Hindawi
Mathematical Problems in Engineering
Volume 2021, Article ID 6221119, 14 pages
https://doi.org/10.1155/2021/6221119

Research Article

An End-to-End Learning-Based Row-Following System for an Agricultural Robot in Structured Apple Orchards

Peichen Huang,[1] **Lixue Zhu**,[2] **Zhigang Zhang**,[3] **and Chenyu Yang**[2]

[1]*College of Automation, Zhongkai University of Agriculture and Engineering, Guangzhou 510225, China*
[2]*College of Electro-mechanical Engineering, Zhongkai University of Agriculture and Engineering, Guangzhou 510225, China*
[3]*Key Laboratory of Key Technology on Agricultural Machine and Equipment, Ministry of Education, South China Agricultural University, Guangzhou 510642, China*

Correspondence should be addressed to Lixue Zhu; zhulixue@zhku.edu.cn

Received 7 May 2021; Accepted 31 August 2021; Published 13 September 2021

Academic Editor: Mohammad Yaghoub Abdollahzadeh Jamalabadi

A row-following system based on end-to-end learning for an agricultural robot in an apple orchard was developed in this study. Instead of dividing the navigation into multiple traditional subtasks, the designed end-to-end learning method maps images from the camera directly to driving commands, which reduces the complexity of the navigation system. A sample collection method for network training was also proposed, by which the robot could automatically drive and collect data without an operator or remote control. No hand labeling of training samples is required. To improve the network generalization, methods such as batch normalization, dropout, data augmentation, and 10-fold cross-validation were adopted. In addition, internal representations of the network were analyzed, and row-following tests were carried out. Test results showed that the visual navigation system based on end-to-end learning could guide the robot by adjusting its posture according to different scenarios and successfully passing through the tree rows.

1. Introduction

With the development of technology, many intensive tasks worked by the human in orchards can be replaced by using agricultural robots [1–8]. However, developing such robots in a real orchard, safely and reliably, is still a challenge. One major challenge for an autonomous agricultural robot in an orchard is row following. Recently, vision sensors have been widely used in agricultural robot navigation since their low cost, high efficiency, and capability to provide huge information [9–17]. In our previous study, a row-following system based on traditional machine vision for an apple orchard was designed, of which navigation was divided into multiple subtasks, such as image binarization, boundaries detection, guidance path generation, coordinate transformation, and low-level motor control [18]. Each subtask was processed independently, and their outputs were integrated as the final control decision. Based on our previous test, traditional development is easy to adjust, optimize, and troubleshoot each module. However, the system's complexity increased at the same time when more and more modules were added to improve the navigation. In addition, it was found that once the environment changed, such as the light intensity and occurrence of deep shadows, the traditional navigation system had to be readjusted to pick out the important features consistently. By contrast, deep learning has the potential to learn for performing many of the complex perception tasks of mobile robot navigation [19, 20]. This study attempted to develop a deep learning network that was directly mapped from pixels to actuation based on the end-to-end learning scheme. As Figure 1 shows, traditional subtasks of the navigation were replaced by a specially designed deep network, which reduces manual programming and simplifies the system.

The ALVINN (autonomous land vehicle in a neural network) study was the first time to prove that end-to-end learning was feasible for unmanned driving. However, limited by the computing power of the time, the ALVINN

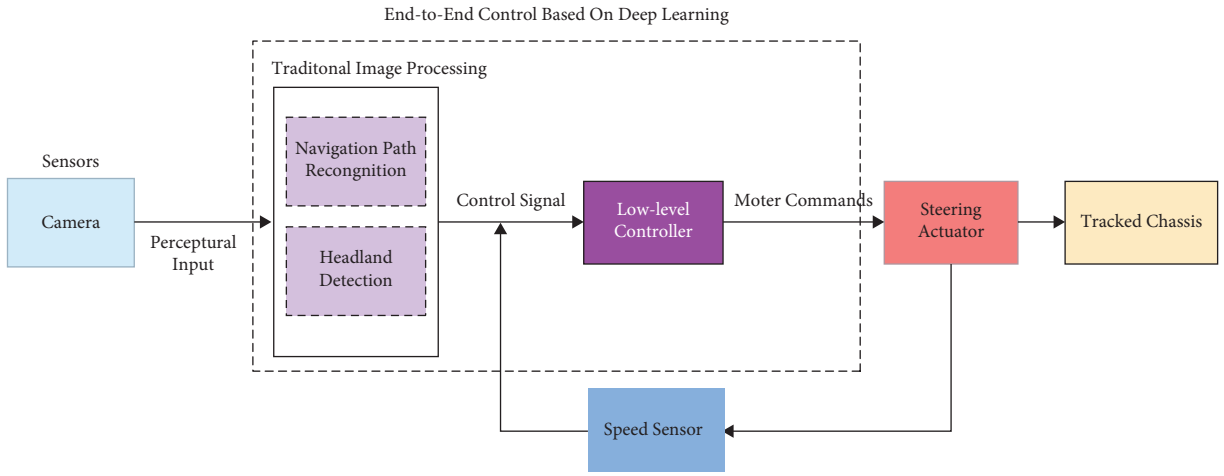


Figure 1: Schematic diagram of vision-based end-to-end learning navigation robot platform.

system used a single hidden layer backpropagation network and a small number of samples for learning, which prevented it from being applied in more complex environments [21]. Recently, with the development of deep learning-based hardware and theories, many breakthroughs have been made in autonomous driving based on end-to-end learning. NVIDIA developed a self-driving car system based on the DAVE-2 network. The network was trained via human driving; samples were captured simultaneously by using three different camera angles. Test results showed that, with less than a hundred hours of training, the car could execute autonomous driving on highways, local roads, and residential neighborhoods in sunny, cloudy, and rainy conditions [22, 23]. Vastly different from the NVIDIA self-driving system, Mobileye divided the self-driving system into three parts: perception, high-precision map, and driving decision; each part was designed on an independent network. Supervised learning techniques and direct optimization were implemented in the recurrent neural network to solve the long-term planning problem. Results showed that, by incorporating adversarial elements into the environment, robust policies could be learned by the designed self-drive system [24]. Large and diverse datasets of driving information, such as steering, braking, and speed, were also considered in some studies and implemented in the end-to-end learning component to achieve the optimal driving strategy. Such a self-drive system was developed by Comma.ai, which expected both speed and steering direction under different driving conditions that could be calculated intelligently [25].

Most of the current research on self-driving (autonomous) systems using deep learning mainly focuses on land-to-road navigation [26–33], while outdoor robot navigation studies remain relatively low. Muller et al. developed a visual-based obstacle avoidance system using deep networks for an outdoor robot. During training, the robot was executed to drive under different terrains, obstacles, and lighting conditions by remote control. Test results showed that the robot exhibited an excellent ability to detect obstacles and navigate around them at speeds of 2 m/s [34]. Hwu et al. applied the IBM NS1e board, which contained the IBM Neurosynaptic System (IBM TrueNorth chip), on the Robotics platform to speed up the CNN (convolutional neural network) training process. Results showed that the designed self-drive system based on CNN enabled the robot to drive along a mountain path with low power processing [35]. Orchard environments are much more complicated than land road environments. Orchard environments have uneven ground surfaces, diversification of trees, and so on, which makes developing an autonomous navigation system based on deep learning challenging. However, studies in this field have not adequately addressed the issue. Bell et al. developed a monocular vision-based row-following system for pergola structured orchards. A fully convolutional network was used to perform semantic segmentation of color images for an abstract class called "traversable space." Test results indicated that the designed self-drive system executed row following better than the existing 3D lidar navigation system. However, the designed system still needs the traditional subtasks such as boundary detection and centerline fitting to generate the final steering decision [36].

The objective of this study is mainly focused on row following and the detection of row ends. A CNN network for the row-following system was developed, which consisted of five convolutional layers and one fully connected layer. Experiments were carried out and results were analyzed. The main novelties of this study are as follows:

(1) A tree row-following system that was directly mapped from pixels to actuation based on the end-to-end learning scheme was developed, which saves much hand programming and simplifies the system compared to traditional methods. In addition, the deep learning-based system has the potential to improve problems such as the fluctuation of light intensity and shadows in complex environments.

(2) A sample collection method for network training was also proposed, by which the robot could automatically drive and collect data without an operator or

remote control. No hand labeling of training samples is required.

(3) Methods such as batch normalization, dropout, data augmentation, and 10-fold cross-validation were adopted to improve the network's generalization ability and visualization analysis has been executed to clearly understand the useful features learned by the network.

The remainder of this paper is divided as follows. Section 2 contains an overview of the vision-based navigation system, the design of the training data collection method, and the CNN network architecture. Section 3 details the results of the simulation and row-following test and discussion. Finally, Section 4 states the conclusion of this study.

2. Materials and Methods

2.1. System Overview. A crawler-type robot platform was used in this research. To keep the navigation system simple and relatively low-cost, a monocular camera (Imaging Source DFK 21AU04) with a frame rate of 30 Hz and an image resolution of 640×480 pixels was used for image acquisition. An industrial computer was used to execute high-level algorithms and a microcontroller was used for low-level control operations. A dual antenna GNSS (Global Navigation Satellite System) and a Trimble BD982 receiver were applied for measuring driving trajectories. Figure 1 demonstrates the schematic diagram of the tree row-following platform.

2.2. Sample Collection. Given a series of N images $x_1, x_2, \ldots, x_N$ sampled at time instances 1, 2,. . ., N, respectively, the goal of the designed deep network is to classify the images as performing one of the discrete commands {move forward, turn left, turn right, stop}. The commands are defined by the following rules: (1) robot moves forward if both sides of tree rows could be perceived in the camera field of view; (2) robot turns left or right if and only if one side of a row could be perceived; and (3) robot stops driving and gets ready for headland turning when the last row is detected.

For navigation based on deep learning, collecting a large number of training samples is a great challenge because a mobile robot sometimes needs to be steered or controlled by an operator for several days or even weeks. To reduce the workload, a collection method was developed based on robot path tracking using GNSS in this study. During the initial stage, to focus more on the performance of the designed method and reduce losses if the robot runs over a barrier of trees, an artificial apple orchard environment was set up. A reference path was defined as the center of the tree rows. A path tracking controller was designed, and the details were presented in [37]. The sample collection process was carried out as follows: First, the robot executed a straight-line path tracking at a speed of 0.3 m/s. Image sequences were then saved and categorized as "move forward." Second, the yaw angle of the camera was adjusted until only the left row could be seen in the camera field of view. Straight-line path tracking was executed again, and image sequences were categorized as "turn right." "Turn left" commands were obtained using a similar process. Finally, images of "move forward," "turn right," and "turn left" actions, which showed the last row, were extracted and categorized as "stop." Some samples are shown in Figure 2.

2.3. Network Configuration and Training

2.3.1. Network Architecture. The designed CNN consists of five convolution layers and one fully connected layer. To speed up the training and real-time navigation, an input image is first resized to 48×36. The output of the network is expected steering commands. As shown in Figure 3, the network mainly consists of convolution layers, activation functions, pooling layers, a fully connected layer, and a SoftMax layer. Five convolutional layers are designed to perform feature extraction and each kernel of the convolutional layer corresponds to a feature map, which will be further compressed and extract key features using pooling layers. Finally, all the features are connected using the fully connected layer and results are output to the classifier for steering command prediction.

2.3.2. Network Training. Optimizing a deep neural network is not trivial due to the gradient vanishing/exploding problem. In addition, optimization may get stuck in a saddle point, resulting in premature termination and inferior low-level features [38]. To improve the training, the following methods were adopted in this study:

(1) Batch normalization was applied right after each convolution layer, which forces the network's activation to generate larger variances across different training samples, accelerating the optimization in the training phase and achieving a better classification performance [39]. Formally, denoting by $\mathbf{x} \in \mathscr{B}$ an input to batch normalization that is from a minibatch $\mathscr{B}$, batch normalization transforms $\mathbf{x}$ according to the following expression:

$$\mathrm{BN}(\mathbf{x}) = \gamma \odot \frac{\mathbf{x} - \widehat{\boldsymbol{\mu}}_{\mathscr{B}}}{\widehat{\boldsymbol{\sigma}}_{\mathscr{B}}} + \beta, \tag{1}$$

where $\widehat{\mu}_{\mathscr{B}}$ is the sample mean and $\widehat{\sigma}_{\mathscr{B}}$ is the sample standard deviation of the minibatch $\mathscr{B}$. The notation $\odot$ represents Hadamard (elementwise) product. γ and β are the scale parameter and shift parameter, respectively; they need to be learned jointly with the other model parameters. $\widehat{\mu}_{\mathscr{B}}$ and $\widehat{\sigma}_{\mathscr{B}}$ in equation (1) can be calculated as follows:

$$\begin{aligned} \widehat{\boldsymbol{\mu}}_{\mathscr{B}} &= \frac{1}{|\mathscr{B}|} \sum_{\mathbf{x} \in \mathscr{B}} \mathbf{x}, \\ \widehat{\sigma}^2_{\mathscr{B}} &= \frac{1}{|\mathscr{B}|} \sum_{\mathbf{x} \in \mathscr{B}} \left(\mathbf{x} - \widehat{\boldsymbol{\mu}}_{\mathscr{B}}\right)^2 + \varepsilon, \end{aligned} \tag{2}$$

where ε, $\varepsilon > 0$, is a small constant which is added to the variance estimate to avoid dividing by zero, even

Figure 2: Training samples. (a) Turn left. (b) Turn right. (c) Move forward. (d) Stop.

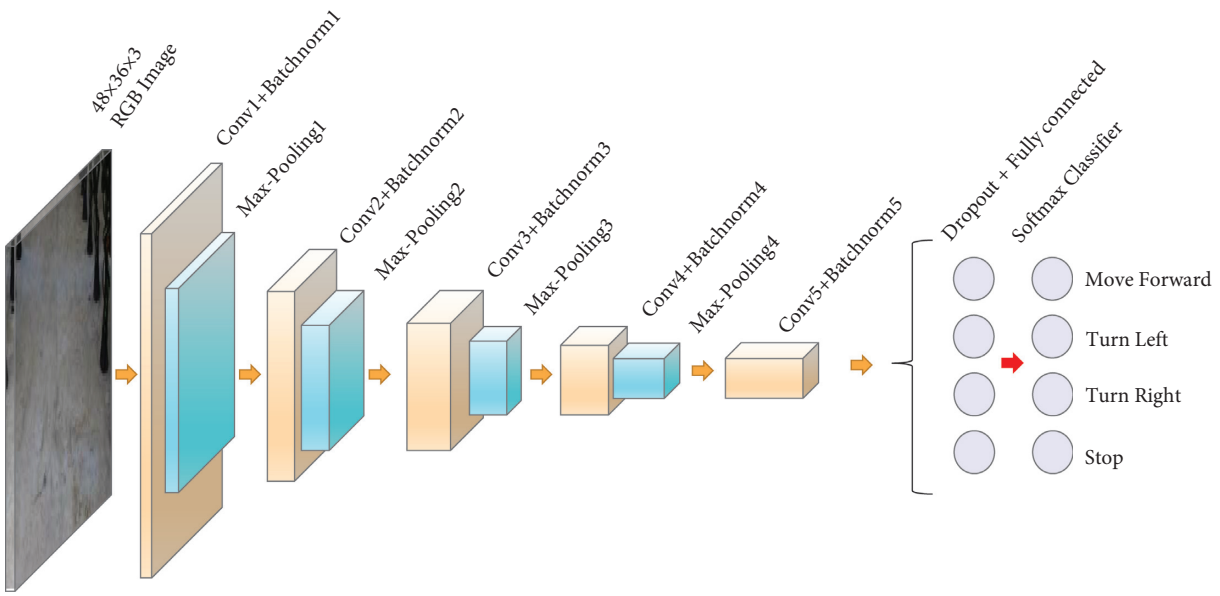


Figure 3: Network architecture.

in cases where the empirical variance estimate might vanish.

(2) Dropout is also adopted during the training to prevent network overfitting [40]. With dropout probability p, each intermediate activation h is replaced by a random variable h' as follows:

$$h' = \begin{cases} 0, & \text{with probability } p, \\ \dfrac{h}{1-p}, & \text{otherwise.} \end{cases} \tag{3}$$

(3) To enrich the training samples, data augmentation was executed. The details, described as "move forward" and "stop," are horizontally flipped. All samples are randomly shifted horizontally or vertically within the range $[-5\ \ 5]$ pixels and randomly rotated counterclockwise or clockwise within the range $[-10\ \ 10]$ degrees. The sample brightness (of each sample) is randomly adjusted to simulate lighting change at different times of the day. Some of the samples after augmentation are displayed in Figure 4.

(4) To minimize the sampling bias when using machine learning models, 10-fold cross-validation is used. The original training data are split into ten nonoverlapping subsets. Then model training and validation are executed ten times, each time training on nine subsets and validating on a different subset (the one not used for training in that round). Finally, the training and validation errors are estimated by averaging the results.

2.3.3. Training Results. A total of 21280 samples were classified into four categories for training. To test the effectiveness in preventing network overfitting when using batch normalization (short as BN) and dropout (short as DP), different types of network structures were executed and compared, e.g., BN plus DP, only BN, and only DP. Figure 5 shows the results of the comparison during network training. The performance was poor when using only DP, with an accuracy of only 37.04%. By contrast, when using BN plus DP or only BN, the accuracy could reach 96.31% and 96.33% on average, respectively, which was much higher than using only DP. Figure 6 further demonstrates the comparison of accuracy and loss rate. By using BN plus DP or only BN, at the early 30th iteration, accuracy had already achieved 97% and the loss rate had dropped down to 0.07. On the contrary, when using only DP, training had to be executed for 210 iterations to achieve the same performance. Overall, BN plus DP and only BN had a similar performance among the test networks; the two methods also had better results than using DP alone.

For the final training run, BN plus DP was chosen to construct the final network since BN could improve the accuracy and speed up the training; DP reduced the number of features in intermediate layers, which could significantly reduce the network's dependence on certain features. 80% of the total samples were randomly selected in each category (17024 images) as the final training set and the remaining samples (4256 images) were selected as the test set. During the final training, the loss function error was reduced to below 0.0055 after executing 330 iterations.

3. Results and Discussion

3.1. Simulation. A video simulation was carried out to evaluate the performance of the network before applying it to the actual test. The robot drove through the tree row for video recording by remote control; a total of 7 videos were recorded. During simulation, the network with BN plus DP and the network using only BN had similar performance, which was consistent with the training results found earlier in the study. The error rate of the network with BN plus DP was 3.8%, while the network with only BN was 4% on average.

To further understand the features learned by the network, visualization was performed. Figure 7 displays activation of the first convolutional layer with eight convolution kernels under different image inputs.

The first and most important task of machine vision-based navigation is perception, which means recognition of tree rows in our case. As Figure 7 shows, white pixels represent strong positive activation, while black pixels represent strong negative activation. A channel that is mostly grey does not activate as strongly on the input image. After training, the regions of the tree row were strongly activated. At the same time, the contours and texture of the tree row were clear, which suggested that the designed CNN network had correctly learned the key features of tree rows. Deep networks would be less powerful and would not be able to learn the complex patterns from the data without activation functions. Traditionally, two widely used nonlinear activation functions are the sigmoid and hyperbolic tangent. A general problem with both the sigmoid and tanh functions is the possibility of saturation. Once saturated, it becomes challenging for the learning algorithm to continue to adapt the weights to improve the performance of the model. Therefore, the ReLU (rectified linear unit) function was adopted, which not only solves the saturation problem but also has the following advantages: computational simplicity, representational sparsity, and linear behavior. The activation of the ReLU function in the first layer is demonstrated in Figure 8.

3.2. Tree Row-Following Test. Two types of row following were carried out. Straight-line row following was first executed, where the length of the row was set to 20 m. To further test the designed network generalization, a curved path section was added between two straight-line row sections. The height and tilt angle of the camera were set to 1.65 m and 15°, respectively. The speed of the robot was set to 0.5 m/s and deviation was measured using GNSS. The experiments were conducted at noon on a sunny day with an occasional breeze. During the experimental test, the robot moved on flat land and surface soils that are naturally hard and dense. To

Figure 4: Samples generation using image augmentation.

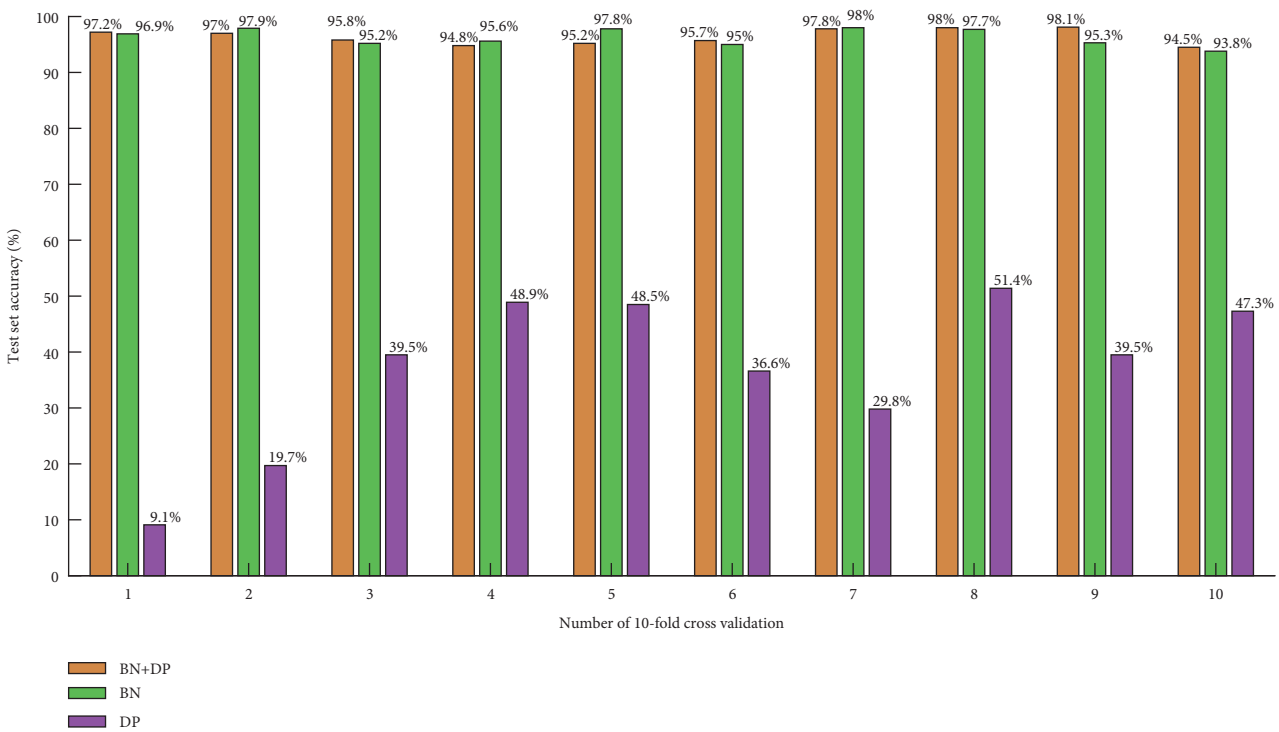
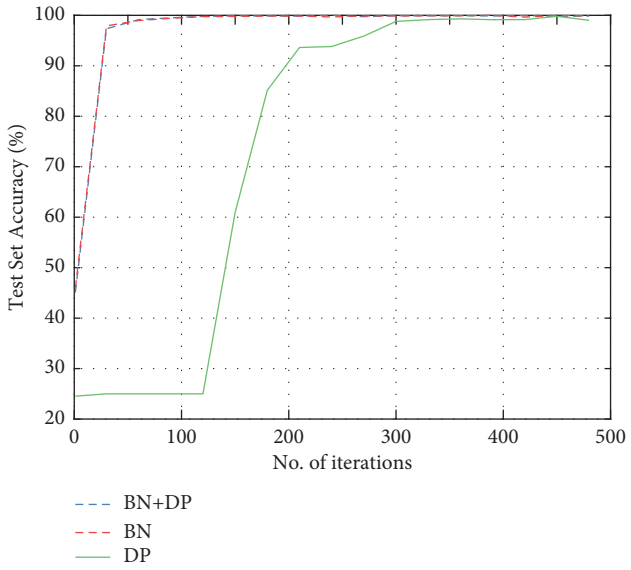


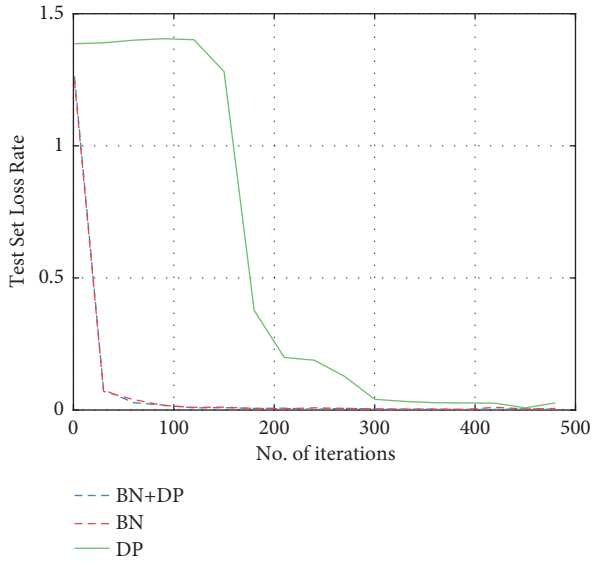
Figure 5: Comparison of test set validation.

reduce the sliding effects, a tracked mobile robot has been adopted in this study since it has better maneuverability in rough terrain and higher friction in turns due to its tracks and multiple points of contact with the surface. The robot platform and the test environment are shown in Figure 9.

During the test, the robot could automatically move forward when both sides of tree rows appeared in the captured image; if only one side of the row appeared, the robot would execute the corresponding turning; once the last row was detected, the robot would stop and get ready for



(a)



(b)

Figure 6: Continued.

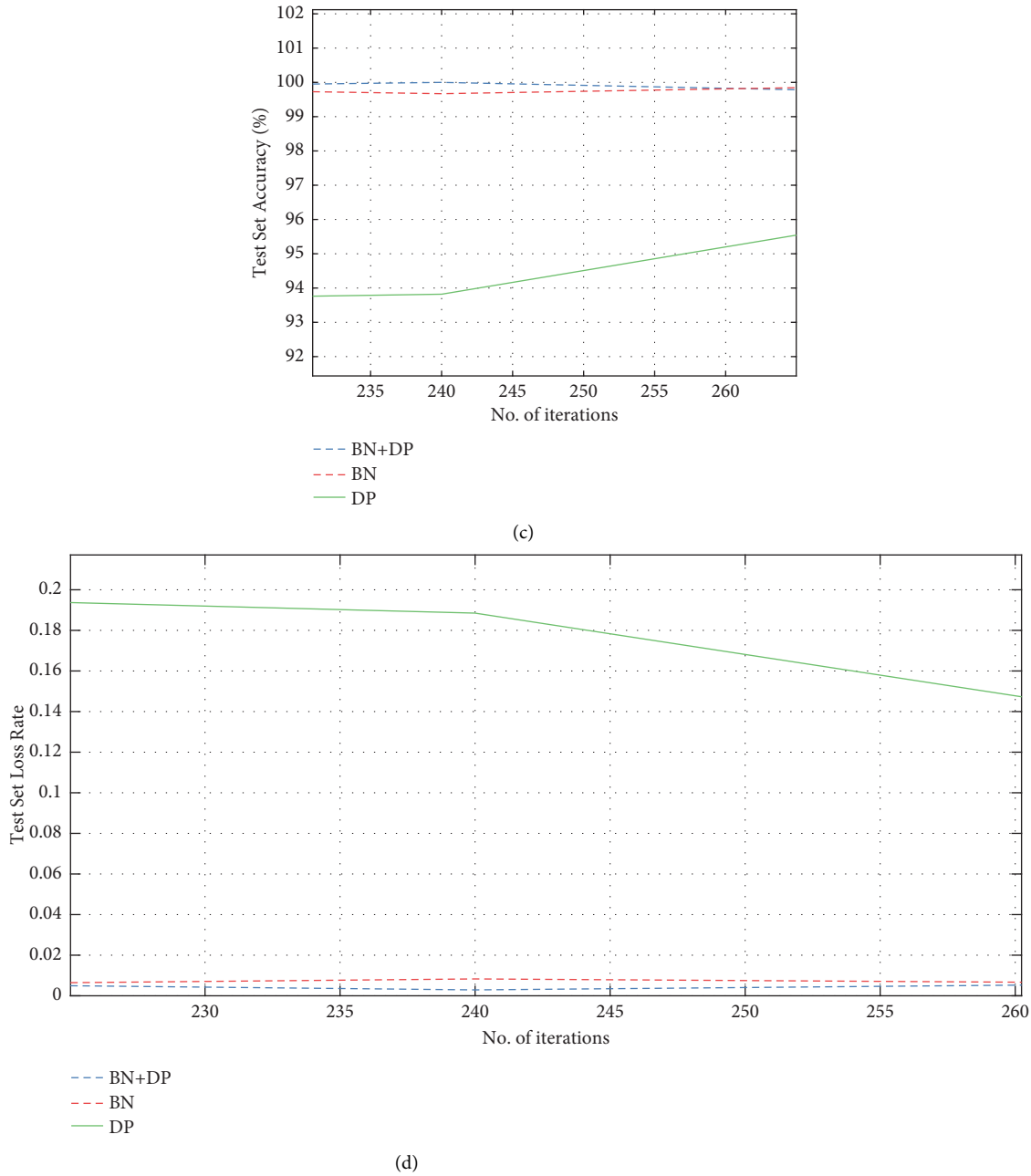
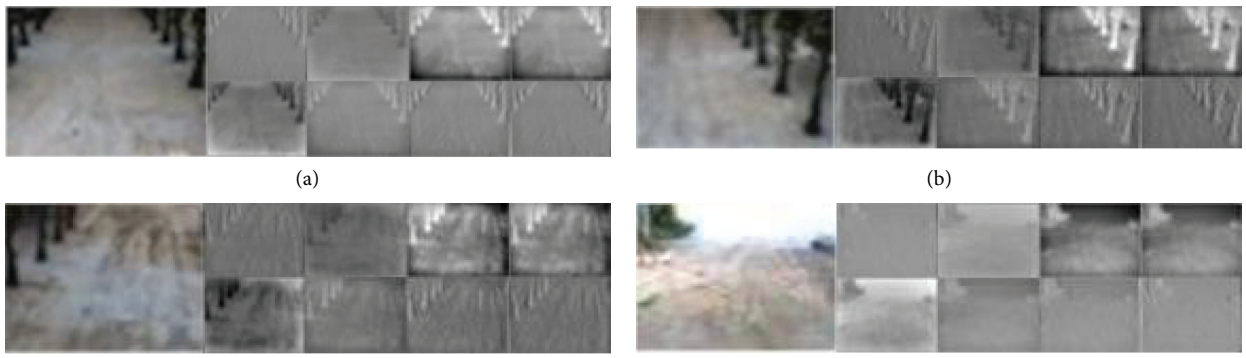


Figure 6: Comparison of test set accuracy and loss rate. (a) Test set accuracy. (b) Test set loss rate. (c) A partial enlarged picture of (a). (d) A partially enlarged picture of (b).

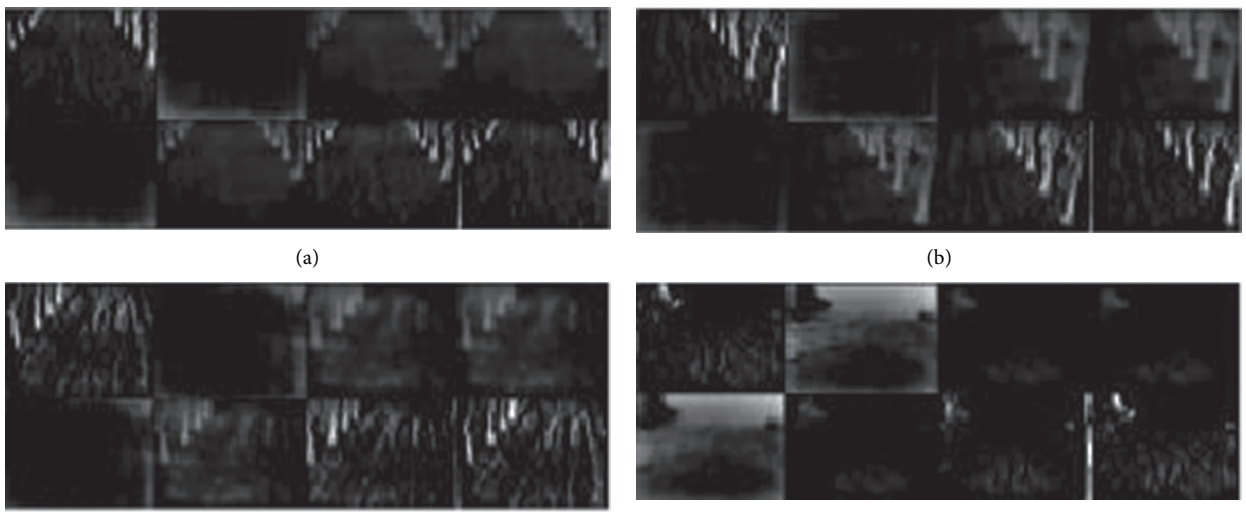
the headland turning. Traditional navigation had also been conducted and compared with the end-to-end method. During the test, images were collected for retraining the end-to-end network after each run. Figure 10 shows a comparison of the straight-line row-following performance and Figure 11 shows the row-following trajectories.

As shown in Figure 10, the lateral errors were 0.14 m and heading errors were 0.8° on average for traditional navigation, while the lateral errors were 0.29 m and heading errors were 1.8° on average for end-to-end navigation. In general, traditional navigation had better performance. However, when the robot drove along trees with large gaps or when trees happened to swing due to wind, traditional navigation became unstable since tree rows could not be fully extracted or were overextracted due to the fluctuation of light intensity. By contrast, the end-to-end row-following system has greatly improved these issues since different light intensities had been simulated using image augmentation during the network training process. Both lateral and heading errors of navigation based on the end-to-end method kept decreasing after each run, which means that performance improved after each network's retraining. Figure 12 demonstrates the poor performance of the end-to-end navigation system. Once a significant difference between



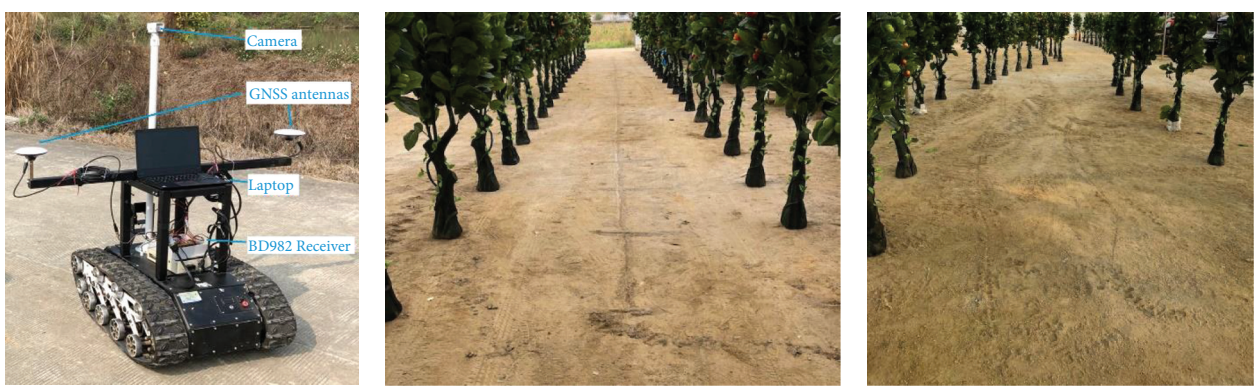
(a) (b) (c) (d)

Figure 7: Visualization of activation of CNN's first convolution layer. (a) Move forward. (b) Turn left. (c) Turn right. (d) Stop.



(a) (b) (c) (d)

Figure 8: Visualization of activation of CNN's first "ReLU" layer. (a) Move forward. (b) Turn left. (c) Turn right. (d) Stop.



(a) (b) (c)

Figure 9: Tree-row-following test environment. (a) Robot platform. (b) Straight path. (c) Curve path.

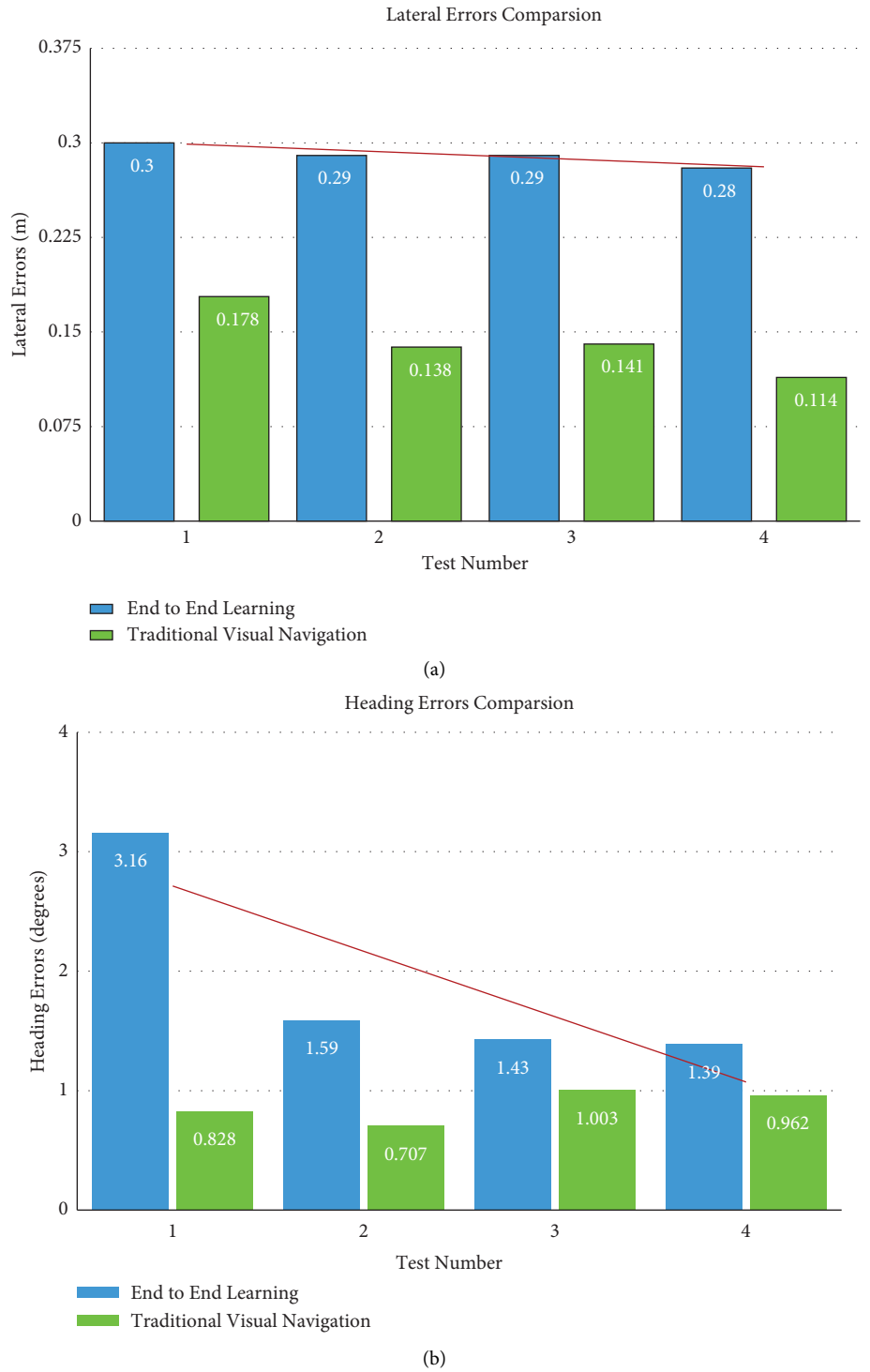


Figure 10: Performance comparison. (a) Lateral errors. (b) Heading errors.

the training sample and the real-time captured image was found, the accuracy of the prediction would decrease. In future work, this problem can be improved by appropriately adding more steering types such as a large turn or a series of small turns. Increasing the amount of training data would also likely increase the accuracy of prediction. Moreover, since the ideal steering control was a continuous control problem, using a regression model to describe the robot movement would be more accurate. Applying other methods such as regularization, early stopping, and network pretraining would improve the network's generalization and should be tested in the future. Overall, even with a certain initial deviation, the robot could still return to the center of the row after a while.

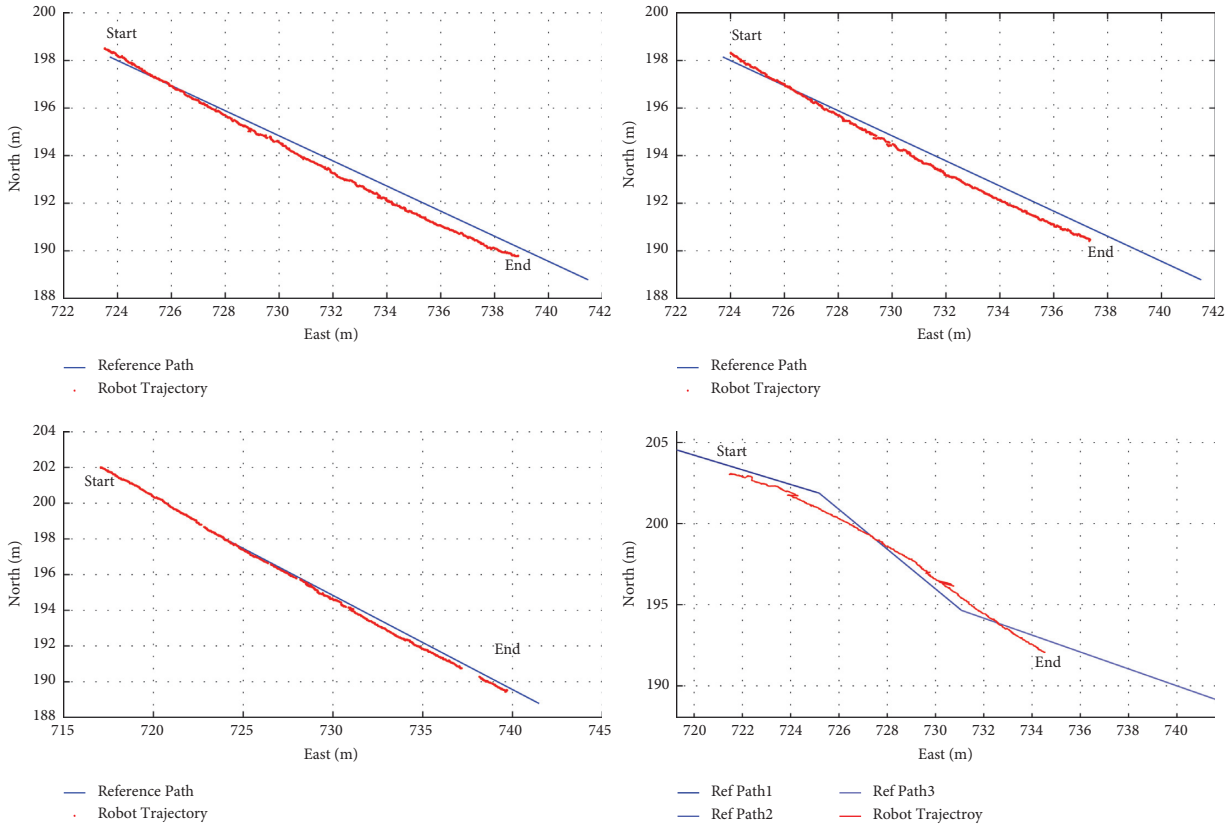
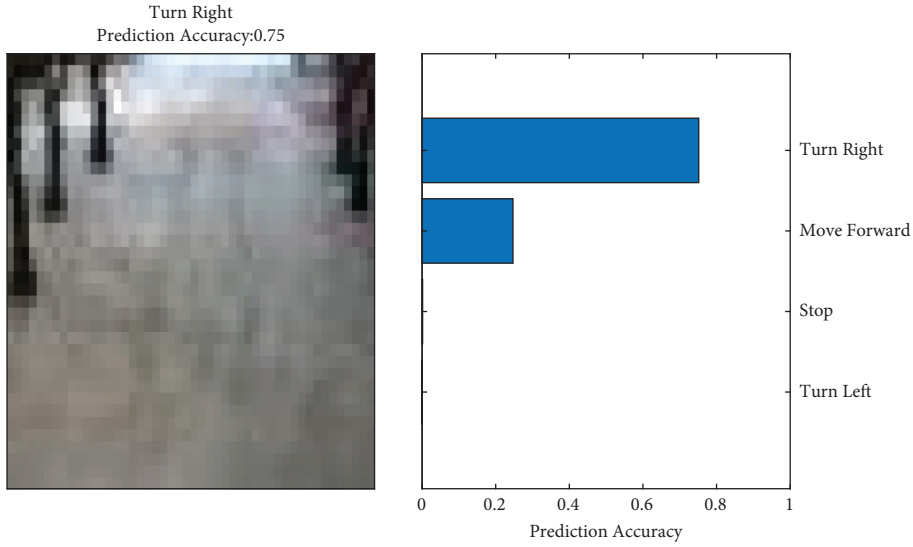


Figure 11: Robot actual trajectory during the test.



(a)

Figure 12: Continued.

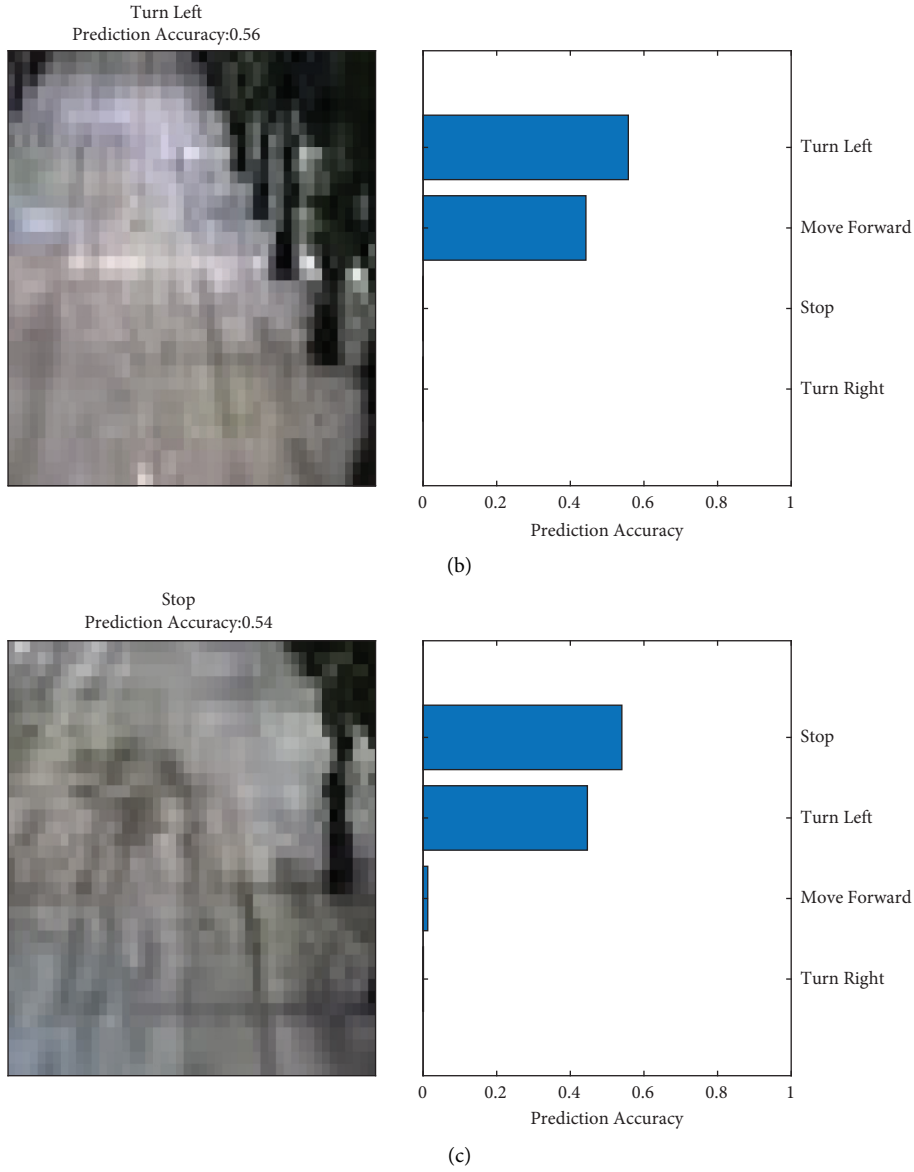


Figure 12: Situations with low prediction accuracy. (a) Case 1. (b) Case 2. (c) Case 3.

4. Conclusions

A visual tree row-following system based on end-to-end learning for an agricultural robot in an apple orchard environment was developed in this paper. The input image was directly mapped to steering commands by the designed CNN. A data collection method without human driving or remote control was also proposed. The CNN network consisted of five convolutional layers and one fully connected layer. To improve the network generalization ability, techniques such as batch normalization, dropout, data augmentation, and 10-fold cross-validation were adopted in the study. Two types of row-following tests were carried out. Test results showed that the robot could adjust its posture according to different situations and drive through the tree row.

A tree row-following system was carried out using a simple landscape with an obvious color contrast and shape structure as a preliminary test. With implementers installed on the mobile robot, this research could be expanded to different agricultural tasks such as planting, spraying, fertilizing, cultivating, harvesting, thinning, weeding, and inspection. In future work, more realistic elements of the orchard navigation will be added. (1) For example, leaves and weeds in a real apple orchard appear as noise in the captured images; this affects the accuracy of visual navigation. In this study, an input image was resized to ensure low resolution, after which noise was reduced while the main regions of tree rows were maintained. In a future study, additional complications will be introduced; e.g., samples under different weather conditions, different tree trunk sizes and colors, canopy shadows, and tree trunk with branches

may be collected and added to the training process to enhance the generalization of the designed model. Moreover, image preprocessing such as noise reduction should also be considered and developed to increase the robustness of the navigation system. (2) Keeping the camera steady during data collection is a great challenge for visual navigation. To simulate this effect, samples were randomly shifted and rotated to simulate camera vibration during training. In future studies, camera position, such as yaw, pitch, and roll angles measured by the IMU (Inertial Measurement Unit), could be used to calibrate the captured images and reduce the camera vibration further. (3) A tracked mobile robot was adopted in this study since it has better maneuverability in rough terrain and higher friction in turns due to its tracks and multiple points of contact with the surface. Furthermore, the sliding effects can be incorporated into an extended kinematics model to make the robot adapt to different terrain conditions. (4) In a real orchard, some apple trees or even an entire tree row may be missed due to the planting plan. This situation should also be considered when executing the sample collection process in future research. (5) Adopting high-performance GPUs and using an embedded development platform such as NVIDIA Jetson TX2, which can realize image and video information processing efficiently, will also be considered. With further development, the system can be combined with longer-term strategies, such as headland turning planning and control.

Data Availability

The data used to support the findings of this study are included within the article.

Conflicts of Interest

The authors declare that there are no conflicts of interest regarding the publication of this study.

Acknowledgments

This work was funded by the Science and Technology R&D Projects in Key Fields of Guangdong Province (2019B020223003), Guangdong Agricultural Technology Research and Development Project (2018LM2167), and Guangdong Province Modern Agricultural Industrial Technology System Innovation Team Project (Guangdong Agricultural Letter (2019) no. 1019). The authors are grateful to the student Mo Dongyan, Zhao Yunxia, and Lin Guchen for their support during the experiments.

References

[1] J. Li, Y. Tang, X. Zou, G. Lin, and H. Wang, "Detection of fruit-bearing branches and localization of litchi clusters for vision-based harvesting robots," *IEEE Access*, vol. 8, pp. 117746–117758, 2020.

[2] H. Kang, H. Zhou, X. Wang, and C. Chen, "Real-time fruit recognition and grasping estimation for robotic apple harvesting," *Sensors*, vol. 20, no. 19, p. 5670, 2020.

[3] W. Zhang, L. Gong, S. Chen, W. Wang, Z. Miao, and C. Liu, "Autonomous identification and positioning of trucks during collaborative forage harvesting," *Sensors*, vol. 21, no. 4, p. 1166, 2021.

[4] J. Chen, Q. Hu, and J. Wu, "Navigation path extraction for greenhouse cucumber-picking robots using the prediction-point Hough transform," *Computers and Electronics in Agriculture*, vol. 180, Article ID 105911, 2021.

[5] L. Gong, X. Du, and K. Zhu, "Pixel level segmentation of early-stage in-bag rice root for its architecture analysis," *Computers and Electronics in Agriculture*, vol. 186, Article ID 106197, 2021.

[6] Y. Majeed, M. Karkee, and Q. Zhang, "Development and performance evaluation of a machine vision system and an integrated prototype for automated green shoot thinning in vineyards," *Journal of Field Robotics*, vol. 38, 2021.

[7] Z. Song, Z. Zhou, and W. Wang, "Canopy segmentation and wire reconstruction for kiwifruit robotic harvesting," *Computers and Electronics in Agriculture*, vol. 181, Article ID 105933, 2021.

[8] R. Urban, M. Štroner, and I. Kuric, "The use of onboard UAV GNSS navigation data for area and volume calculation," *Acta Montanistica Slovaca*, vol. 25, pp. 361–374, 2020.

[9] C. Wang, Y. Tang, X. Zou, L. Luo, and X. Chen, "Recognition and matching of clustered mature litchi fruits using binocular charge-coupled device (CCD) color cameras," *Sensors*, vol. 17, no. 11, p. 2564, 2017.

[10] M. Chen, Y. Tang, X. Zou, K. Huang, L. Li, and Y. He, "High-accuracy multi-camera reconstruction enhanced by adaptive point cloud correction algorithm," *Optics and Lasers in Engineering*, vol. 122, pp. 170–183, 2019.

[11] G. Lin, Y. Tang, X. Zou, J. Cheng, and J. Xiong, "Fruit detection in natural environment using partial shape matching and probabilistic hough transform," *Precision Agriculture*, vol. 21, no. 1, pp. 160–177, 2020.

[12] H. Wang, L. Dong, and H. Zhou, "YOLOv3-Litchi detection method of densely distributed litchi in large vision scenes," *Mathematical Problems in Engineering*, vol. 2021, Article ID 8883015, 11 pages, 2021.

[13] G. Lin, Y. Tang, and X. Zou, "Three-dimensional reconstruction of guava fruits and branches using instance segmentation and geometry analysis," *Computers and Electronics in Agriculture*, vol. 184, Article ID 106107, 2021.

[14] Z. Yang, L. Gong, and C. Liu, "Efficient TCP calibration method for vision guided robots based on inherent constraints of target object," *IEEE Access*, vol. 9, pp. 8902–8911, 2021.

[15] Y. Tang, M. Chen, C. Wang et al., "Recognition and localization methods for vision-based fruit picking robots: a review," *Frontiers of Plant Science*, vol. 11, p. 510, 2020.

[16] M. Chen, Y. Tang, and X. Zou, "3D global mapping of large-scale unstructured orchard integrating eye-in-hand stereo vision and SLAM," *Computers and Electronics in Agriculture*, vol. 187, Article ID 106237, 2021.

[17] M. Sága, V. Bulej, N. Čuboňova, and I. Kuric, "Case study: performance analysis and development of robotized screwing application with integrated vision sensing system for automotive industry," *International Journal of Advanced Robotic Systems*, vol. 17, pp. 1–23, 2020.

[18] P. Huang, Z. Zhang, and X. Luo, "Monocular visual navigation based on scene. model of differential-drive robot in corridor-like orchard environments," *International Agricultural Engineering Journal*, vol. 28, no. 1, pp. 310–316, 2019.

[19] G. E. Hinton and S. Osindero, "A fast learning algorithm for deep belief nets," *Neural Computation*, vol. 18, no. 7, pp. 1527–1554, 2006.
[20] Y. LeCun, Y. Bengio, and G. Hinton, "Deep learning," *Nature*, vol. 521, no. 7553, pp. 436–444, 2015.
[21] D. A. Pomerleau, "Alvinn: an autonomous land vehicle in a neural network," *Advances in Neural Information Processing Systems*, pp. 305–313, 1989.
[22] M. Bojarski, D. Del Testa, and D. Dworakowski, "End to end learning for self-driving cars," 2016, https://arxiv.org/abs/1604.07316.
[23] M. Bojarski, P. Yeres, A. Choromanska, and K. Choromanski, "Explaining how a deep neural network trained with end-to-end learning steers a car," 2017, https://arxiv.org/abs/1704.07911.
[24] S. Shalev-Shwartz, N. Ben-Zrihem, and A. Cohen, "Long-term planning by short-term prediction," 2016, https://arxiv.org/abs/1602.01580.
[25] E. Santana and G. Hotz, "Learning a driving simulator," 2016, https://arxiv.org/abs/1608.01230.
[26] S. Grigorescu, B. Trasnea, and T. Cocias, "A survey of deep learning techniques for autonomous driving," *Journal of Field Robotics*, vol. 37, no. 3, pp. 362–386, 2020.
[27] S. Chen, Y. Leng, and S. Labi, "A deep learning algorithm for simulating autonomous driving considering prior knowledge and temporal information," *Computer-Aided Civil and Infrastructure Engineering*, vol. 35, no. 4, pp. 305–321, 2020.
[28] J. Hu, X. Zhang, and S. Maybank, "Abnormal driving detection with normalized driving behavior data: a deep learning approach," *IEEE Transactions on Vehicular Technology*, vol. 69, no. 7, pp. 6943–6951, 2020.
[29] M. Gjoreski, M. Ž Gams, and M. Luštrek, "Machine learning and end-to-end deep learning for monitoring driver distractions from physiological and visual signals," *IEEE Access*, vol. 8, pp. 70590–70603, 2020.
[30] J. Ni, Y. Chen, and Y. Chen, "A survey on theories and applications for self-driving cars based on deep learning methods," *Applied Sciences*, vol. 10, no. 8, p. 2749, 2020.
[31] D. Shin, H. Kim, and K. Park, "Development of deep learning based human-centered threat assessment for application to automated driving vehicle," *Applied Sciences*, vol. 10, no. 1, p. 253, 2020.
[32] Z. Guo, Y. Huang, and X. Hu, "A survey on deep learning based approaches for scene understanding in autonomous driving," *Electronics*, vol. 10, no. 4, p. 471, 2021.
[33] G. Li, Y. Yang, and X. Qu, "A deep learning based image enhancement approach for autonomous driving at night," *Knowledge-Based Systems*, vol. 213, Article ID 106617, 2021.
[34] U. Muller, J. Ben, and E. Cosatto, "Off-road obstacle avoidance through end-to-end learning," in *Advances in Neural Information Processing Systems 18, Proceedings of the 2005 Conference*, pp. 739–746, Vancouver, Canada, December 2005.
[35] T. Hwu, J. Isbell, and N. Oros, "A self-driving robot using deep convolutional neural networks on neuromorphic hardware," 2016, https://arxiv.org/abs/1611.01235.
[36] J. Bell, B. A. MacDonald, and H. S. Ahn, "Row following in pergola structured orchards by a monocular camera using a fully convolutional neural network," in *Proceedings of 2016 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, pp. 640–645, Daejeon, Korea, October 2016.
[37] P. Huang, L. Zhu, and Z. Zhang, "Row end detection and headland turning control for an autonomous banana-picking robot," *Machines*, vol. 9, no. 5, 2021.
[38] R. Lingyan, Z. Yanning, and Z. Qilin, "Convolutional neural network-based robot navigation using uncalibrated spherical images," *Sensors*, vol. 17, no. 6, p. 1341, 2017.
[39] S. Ioffe and C. Szegedy, "Batch normalization: accelerating deep network training by reducing internal covariate shift," 2015, https://arxiv.org/abs/1502.03167.
[40] N. Srivastava, G. Hinton, and A. Krizhevsky, "Dropout: a simple way to prevent neural networks from overfitting," *Journal of Machine Learning Research*, vol. 15, no. 1, pp. 1929–1958, 2014.