AgriEngineering

Article

A Deep Learning-Driven Binocular Vision Path Detection Approach for Orchard Robots

Xiongchu Zhang *, Zhongle Zhou and Zhengtong Liu

College of Mechanical Engineering, Shenyang Ligong University, No. 6, Nanping Middle Road, Hunnan District, Shenyang 110159, China; 15315447551@163.com (Z.Z.); 13147800260@163.com (Z.L.)
* Correspondence: zhangxiongchu@gmail.com

Abstract

Autonomous driving relying on visual navigation plays a vital role in promoting automation within the jujube industry. Conventional visual navigation strategies fail to satisfy the demands of straddle-type jujube harvesters owing to their unique row-straddling configuration and complex orchard environments. Accordingly, this paper proposes a novel binocular vision-based path detection algorithm for autonomous jujube harvesters. In the proposed method, target trunks detected from binocular images are used to generate a navigation path that better aligns with the operating trajectory of harvester. A Single Shot MultiBox Detector (SSD) deep learning model is employed to detect trunk bounding boxes. To mitigate interference induced by false detections from the deep learning model, a curve-fitting-based path calibration strategy is implemented. Experimental results demonstrate that the proposed algorithm achieves a detection speed of 14.38 fps with a false detection rate of 3.64%, satisfying the operational demands for autonomous driving of jujube harvesters. Furthermore, this algorithm can be extended to other orchard mobile robots that execute row-straddling operations similar to jujube harvesters.

Keywords: jujube harvester; visual navigation; binocular vision; navigation path detection; deep learning

1. Introduction

The development of orchard mobile robots offers promising opportunities for improving automation, efficiency, and sustainability in fruit production [1]. To improve the automation level of the jujube industry and promote industrial transformation, a straddle-type jujube harvester was designed and developed by the laboratory at Shihezi University. The harvester adopts a row-straddling operating configuration, allowing it to straddle jujube tree rows and navigate along tree trunks during harvesting operations. To further reduce operational costs and improve the level of automation, developing an autonomous navigation system for the harvester has become a critical requirement. Furthermore, effective navigation path planning is essential for achieving autonomous operation of the jujube harvester [2]. However, conventional navigation methods are unsuitable for jujube harvesters due to the complex orchard environment and their unique row-straddling operating configuration. Therefore, this paper proposes a novel navigation path detection algorithm to enable autonomous operation of the jujube harvester.

Existing navigation path detection methods for orchard mobile robots can be broadly classified into three categories based on sensing technologies: GNSS (Global Navigation

Check for updates

Academic Editors: Ray E. Sheriff and Gang Yuan

Received: 27 April 2026
Revised: 7 August 2026
Accepted: 31 August 2026
Published: 2 September 2026

Copyright: © 2026 by the authors. Licensee MDPI, Basel, Switzerland. This article is an open access article distributed under the terms and conditions of the Creative Commons Attribution (CC BY) license.

Satellite System)-based methods, LiDAR (Light Detection And Ranging)-based methods, and Computer Vision-based methods [3].

With the availability of GNSS for civilian applications, agricultural production has been significantly advanced, particularly in the field of precision agriculture [4]. Currently, RTK-DGP-based navigation systems can achieve centimeter-level positioning accuracy and have been successfully implemented in agricultural fields, such as sugar beet production [5]. However, the reliability of GNSS-based navigation is significantly affected in orchard environments, where satellite signals may become unstable, unavailable, or inaccurate due to obstructions caused by tree canopies and complex terrain conditions [6–8].

To overcome the limitations of GNSS-based navigation in orchard environments, LiDAR has been widely investigated as a local perception sensor for autonomous navigation due to its ability to extract geometric features from surrounding scenes [9]. Several studies have explored LiDAR-based navigation methods for orchard mobile robots. A navigation method equipped with a 2D LiDAR scanner was developed by integrating Particle filter and Kalman filter algorithms [10]. Furthermore, a line feature extraction-based navigation approach was proposed using the PEARL algorithm on 2D point clouds [11]. Nevertheless, LiDAR-based navigation still faces challenges in complex orchard environments because sufficient and distinctive geometric features or reliable landmarks are difficult to identify [3]. Moreover, the relatively high cost of LiDAR sensors limits their widespread application in agricultural robotic systems [12].

With the rapid development of digital image processing technologies, computer vision has been widely adopted in agricultural production due to its intuitive perception capability and high accuracy [13]. In recent years, vision-based navigation of orchard mobile robots has attracted considerable research interest [14,15]. According to the number of cameras employed, existing visual navigation approaches can generally be categorized into monocular vision-based and stereo vision-based methods.

Monocular vision-based navigation approaches mainly rely on detecting navigation paths from in-row regions [16–18] or tree canopies [19]. Nevertheless, their performance is often affected by illumination variations and occlusions, resulting in limited navigation accuracy [20]. In addition, the canopy characteristics of jujube trees during the harvesting season are relatively inconspicuous and highly similar to the surrounding ground background, which increases the difficulty of image segmentation. More importantly, since jujube harvesters perform row-straddling operations and require navigation according to trunk positions, monocular vision-based approaches are inadequate for autonomous navigation because canopy information cannot reliably represent the spatial distribution of tree trunks.

Compared with monocular vision, stereo vision, especially binocular vision, exhibits superior performance in complex environments with uneven illumination and low contrast and has been widely used for terrain mapping and environmental perception [21,22]. In general, binocular vision systems estimate depth information by constructing disparity maps from the correspondence between left and right images, which enables 3D reconstruction and provides reliable information for navigation path generation [12,23–25]. Furthermore, binocular vision is particularly suitable for orchard mobile robots with limited payload and power capacity due to its passive sensing capability and relatively low hardware cost [20,26]. However, conventional binocular vision-based navigation approaches cannot meet the requirements of jujube harvesters because the fixed installation configuration of stereo cameras makes it difficult to directly obtain the trunk-related position navigation information required for row-straddling operations.

Based on the aforementioned analysis, this paper proposes a novel binocular vision-based navigation path detection algorithm for autonomous jujube harvesters. The proposed method employs deep learning techniques to estimate the positions of target trunks on

both sides of the harvester and generate the corresponding navigation path. Experimental results demonstrate that the proposed algorithm can accurately determine navigation paths for jujube harvesters and meet practical operational requirements. This study provides a theoretical foundation for implementing autonomous navigation in jujube harvesting systems. The main contribution of this work is the development of a trunk-guided navigation path detection algorithm based on a specially designed binocular vision system, which addresses the unique requirements of straddle-type jujube harvesters.

2. Materials and Methods

2.1. Experimental Platform and Equipment

The objective of this study is to develop a binocular vision-based navigation path detection algorithm for autonomous navigation of the jujube harvester based on tree trunk detection. The experimental platform and equipment, including the jujube harvester and binocular cameras, are described in the following sections.

2.1.1. Jujube Harvester

The proposed method was developed on the jujube harvester shown in Figure 1, which was independently designed and manufactured by Shihezi University. The harvester achieves automatic jujube harvesting by shaking tree branches and collecting the falling fruits. Figure 2 presents the top-view schematic of the jujube harvester during its row-straddling operation. The harvester moves along the tree row while straddling the jujube trees, and the binocular cameras are mounted on the lower left and lower right sides of the machine, respectively.



Figure 1. The jujube harvester designed by Shihezi University.

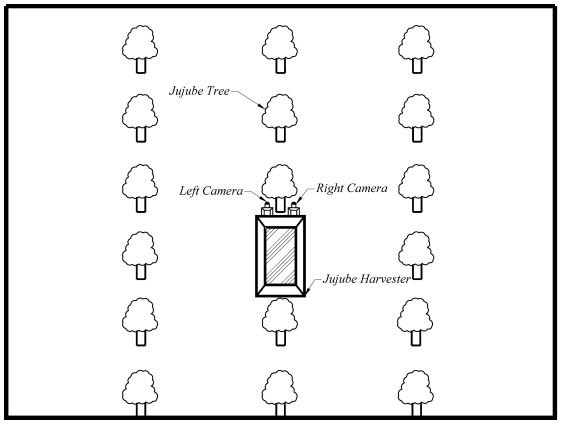


Figure 2. The top view diagram of the straddle-type jujube harvester operation.

2.1.2. Binocular Cameras

Figure 3 illustrates the binocular camera configuration on the jujube harvester. The cameras are symmetrically mounted on the left and right transmission devices, with a height of 40 cm above the ground and a baseline distance of 50 cm. The optical axes are oriented at 45° to the forward direction of the harvester. The main camera specifications are provided in Table 1.

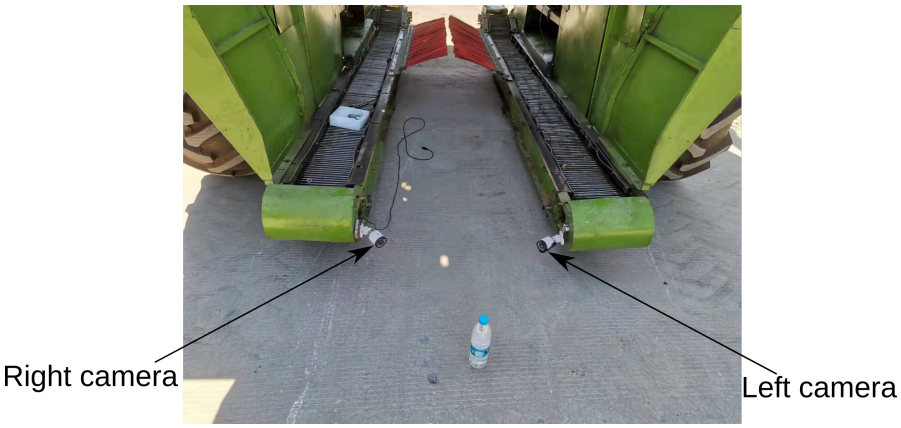


Figure 3. Binocular cameras installation in this paper.

Table 1. Main parameters of the camera used for image acquisition.

Resolution	Interface Type	Product Model	Temperature
640 × 480 VGA MJPEG	USB2.0 High Speed	RER-USBFHD01M-LS36	−20∼70 °C

Existing navigation path estimation methods cannot be directly applied to the jujube harvester due to its unique binocular camera configuration. Unlike conventional binocular systems with closely spaced cameras (Figure 4), the proposed system adopts a wider camera baseline, resulting in the following challenges:

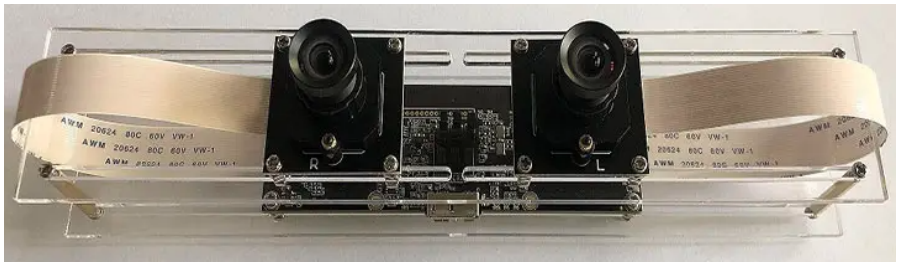


Figure 4. Conventional binocular cameras.

First, Figure 5 presents the binocular images acquired by the cameras mounted on the jujube harvester. Rectangular boxes with identical colors denote the same trunk in both views. As shown in Figure 6b, the overlapping field of view between the cameras is limited and changes during harvesting. Conventional stereo vision methods require stable camera geometry and epipolar constraints, which are difficult to preserve under field conditions. Although the cameras are rigidly mounted on the chassis, harvesting-induced vibrations continuously alter their relative pose and invalidate the calibrated epipolar geometry required by stereo matching algorithms such as ELAS (Efficient Large Scale Stereo Matching) [27]. Therefore, static calibration-based stereo methods cannot provide reliable trunk detection for jujube harvesting, motivating the proposed approach.



Figure 5. The actual image from binocular cameras in this paper.

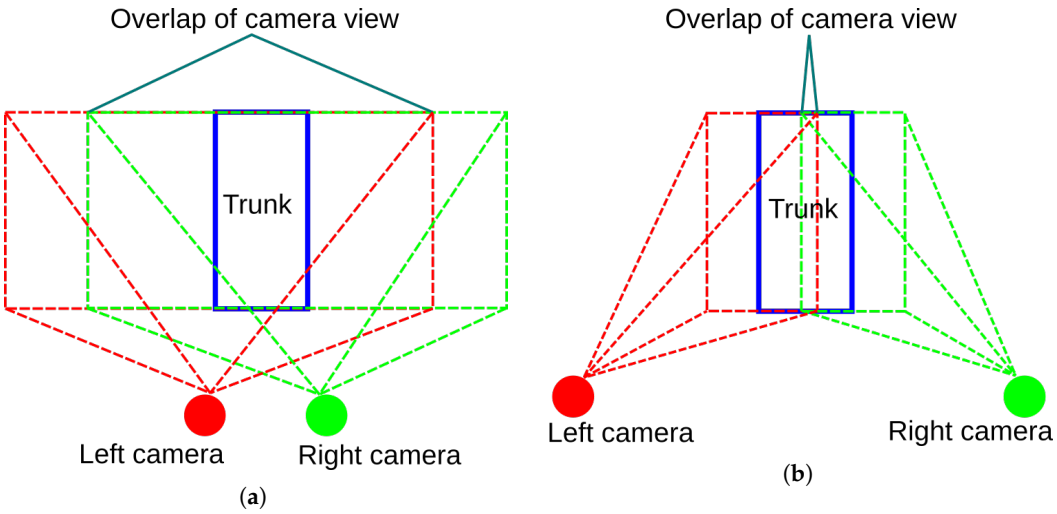


Figure 6. The overlap of camera view from binocular cameras in this paper: (**a**) overlap of camera view from conventional binocular cameras; (**b**) overlap of camera view from binocular cameras in this paper.

2.2. Image Acquisition

The video data were collected from 11 November 2021 to 1 December 2021, between 8:00 a.m. and 6:00 p.m. under clear weather conditions in a jujube orchard in A'er, Xinjiang, China. The cameras were installed according to the configuration described in Section 2.1.2. A worker manually drove the jujube harvester at an operating speed of 2.2 km/h to record the video sequences. Figure 7 shows the video acquisition process in the orchard, and the collected dataset is summarized in Table 2.



Figure 7. The scene of the jujube harvester collecting images in jujube orchards.

Table 2. The information of videos collected in jujube orchards.

Camera	Video Number	Frame Number	Frame Rate
left	24	7594	15
right	24	7594	15

2.3. Navigation Path

The objective of this study is to develop a trunk-guided navigation path detection method for autonomous jujube harvesting. Considering the unique binocular camera configuration of the jujube harvester, a customized binocular vision-based approach is proposed to generate suitable navigation paths.

Figure 8 illustrates the image configuration used for navigation path calculation. The distances d_l and d_r between the center (red cross) of the target trunk bounding box (green rectangle) and the left and right image boundaries are used to calculate the navigation path C in Equation (1). The target trunk is selected as the closest trunk detected by the object detection model.

$$C = \begin{cases} Turn\ right & d_r > d_l \\ Keep & d_r = d_l \\ Turn\ left & d_r < d_l \end{cases} \quad (1)$$

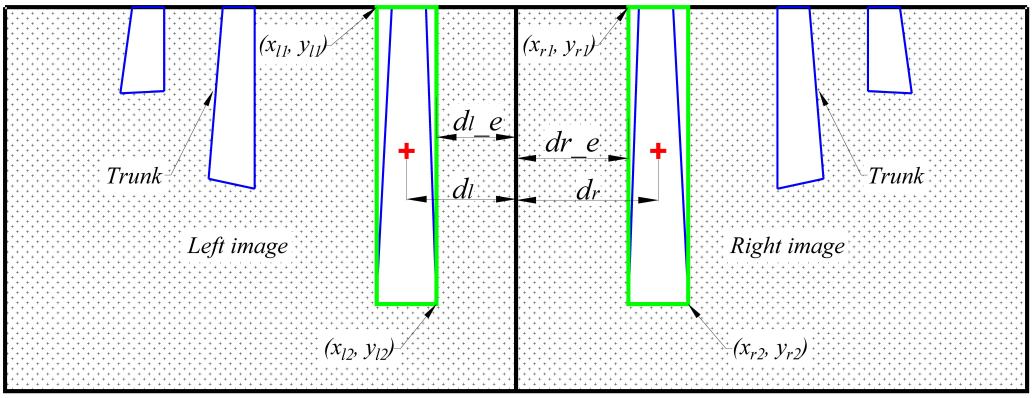


Figure 8. The diagram of the navigation path.

Equation (2) defines the calculation of d_l and d_r. The coordinates (x_{l1}, y_{l1}) and (x_{l2}, y_{l2}) denote the top-left and bottom-right corners of the target trunk bounding box in the left image, while (x_{r1}, y_{r1}), (x_{r2}, y_{r2}) represent the corresponding corners in the right image. w is the width of the left image. The top-left corner is set as the coordinate origin, and the horizontal and vertical directions are defined as the *x-axis* and the *y-axis*, respectively.

$$\begin{cases} d_l = w - \dfrac{x_{l1} + x_{l2}}{2} \\ d_r = \dfrac{x_{r1} + x_{r2}}{2} \end{cases} \quad (2)$$

Additionally, d_l_e and d_r_e denote the distances between the target trunk bounding box edges and the image boundaries, as calculated in Equation (3). They are used to determine whether the trunk reaches the image boundary.

$$\begin{cases} d_l_e = w - x_{l2} \\ d_r_e = x_{r1} \end{cases} \quad (3)$$

2.4. Trunk Boxes Obtained

The first step of the proposed method is to detect jujube tree trunks from binocular images. A deep learning-based object detection model is employed to generate trunk bounding boxes due to its effectiveness in object detection tasks.

2.4.1. Dataset Generation

The dataset consists of 15,188 images collected from jujube orchards. A total of 3155 images were randomly selected and manually annotated using LabelImg (https://github.com/heartexlabs/labelImg (accessed on March 6 2022)). The annotated dataset was randomly divided into training, testing, and validation sets with ratios of 60%, 20%, and 20%, containing 1893, 631, and 631 images, respectively.

2.4.2. Model Selection Method

With the rapid advancement of deep learning technologies, deep learning-based visual navigation has attracted increasing attention in orchard mobile robot applications [28–30]. Deep learning enables computational models to learn hierarchical data representations through multiple layers [31].

Three representative detectors, Faster R-CNN, YOLO V3, and SSD, were selected for comparison, representing two-stage high-accuracy, single-stage high-speed, and balanced single-stage architectures, respectively. These models were chosen to evaluate the adaptability of classical object detection frameworks for jujube trunk recognition. Lightweight detectors such as YOLO V7 and YOLO V8 were not included because the embedded computing platform of the harvester lacks compatibility with their required operators and frameworks. Moreover, this study focuses on robust path calibration rather than network backbone improvement. The deployment of advanced lightweight detectors on agricultural robotic platforms will be explored in future work.

Furthermore, a study [32] demonstrated that Faster R-CNN [33] outperformed YOLO V3 [34] in trunk detection and was successfully applied to orchard robot navigation under varying illumination conditions using thermal images. The SSD model [35] provides a favorable trade-off between detection accuracy and speed and has been widely adopted in industrial applications. Therefore, Faster R-CNN, YOLO V3, and SSD were selected for comparative evaluation of jujube trunk detection.

The model selection strategy was developed based on [36]. K-fold cross validation was employed to evaluate model performance [37,38]. The mean Average Precision (mAP) was selected as the primary evaluation metric for object detection [39–42]. Considering the requirement for accurate trunk localization, the IoU threshold was set to 0.9, and mAP@0.9 was adopted for evaluation. Detection speed was also included due to its importance in practical operations. In addition, batch size and learning rate were optimized as critical hyperparameters affecting model performance [43]. The model selection procedure is described as follows.

2.4.3. Image Augmentation

Image augmentation is widely used in deep learning training to increase data diversity, improve model generalization, and reduce overfitting [44,45]. Traditional augmentation techniques include affine transformations and color modifications [46]. Since trunk position is essential for navigation, rotation augmentation was excluded in this study. The applied augmentation methods included RGB-HSV color conversion, random contrast, random brightness, random cropping, expansion, random mirroring, and random light noise addition [47].

2.4.4. Semi-Supervised Training

Manual annotation is labor-intensive and time-consuming [48]. To reduce annotation requirements, semi-supervised learning utilizes both labeled and unlabeled data during model training [49,50]. Self-training is a widely used semi-supervised strategy for object detection, where pseudo-labels generated by model predictions are used to expand the training set [49,51]. Following [52], a pseudo-label-based self-training method was adopted in this study. The initial model M was trained using labeled data, and the iterative procedure is described as follows.

1. Generate pseudo labels for randomly selected unlabeled images using model M with confidence threshold c.
2. Combine pseudo-labeled images with the original training data.
3. Train a new model M' using the updated dataset.
4. Repeat Steps 2 to 3 until all unlabeled images are utilized.
5. Select the model with the highest $mAP@0.9$ on the test dataset.

2.5. Finding Target Trunk Boxes

Since multiple trees may appear in a single image, multiple trunk bounding boxes can be detected simultaneously. The trunk closest to the jujube harvester is selected as the target trunk. According to Section 2.3, this trunk corresponds to the bounding box closest to the image boundary in each binocular image. Therefore, d_l or d_r is calculated for all detected trunk boxes, and the box with the minimum value is selected as the target trunk box.

2.6. Path Calibration Method

Since deep learning models may produce detection errors, abnormal predictions must be handled before generating navigation commands. A filtering strategy is applied to remove frames with abnormal d_l or d_r values, and the trunk position is subsequently estimated to maintain continuous and stable control.

2.6.1. Finding the False Frame

A false frame refers to a frame with an incorrect detection result from the deep learning model. Figure 9a illustrates the correctly estimated navigation path, which provides real-time steering commands for jujube harvester direction adjustment.

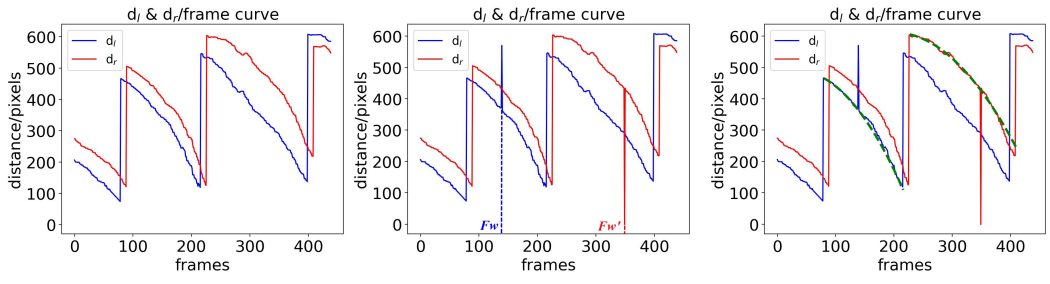


Figure 9. The diagram of path calibration: (**a**) The correct navigation path. (**b**) The false navigation path. (**c**) The navigation path after calibration.

As shown in Figure 9b, false detections occur in the frame of F_w, and F'_w. In frame F_w, the target trunk is present but missed by the SSD model in the left image (Figure 10b). In frame F'_w, the target trunk is missed in the right image, resulting in no detected trunk bounding box (Figure 10a). Equation (4) defines the criterion for false frame identification, where L denotes the label of the current frame.

$$L = \begin{cases} False & d > p \text{ and } d_e < k \\ True & besides \end{cases} \tag{4}$$

where

$$\begin{cases} d = d_l \text{ and } d_e = d_l_e & in\ left\ image \\ d = d_r \text{ and } d_e = d_r_e & in\ right\ image \end{cases} \tag{5}$$

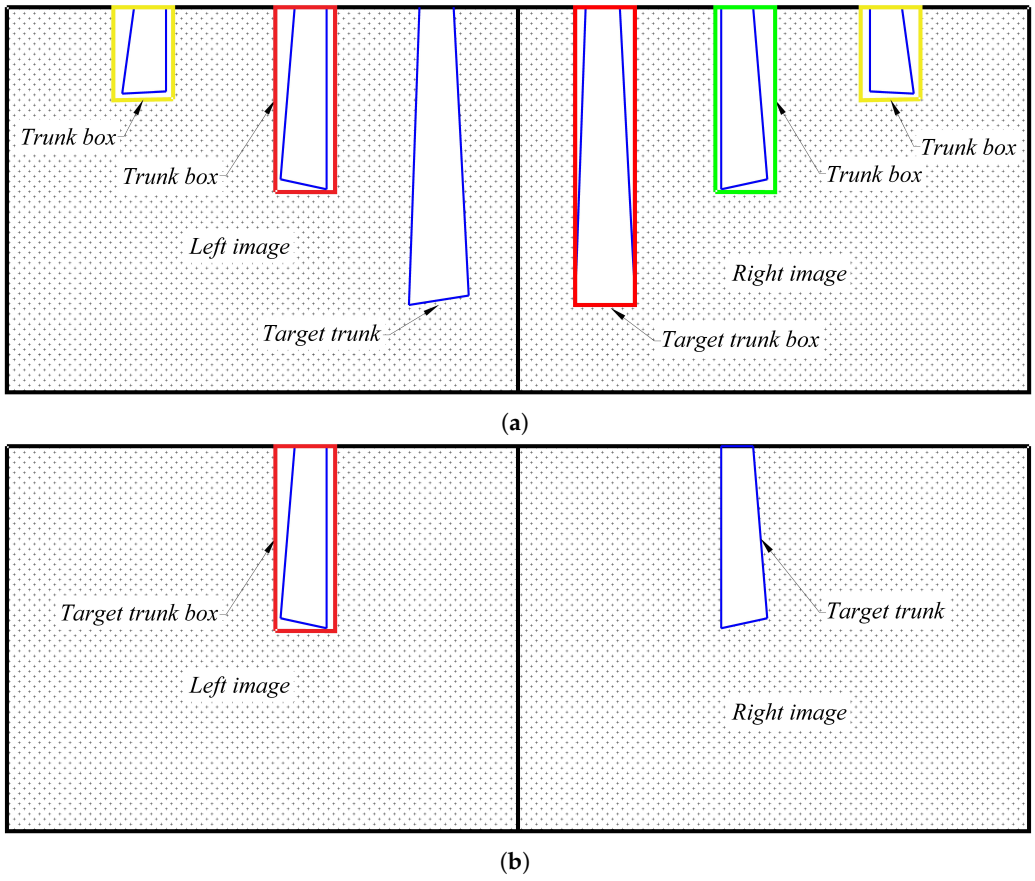


Figure 10. Diagrams of false frames: (**a**) F_w frame. (**b**) F'_w frame.

2.6.2. Achieve the Estimated Value

After identifying false frames, the missing values are estimated to calculate the navigation path. The curve function of the interval containing F_w or F'_w is defined in Equation (6), where each interval is approximated by a conic section. The coefficients a_0, a_1, and a_2 are obtained using the least squares method based on valid points before the false frame. The missing value d is then estimated by substituting F_w or F'_w into Equation (6).

$$d = a_0 + a_1 \cdot F^1 + a_2 \cdot F^2 \tag{6}$$

The fitted curve is plotted as the green dashed line in Figure 9. As illustrated, d_l and d_r defined under the F_w and F'_w frames can be adopted to compute the navigation path after path calibration.

To summarize, the work flow of the navigation system is shown in Figure 11. And the *control system* part of them is not introduced in this paper.

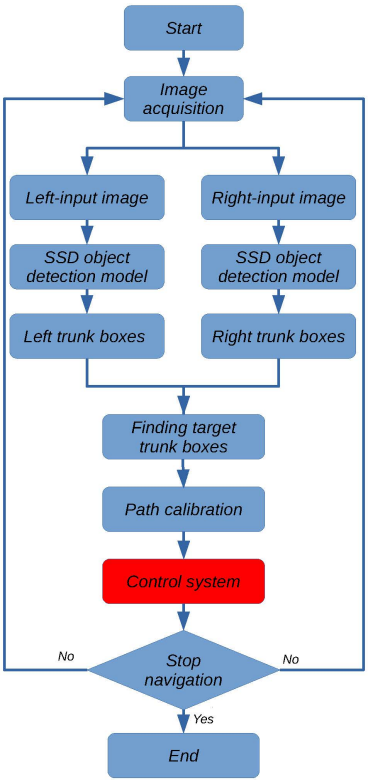


Figure 11. The work flow of the navigation system.

3. Results

All algorithms and data processing were implemented on the $Linux\ x64\ 5.4.0-131-generic$ system using *Visual Studio Code* (*V* 1.72.2) and *Anaconda* 3. The host computer is equipped with an $Intel(R)Core(TM)\ i7-4770CPU@3.40\ GHz$, 16 *GB* RAM, and a $GeForceRTX\ 2080Ti$ graphics card.

3.1. Deep Learning Model Selection

Figure 12 shows validation loss curves for models with varied hyperparameters trained using $5-foldcross-validation$, where *bz* and *lr* represent batch size and learning rate. The SSD model trained at learning rates of 1×10^{-2} and 1×10^{-3} cannot be applied to this study.

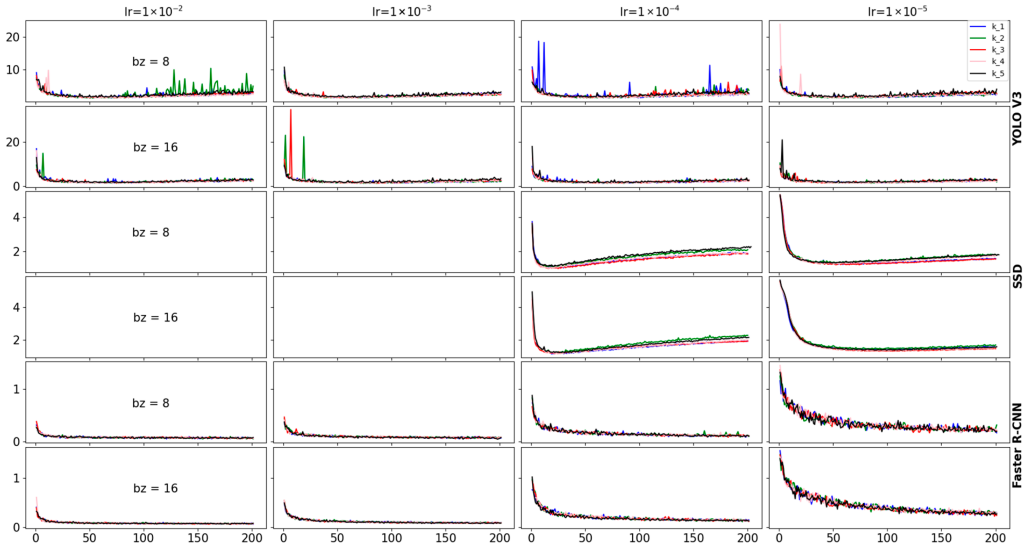


Figure 12. The training and validation loss curve of different models with different parameters.

Table 3 summarizes the model evaluation metrics. Faster R-CNN obtains the optimal $mAP@0.9$ ($(24.93 \pm 6.22)\%$), yet its detection speed ($(10.13 \pm 0.03) fps$) fps cannot satisfy the task's real-time demand. SSD outperforms YOLO V3 slightly in experiments and is therefore adopted as the deep learning backbone.

Table 3. The performance of different models trained by different hyperparameters.

Model	Learning Rate	Batch Size	mAP90 (%)	Speed (fps)	Recall (%)	Precision (%)	F1-Score (%)
Yolo V3	1×10^{-2}	8	21.18 ± 5.07	48.96 ± 2.15	42.08 ± 5.48	39.82 ± 5.41	40.91 ± 5.44
		16	19.76 ± 6.13	49.83 ± 0.45	40.08 ± 5.75	38.00 ± 5.85	39.01 ± 5.80
	1×10^{-3}	8	22.98 ± 3.82	49.50 ± 0.65	43.89 ± 3.89	40.30 ± 4.57	42.01 ± 4.26
		16	19.71 ± 5.19	49.89 ± 0.56	39.66 ± 5.17	37.52 ± 5.38	38.56 ± 5.29
	1×10^{-4}	8	23.10 ± 7.00	43.58 ± 0.39	42.70 ± 7.39	40.53 ± 7.14	41.59 ± 7.26
		16	21.22 ± 9.08	44.06 ± 1.06	42.42 ± 9.71	40.27 ± 9.33	41.32 ± 9.51
	1×10^{-5}	8	21.89 ± 4.17	8.73 ± 1.09	42.38 ± 4.49	39.61 ± 4.96	40.94 ± 4.74
		16	20.00 ± 5.15	42.70 ± 1.18	40.50 ± 5.96	38.12 ± 5.88	39.28 ± 5.91
SSD	1×10^{-2}	8	nan	nan	nan	nan	nan
		16	nan	nan	nan	nan	nan
	1×10^{-3}	8	nan	nan	nan	nan	nan
		16	nan	nan	nan	nan	nan
	1×10^{-4}	8	23.98 ± 2.74	87.04 ± 4.78	44.27 ± 2.12	43.92 ± 1.95	44.10 ± 2.03
		16	21.52 ± 3.28	85.08 ± 1.19	42.14 ± 3.17	41.86 ± 2.97	42.00 ± 3.07
	1×10^{-5}	8	15.53 ± 1.55	76.8 ± 4.87	34.90 ± 1.91	34.84 ± 1.87	34.87 ± 1.88
		16	14.66 ± 1.41	66.80 ± 5.69	33.10 ± 1.87	33.28 ± 1.65	33.19 ± 1.76
Faster R-CNN	1×10^{-2}	8	23.99 ± 7.34	10.22 ± 0.12	48.41 ± 7.45	41.67 ± 6.42	44.78 ± 6.90
		16	24.93 ± 6.22	10.13 ± 0.03	49.64 ± 6.73	41.48 ± 5.33	45.19 ± 5.94
	1×10^{-3}	8	23.48 ± 2.71	10.25 ± 0.05	48.67 ± 3.22	41.70 ± 2.91	44.92 ± 3.06
		16	19.89 ± 2.81	10.25 ± 0.09	43.89 ± 3.55	37.99 ± 3.16	40.72 ± 3.31
	1×10^{-4}	8	19.03 ± 1.00	10.21 ± 0.11	42.52 ± 1.28	36.74 ± 0.96	39.42 ± 1.10
		16	16.67 ± 2.87	9.64 ± 1.16	38.75 ± 3.29	33.11 ± 2.76	35.71 ± 3.00
	1×10^{-5}	8	10.84 ± 1.00	10.11 ± 0.07	28.65 ± 1.55	24.33 ± 1.67	26.31 ± 1.63
		16	7.36 ± 0.88	10.13 ± 6.86	22.94 ± 1.01	19.14 ± 1.12	20.87 ± 1.08

3.2. Semi-Supervised Learning

Two experiments were carried out to improve model performance using pseudo-labels. As a mature training scheme, Early Stopping frequently surpasses regularization methods and gains extensive application due to easy implementation [53]. The Early Stopping hyperparameters were configured with Patience of 20 and validation loss as the monitoring indicator.

The first experiment investigates the optimal quantity of pseudo-labeled samples for each training iteration. Experimental results are summarized in Table 4. As observed, the optimal pseudo-label count is 100, which yields the highest $mAP@0.9$ of 52.86% on the test set.

Table 4. The best number of pseudo-labels for every training set.

Number	mAP90 (%)
100	52.86
200	48.46
300	47.76
400	49.63
500	47.03
600	50.17
700	51.72
800	43.64
900	44.68
1000	46.09

Second, as unlabeled data far outnumber labeled samples, we further explore the optimal quantity of pseudo-labeled images. Figure 13 indicates that the model trained with approximately 1400 pseudo-labeled images achieves the best performance, and this configuration is adopted for our final model.

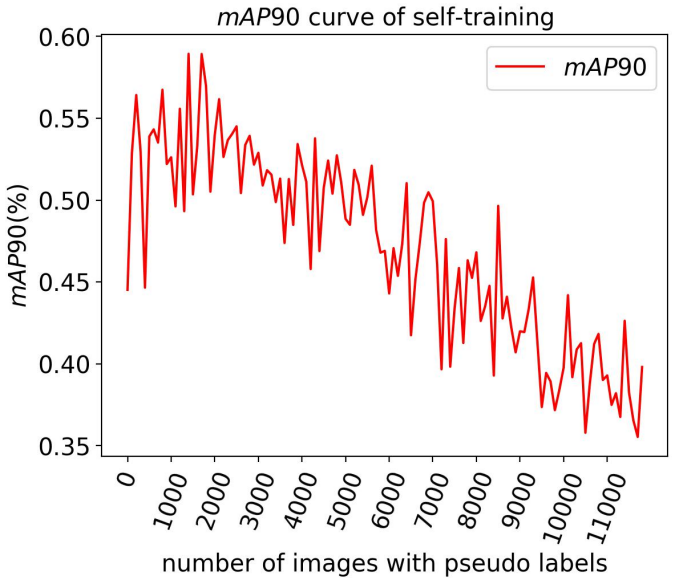


Figure 13. The *mAP*@0.9 curve of self-training under different numbers of unlabeled images.

3.3. Performance Improvement Result

Model performance after image augmentation and self-training is shown in Table 5. The *mAP*@0.9 metric rises by 20% with image augmentation and further improves by 14% via pseudo-label self-training. Accordingly, both image augmentation and pseudo-label self-training boost the *SSD* model performance in this task.

Table 5. The performance of models under different conditions.

Processing	mAP90 (%)
without image augmentation	23.98
with image augmentation	44.52
self-training	58.93

The 23.98% *mAP*@0.9 is the hyperparameter-optimized SSD from Table 3, serving as the unified baseline for all ablation tests in Table 5.

Although the *mAP*@0.9 of 58.93% is lower than that of dataset-oriented detectors, it meets the requirements of orchard navigation for three reasons:

1. Under heavy orchard interference, our optimizations raise the SSD baseline from 23.98% by over 34 percentage points.
2. Embedded path calibration reconstructs navigation paths during temporary detection failures discussed in Section 2.6.
3. The low traveling speed of the harvester enables the control system to compensate occasional detection deviations.

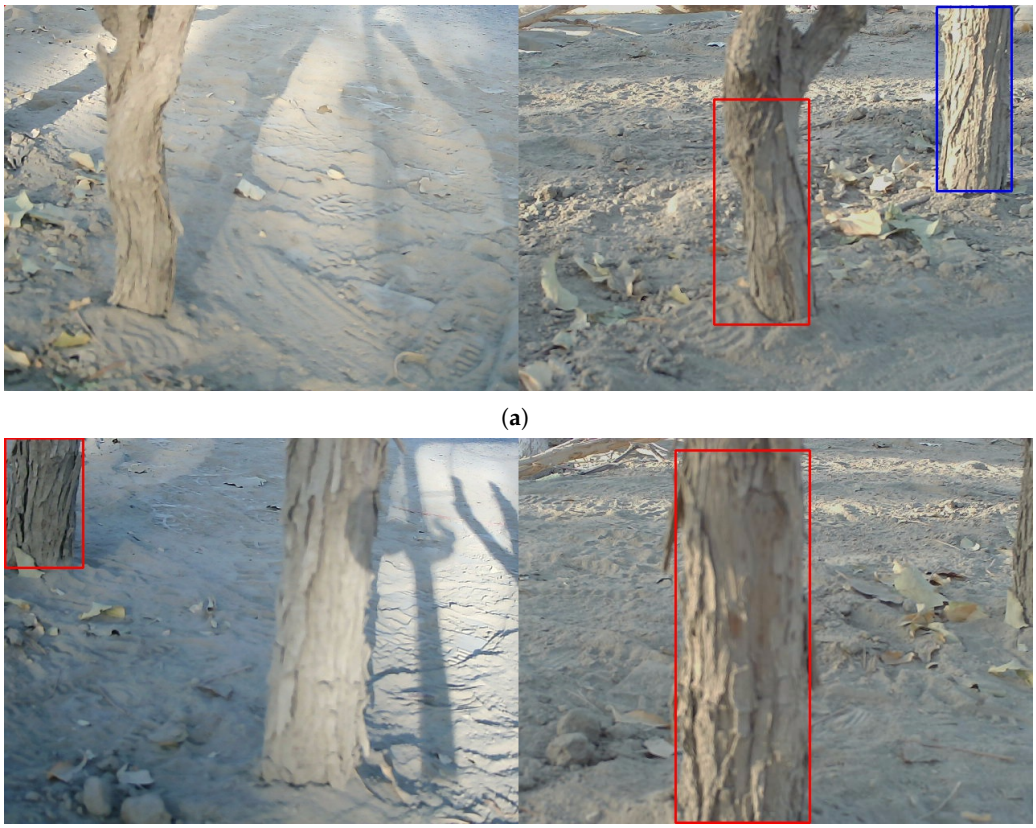
3.4. False Detection Analysis

Typical detection failure cases are presented below. In all subsequent figures, red boxes denote detected target trunks in binocular image pairs. In Figure 14a, the *SSD* model fails to output trunk bounding boxes on the left image, producing no valid navigation

path for this frame. In Figure 14b, the target trunk near the right edge of the left image is undetected, which yields an erroneous navigation path.

Furthermore, experiments on the navigation path false detection rate r_d are carried out using randomly sampled video clips from the collected dataset. The metric r_d is computed via Equation (7), where w_f and t_f denote the number of falsely detected frames and the total number of frames, respectively. Experimental results are summarized in Table 6.

$$r_d = \frac{w_f}{t_f} \times 100\% \tag{7}$$



(**a**)

(**b**)

Figure 14. The false detection caused by the SSD model.

Table 6. Experimental results of the false detection rate.

Speed (fps)	t_f (Frames)	w_f (Frames)	r_d (%)
14.38	11,575	422	3.64

The false detection rate reaches 3.64% and the inference speed is 14.38 fps, satisfying the autonomous navigation requirements of the jujube harvester.

Two frame-rate metrics must be differentiated in this work. High frame rates in the model comparison table denote pure single-image backbone inference, excluding preprocessing and post-processing. By contrast, the integrated system speed of 14.38 fps covers the full binocular pipeline: dual-camera synchronous *SSD* inference, valid trunk target filtering, path calibration interpolation for detection missing, and navigation path command generation for the harvester control unit. This full-stack frame rate characterizes the practical real-time performance on the physical harvester.

3.5. Path Calibration

In Figure 15, the first row Figure 15(a1–d1) plots raw d_l and d_r curves without path calibration from test videos. The second row Figure 15(a2–d2) presents the corresponding curves with outliers corrected via path calibration.

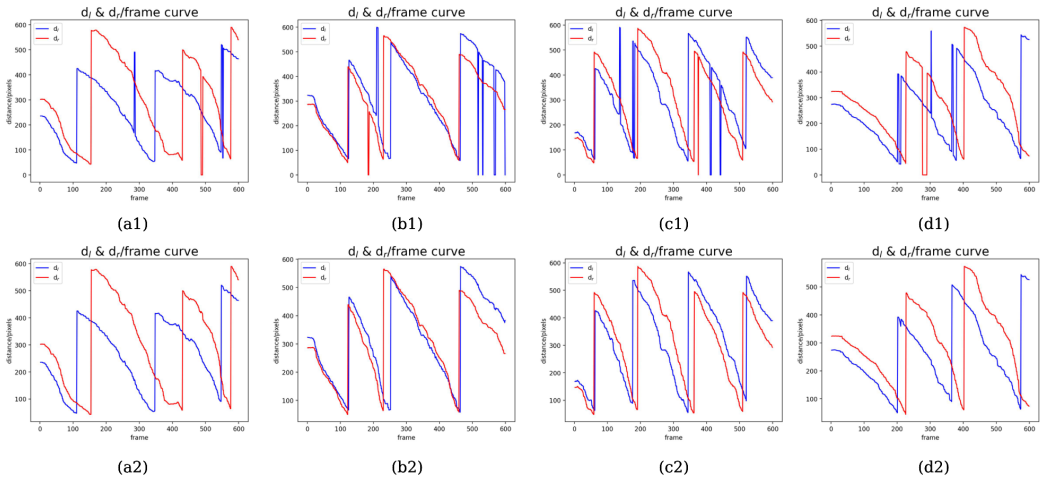


Figure 15. The curve before and after path calibration.

As observed, calibrated d_l and d_r can serve as navigation paths for autonomous jujube harvester navigation amid false detections.

The proposed conic interpolation for missing detection compensation has inherent limitations. The lightweight single conic fitting suits jujube harvesters navigating slowly along straight or gently curved rows over mildly undulating terrain, where smooth trajectory assumptions are valid. Under extreme conditions including sharp steering and rough terrain, abrupt trajectory variations cannot be precisely represented by one quadratic curve and produce large path deviations. Future work will employ segmented piecewise curve fitting to adapt to varying trajectory curvatures.

3.6. Ablation Study

It should be emphasized that the following comparison between SSD and Detectron2 only conducts a relative consistency test of trunk mass center outputs, rather than an absolute positioning accuracy assessment against manually annotated ground truth. This group of experiments aims to verify whether lightweight SSD can generate trunk center coordinates consistent with the high-precision segmentation model Detectron2 [54], rather than quantifying the real geometric error of either model against actual trunk positions. Figure 16 displays the detection result leaned on the Detectron 2 deep learning model designed, which is one of the state-of-the-art instance segmentation models. In this picture, for the same trunk, the blue-connected area and the red rectangle box is generated from the Detectron 2 model.

Although instance segmentation delivers higher detection accuracy, its low inference speed limits deployment on the jujube harvester. To verify whether general object detection suits this scenario, we calculate the distance between the two mass centers.

Equation (8) computes the mass center (c_i_x, c_i_y) of the blue connected area; (x_k, y_k) is the coordinate of the $k-th$ pixel and n is the total pixel count inside the region. Equation (9) yields the mass center (c_o_x, c_o_y) of the red bounding box, defined by its top-left (x_1, y_1) and bottom-right (x_2, y_2) corners. Errors are calculated via Equation (10), and results are listed in Table 7.

$$\begin{cases} c_i_x = \dfrac{\sum_{k=1}^{n} x_k}{n} \\ c_i_y = \dfrac{\sum_{k=1}^{n} y_k}{n} \end{cases} \tag{8}$$

$$\begin{cases} c_o_x = \dfrac{x_1 + x_2}{2} \\ c_o_y = \dfrac{y_1 + y_2}{2} \end{cases} \tag{9}$$

$$d = |c_i_x - c_o_x| \tag{10}$$

Table 7. Distance between object detection and instance segmentation.

Num	Frame	Instances Number	Total Error	Average Error (Pixels)
1	1000	1670	4601.866	2.755
2	1000	1699	4511.212	2.655
3	1000	1747	4562.463	2.611
4	1000	1685	4337.054	2.576
5	1000	1713	4590.470	2.679

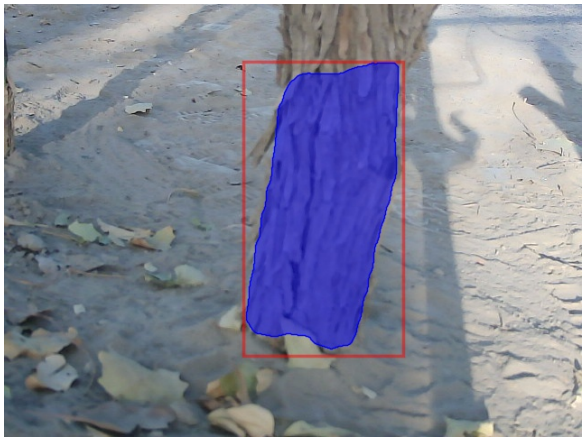


Figure 16. Results of the instance segmentation and the object detection based on the Detectron 2 model.

As listed in the table, the average error ranges from 2.5 to 2.7 pixels and stays within acceptable bounds. Thus, the object detection accuracy meets the autonomous navigation requirements of jujube harvester. Nevertheless, this comparative evaluation carries inherent limitations. The slight coordinate offset between SSD and Detectron2 merely proves prediction consistency between the two models, rather than the absolute deviation from the real trunk ground truth. Future work will adopt the manually annotated ground truth to measure absolute positioning error and further quantify the geometric precision of the lightweight SSD.

4. Limitations

The limitations of this work are illustrated via the d_l and d_r curves in Figure 15. Obvious discrepancies arise on the curves when detection switches from one target trunk to the next, accompanied by substantial divergence between d_l and d_r. This phenomenon mainly stems from inconsistent mounting heights and installation angles of the two cameras. In addition, the algorithm is validated only on collected datasets; further field experiments in jujube orchards will be conducted in future work.

5. Conclusions

This work develops a deep learning binocular vision navigation algorithm for autonomous jujube harvesters. Comparative and ablation experiments validate three key contributions: SSD surpasses $YOLO\ V3$ and $Faster\ R-CNN$ for trunk detection under complex orchard conditions; pseudo-label self-training improves baseline SSD accuracy; and conic-section path calibration boosts system robustness against detection loss and false positives. The binocular vision system steadily outputs real-time navigation paths in natural orchards. This lightweight anti-interference framework supports orchard autonomous navigation and offers an algorithmic basis for unmanned jujube harvesters.

Future work covers two aspects. Algorithmically, we improve trunk detection accuracy and long-sequence stability for extreme orchard environments. For system integration, the vision module will connect to the vehicle controller for closed-loop field tests, with continuous in-row trials verifying real-time performance and practical reliability.

In conclusion, this paper constructs a full binocular vision pipeline adapted to vibration-disturbed jujube orchards. Comparative tests, pseudo-label self-training ablation studies and missing-path calibration confirm that the integrated algorithm reliably generates navigation trajectories for autonomous harvesters, delivering a viable visual perception scheme for unmanned orchard harvesting equipment.

Author Contributions: Conceptualization, X.Z.; methodology, X.Z.; software, X.Z.; validation, Z.Z.; formal analysis, X.Z. and Z.L.; investigation, X.Z.; data curation, Z.L. and Z.Z.; writing—original draft preparation, X.Z.; writing—review and editing, X.Z.; visualization, X.Z.; supervision, X.Z. All authors have read and agreed to the published version of the manuscript.

Funding: This work was supported by the National Key R & D Program of China (No. 2020YFB1711703 and No. 2020YFB2007700), the Key Projects of National Natural Science Foundation of China (No. 52035002), the National Natural Science Foundation of China (No. 62173356), and the Science and Technology Development Fund (FDCT) of Macau, China (No. 0019/2021/A).

Institutional Review Board Statement: Not applicable.

Informed Consent Statement: Not applicable.

Data Availability Statement: Data are not available for sensibility reasons.

Acknowledgments: The authors would like to thank the editor, associate editor and anonymous reviewers for their constructive comments and suggestions that significantly improve this paper.

Conflicts of Interest: The authors declare no conflicts of interest.

References

1. Bayar, G.; Bergerman, M.; Koku, A.B.; ilhan Konukseven, E. Localization and control of an autonomous orchard vehicle. *Comput. Electron. Agric.* **2015**, *115*, 118–128. [CrossRef]
2. Chen, J.; Wang, Z.; Long, T.; Wu, J.; Cai, G.; Zhang, H. Research on Navigation Line Extraction of Garden Mobile Robot Based on Edge Detection. *J. Intell. Robot. Syst.* **2022**, *105*, 27. [CrossRef]
3. Li, X.; Qiu, Q. Autonomous Navigation for Orchard Mobile Robots: A Rough Review. In Proceedings of the 2021 36th Youth Academic Annual Conference of Chinese Association of Automation (YAC), Nanchang, China, 28–30 May 2021; pp. 552–557. [CrossRef]
4. Rovira-Más, F.; Chatterjee, I.; Sáiz-Rubio, V. The role of GNSS in the navigation strategies of cost-effective agricultural robots. *Comput. Electron. Agric.* **2015**, *112*, 172–183. [CrossRef]
5. Bakker, T.; van Asselt, K.; Bontsema, J.; Müller, J.; van Straten, G. Autonomous navigation using a robot platform in a sugar beet field. *Biosyst. Eng.* **2011**, *109*, 357–368. [CrossRef]
6. Jones, M.H.; Bell, J.; Dredge, D.; Seabright, M.; Scarfe, A.; Duke, M.; MacDonald, B. Design and testing of a heavy-duty platform for autonomous navigation in kiwifruit orchards. *Biosyst. Eng.* **2019**, *187*, 129–146. [CrossRef]
7. Gan, H.; Lee, W. Development of a Navigation System for a Smart Farm. *IFAC-PapersOnLine* **2018**, *51*, 1–4. [CrossRef]

8. Aguiar, A.S.; dos Santos, F.N.; Cunha, J.B.; Sobreira, H.; Sousa, A.J. Localization and Mapping for Robots in Agriculture and Forestry: A Survey. *Robotics* **2020**, *9*, 97. [CrossRef]
9. Nehme, H.; Aubry, C.; Solatges, T.; Savatier, X.; Rossi, R.; Boutteau, R. LiDAR-based Structure Tracking for Agricultural Robots: Application to Autonomous Navigation in Vineyards. *J. Intell. Robot. Syst.* **2021**, *103*, 61. [CrossRef]
10. Blok, P.M.; van Boheemen, K.; van Evert, F.K.; IJsselmuiden, J.; Kim, G.H. Robot navigation in orchards with localization based on Particle filter and Kalman filter. *Comput. Electron. Agric.* **2019**, *157*, 261–269. [CrossRef]
11. Malavazi, F.B.; Guyonneau, R.; Fasquel, J.B.; Lagrange, S.; Mercier, F. LiDAR-only based navigation algorithm for an autonomous agricultural robot. *Comput. Electron. Agric.* **2018**, *154*, 71–79. [CrossRef]
12. Li, Y.; Wang, X.; Liu, D. 3D autonomous navigation line extraction for field roads based on binocular vision. *J. Sens.* **2019**, *2019*, 6832109. [CrossRef]
13. Liu, S.; Wang, X.; Li, S.; Chen, X.; Zhang, X. Obstacle avoidance for orchard vehicle trinocular vision system based on coupling of geometric constraint and virtual force field method. *Expert Syst. Appl.* **2022**, *190*, 116216. [CrossRef]
14. Zhou, J.; Geng, S.; Qiu, Q.; Shao, Y.; Zhang, M. A Deep-Learning Extraction Method for Orchard Visual Navigation Lines. *Agriculture* **2022**, *12*, 1650. [CrossRef]
15. Opiyo, S.; Okinda, C.; Zhou, J.; Mwangi, E.; Makange, N. Medial axis-based machine-vision system for orchard robot navigation. *Comput. Electron. Agric.* **2021**, *185*, 106153. [CrossRef]
16. Yang, Z.; Ouyang, L.; Zhang, Z.; Duan, J.; Yu, J.; Wang, H. Visual navigation path extraction of orchard hard pavement based on scanning method and neural network. *Comput. Electron. Agric.* **2022**, *197*, 106964. [CrossRef]
17. Sharifi, M.; Chen, X. A novel vision based row guidance approach for navigation of agricultural mobile robots in orchards. In *Proceedings of the 2015 6th International Conference on Automation, Robotics and Applications (ICARA)*; IEEE: New York, NY, USA, 2015; pp. 251–255. [CrossRef]
18. Lyu, H.K.; Park, C.H.; Han, D.H.; Kwak, S.W.; Choi, B. Orchard Free Space and Center Line Estimation Using Naive Bayesian Classifier for Unmanned Ground Self-Driving Vehicle. *Symmetry* **2018**, *10*, 355. [CrossRef]
19. Radcliffe, J.; Cox, J.; Bulanon, D.M. Machine vision for orchard navigation. *Comput. Ind.* **2018**, *98*, 165–171. [CrossRef]
20. Du, Y.; Fan, B.; Han, J.; Tang, Y. Binocular Based Moving Target Tracking for Mobile Robot. In *Proceedings of the Intelligent Robotics and Applications*; Xie, M., Xiong, Y., Xiong, C., Liu, H., Hu, Z., Eds.; Springer: Berlin/Heidelberg, Germany, 2009; pp. 929–935. [CrossRef]
21. Tang, Y.; Chen, M.; Wang, C.; Luo, L.; Li, J.; Lian, G.; Zou, X. Recognition and Localization Methods for Vision-Based Fruit Picking Robots: A Review. *Front. Plant Sci.* **2020**, *11*, 510. [CrossRef]
22. Rovira-Más, F.; Zhang, Q.; Reid, J.F. Stereo vision three-dimensional terrain maps for precision agriculture. *Comput. Electron. Agric.* **2008**, *60*, 133–143. [CrossRef]
23. Zhang, S.; Wang, Y.; Zhu, Z.; Li, Z.; Du, Y.; Mao, E. Tractor path tracking control based on binocular vision. *Inf. Process. Agric.* **2018**, *5*, 422–432. [CrossRef]
24. Takahashi, T.; Zhang, S.; Fukuchi, H. Acquisition of 3-D information by binocular stereo vision for vehicle navigation through an orchard. In *Automation Technology for Off-Road Equipment*; American Society of Agricultural and Biological Engineers: St. Joseph, MI, USA, 2002; p. 337.
25. Zhai, Z.; Zhu, Z.; Du, Y.; Song, Z.; Mao, E. Multi-crop-row detection algorithm based on binocular vision. *Biosyst. Eng.* **2016**, *150*, 89–103. [CrossRef]
26. Hu, P.; Hao, X.; Li, J.; Cheng, C.; Wang, A. Design and implementation of binocular vision system with an adjustable baseline and high synchronization. In *Proceedings of the 2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*; IEEE: New York, NY, USA, 2018; pp. 566–570. [CrossRef]
27. Geiger, A.; Roser, M.; Urtasun, R. Efficient Large-Scale Stereo Matching. In *Proceedings of the Computer Vision—ACCV 2010*; Kimmel, R., Klette, R., Sugimoto, A., Eds.; Springer: Berlin/Heidelberg, Germany, 2011; pp. 25–38. [CrossRef]
28. Kim, W.S.; Lee, D.H.; Kim, Y.J.; Kim, T.; Lee, H.J. Path detection for autonomous traveling in orchards using patch-based CNN. *Comput. Electron. Agric.* **2020**, *175*, 105620. [CrossRef]
29. Han, S.H.; Kang, K.M.; Choi, C.H.; Lee, D.H. Deep learning-based path detection in citrus orchard. In *Proceedings of the 2020 ASABE Annual International Virtual Meeting*; American Society of Agricultural and Biological Engineers: St. Joseph, MI, USA, 2020; p. 1. [CrossRef]
30. Shang, G.; Liu, G.; Zhu, P.; Han, J.; Xia, C.; Jiang, K. A Deep Residual U-Type Network for Semantic Segmentation of Orchard Environments. *Appl. Sci.* **2021**, *11*, 322. [CrossRef]
31. Pouyanfar, S.; Sadiq, S.; Yan, Y.; Tian, H.; Tao, Y.; Reyes, M.P.; Shyu, M.L.; Chen, S.C.; Iyengar, S.S. A survey on deep learning: Algorithms, techniques, and applications. *ACM Comput. Surv. (CSUR)* **2018**, *51*, 1–36. [CrossRef]
32. Jiang, A.; Noguchi, R.; Ahamed, T. Tree Trunk Recognition in Orchard Autonomous Operations under Different Light Conditions Using a Thermal Camera and Faster R-CNN. *Sensors* **2022**, *22*, 2065. [CrossRef]

33. Ren, S.; He, K.; Girshick, R.B.; Sun, J. Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks. *Adv. Neural Inf. Process. Syst.* **2015**, 28, 91–99.
34. Redmon, J.; Farhadi, A. YOLOv3: An Incremental Improvement. *arXiv* **2018**, arXiv:1804.02767. [CrossRef]
35. Liu, W.; Anguelov, D.; Erhan, D.; Szegedy, C.; Reed, S.; Fu, C.Y.; Berg, A.C. SSD: Single Shot MultiBox Detector. In *Proceedings of the Computer Vision—ECCV 2016*; Leibe, B., Matas, J., Sebe, N., Welling, M., Eds.; Springer: Cham, Switzerland, 2016; pp. 21–37. [CrossRef]
36. Xu, Y.; Goodacre, R. On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning. *J. Anal. Test.* **2018**, *2*, 249–262. [CrossRef]
37. Saud, S.; Jamil, B.; Upadhyay, Y.; Irshad, K. Performance improvement of empirical models for estimation of global solar radiation in India: A k-fold cross-validation approach. *Sustain. Energy Technol. Assess.* **2020**, *40*, 100768. [CrossRef]
38. Wong, T.T.; Yeh, P.Y. Reliable accuracy estimates from k-fold cross validation. *IEEE Trans. Knowl. Data Eng.* **2019**, *32*, 1586–1594. [CrossRef]
39. Everingham, M.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes (voc) challenge. *Int. J. Comput. Vis.* **2010**, *88*, 303–338. [CrossRef]
40. Everingham, M.; Eslami, S.; Van Gool, L.; Williams, C.K.; Winn, J.; Zisserman, A. The pascal visual object classes challenge: A retrospective. *Int. J. Comput. Vis.* **2015**, *111*, 98–136. [CrossRef]
41. Russakovsky, O.; Deng, J.; Su, H.; Krause, J.; Satheesh, S.; Ma, S.; Fei-Fei, L. ImageNet large scale visual recognition. *arXiv* **2014**, arXiv:1409.0575.
42. Henderson, P.; Ferrari, V. End-to-End Training of Object Class Detectors for Mean Average Precision. In *Proceedings of the Computer Vision—ACCV 2016*; Lai, S.H., Lepetit, V., Nishino, K., Sato, Y., Eds.; Springer: Cham, Switzerland, 2017; pp. 198–213. [CrossRef]
43. Kandel, I.; Castelli, M. The effect of batch size on the generalizability of the convolutional neural networks on a histopathology dataset. *ICT Express* **2020**, *6*, 312–315. [CrossRef]
44. Shorten, C.; Khoshgoftaar, T.M. A survey on image data augmentation for deep learning. *J. Big Data* **2019**, *6*, 60. [CrossRef]
45. Bloice, M.D.; Stocker, C.; Holzinger, A. Augmentor: An image augmentation library for machine learning. *arXiv* **2017**, arXiv:1708.04680. [CrossRef]
46. Mikołajczyk, A.; Grochowski, M. Data augmentation for improving deep learning in image classification problem. In Proceedings of the 2018 International Interdisciplinary PhD Workshop (IIPhDW), Świnoujście, Poland, 9–12 May 2018, pp. 117–122. [CrossRef]
47. Khalifa, N.E.; Loey, M.; Mirjalili, S. A comprehensive survey of recent trends in deep learning for digital images augmentation. *Artif. Intell. Rev.* **2021**, *55*, 2351–2377. [CrossRef]
48. Hu, N.; Dannenberg, R.B. A Bootstrap Method for Training an Accurate Audio Segmenter. In Proceedings of the ISMIR, London, UK, 11–15 September 2005; pp. 223–229.
49. Rosenberg, C.; Hebert, M.; Schneiderman, H. Semi-supervised self-training of object detection models. In Proceedings of the 2005 Seventh IEEE Workshops on Applications of Computer Vision (WACV/MOTION'05), Breckenridge, CO, USA, 5–7 January 2005. [CrossRef]
50. Albert, P.; Ortego, D.; Arazo, E.; O'Connor, N.; McGuinness, K. ReLaB: Reliable Label Bootstrapping for Semi-Supervised Learning. In Proceedings of the 2021 International Joint Conference on Neural Networks (IJCNN), Shenzhen, China, 18–22 July 2021; pp. 1–8. [CrossRef]
51. Tanha, J.; Van Someren, M.; Afsarmanesh, H. Semi-supervised self-training for decision tree classifiers. *Int. J. Mach. Learn. Cybern.* **2017**, *8*, 355–370. [CrossRef]
52. Lee, D.H. Pseudo-label: The simple and efficient semi-supervised learning method for deep neural networks. In Proceedings of the Workshop on Challenges in Representation Learning, ICML, Atlanta, GA, USA, 16–21 June 2013; Volume 3, p. 896.
53. Prechelt, L. Automatic early stopping using cross validation: Quantifying the criteria. *Neural Netw.* **1998**, *11*, 761–767. [CrossRef]
54. Wu, Y.; Kirillov, A.; Massa, F.; Lo, W.Y.; Girshick, R. Detectron2. 2019. Available online: https://github.com/facebookresearch/detectron2 (accessed on 23 April 2022).

Disclaimer/Publisher's Note: The statements, opinions and data contained in all publications are solely those of the individual author(s) and contributor(s) and not of MDPI and/or the editor(s). MDPI and/or the editor(s) disclaim responsibility for any injury to people or property resulting from any ideas, methods, instructions or products referred to in the content.