclidean distance of each robot position measured by the Leica to the ground truth. The average of these distances is considered as the robot translation error on that segment. For the whole row, we again average all the translation errors of all segments. The reason we choose this metric is that it provides insight into how the robot performs on each segment of a row.

(b) *Distance to poles error*: We measure the distances from the center of the robot body to the poles on both of its sides when passing them. The distance from the robot body center to each pole on the left and right sides of the robot are taken by a laser measurement Uni-T UT390B^+ with ±2 mm accuracy. These measurements are used to evaluate how well the robot stays in-between poles.

We collect our metric measurements by letting the robot run autonomously through the mock-up tunnel 10 times. The final result is averaged over these 10 trials.

The result of ground truth comparison is shown in Figure 14. The ground truth trajectory (blue line) consists of line segments representing the ideal trajectory that stays exactly in the middle of rows. The translation errors are shown in Table 1. We ignore the robot trajectory that is outside rows. Given the average path of each

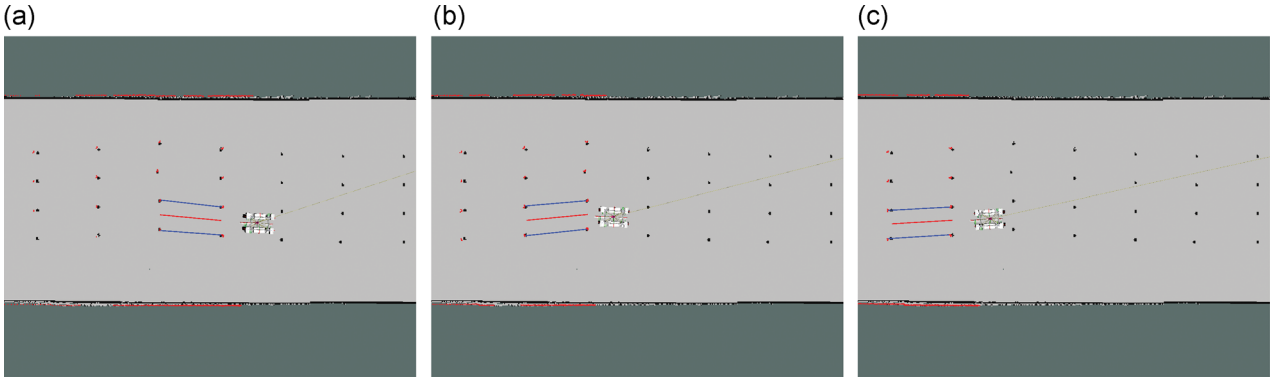

FIGURE 10 Snapshots of the robot movement in simulation. (a) Clockwise turning movement; (b) anticlockwise turning movement; (c) return to straight-line movement [Color figure can be viewed at wileyonlinelibrary.com]

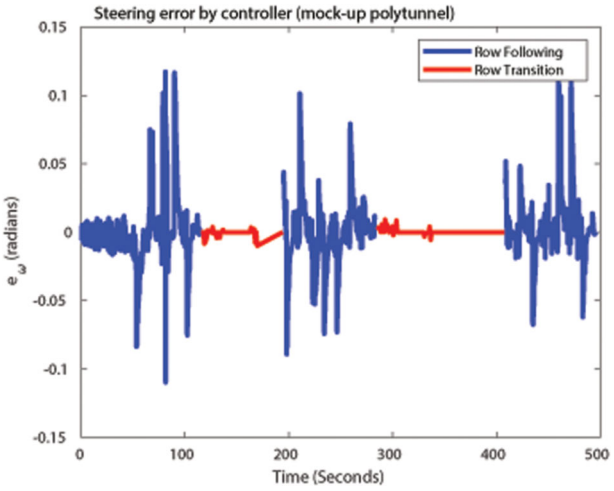

FIGURE 11 Steering controller error in mock-up polytunnel. Steering errors when the robot follows rows are shown in blue. Steering errors when the robot changes rows are shown in red. Best view in color [Color figure can be viewed at wileyonlinelibrary.com]

row is 21 m, the maximum mean error is only 29.3 cm, which yields a relative small 1.4% drift over a traveled distance.

In Table 2, the results of staying in-between poles are presented. The distance between two poles on each side of a robot is 1.5 m. The robot width is 1 m. It means the robot needs to stay at least 25 cm away from poles on each side. The maximum mean distance to poles on the right side is approximately 25.1 cm, and to the left side is 28.2 cm. This shows that the closest distances the robot gets to a left and a right pole are approximately 24.9 and 22.8 cm, respectively, which are well within the ideal safety distance. Also, the maximum difference between the mean distances on both sides of the robot is 8.9 cm. It shows that the robot well maintains its position at the center along rows by keeping the same distances to poles on both sides of a row.

The pure pursuit controller in field tests behaves in a similar way to the simulation. Unlike the simulated environment, the poles are not aligned perfectly straight with respect to each other in real fields. The lines that are detected from line detection algorithm are not exactly straight as well; hence the steering error in the real-field tests is higher than the one in the simulation. As in Figure 11, the curved areas are evident in which the steering error peaks in each row along the mock-up polytunnel. In the row transition module, the steering error is kept to a low value even in the uneven terrain that are similar to simulations (Figures 12–14).

We also compare our proposed method with Krajník et al. (2014a). The topological navigation approach proposed in Krajník et al. (2014a) is currently being used in our RASBerry project.[5] To use topological navigation, one needs to manually create *topological nodes* that connect each other to form a topological map as shown in Figure 15. Given a topological map, the robot can move from one arbitrary node to another. This method relies on AMCL for localization, which is similar to ours.

We run 10 trial tests and collect the same metric measurements for comparison. The final result of topological navigation is averaged over 10 runs and shown in Tables 1 and 2 altogether with our proposed system for comparison. Bold numbers indicate better results. Our method on average achieves better result in both metrics.

We also note that the topological navigation requires creating topological nodes, which must be done manually and therefore unsuitable for a large polytunnel. This is one of the motivations of our proposed method. One might argue that a *sparse* topological map would be easier to make. However, we found that in practice, a tightly constrained space requires a *dense* topological map for the robot to travel safely. In addition, by relying on a *cost map* for planning, the robot is prone to make dangerous path planning such as in Figure 16, whereas our method does not. Furthermore, the use of *cost map* makes the system more sensitive to faulty reading from sensor. As shown in Figure 16, an artificial obstacle due to noisy laser scanner was added to the *cost map* and forced the robot to move out of the row. This is a dangerous behavior. The robot is likely to collide with other poles because there is not enough space for rotation. Hence, we claim that our proposed system is better suited for polytunnel environment.

A video of the robot moving in the mock-up polytunnel is available online: https://youtu.be/xkSpEkcBXaU.

[5]https://rasberryproject.com/

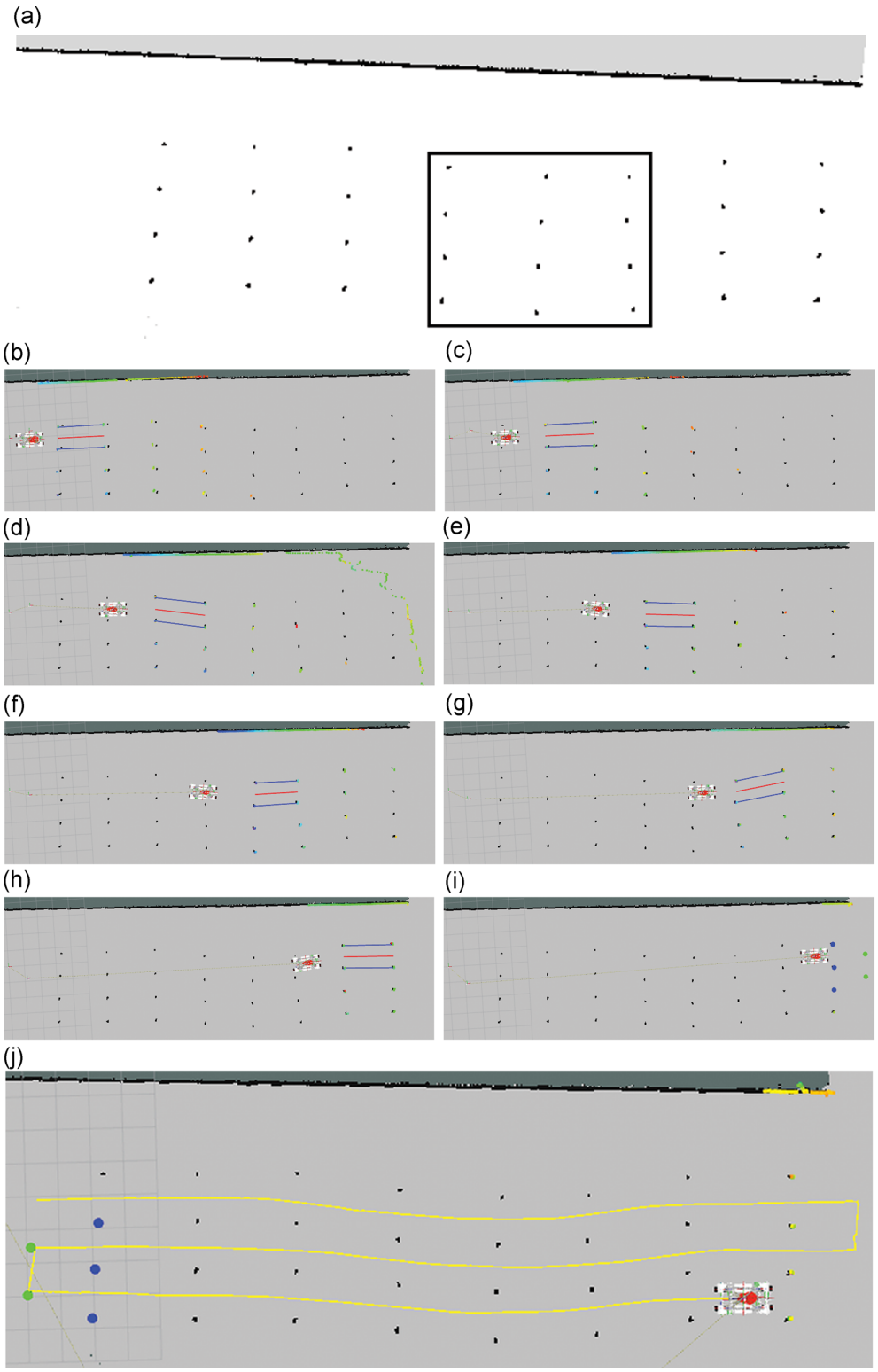

FIGURE 12 Results of autonomous navigation between rows in a mock-up polytunnel. Blues lines are virtual lines between detected poles. Red lines are the central paths between rows. Yellow line is the complete actual trajectory [Color figure can be viewed at wileyonlinelibrary.com]

4.3 | Discussion

One key aspect of our system is low cost. However, we have shown that autonomous navigation in tightly constrained agricultural domains such as polytunnels can be carried out efficiently. Our system works well in the challenging environment of polytunnels, where features for laser scanner detection are sparse. Our system can be easily adapted to different types of polytunnels without much

FIGURE 13 Robot setup, a 2D Hokuyo laser scanner UST-20 LX is mounted in front of the robot as shown in the highlight area [Color figure can be viewed at wileyonlinelibrary.com]

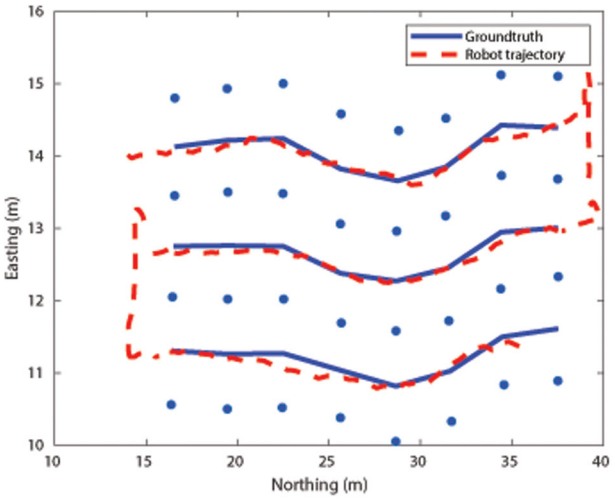

FIGURE 14 Qualitative analysis of trajectories [Color figure can be viewed at wileyonlinelibrary.com]

TABLE 1 Translation errors per row

	Translation errors (m)					
	Minimum		Average		Maximum	
	Ours	Krajník et al. (2014a)	Ours	Krajník et al. (2014a)	Ours	Krajník et al. (2014a)
Row 1	0.052	0.024	0.117	0.128	**0.165**	0.169
Row 2	0.101	0.101	0.213	0.187	0.293	**0.292**
Row 3	0.063	0.131	0.154	0.237	**0.203**	0.386

Note: Bold numbers indicate best results.

effort. For example, plant trays might be hanged using cables instead of sitting on poles. In this case, if those cables are smalls and cannot be reliably detected by the laser scanner, we can adjust to mount the laser scanner to directly detect the trays. Our system can continue to work normally without any further changes. The line detection will be easier since laser scanner detects more points from trays.

TABLE 2 Mean errors of distance from the center of the robot body to poles on both sides

	Distance to poles (m)					
	To left poles d_ℓ		To right poles d_r		$\lvert d_\ell - d_r \rvert$	
	Ours	Krajník et al. (2014a)	Ours	Krajník et al. (2014a)	Ours	Krajník et al. (2014a)
Row 1	0.257	0.272	0.226	0.215	**0.031**	0.057
Row 2	0.282	0.214	0.193	0.257	0.089	**0.043**
Row 3	0.235	0.321	0.251	0.147	**0.016**	0.174

Note: Bold numbers indicate best results.

Another practical consideration is how to determine the window size for pole detection. In our implementation, the window size is fixed and its value is set upon the applied standard polytunnel structure, that is, the distance between two consecutive poles in a row is approximately 3 m, the row width is 1.5 m. These values can be preset once with respect to the actual environment before letting the robot move. It might sound preferable to have an automatically adaptive window size, but in fact, we rarely see a polytunnel with different spacing between poles. For most cases, polytunnels are built in compliance with a standard, for which we argue that a corresponding fixed size window is adequate.

It is obvious that our system makes strong assumptions about the environment such as the distance between poles is constant, number of poles on each side of a row is equal. Our proposed system may fail to operate if those assumptions are not satisfied. However, we argue that those assumptions are reasonable, that is, it is uncommon to find a polytunnel with asymmetrical structure. Therefore our proposed system is useful for most cases.

5 | CONCLUSIONS

In this paper, a simple but efficient navigation solution of row following for an agricultural robot is presented. Our main goal is to develop an efficient but also cost effective system that can work reliably in polytunnels, which are the typical space constrained agricultural environment. We deliberately employ one 2D laser scanner. Using only this type of sensor, the robot is able to move between rows while keeping equidistant to both sides of a row. We claim this is important for several tasks, in which the robot must stay in the middle of a row, such as delivering UV light treatment, or autonomous transporting harvested products in and out of a polytunnel.

Experimental testing in both simulation and a mock-up polytunnel were performed to evaluate the quality of navigation. The results show a small drift of 1.4% over total traveled distance per row and the robot maintains the same distance to poles on both sides. We

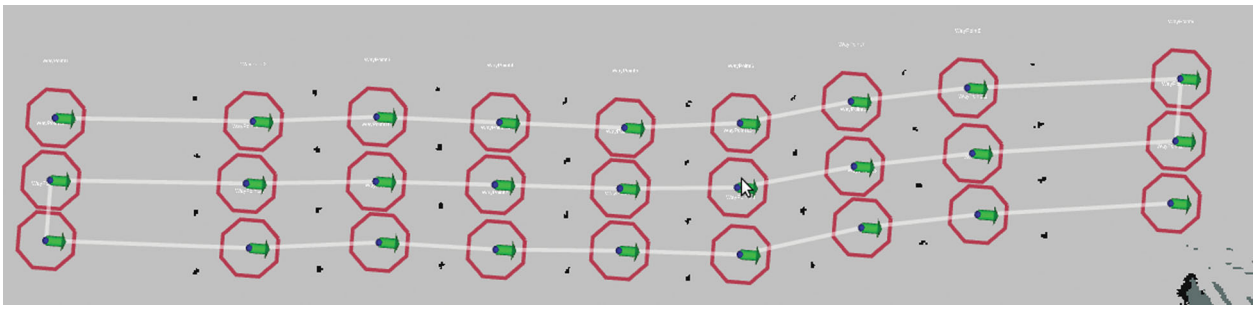

FIGURE 15 Topological map for navigation. This method proposed by Krajník et al. (2014a) relies on *move_base* for motion planning. All the topological nodes were manually created [Color figure can be viewed at wileyonlinelibrary.com]

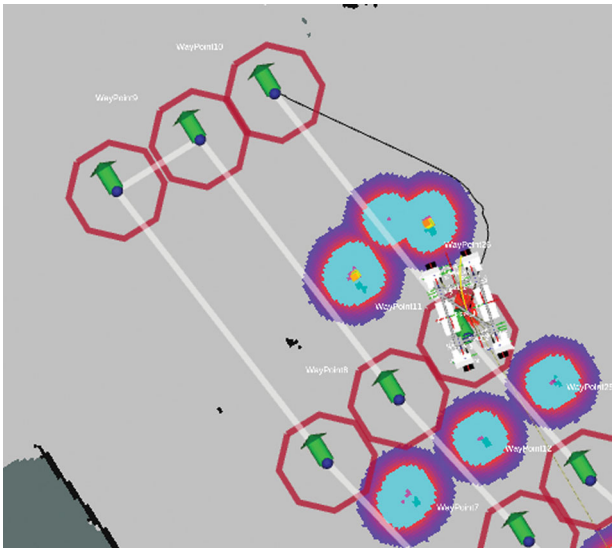

FIGURE 16 Topological navigation attempts a dangerous movement. The black line shows the planned trajectory. Notice the artificial obstacle between the poles was incorrectly perceived due to noisy laser scanner readings. It forces the robot to move around [Color figure can be viewed at wileyonlinelibrary.com]

compare our proposed row-following method with an existing one in Krajník et al. (2014a). We show that our method achieves better results.

We have discussed how our system can be easily adapted to different types of polytunnels. In rare cases, where the structure of a polytunnel is irregular, that is, distances between poles are different, number of poles on each side of a row are not equal, our system will fail. However, it is unusual to have a structure like that. Our proposed solution replaces the traditional and other way point based methods such as topological navigation (Krajník et al., 2014b) and thus simplifies the robot operation process. For future work, we aim to develop a full-scale SLAM-based navigation system. More navigation and safety sensors will also be used on the robot for the human aware navigation in the future that can cooperate along human laborers and also in respect to the safety standards in the polytunnels.

ACKNOWLEDGMENT

We would like to thank Øystein Sund from SAGA Robotics for designing UV light holder systems.

REFERENCES

Åstrand, B., & Baerveldt, A.-J. (2005). A vision based row-following system for agricultural field machinery. *Mechatronics*, *15*(2), 251–269.

Bakker, T., Wouters, H., VanAsselt, K., Bontsema, J., Tang, L., Müller, J., & vanStraten, G. (2008). A vision based row detection system for sugar beet. *Computers and Electronics in Agriculture*, *60*(1), 87–95.

Bergerman, M., Maeta, S. M., Zhang, J., Freitas, G. M., Hamner, B., Singh, S., & Kantor, G. (2015). Robot farmers: Autonomous orchard vehicles help tree fruit production. *IEEE Robotics and Automation Magazine*, *22*(1), 54–63.

Bergerman, M., Singh, S., & Hamner, B. (2012). Results with autonomous vehicles operating in specialty crops. *2012 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 1829–1835.

Biber, P., Weiss, U., Dorna, M., & Albert, A. (2012). Navigation system of the autonomous agricultural robot bonirob. *Workshop on Agricultural Robotics: Enabling Safe, Efficient, and Affordable Robots for Food Production (Collocated with IROS 2012)*, Vilamoura, Portugal.

Cariou, C., Lenain, R., Thuilot, B., & Berducat, M. (2009). Automatic guidance of a four-wheel-steering mobile robot for accurate field operations. *Journal of Field Robotics*, *26*(6-7), 504–518.

Fischler, M. A., & Bolles, R. C. (1987). Random sample consensus: A paradigm for model fitting with applications to image analysis and automated cartography. *Readings in Computer Vision*, 726–740.

Grimstad, L., & From, P. J. (2017a). Thorvald ii-a modular and re-configurable agricultural robot. *IFAC-PapersOnLine*, *50*(1), 4588–4593.

Grimstad, L., & From, P. J. (2017b). The Thorvald II agricultural robotic system. *Robotics*, *6*(4), 24.

Grimstad, L., Pham, C. D., Phan, H. T., & From, P. J. (2015, June). On the design of a low-cost, light-weight, and highly versatile agricultural robot. *In 2015 IEEE Workshop on Advanced Robotics and its Social Impacts (ARSO 2015)*, IEEE, pp. 1-6.

Grimstad, L., Skattum, K., Solberg, E., Loureiro, G. D. S. M., & From, P. J. (2017). Thorvald II configuration for wheat phenotyping. *In Proceedings of the IROS Workshop on Agri-Food Robotics: Learning from Industry* (Vol. 4).

Grimstad, L., Zakaria, R., Le, T. D., & From, P. J. (2018, October). A novel autonomous robot for greenhouse applications. *IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, IEEE, pp. 1-9.

Hiremath, S., van Evert, F., van der Heijden, G., ter Braak, C., & Stein, A. (2012, October). Image-based particle filter for robot navigation in a maize field. *In Proceedings of the Workshop on Agricultural Robotics (IROS 2012), Vilamoura, Portugal*, pp. 7-12.

Krajník, T., Fentanes, J. P., Mozos, O. M., Duckett, T., Ekekrantz, J., & Hanheide, M. (2014a, September). Long-term topological localisation for service robots in dynamic environments using spectral maps. *In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 4537–4542.

Krajník, T., Fentanes, J. P., Mozos, O. M., Duckett, T., Ekekrantz, J., & Hanheide, M. (2014b, September). Long-term topological localization for service robots in dynamic environments using spectral maps. *In 2014 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 4537-4542.

Lázaro, M. T., Grisetti, G., Iocchi, L., Fentanes, J. P., & Hanheide, M. (2017, November). A lightweight navigation system for mobile robots. *In Iberian Robotics conference*, Springer, Cham, pp. 295-306.

Moorehead, S. J., Wellington, C. K., Gilmore, B. J., & Vallespi, C. (2012, October). Automating orchards: A system of autonomous tractors for orchard maintenance. IEEE/RSJ International Conference on Intelligent Robots and Systems Workshop on Agricultural Robotics, Vilamoura, Portugal.

Reid, J. F. (2011). The impact of mechanization on agriculture. *Bridge*, *41*(3), 22–29.

Coulter, R. Craig (1992). Implementation of the Pure Pursuit Path Tracking Algorithm. *Technical Report*, *The Robotics Institute*, *Carnegie Mellon University*, *1*, 1–15.

Riggio, G., Fantuzzi, C., & Secchi, C. (2018, May). A low-cost navigation strategy for yield estimation in vineyards. *In 2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 2200–2205.

Stoll, A., & Kutzbach, H. D. (2000). Guidance of a forage harvester with gps. *Precision Agriculture*, *2*(3), 281–291.

Subramanian, V., Burks, T. F., & Arroyo, A. (2006). Development of machine vision and laser radar based autonomous vehicle guidance systems for citrus grove navigation. *Computers and Electronics in Agriculture*, *53*(2), 130–143.

Tarsha-Kurdi, F., Landes, T., & Grussenmeyer, P. (2007, September). Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. *In ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, 36, pp. 407–412.

Thrun, S. (2002). Probabilistic robotics. *Communications of the ACM*, *45*(3), 52–57.

Ting, K., Abdelzaher, T., Alleyne, A., & Rodriguez, L. (2011). Information technology and agriculture global challenges and opportunities. *Bridge*, *41*(3), 6–13.

Xiong, Y., From, P. J., & Isler, V. (2018, May). *Design and evaluation of a novel cable-driven gripper with perception capabilities for strawberry picking robots. In 2018 IEEE International Conference on Robotics and Automation (ICRA)*, IEEE, pp. 7384-7391.

York, D. (1966). Least-squares fitting of a straight line. *Canadian Journal of Physics*, *44*(5), 1079–1086.

Zhang, J., Chambers, A., Maeta, S., Bergerman, M., & Singh, S. (2013, November). 3d perception for accurate row following: Methodology and results. *In 2013 IEEE/RSJ International Conference on Intelligent Robots and Systems*, IEEE, pp. 5306–5313.

How to cite this article: Le TD, Ponnambalam VR, Gjevestad JGO, From PJ. A low-cost and efficient autonomous row-following robot for food production in polytunnels. *J. Field Robotics*. 2019;1–13. https://doi.org/10.1002/rob.21878