agronomy

Article

Machine Learning Feature Extraction Based on Binary Pixel Quantification Using Low-Resolution Images for Application of Unmanned Ground Vehicles in Apple Orchards

Hong-Kun Lyu [1,*], Sanghun Yun [1] and Byeongdae Choi [1,2]

1 Division of Electronics and Information System, ICT Research Institute, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu 42988, Korea; shyun@dgist.ac.kr (S.Y.); bdchoi1@dgist.ac.kr (B.C.)
2 Department of Interdisciplinary Engineering, Daegu Gyeongbuk Institute of Science and Technology (DGIST), Daegu 42988, Korea
* Correspondence: hklyu@dgist.ac.kr; Tel.: +82-53-785-3550

Received: 20 November 2020; Accepted: 4 December 2020; Published: 8 December 2020

Abstract: Deep learning and machine learning (ML) technologies have been implemented in various applications, and various agriculture technologies are being developed based on image-based object recognition technology. We propose an orchard environment free space recognition technology suitable for developing small-scale agricultural unmanned ground vehicle (UGV) autonomous mobile equipment using a low-cost lightweight processor. We designed an algorithm to minimize the amount of input data to be processed by the ML algorithm through low-resolution grayscale images and image binarization. In addition, we propose an ML feature extraction method based on binary pixel quantification that can be applied to an ML classifier to detect free space for autonomous movement of UGVs from binary images. Here, the ML feature is extracted by detecting the local-lowest points in segments of a binarized image and by defining 33 variables, including local-lowest points, to detect the bottom of a tree trunk. We trained six ML models to select a suitable ML model for trunk bottom detection among various ML models, and we analyzed and compared the performance of the trained models. The ensemble model demonstrated the best performance, and a test was performed using this ML model to detect apple tree trunks from 100 new images. Experimental results indicate that it is possible to recognize free space in an apple orchard environment by learning using approximately 100 low-resolution grayscale images.

Keywords: machine learning (ML); unmanned ground vehicle (UGV); orchard; binary image; feature extraction; ensemble model

1. Introduction

Fruits and vegetables are important foods for human health and a balanced diet. The U.S. Department of Agriculture and the U.S. Department of Health and Human Services publish dietary guidelines every five years, including recommendations to increase fruit and vegetable intake. In the United States, it is recommended that children consume 1 to 1.5 cups of fruit daily and adults 1.5 to 2 cups, depending on age and gender. Apples contains many essential nutrients, e.g., vitamin C, potassium, and fiber [1]. Apples are considered a superfood because they have high nutritional value. Apples also contain antioxidants, e.g., flavonoids and polyphenols, and eating apples can reduce cholesterol and sugar diabetes, as well as make your skin healthier [2]. As human life has been enriched in recent decades, the consumption of apples has increased, primarily among health-conscious individuals. The fruit market, including apples, is expected to continue to grow as the demand for superfoods increases [2]. According to the Food and Agriculture Organization's 2018 World Fruit Production Statistics, globally, apples represent the third largest fruit production [3]. The highest producing countries are China, the United States, India, and Turkey [3,4].

Self-driving vehicle technology has entered the commercialization stage with the advancement of machine learning (ML) technologies. In addition, practical technology development for various industrial applications, including agricultural applications, is ongoing. Deep learning and machine learning technologies are being actively researched for application in the agricultural field, and a representative technical field is image-based object recognition technology [5–8]. Here, the target objects include all things of industrial interest, e.g., trees, people, cars, roads, buildings, various obstacles, objects, numbers, and letters. An important technology along with object recognition technology for unmanned ground vehicles (UGVs) in agricultural environments is free space recognition technology to recognize the area in which UGVs can move. Unlike public roads, there are no lanes and traffic lights in farmland, so there was a need for an algorithm to detect such free space, and studies have been reported in several research groups [9–15]. A method of using color image-based histogram of oriented gradients (HOG) feature extraction process and support vector machine (SVM) classification process to recognize tree trunk [9], HOG-SVM technology based on multiple cameras and ultrasonic sensors [10] has been proposed, but the HOG feature is inevitably more computational than the ML feature extraction from the binary image proposed in this paper. The tree trunk detection technology based on the lidar sensor module shows relatively good performance, but there is a limitation in applying the expensive lidar sensor [11,12]. Among the methods of generating autonomous paths in an orchard based on monocular color images, the technique of converting continuous frame images to the local-structure of the tree rows has a disadvantage of increasing the complexity of the algorithm in the process of comparing multiple frame images [13]. In a horizontal projection method, there is a problem that performance cannot be guaranteed if the ground pattern is complicated due to weeds and soil [14]. As an alternative to this problem, an algorithm using a machine vision technique based on the sky image has been proposed, but it is difficult to apply in seasons with few leaves [15]. It is still difficult to accurately recognize tree trunks and locate trees in an orchard. Meanwhile, with machine learning technology, the developer defines feature data that well express the features of the detection target, uses the features to train the ML model, verifies the trained ML model, and uses the trained ML model as a detector for a new input. Deep learning technology is based on an algorithm that can directly find a feature that expresses the characteristics of a target object for recognition and evaluation. Recently, to expand application and service areas, the demand for machine learning and deep learning technologies to run in mobile and embedded edge computing environments has increased in the image analysis field.

Specifically, image analysis technology is required in low-end devices, e.g., IoT devices, devices with power limitations, e.g., drones and smartphones, marine and forested areas, and agricultural environments where Internet connections are not supported. In line with these requirements, the recent development of technologies that can operate mobile/embedded environments is centered on deep learning technologies, which are collectively referred to as lightweight deep learning technology. Lightweight deep learning technology has been researched relative to model weight reduction, quantization, pruning, depth-wise separable convolution, etc., through efficient layer design in deep learning models. However, deep learning is an artificial neural network comprising several hidden layers between the input and output layers [16,17]. However, with multiple hidden layers, a computer must perform many numbers of simple iterative calculations; thus, a GPU environment is required for industrial application of deep learning technology [18–21].

Therefore, in this paper, we propose a machine learning modeling method that distinguishes free space in which UGVs can move. The proposed method is expected to be used as an element technology for the development of object recognition technology applicable to small smart farms, e.g., orchard environments. In our previous work, the coordinates of the lowest part of the image binarized with the ML function were extracted, and the free space centerline between the two rows of apple trees was finally estimated using Naive Bayesian classification [7]. However, in this study, we used the same binarized image as in the previous study, but we intend to detect the location of the tree with high accuracy by proposing the ML characteristic variable based on the binary pixel quantification. Generally, relatively low-cost lightweight processors are required for computer systems used in small smart farm IT devices to ensure price competitiveness. Thus, to implement an algorithm to classify trees and free space in an orchard environment, we applied ML technology to operate in a lightweight computer system. In addition, this study was conducted by quantizing or binarizing the input data to facilitate simple implementation of ML feature extraction.

Essentially, there are two main research areas relative to implementing UGV autonomous driving in orchards. The first is fruit tree detection. In other words, this technology identifies and positions fruit trees based on information collected by sensors. The second area is path planning, which allows UGVs to autonomously drive in the free space between two tree lines. Therefore, in order to realize autonomous UGV driving in an orchard, a fruit tree detection technology that must be realized before the path planning technology is proposed. The overall workflow is shown in Figure 1. The image data for ML used in this study comprised 229 consecutive images collected while moving along an alley in an apple orchard in Gyeongsangbuk-do, Korea. These 229 images were acquired with a CMOS image sensor (CIS) camera equipped with a NIR (750–850 nm) pass NIR filter, and the image resolution was 320 × 240 [7]. Among the 229 images, 129 images were used for ML model training. The remaining 100 images were used to verify the performance of the feature extraction and trained ML model. In addition, 129 images were used as training data, and image binarization (data engineering), feature extraction, model training, analysis, and evaluation were performed in order. Then, practical prediction was performed in the final step by inputting 100 raw images.

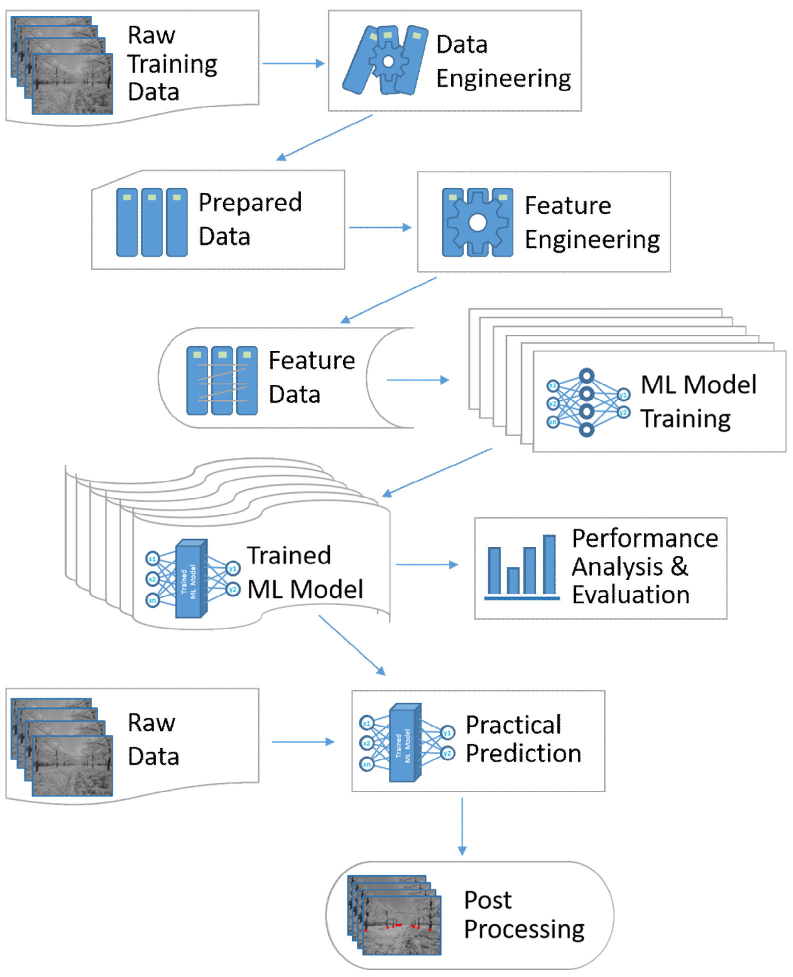


Figure 1. Workflow: feature extraction, training machine learning model, analysis, and practical prediction.

2. Feature Extraction and Model Training

2.1. Local-Lowest Point in Segments

For a lightweight computer system, the input data size is minimized using mono images (320 × 240 pixels). Figure 2a shows an image corresponding to the UGV view angle in an apple orchard, and Figure 2b shows the result of converting the image inside the red dotted line in Figure 2a to quantization or binarization through simple preprocessing. In addition, the red dot in Figure 2b represents the local-lowest points of the segments created in the binary images. Note that local-lowest points can have multiple local-lowest points in a single segment, e.g., points 1 and 2 in the yellow circle (Figure 2c).

As a result, one or more local-lowest points exist in a single segment formed in the binary image. As shown in Figure 2d, the local-lowest extraction method uses the left-handed rule algorithm or right-handed rule algorithm with the top left corner as the starting point, and then the point of interest is moved along the outermost line of the segment and starting point. Originally, this algorithm is known to be effective in escaping a maze made of walls. At the starting point, attach your left or right hand to the wall of the maze and keep moving, you will arrive at the maze exit at the end or return to the original starting position if there is no exit [22]. In Figure 2d, LLP means local-lowest point, LTP means local-top point, and h is the difference in the y-axis between the LTP and LLP. As the location of interest moves along the outermost line, the location information of one or more local-lowest points is stored and used as the local-lowest point data of the segment. As shown in Figure 2b, there can be more than one segment in a single image and one or more local-lowest points in a single segment.

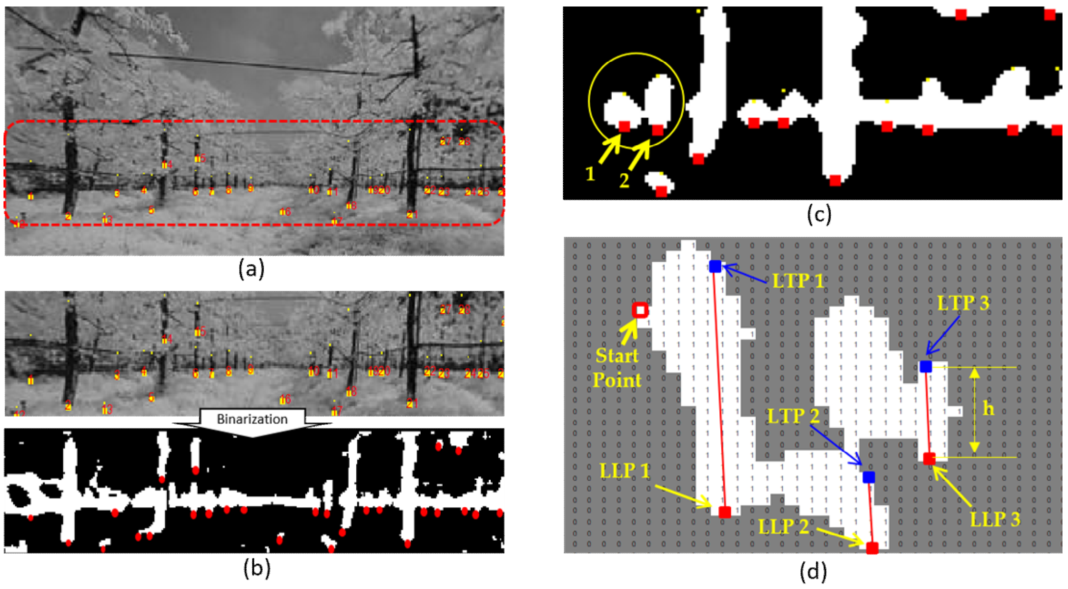


Figure 2. Raw image and local-lowest points: (**a**) apple orchard environment, (**b**) results of converting the image inside the red dotted line to quantization or binarization, in the actual experimental process for this paper, binarization was performed on the entire image, (**c**) points 1 and 2 in the yellow circle, and (**d**) the starting point, local-lowest point (LLP), and local-top point (LTP).

2.2. Predictor Variables (Feature) for Machine Learning

The feature proposed to detect the lower part of an orchard tree using ML is extracted based on the local-lowest points extracted from the binary images. Pixel data in the vicinity, including the local-lowest points, are defined as 33 variables, which are used as feature data for ML model training. The 33 variables include four coordinate variables, five segment-block variables, 12 upper block variables, and 12 lower block variables. The four coordinate variables are defined as local-lowest point coordinates (x1, y1) and upper point markers (x2, y2) in the segment corresponding to the LLP.

Segment-block variables are the value obtained by dividing the number of LLPs in the segment containing the LLP by the number of segment pixels (segment size), and three values derived from the height (i.e., the difference of the y value) and the height between LLP and LTP. As shown in Figure 3, the upper blocks consist of 12 blocks, and features were extracted from each block. The height between LLP and LTP is divided into four, and the width of each block is one-quarter of the height, and three blocks are arranged around the ×1 point. The value obtained by dividing the number of white pixels in each of the 12 blocks by the total number of pixels in the block is defined as a variable of the corresponding block. The lower (bottom) block feature is also defined in the same manner as the upper block feature variables, and the location is positioned below the upper block, as shown in Figure 3.

For example, if there are *i* segments in a single image and j local-lowest points in each segment, the number of extracted feature datasets is $i \times j$. As shown in Figure 2b, it can be expected that the entire image may be a single segment in a binary image. Here, to effectively use the proposed ML features and ML models, it is necessary to preprocess the entire image such that the tree trunk's segments can be segmented effectively from the other segments in the process of converting the input image into to a binary image.

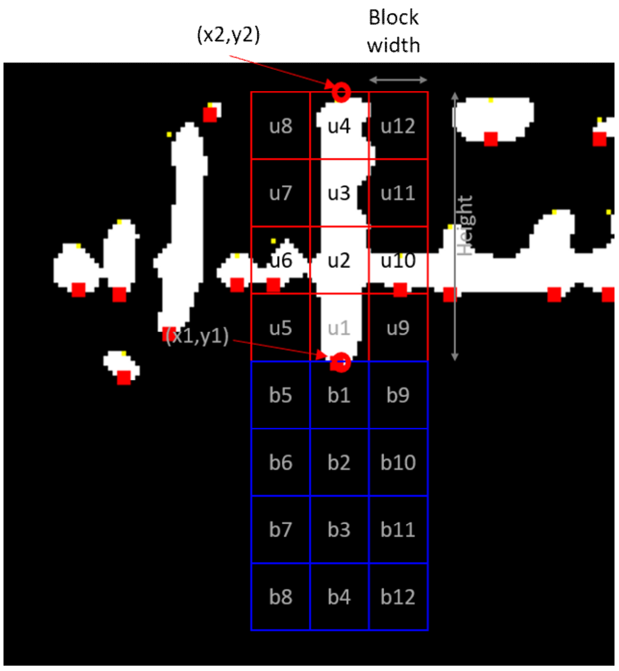


Figure 3. Feature definition (12 upper and 12 lower blocks).

2.3. *Preparing Training Data*

From the 229 apple orchard images, the predictor variables described above were extracted from 129 images, and the final extracted feature datasets were 12,300. Figure 4 shows an example of the local-lowest points corresponding to the feature data extracted from one image. Here, Figure 4a shows the raw grayscale image used as input. As can be seen, 172 local-lowest points are indicated by red dots, and 12 LLPs corresponding to the apple trunk bottom are indicated by blue squares. The ML proposed model should be trained to detect apple trunk bottom points indicated by the blue squares in Figure 4. Thus, for the extracted points in Figure 4, the data at the bottom of the apple trunk are labeled as True, and the other points are labeled as False. As can be seen, the difference between the number of true labeled data and the number of false labeled data is large. This imbalance problem arising must be solved when preparing data for ML.

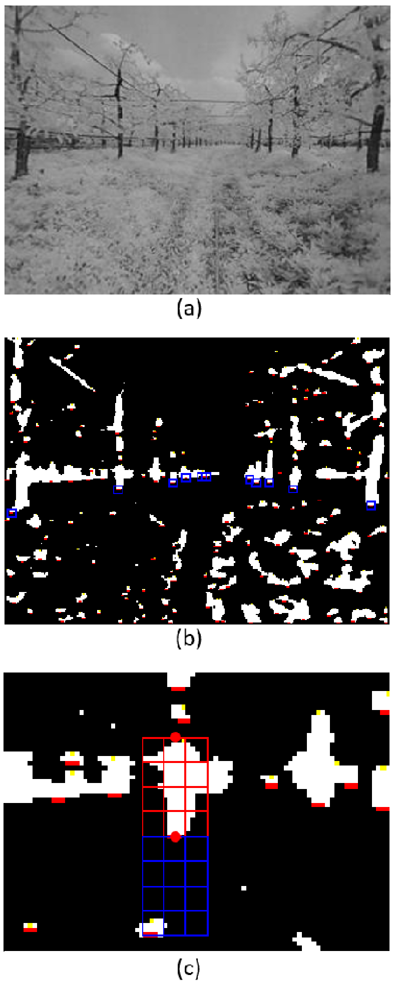


Figure 4. Feature data extraction process: (**a**) raw grayscale image, (**b**) extracted points in binarized image, and (**c**) 24 upper/lower blocks for feature data extraction.

Here, under-sampling was performed using a method similar to one-side selection, which combines Tomek links and condensed nearest neighbor methods to obtain 2563 balanced feature datasets [23,24].

2.4. Model Extraction

To select an ML model suitable for trunk bottom detection, six ML models were trained, and their performance was analyzed. The development language used in this study was MATLAB$^{®®}$, and the six ML models were the decision tree, discriminant, Naïve Bayes, SVM, KNN, and Ensemble provided as functions in MATLAB$^{®®}$. SVM algorithm constructs a hyperplane or hyperplane set in a high-dimensional or infinite-dimensional space for a vector of feature data [25], and then, the k-nearest neighbors (KNN) algorithm classifies points of interest into multiple neighbor votes and assigns points of interest to the most common class of k nearest neighbors. Where k is the user-defined constant [26]. These ML algorithms are suitable for classifying the location of trees, which is the purpose of this study.

From the 2563 balanced datasets, 1800 data were selected to maintain a true-false balance, these data were used to train the six ML models. In this experiment, K-fold cross-validation was performed to train the six models [27,28]. The receiver operating characteristic (ROC) curve, area under the curve (AUC), accuracy, a confusion matrix, recall, and precision were calculated and compared to analyze the performance the trained ML models. Among the various performance evaluation metrics, recall and precision are considered the most important in this study.

3. Validation

The 12,300 feature datasets are imbalanced data that cannot be used to train an ML model. To solve this problem, 2563 true–false balanced feature datasets were prepared by performing under-sampling on the 12,300 feature data. Here, 1800 datasets were used to train the six ML models to obtain the performance results shown in Table 1. As can be seen, the ensemble model demonstrates the best results with high recall and precision are high; thus, the accuracy of this model was the highest.

Table 1. Performance trained machine learning (ML) models, area under the curve (AUC), true positives (TP), false positives (FP), true negative (TN), and false negative (FN).

ML Models	AUC	TP	FP	TN	FN	Recall	Precision	Accuracy
Decision Tree	0.977	912	26	831	31	0.967	0.972	0.968
Discriminant	0.955	877	136	721	66	0.930	0.866	0.888
Naïve Bayes	0.930	936	175	682	7	0.993	0.842	0.899
SVM	0.966	911	81	776	32	0.966	0.918	0.937
KNN	0.970	915	97	760	28	0.970	0.904	0.931
Ensemble	0.994	932	33	824	11	0.988	0.966	0.976

Other performance evaluation experiments were performed on the trained ML models. Here, verification was performed using 763 feature datasets not used in ML model training; thus, the experimental conditions closely represent raw image input collected in the field. The evaluation metrics results are summarized in Table 2. The results indicate a similar trend to the cross-validation results of ML model training. To perform this validation process, we also used the cross-validation function provided as functions in MATLAB®. In addition, the decision tree model obtained the best precision results; however, the ensemble model obtained even performance relative to recall, precision, and accuracy.

Table 2. Performance of trained machine learning (ML) models using new data, true positives (TP), false positives (FP), true negative (TN), and false negative (FN).

ML Models	TP	FP	TN	FN	Recall	Precision	Accuracy
Decision Tree	337	15	360	51	0.869	0.957	0.913
Discriminant	352	52	323	36	0.907	0.871	0.885
Naïve Bayes	383	77	298	5	0.987	0.833	0.893
SVM	366	47	328	22	0.943	0.886	0.910
KNN	346	59	316	42	0.892	0.854	0.868
Ensemble	377	22	353	11	0.972	0.945	0.957

Trunk Bottom Detection

We evaluated the performance of the trained ML models on 763 feature datasets not used for ML model training. This test was performed on 100 new images of an apple orchard environment. The implementation procedure is shown in Figure 5, which shows a block diagram of trunk the bottom point decision process. In this experiment, the apple tree trunk bottom points detected by the trained ML (Figure 6) were displayed on the raw grayscale image used as input.

The first step in detecting the apple tree trunk bottom points was to take 100 orchard environment images as input and create a binary image after preprocessing each image. Here, the local-lowest points were detected in segments created in the binary image, and the LTP was detected based on the LLP. Then, the 33 predictor variables were extracted based on the LLP and LTP coordinates. The extracted predictor variable feature datasets were used as the inputs to the trained ML models to detect trunk bottom points. Here, to detect trunk bottom points in the actual input image, only the trained ensemble model was used. The detected trunk bottom points are shown as red dots with the input image as the background in Figure 5.

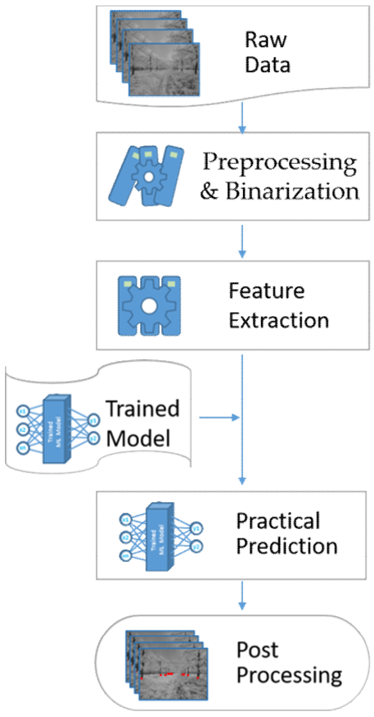


Figure 5. Block diagram of trunk bottom point decision process.

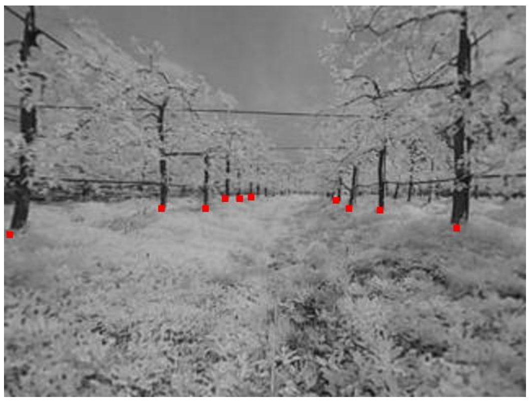


Figure 6. Apple tree trunk bottom points as red dot detected by trained machine learning model.

4. Results and Discussion

We confirmed that apple trunk bottom points could be detected (Figure 6), and approximately 90% detection accuracy was obtained, as shown in Table 2. When the proposed ML feature extraction was performed on an orchard environment image, there were approximately 10 real positive points corresponding to the apple tree trunk bottom and greater than 100 real negative points representing other non-trunk bottom points. To analyze the performance of the six trained ML models, performance evaluation data were collected to calculate and compare the ROC curve, AOU, accuracy, confusion matrix, recall, and precision. The confusion matrix, recall, precision, AOU, and accuracy results are summarized in Table 1. Generally, accuracy is the most common metric used to evaluate a trained model; however, in this study, precision, which represents the ratio of the real trunk bottom among the results predicted as trunk bottom points, and recall, which indicates the ratio of the predicted trunk bottom points among real trunk bottoms, are also important because for ML models that predict free space in an orchard environment, the ability to detect tree trunks is more important than predicting non-tree trunks. In addition, the trained ML model should demonstrate balanced precision and recall performance [29,30].

As shown in Table 1, no significant difference was observed between the recall and AOU values. However, for precision, the decision tree and ensembles models obtained precision values of 0.972 and 0.966, respectively; thus, the corresponding accuracy of these models was also relatively high. The precision and accuracy values of the other ML models were 0.9 or less. Among the ML model trained using 1800 training feature datasets (Table 1), the decision tree and ensemble models demonstrated excellent performance.

An additional verification was conducted after comparing the performance of the ML models trained on 1800 training feature datasets. Here, the performance of the six ML models was evaluated on 763 new datasets not used for ML model training. The results are summarized in Table 2. These feature datasets were not used for ML model training; thus, we expected that the 1800 new data would have similar characteristics to images captured the field. As shown in Table 2, the decision tree model showed the best prediction results (except for the ensemble model).

Theoretically, the decision tree model is known to have several disadvantages, e.g., small fluctuations in the training data significantly affects the final results and a small number of noise data. Figure 7 shows the difference between the decision tree and ensemble models, where Figure 7a shows that prediction is obtained using a single decision tree. Figure 7b shows that the final prediction result is determined by integrating the prediction results of multiple (n-trees) decision tree models. The ensemble model employed in this study uses the decision tree model as a base model and improves accuracy of prediction by integrating the results of several base models using the rule of majority vote or average. This conceptual method is referred to as the ensemble model, and a random forest ensemble model based on the decision tree model was used in this study.

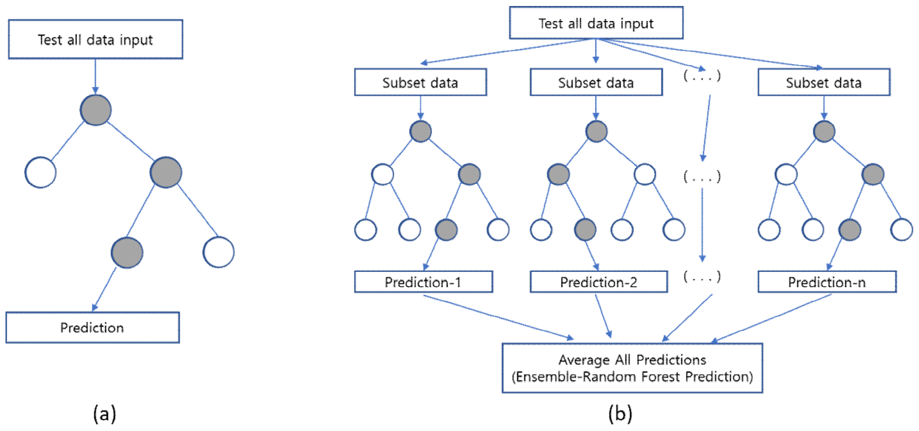


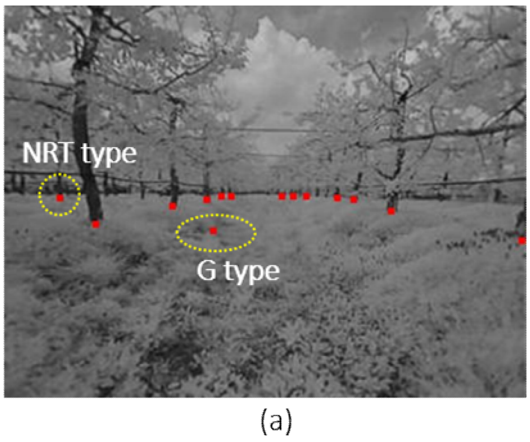
Figure 7. Conceptual difference between decision tree and ensemble models, (**a**) a single decision tree, (**b**) a random forest ensemble model.

The results shown in Table 2 are those obtained in additional verification experiment on 763 new datasets not used for model training. As can be seen, recall, precision, and accuracy gradually decreased with all six trained ML models. Here, the decision tree model obtained the best precision results; however, the ensemble model demonstrate even performance relative to recall, precision, and accuracy.

A test corresponding to the practical prediction (Figure 1, bottom) was performed by inputting 100 gray images. The trunk bottom points detected by the ML classifier using the trained ensemble model are indicated by the red dots in Figure 8a,b. Table 3 shows false positives (FP) and true positives (TP) to confirm the performance of the learned ML model with 100 images not used for model training as inputs. Note that precision is the only performance metric that can be confirmed by TP and FT, and the precision results are shown Table 3. Here, the "strict" item in Table 3 represents the next row tree (NRT) type, grass, or ground (G) type, and branch (B) type points shown in Figure 8 that were classified as FPs. As shown in Table 3, precision is 0.85, which is less than the previous result. This difference can be explained by the fact that the feature data of the actual image have different features from the data used to train the ML model. In addition, the ML model was trained on feature datasets extracted from approximately 100 images; thus, this performance differences may have been caused a lack of sufficient training data.

Table 3. True positives (TP), false positives (FP), and precision with new 100 images.

Standard	TP	FP	Precision
Strict	1120	191	0.85
Liberal	1239	72	0.95



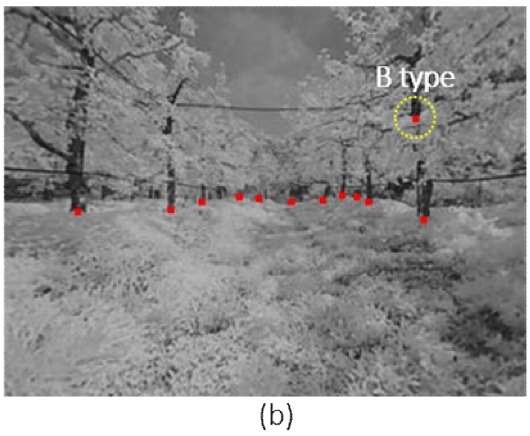


Figure 8. Trunk bottom points, next row tree (NRT) point, grass, or ground (G) point, and branch (B) point: (**a**) NRT type error and G type error, and (**b**) B type error shown as red dots in yellow dotted circles.

On the other hand, cases in which the NRT type error in Figure 8a is reclassified as true positive (TP) are summarized in the "liberal" item in Table 3, and the result was also calculated as Precision. Since the NRT error type is a case where the tree in the next row is classified as positive, the effect on free space prediction is not significant. The detection performance was 95%, which is greater than the strict standard.

In our previous study, the coordinates of the lowest part of the image binarized with the ML function were extracted, and the centerline of the free space between two rows of the apple tree was finally estimated using Naive Bayesian classification [7]. Although in this work we used the same binarized image as the previous study, the ML feature variable based on the binary pixel quantification proposed, can be used to detect the position of the tree with high accuracy. Using the position, we can estimate free space, calculate such as the distance between the tree and the camera and the distance between the trees.

5. Conclusions

In this paper, we have proposed an orchard environment free space recognition technology suitable for developing small-scale agricultural UGV autonomous mobile equipment using a low-cost, lightweight processor. The proposed method extracts feature data required for ML training from segment images by converting low-resolution grayscale (320 × 240 pixels) to binary images to detect trees in orchards. The overall workflow of the proposed technology is summarized as follows. First, feature data were extracted from the input images using the proposed ML feature extraction method. Then, six ML models were trained using the extracted feature data. The optimal trained ML model was selected through a performance analysis. Then, apple tree trunk bottom point detection using the selected trained ML model was evaluated. Finally, performance of the trained ML model was verified on new image data. The experimental results demonstrate that the ensemble model obtained good, even performance to recall, precision, and accuracy.

In summary, the proposed method could detect apple tree trunks in low-resolution grayscale images, and the accuracy was approximately 90%. To evaluate the characteristics of tree trunk bottom point detection in 100 new images (not used for model training), the trained ensemble model was used to confirm performance, and precision of 0.85 was obtain, which is less than the performance obtained in the model training process. There may be various reasons for this result; however, the primary cause appears to be due to the fact that the number of images used for training was only 100, which is insufficient ML model training. Thus, in future, we plan to obtain more diverse orchard environment images and use them to further study the proposed ML feature extraction method for deep learning technologies.

Author Contributions: Conceptualization, H.-K.L. and B.C.; software, H.-K.L. and S.Y.; investigation, H.-K.L. and S.Y.; writing—review and editing, H.-K.L., S.Y. and B.C. All authors have read and agreed to the published version of the manuscript.

Funding: This research received no external funding.

Acknowledgments: This study was supported by the Basic Research Program (20-IT-09, and 20-IT-10-01) through the Daegu Gyeongbuk Institute of Science and Technology (DGIST), and funded by the Ministry of Science, ICT, and Future of Planning of Korea.

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

References

1. Statistica Research Department. Fruit Consumption—Statistics & Facts. Available online: https://www.statista.com/topics/1475/fruit-consumption/ (accessed on 23 September 2020).
2. Maida, J. Global Apple Market 2020–2024. Available online: https://www.businesswire.com/news/home/20191202005540/en/Global-Apple-Market-2020-2024-Evolving-Opportunities-Borton (accessed on 23 September 2020).
3. Shahbandeh, M. Global Fruit Production in 2018, by Selected Variety (in Million Metric Tons). Available online: https://www.statista.com/statistics/264001/worldwide-production-of-fruit-by-variety/ (accessed on 22 September 2020).
4. Production of Apples by Country. Available online: https://www.yara.kr/crop-nutrition/apple-pear/key-facts/world-production/ (accessed on 29 September 2020).
5. Stefas, N.; Bayram, H.; Isler, V. Vision-Based UAV Navigation in orchards. *IFAC-PapersOnLine* **2016**, *49*, 10–15. [CrossRef]
6. Gao, G.; Xiao, K.; Jia, Y. A spraying path planning algorithm based on colour-depth fusion segmentation in peach orchards. *Comput. Electron. Agric.* **2020**, *173*, 105412. [CrossRef]
7. 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]
8. Kim, W.-S.; Lee, D.-H.; Kim, Y.-J.; Kim, T.; Hwang, R.-Y.; Lee, H.-J. Path detection for autonomous traveling in orchards using patch-based CNN. *Comput. Electron. Agric.* **2020**, *175*, 105620. [CrossRef]

9. Cheein, F.A.; Steiner, G.; Paina, G.P.; Carelli, R. Optimized EIF-SLAM algorithm for precision agriculture mapping based on stems detection. *Comput. Electron. Agric.* **2011**, *78*, 195–207. [CrossRef]
10. Chen, X.; Wang, S.; Zhang, B.; Luo, L. Multi-feature fusion tree trunk detection and orchard mobile robot localization using camera/ultrasonic sensors. *Comput. Electron. Agric.* **2018**, *147*, 91–108. [CrossRef]
11. Bargoti, S.; Underwood, J.; Nieto, J.I.; Sukkarieh, S. A Pipeline for Trunk Detection in Trellis Structured Apple Orchards. *J. Field Robot.* **2015**, *32*, 1075–1094. [CrossRef]
12. Bell, J.; MacDonald, B.A.; Ahn, H.S. Row Following in Pergola Structured Orchards by a Monocular Camera Using a Fully Convolutional Neural Network. In Proceedings of the Australasian Conference on Robotics and Automation (ACRA), Sydney, Australia, 11–13 December 2017; pp. 133–140.
13. Zhang, J.; Kantor, G.; Bergerman, M.; Singh, S. Monocular visual navigation of an autonomous vehicle in natural scene corridor-like environments. In Proceedings of the 2012 IEEE/RSJ International Conference on Intelligent Robots and Systems, Algarve, Portugal, 7–12 October 2012; pp. 3659–3666.
14. He, B.; Liu, G.; Ji, Y.; Si, Y.; Gao, R. Auto Recognition of Navigation Path for Harvest Robot Based on Machine Vision. In Proceedings of the International Conference on Computer and Computing Technologies in Agriculture, Zhangjiajie, China, 19–21 October 2011; pp. 138–148.
15. Radcliffe, J.; Cox, J.; Bulanon, D.M. Machine vision for orchard navigation. *Comput. Ind.* **2018**, *98*, 165–171. [CrossRef]
16. Bengio, Y.; Courville, A.; Vincent, P. Representation Learning: A Review and New Perspectives. *IEEE Trans. Pattern Anal. Mach. Intell.* **2013**, *35*, 1798–1828. [CrossRef]
17. Schmidhuber, J. Deep learning in neural networks: An overview. *Neural Netw.* **2015**, *61*, 85–117. [CrossRef]
18. Wang, Y.E.; Wei, G.Y.; Brooks, D. Benchmarking TPU, GPU, and CPU platforms for deep learning. *arXiv* **2019**, arXiv:1907.10701.
19. Wang, Y.; Wang, Q.; Shi, S.; He, X.; Tang, Z.; Zhao, K.; Chu, X. Benchmarking the performance and power of AI accelerators for AI training. *arXiv* **2019**, arXiv:1909.06842.
20. Dai, W.; Berleant, D. Benchmarking Contemporary Deep Learning Hardware and Frameworks: A Survey of Qualitative Metrics. In Proceedings of the 2019 IEEE First International Conference on Cognitive Machine Intelligence (CogMI), Los Angeles, CA, USA, 12–14 December 2019; pp. 148–155.
21. Kang, H.; Zhou, H.; Chen, C. Visual Perception and Modeling for Autonomous Apple Harvesting. *IEEE Access* **2020**, *8*, 62151–62163. [CrossRef]
22. Kumar, R.; Jitoko, P.; Kumar, S.; Pillay, K.; Prakash, P.; Sagar, A.; Singh, R.; Mehta, U. Maze Solving Robot with Automated Obstacle Avoidance. *Procedia Comput. Sci.* **2017**, *105*, 57–61. [CrossRef]
23. Bekkar, M.; Alitouche, T.A. Imbalanced data learning approaches review. *Int. J. Data Min. Knowl. Manag. Process* **2013**, *3*, 15. [CrossRef]
24. Batista, G.E.; Prati, R.C.; Monard, M.C. A study of the behavior of several methods for balancing machine learning training data. *ACM SIGKDD Explor. Newsl.* **2004**, *6*, 20–29. [CrossRef]
25. Cortes, C.; Vapnik, V. Support-vector networks. *Mach. Learn.* **1995**, *20*, 273–297. [CrossRef]
26. Altman, N.S. An Introduction to Kernel and Nearest-Neighbor Nonparametric Regression. *Am. Stat.* **1992**, *46*, 175–185. [CrossRef]
27. Rodríguez, J.D.; Perez, A.; Lozano, J.A. Sensitivity Analysis of k-Fold Cross Validation in Prediction Error Estimation. *IEEE Trans. Pattern Anal. Mach. Intell.* **2010**, *32*, 569–575. [CrossRef]
28. Bengio, Y.; Grandvalet, Y. No unbiased estimator of the variance of k-fold cross-validation. *J. Mach. Learn. Res.* **2004**, *5*, 1089–1105.
29. Saito, T.; Rehmsmeier, M. The Precision-Recall Plot Is More Informative than the ROC Plot When Evaluating Binary Classifiers on Imbalanced Datasets. *PLoS ONE* **2015**, *10*, e0118432. [CrossRef] [PubMed]
30. Davis, J.; Goadrich, M. The relationship between Precision-Recall and ROC curves. In Proceedings of the 23rd International Conference on Machine learning, Pittsburgh, PA, USA, 25–29 June 2006; pp. 233–240.