AgriEngineering

MDPI

Article

Development of a Cross-Platform Mobile Application for Fruit Yield Estimation

Brandon Duncan [1,†], Duke M. Bulanon [2,*,†], Joseph Ichiro Bulanon [1,†] and Josh Nelson [2,†]

[1] Department of Mathematics and Computer Science, Northwest Nazarene University, 623 S University Blvd, Nampa, ID 83686, USA
[2] Department of Physics and Engineering, Northwest Nazarene University, 623 S University Blvd, Nampa, ID 83686, USA
* Correspondence: dbulanon@nnu.edu; Tel.: +1-208-467-8047
† These authors contributed equally to this work.

Abstract: The Fruit Harvest Helper, a mobile application developed by Northwest Nazarene University's (NNU) Robotics Vision Lab, aims to assist farmers in estimating fruit yield for apple orchards. Currently, farmers manually estimate the fruit yield for an orchard, which is a laborious task. The Fruit Harvest Helper seeks to simplify their process by detecting apples on images of apple trees. Once the number of apples is detected, a correlation can then be applied to this value to obtain a usable yield estimate for an apple tree. While prior research efforts at NNU concentrated on developing an iOS app for blossom detection, this current research aims to adapt that smart farming application for apple detection across multiple platforms, iOS and Android. Borrowing ideas from the former iOS app, the new application was designed with an intuitive user interface that is easy for farmers to use, allowing for quick image selection and processing. Unlike before, the adapted app uses a color ratio-based image-segmentation algorithm written in C++ to detect apples. This algorithm detects apples within apple tree images that farmers select for processing by using OpenCV functions and C++ code. The results of testing the algorithm on a dataset of images indicate an 8.52% Mean Absolute Percentage Error (MAPE) and a Pearson correlation coefficient of 0.6 between detected and actual apples on the trees. These findings were obtained by evaluating the images from both the east and west sides of the trees, which was the best method to reduce the error of this algorithm. The algorithm's processing time was tested for Android and iOS, yielding an average performance of 1.16 s on Android and 0.14 s on iOS. Although the Fruit Harvest Helper shows promise, there are many opportunities for improvement. These opportunities include exploring alternative machine-learning approaches for apple detection, conducting real-world testing without any human assistance, and expanding the app to detect various types of fruit. The Fruit Harvest Helper mobile application is among the many mobile applications contributing to precision agriculture. The app is nearing readiness for farmers to use for the purpose of yield monitoring and farm management within Pink Lady apple orchards.

Keywords: precision agriculture; farm automation; farm management; apple detection; fruit detection; agricultural technology; Pink Lady; Pink Lady apples; smart agriculture; Internet of Things in agriculture

check for updates

Citation: Duncan, B.; Bulanon, D.M.; Bulanon, J.I.; Nelson, J. Development of a Cross-Platform Mobile Application for Fruit Yield Estimation. *AgriEngineering* **2024**, *6*, 1807–1826. https://doi.org/10.3390/agriengineering6020105

Academic Editor: Lin Wei

Received: 3 April 2024
Revised: 12 June 2024
Accepted: 13 June 2024
Published: 19 June 2024

Copyright: © 2024 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 (https://creativecommons.org/licenses/by/4.0/).

1. Introduction

In the current farming practices, precision agriculture is becoming more critical since new technologies are helping farmers reduce their costs, increase their profits, and even produce more crops. According to a study conducted by the Islamia University of Bahawalpur in Pakistan, when comparing cotton farms that adopted technology with those that did not, the adopters saw up to a 22% higher crop yield per acre and a 27% higher profit per acre [1]. These findings suggest that farms that utilize precision agriculture technologies will be significantly more sustainable and will be able to outperform their competitors. Precision

agriculture refers to any farming practice that uses technology, data analysis, or spatial variability-management techniques to optimize crop production efficiency, sustainability, and profitability. Mobile applications are among the many precision agriculture tools being created for use in agriculture. As shown in a survey by Oteyo et al., 2021, mobile applications for intelligent agriculture span various agricultural areas, including irrigation management, data collection, farm management, crop health, and yield monitoring. Each of these mobile applications was made to solve a specific agricultural task, helping farms meet demands and outperform their competitors [2].

Yield monitoring is an area of precision agriculture that focuses on measuring and analyzing the quantity of crops produced by a field to provide valuable information to farmers. The information gained from yield monitoring helps farmers increase their productivity and inform their decision-making. As found by a recent study, yield monitoring systems that farmers have adopted provide row crop farmers with valuable insights into the performance of different areas of their fields. However, these methods, usually based on measuring the weight of harvested crops, are most successful for row crops like corn and soybeans. They are less successful for specialty crops. Specialty crops are widespread and consist of fruits, vegetables, and nuts. These weight methods do not work for specialty crops because various challenges arise due to their geometric parameters [3]. This issue has shed more light on the need for yield monitoring systems designed for specialty crops. A book chapter published in 2022 discovered that most yield monitoring technologies do not specialize in fruit crops, which still need more thorough analysis [4].

While lacking thorough analysis, some fruit yield monitoring technologies have already been developed. According to a research paper released in 2013, their team of researchers developed an Android smartphone application that uses a Java fruit-counting algorithm to estimate citrus yield. The researchers even claim that the application can assist farmers in assessing the fruit yield for an individual citrus tree [5]. There is also another Android mobile application for apple yield estimation, as proposed by Qian et al. (2017), which uses an artificial neural network to detect Fuji apples accurately. As they mentioned, their application only works on Android devices, and for the Fuji apple variety, a variety known to have deeper colors of red and very few green patches [6]. So far, research has yet to be done to create a fruit yield-estimation mobile application for apples of the Pink Lady variety, which have distinct characteristics compared to Fuji apples. Pink Lady apples are known to have more of a pink color than red, like some other varieties. Also, Pink Lady apples tend to have more green and yellow colors and along with other differences. Given the differences in color, shape, size, and texture between Fuji and Pink Lady apples, utilizing a Fuji apple-detection app would not be as effective for detecting Pink Lady apples as an app specifically tailored to this variety. Seeing there was no fruit yield-estimation app specifically for Pink Lady apples, this urged Northwest Nazarene University's (NNU) Robotics Vision Lab to look further into the task of fruit yield estimation.

Traditional fruit farmers face the obstacle of manually counting fruits on trees during harvesting season to estimate the fruit yield for the entire orchard. Currently, the fruit yield-estimation process consists of selecting a group of trees, manually counting the fruits on each tree, calculating the average fruits found on a tree, and then applying this average to a large group of trees in the orchard or the entire orchard [7]. The issue with this manual method is that it is time-consuming, labor-intensive, and inaccurate, which are all things that technology can solve. As mentioned earlier, there is also a need for a mobile application that helps estimate fruit yield in Pink Lady apple orchards. Recognizing this gap, the researchers at NNU's Robotics Vision Lab invented a solution by developing a mobile application capable of detecting apples on Pink Lady trees to facilitate fruit yield estimation.

In other words, the main objective of this research was to design a mobile application that could detect Pink Lady apples on trees, helping to automate the fruit yield-estimation task for farmers. Or rather, the study aimed to adapt an existing blossom-detection iOS application into a cross-platform tool. To elaborate, this meant that the NNU team tran-

sitioned from a blossom-counting fruit yield-estimation approach to an apple-counting approach while ensuring accessibility to farmers using iOS and Android devices [8]. The last objective of this research was to validate the newly developed algorithm's effectiveness through mathematical analysis and by establishing a correlation between the number of detected apples and the actual number of apples. A similar validation process was conducted by Bargoti and Underwood, who also compared the apples detected on the images to the count of physical apples in order to obtain a correlation between the two variables [9].

Considering those objectives, the scope of this research was exclusively to develop a functional mobile application for iOS and Android platforms and to assess whether the application was a technology that was viable for farmers to adopt. At Fruit Harvest Helper's current stage, the mobile application has yet to be released for public use since the algorithm needs further testing and improvement to work in the real world. However, the potential benefits of the Fruit Harvest Helper are significant for farmers. By helping automate the central aspect of the fruit yield-estimation task, this yield monitoring technology would be much more efficient than manually counting apples, saving farmers hours of valuable time and reducing the number of fruit pickers and labor costs. Currently, no mobile application for assisting farmers with estimating fruit yield in Pink Lady apple orchards exists, making the Fruit Harvest Helper a tool that farmers greatly need. Improving this mobile application slightly and publishing it would impact the farming industry. All farmers with Pink Lady apple orchards would have an intelligent agriculture tool for automating most of the fruit yield estimation. Also, with improvements made to the Pink Lady-detection algorithm, this precision agriculture system could surpass the accuracy of manually counting due to the inaccuracies of the humans that do the counting. Although the Fruit Harvest Helper is not at the stage where it is viable for apple yield estimation in Pink Lady orchards, advancements to its detection algorithm would enable farmers to more efficiently and possibly more accurately estimate fruit yield, helping farmers anticipate labor and farm-management costs for the upcoming harvest.

2. Materials and Methods

2.1. App Conceptualization and Development Process

The concept of the Fruit Harvest Helper was thought of after researching and identifying what agricultural research tasks needed to be solved. One of the studies found through this research was a study conducted in 2015 by Karkhile and Ghuge, which showcased a multi-purpose precision agriculture tool. The precision agriculture tool was an Android application that informed farmers to help them increase their profits. It gave them information such as the current weather in their area, news updates, and recent market trade deals. This app demonstrated how mobile applications could be used for precision agriculture and clearly explained how mobile computing can help farmers in this new age [10]. As a logical progression of reading about this Android application, NNU's Robotics Vision Lab became interested in developing its own mobile application for precision agriculture.

Doing more research led to the discovery that fruit yield monitoring technologies were an area that needed to be studied more. Some mobile apps for fruit yield estimation did exist, including apps that detected fruits like citrus, kiwis, and apples. Similarly, the proposed mobile application, the Fruit Harvest Helper, is also used to detect apples and estimate fruit yield. However, the Fruit Harvest Helper detects a specific variety of apples that still need to be researched. The mobile application for kiwi detection, called the KiwiDetector, was an Android application that used a few different deep-learning approaches to quickly and accurately detect kiwis for kiwi yield estimation [11]. Another existing application was the app developed by Qian et al. that uses an artificial neural network to accurately detect Fuji apples for yield estimation, as discussed earlier [6]. The main advantage of Fruit Harvest Helper over this existing application is that it has a new algorithm designed specifically for Pink Lady apple detection. Pink Lady apples are unlike Fuji apples in color, shape, size, and texture, requiring a specific algorithm for successful

detection. The primary difference is in color, as Pink Lady apples have more pink and less red than some other varieties of apples, as well as more yellows and greens.

Furthermore, the Fuji-detection app was only for Android and lacked an iOS counterpart to help reach more farmers. While any fruit yield-estimation mobile app could be adapted for Pink Lady apples, a novel algorithm would need to be created for this variety of apples as Pink Lady apples have their own set of characteristics. Developing an entire algorithm for Pink Lady apples would require much work. Learning about existing mobile applications helped narrow the focus when deciding on what specific research would be done. No research on mobile applications for the Pink Lady variety was found. In noticing this, it was clear that predicting the apple yield for Pink Lady apples with a mobile device was an agricultural task that needed attention.

Since a mobile device capable of predicting apple yield for Pink Lady apples is something farmers need, NNU set out to develop a mobile application for this purpose. In order to be a convenient tool for farmers, this mobile application will eventually need an accurate enough algorithm and will have to be publicly available for them to download. A relatively simple image-processing algorithm was chosen for this study to serve as just a starting place for Pink Lady apple detection. The sole focus of this research was developing an end-to-end mobile application that could be improved upon. With the transition to a more advanced detection algorithm in the future, the Fruit Harvest Helper will be able to save farmers time and costs, which is very important to farmers. The Fruit Harvest Helper will also increase farmers' productivity by making fruit yield-estimation magnitudes faster and could provide more accurate estimations than manual methods, which are prone to error. More accurate estimations would improve the quality of decision-making in agricultural practices since there would be better information to work with. Farmers need a Pink Lady apple yield-estimation tool, meaning the Fruit Harvest Helper will help them. Given there is a practical use of this innovation, farmers would be interested in downloading and adopting this precision agriculture technology to use in their orchards.

The last step before development began was picking the platform(s) for creating the mobile app. A study on mobile applications for agriculture shows that most of the apps generated come from iOS and Android platforms. After all, these are the two most commonly used platforms, so this makes sense. Some of the applications this study included were farming apps for business and financial data, pests and diseases, agricultural machinery, and farm management. According to the survey, in 2016, there were 91 iOS and 69 Android mobile applications for farm management; the category fruit yield estimation falls under [12]. Since iOS was the more popular choice for mobile apps in the farm management category, NNU's Robotics Vision Lab initially developed a mobile application only for iOS devices in a previous research endeavor. Another reason NNU only made an iOS app was that creating an application for one platform was easier. This first mobile application for iOS was designed with a blossom-detection algorithm to help estimate the apple yield for a Pink Lady orchard. Once this mobile application was functional, many improvements still needed to be made to the blossom-detection method. Due to this, the researchers decided to switch to apple detection before coming back and making various improvements to the blossom-detection algorithm [8]. For the new app, the Fruit Harvest Helper, the researchers also made the change of developing the Android platform to reach more farmers needing a tool that automates the main part of the fruit yield-estimation task. According to a study conducted by the University of Salamanca, Samsung smartphones led in worldwide sales for both years studied, 2012 and 2013 [13]. Given that more farmers would likely have an Android application, the scope of the research changed, leading the researchers to develop an Android mobile application, too, in this study. Using React Native, the researchers began developing iOS and Android mobile applications. View Figure 1 to see this application being used.

In the development phase of the Fruit Harvest Helper app, the user interface was made using the React Native framework since this was a convenient option. React Native is a framework built directly on top of the JavaScript programming language used to create

cross-platform mobile applications. As the front end took shape, the main functionality, or the back end, began to be developed for the application, a task executed separately for the iOS and Android platforms. The iOS application relied on C++ code for the back end, whereas the Android counterpart used a combination of Java and C++. Despite these differences, both platforms shared the same C++ apple-detection algorithm comprising a wide range of functions from OpenCV as well as unique classification equations and C++ code. To provide context, OpenCV is a widely used machine vision library that provides the necessary functionality to create the image processing component of the application. The performance of the library's functions and the entire algorithm was reliable across both platforms. The researchers used the latest version of Visual Studio Code, Android Studio, and Xcode for the development phase. Android Studio and Xcode are the Android and iOS applications for mobile development that allow for all things related to mobile development.



Figure 1. The Fruit Harvest Helper Android mobile application is being used to detect apples on a real apple tree.

Once the development phase for the mobile application was complete, the researchers needed to test the app to evaluate the efficacy of the Fruit Harvest Helper's functionality. The researchers collected a dataset of images utilizing 40 Pink Lady apple trees from Symm's Fruit Ranch. The device used to capture these images was a Samsung Galaxy Tab S7 with a dual-lens rear camera and an image resolution of 13 megapixels. One image was taken from each side of an apple tree unless it was obstructed due to circumstances out of the researchers' control. As a result of this process, 73 images of mature Pink Lady apple trees were gathered a few weeks before the harvesting season. The researchers used these images to test the algorithm's detection capabilities and performance. The first part of this testing phase aimed to verify the application's functionality by evaluating the apple-detection algorithm outside the application in a separate environment. Testing the algorithm outside the mobile application was much quicker since the researchers did not have to wait for the app to build each time, allowing them to fine-tune the algorithm

and evaluate its detection abilities in a fraction of the time. After testing the algorithm's detection capabilities, it was determined that it was not effective enough for farmers to use for apple yield estimation and still required improvements. Next, the other part of the testing phase was conducted, utilizing five images from the dataset to test the algorithm's performance on each platform. A timer was used to report the processing time of the algorithm when running for each of the five images selected. The average processing time was calculated for iOS and Android, indicating that both platforms processed images reasonably quickly, but the iOS application performed much better. What could have been included in the testing phase of the mobile application was the comparison between a fruit yield estimation for the entire orchard found with the assistance of the app and the actual fruit yield of the Pink Lady apple orchard reported by the farmers. Once the algorithm is improved in the future, this will need to be assessed to determine the app's efficacy in assisting with the fruit yield-estimation task.

2.2. Algorithm Development

The novel Pink Lady apple-detection algorithm created for the Fruit Harvest Helper was a color ratio-based image-segmentation algorithm. Many machine-learning algorithms could detect apples in images of apple trees, but this was the one the researchers settled on. Studies in the past have explored many apple-detection algorithms, such as a color index-based image-segmentation method presented in 2022. While some algorithms are better than others, each has its own set of strengths and limitations [14].

Like any algorithm, the unique color ratio-based algorithm developed from the ground up by the researchers can be considered a series of operations. The series of operations for this algorithm has three main aspects: pre-processing, processing, and post-processing. Data pre-processing is the stage that prepares the data for processing, processing is the actual modification of the data, and post-processing is when the processed data is cleaned up to make it more effective. Before any pre-processing could be done, the algorithm received an image that the farmer captured or selected. Once the image was obtained, the algorithm executed a series of steps to detect Pink Lady apple clusters in an apple tree image. These high-level steps are shown in Figure 2. At the beginning of the pre-processing stage of the algorithm steps, the researchers made a copy of the input image to preserve the original image when it needed to be displayed. All of the following calculations were done to the copied image or on information that was derived from the copied image. An OpenCV C++ function was used to copy the original image. In its entirety, this algorithm used a variety of OpenCV functions as almost every step uses OpenCV functions in combination with the unique C++ code written by the researchers. The OpenCV library represented the apple tree image as a multi-dimensional matrix of pixels containing a dimension for rows, columns, and color channels. Now that the image was copied, the next step consisted of splitting the image into three color channels: Blue, Green, and Red. Blue, Green, Red, or BGR is the format in which OpenCV stores images. The image was separated into color channels because this would eventually help the researchers obtain more information as it allowed each color channel to be analyzed individually. More specifically, separating the image into color channels would be crucial for deriving the values used in the algorithm's classification equations, which were the critical component of the processing stage of the algorithm.

Now that the image was split into color channels, further preparation was needed before the values used in the classification equations could be calculated. This preparation involved converting a duplicate version of the copied image and all three color channels to grayscale. A duplicate version of the copied image was made since an unaltered image would be needed to draw bounding boxes in post-processing. Since each color channel and the entire image were represented in the grayscale format at this point, the intensity of an individual color channel could be easily compared to the intensity of the whole image, resulting in a ratio indicating the intensity of the color in the apple tree image. The researchers obtained a red, green, and blue color ratio by comparing each grayscale color channel to

the full grayscale image. Once again, these ratios indicated how much of a particular color was in the image the farmer selected. These ratios formed the heart of the apple-detection algorithm since they were the most crucial aspect of the color ratio classification equations that classified every pixel as an apple or non-apple. Equations (1) and (2) are the Pink Lady apple equations that the researchers created from scratch for classifying pixels:

$$pixelEval1 = 0.4 * redRatio - 0.2 * greenRatio - 0.3 * blueRatio \quad (1)$$

$$pixelEval2 = 0.35 * redRatio - 0.15 * greenRatio - 0.3 * blueRatio - 0.02 \quad (2)$$

In calculating the values for the classification equations, pre-processing was concluded, and processing began. These equations were used for the processing stage of the algorithm in which image pixels were classified. Each pixel was iterated through and classified based on the two equations. Pixels classified as apple pixels were modified to be white pixels, and those classified as non-apple pixels were altered to black pixels. The ratios within these classification equations were weighted to give specific colors, like red, more influence in determining if pixels were apple pixels. In essence, if there was significantly more red in a pixel than blue and green combined, the equations classified a pixel as an apple pixel; otherwise, it was classified as a non-apple pixel. The researchers empirically selected and fine-tuned these weights or hyperparameters through much testing to achieve the optimal detection of apples across the image dataset. As pointed out in a research article discussing fine-tuning hyperparameters, a machine-learning algorithm's hyperparameters depend on the dataset used for fine-tuning them [15]. The research presented in this article suggests that the hyperparameters for this image-segmentation algorithm would likely function best with the dataset the researchers collected. Further testing with data from multiple Pink Lady apple orchards would need to be conducted to determine the algorithm's effectiveness across different datasets.

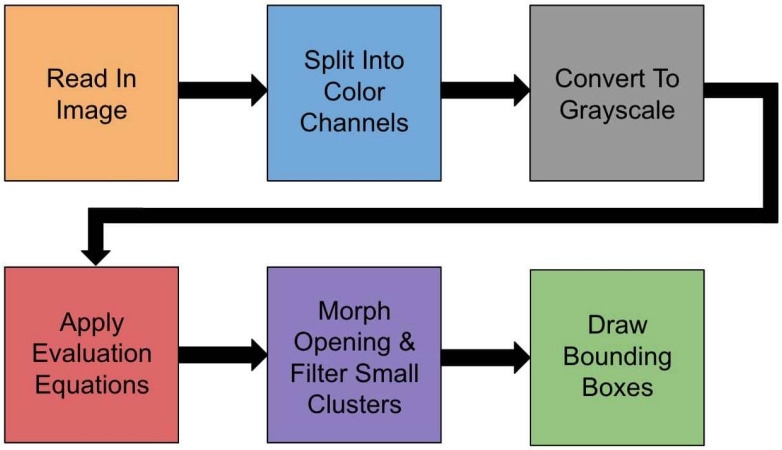


Figure 2. A process flow diagram illustrating the sequence of steps within the apple-detection algorithm. Only the key steps are shown.

After the pixels were classified during processing, the image-segmentation results still needed to be refined and displayed. The process of refining an algorithm's results can be described as post-processing. The post-processing operations of the Fruit Harvest Helper's algorithm were applied to the binary image generated from the processing stage. A binary image is an image that contains only two colors, typically black and white. The white pixels in the binary image indicated pixels the algorithm marked as apples, whereas the black pixels were what the algorithm deemed as non-apple pixels. The first post-processing operation performed on the binary image was a morphological opening with a structuring element. According to "Digital Image Processing", a morphological opening is a widespread image processing technique that consists of an erosion operation on a region of pixels followed by a dilation operation performed on that result [16]. Erosion and

dilation are the shrinking and expanding of an area of pixels, also known as a cluster. The purpose of a morphological opening is that it is a filtration operation that filters out small regions surrounding identified clusters; in this case, the clusters represent apples [17].

In the next step, clusters below a minimum size threshold were filtered out, removing clusters likely too small to truly be apples. Using this minimum cluster size threshold reduced false positives because most clusters under the threshold tended to be falsely classified as apple clusters. This minimum size threshold was a hyperparameter set and fine-tuned to improve the algorithm's detection. Finally, the remaining contours in the binary image represented only the significant apple clusters above the size threshold. Since these clusters or contours were substantial, they could be drawn onto the copied image. These contours represented where the algorithm "thought" there were apples based on the previously described steps. In the drawing step, green bounding boxes were drawn onto the apple tree image, highlighting regions identified as apple clusters for visualization. The finalized processed image was ready to be displayed to the user after this step was complete.

Initially, the image-segmentation algorithm was developed with the latest version of MatLab. The algorithm was initially developed in MatLab to facilitate the creation of the Pink Lady apple classification equations. Since MatLab is built for math-based tasks, creating the classification equations from scratch here was convenient. Later, the MatLab algorithm was converted to the OpenCV C++ algorithm just described to put it into a language compatible with the Fruit Harvest Helper mobile app. Utilizing C++ enabled the detection algorithm to work with iOS and Android platforms.

In order to gather the necessary data for algorithm development and testing, the researchers took pictures of apple trees at Symm's Fruit Ranch in October 2023. This orchard was the only apple orchard from which data was collected and, therefore, the only field used for testing. An Android device, more specifically a Samsung Galaxy Tab S7, was used to capture images from both the west-facing and east-facing sides of the Pink Lady apple trees. The rear camera of the Samsung Galaxy Tab S7 was used, a dual-lens camera that captures images at a resolution of 13 megapixels. The researcher who captured these images used the default zoom and stood approximately 2.5 m from the trees. No filters were used to help manage variations in light intensity since the researchers needed to mimic the process a farmer would use to take images. To clarify, the images captured for data were precisely the same as those a farmer would take when using the mobile app to assist with fruit yield estimation.

In total, 40 trees were used in the data collection process. These trees were selected as these were all of the trees NNU was permitted to do research on by Symm's Fruit Ranch. All of these trees belonged to a single row of the Pink Lady apple tree orchard and sat next to each other. Unfortunately, only 39 east- and 34 west-side images were collected due to unforeseen circumstances. Considering this, the dataset consisted of 73 apple tree images altogether. Although challenges existed with image collection, the researchers still had adequate information or photos to refine the algorithm through tests. This fine-tuning significantly improved how well the algorithm could detect apples based on color.

At the beginning of the testing process, the hyperparameters of the detection algorithm would get tweaked, the researchers would visually assess its ability to detect apples across the dataset, and then the hyperparameters would get tweaked again. This tweaking of the hyperparameters helped fine-tune the algorithm to identify apple clusters better. Once the fine-tuning stage of testing was complete, the next step could be carried out: assessing the ability of the Pink Lady apple-detection algorithm.

The evaluation method for assessing the algorithm's detection abilities involved calculating the Mean Absolute Percentage Error (MAPE). In conducting testing to find multiple MAPE values, the researchers discovered that the algorithm detects apples best when two photos of the apple tree are taken; one from each side of the tree. So, for a farmer to get the best results with this app, a farmer must take a picture from both sides of an apple tree and combine the number of detected apples, not apple clusters, from each image.

After the detected apples from both images are combined, a correlation between detected and actual apples derived in this study can then be applied to get an estimation for the fruit yield on that tree. A farmer would then be able to repeat this process on as many trees as they like to obtain an average that they can apply to get a fruit yield estimation for the entire orchard.

The researchers calculated the MAPE value associated with using a photo from each side of an apple tree by utilizing data from 34 apple trees within the dataset, each containing both east and west images. While 40 trees were used for the apple tree image dataset, only 34 trees contained images from both sides. For each tree, the researchers had to evaluate the two original and two processed apple tree images, or four images in total. An original apple tree image was assessed by manually spotting the number of apples on the tree of interest, serving as a ground truth number against which the algorithm's detections could be compared. On the other hand, a processed apple tree image was evaluated by manually counting the number of apples detected by the algorithm on the tree of interest, requiring the examination of the bounding boxes. The following process that was carried out on these images is demonstrated by a pair of original and processed images in Figure 3.



(**a**) An original image taken by the mobile app with the tree of interest outlined to distinguish it from the surrounding trees. The apples on the tree were marked during the counting step to get the ground truth number of apples.

(**b**) The processed version of the original image. Instead of counting the ground truth, this processed image was used to count the number of apples detected by the image-segmentation algorithm that the researchers developed.

Figure 3. An original and processed image after they were used for counting to test the algorithm.

Throughout the evaluation process what was kept the same across both images was that only apples within the bounds of the tree of interest were counted. The latest version of Microsoft Paint was used to draw a perimeter around the tree of interest to create bounds for the tree, delineating it from neighboring trees and other elements. Since the apple trees in Symm's Fruit Ranch are so close together, drawing a perimeter around the tree of interest was necessary to properly evaluate the algorithm's ability. Next, the researchers counted the apples inside the perimeter within the original and processed images. When counting the apples for a processed image, the researchers assumed the algorithm detected each apple within the green bounding boxes. Although these bounding boxes represent an apple cluster and not an apple, this method seemed fair considering the detection algorithms' tendency to spot hard-to-find apples. For the original and processed image from one side of an apple tree, this procedure carried out on the images is illustrated in Figure 3 as mentioned. However, as mentioned, this was done for both sides of each apple tree to get the MAPE value. The entire process of examining four images was repeated for each of

the 34 apple trees. Finding the counts for the original and processed images allowed the researchers to compare the human observations within the image or the ground truth for the tree of interest with the apples that the algorithm detected. In finding these values, the researchers could analyze the algorithm's ability by calculating the MAPE and finding a correlation between detected and actual apples.

Now that the number of detected apples for each of the studied trees was found, these could be compared to the actual number of apples on a tree. By comparing these two variables, a correlation or relationship between detected apples and actual apples would be the result. To obtain the actual number of apples on all 40 of the trees studied, the researchers had to manually count the number of apples on each tree. This manual count was done about a month before the apples were harvested. While it would have been most accurate to harvest and count the apples immediately after collecting the images of the apple trees, this was not the strategy used. Instead, the researchers counted the apples on one of the trees by starting at the top of the tree and working their way down, counting all of the apples on a single branch before moving on to the next branch. A clicker was used by the researchers to keep the count so they could focus on correctly counting the apples to the best of their ability. While the manual counts were reasonably accurate, the values obtained were just indices, serving as relative indicators rather than exact measures. The counts distinguished the studied trees from one another, providing the researchers with unique values for each tree that were not too inaccurate. Having a distinct apple count for each tree was great for deriving a correlation. However, this method could be improved in the future when testing the next algorithm, as the researchers could obtain an apple count by harvesting the apples on the studied trees.

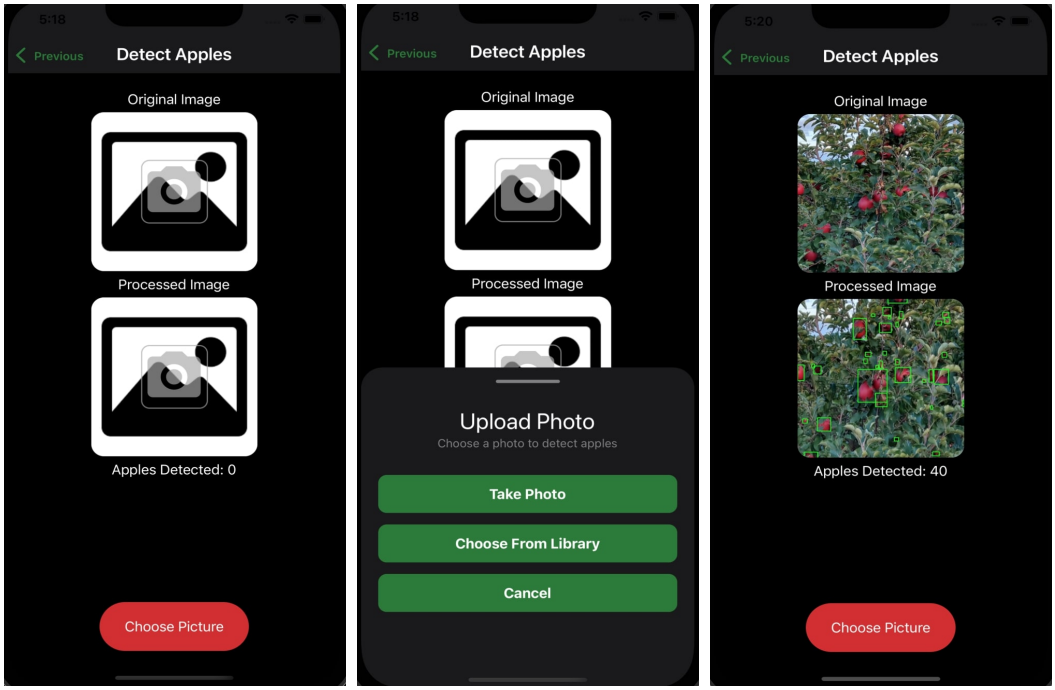
2.3. Front-End Development

According to a study by Brambilla et al., when designing any application, two large-picture things should be considered: the front end and the back end. An application's front end is responsible for its appearance and feel. Having a well-thought-out user interface (UI) ensures that the user of the technology has a positive experience [18]. Furthermore, an excellent front end should keep the users engaged and satisfied by serving its purpose and nothing else. In the case of the Fruit Harvest Helper app, creating a UI tailored to farmers was essential for keeping them satisfied as they used the app. Tailoring the app to them involved making the design simple and easy to use since farmers likely prefer functionality over complexity. As depicted in Figure 4, the layout of the Fruit Harvest Helper mobile application was straightforward and lacked any unnecessary features or elements.

To develop the front end, the researchers used the React Native framework, which takes an approach to simultaneously developing both iOS and Android platforms. Designing both applications at the same time was much more efficient than individually building out the front ends. Considering what design farmers would want, they first planned the information layout for display. By incorporating the same elements from the existing blossom-detection app, created by the NNU Robotics Vision Lab previously, the process of designing the UI for this new application became easy.

During the design phase of the UI, the researchers added four main components to display to the farmers by using React Native. Firstly, showing the selected or taken image of a Pink Lady apple tree helps farmers confirm their image choice, providing a sense of direction for them. The processed image, generated through the process of the algorithm analysis of the photo, offers farmers valuable insights into the apple-detection results. Therefore, it was also chosen for display. These two elements are displayed in vertically aligned boxes with the regular image placed above the processed one. Also, the UI shows the number of apples detected as an output measure for farmers to estimate the apple yield for their orchards. Before they could calculate an estimate for apple yield, though, additional calculations like applying a correlation between detected and actual Pink Lady apples would be necessary. The researchers found a correlation in this study, which will be discussed later on, but at the current stage of the application, it cannot be

applied directly to this output of apples number. Considering there are different varieties of apple trees, creating a correlation for each specific variety being analyzed would be vital for precise estimations across these different types of trees. Lastly, there is a button to select new images for processing. After clicking the button, an easy-to-use pop-up interface provides options to choose images from the photo library, capture a photo, and cancel the action. Once an apple tree image is selected, it gets transmitted to the back end, where it is processed. The UI is then instantly refreshed to show the altered Pink Lady apple tree image and the number of apple clusters identified and displayed.



(**a**) The default user interface is shown with image placeholders and a choose picture button to make an image selection.

(**b**) This image displays the pop-up interface with the options for selecting an image to detect apples on.

(**c**) The user interface after an image has been selected and processed, showing both images and detected apple clusters.

Figure 4. A visualization of the three different stages of the Fruit Harvest Helper's front-end or UI.

2.4. iOS Back-End Development

The functionality that lies at the core of an application, called the back end of an application, serves the purpose of handling the user's input or selections and then processing that input to generate an output. In the context of the Fruit Harvest Helper mobile app, which focuses on detecting apples, a farmer picks an image of a Pink Lady apple tree, which is the user selection. After processing this image on the back end, a processing image along with a tally of the identified apple clusters are returned to the user on the front end, as just discussed. To bring this back-end functionality to life, the researchers utilized the C++ programming language, C++ native libraries, and the OpenCV library for processing.

The Fruit Harvest Helper's architectural design followed a pattern described in a research study by Thakur and Pandey, the Model View Controller (MVC) pattern, which is an embraced approach in software engineering. The MVC pattern segmented the application into three parts: the Model managed input data, the View handled the front-end presentation for users or farmers, and the Controller processed data behind the scenes. This separation of concerns made the code more modular and maintainable for future researchers who make additions or changes to this application [19]. Enabling communication between React Native and the C++ back end involved integrating Djinni, a deprecated tool that can be used for communication between the front end and back end in React Native

applications. The researchers initially selected Djinni for its ability to generate a back-end file where the functionality for both iOS and Android could be written to process apple tree images. Unfortunately, the researchers faced challenges in making this method compatible with Android devices, but it still worked as a solution for the iOS back end.

To overcome the difficulties of sending the Pink Lady apple tree images to the C++ back end using Djinni, the research team developed a workaround. This workaround was just a matter of saving the apple tree image in the cache of the iOS device on the React Native front end and later retrieving it by the C++ back end through the use of the iOS device's file system. Once the image was received and it was confirmed, the OpenCV C++ image-segmentation algorithm was applied, which used the machine learning techniques outlined in the algorithm section. The edited or processed image and a count of the detected Pink Lady apples were then sent back to the front end to display to the farmers.

2.5. Android Back-End Development

The development of the Android back end consisted of using a mixture of Java and C++. Java is the native back-end language for Android, meaning it had to be used, and C++ needed to be used to use the same color ratio-based OpenCV C++ algorithm. Recalling the iOS app created a pathway to send information from the application's front end to the back end using Djinni. Still, the researchers needed help to make this method successful for Android. After exploring this option enough, a more well-known method of communicating with the Android back end was tried. Typically, Android Native Modules can provide back-end functionality for Android apps when working with the React Native framework. By using a more popular method, documentation was easier to find and, therefore, followed for the creation of the Android back end. Android Native Modules provide a back-end environment written in Java that can be used for mobile applications.

The React Native front end establishes a connection with an Android Native Module by calling a Java function that developers create within the module. Setting up this connection required the researchers to undergo various configuration steps. Initially, a JavaScript wrapper was made to wrap around the module on the React Native front end. Also, methods that needed to be called in the Android Native Module from the front end had to be labeled as callable so they could be found when an image was selected for processing. The last step was registering the native module for usage within the application, making it recognizable to Android devices. These configuration steps marked the first phase of handling the image processing for Android.

The color ratio-based image-segmentation algorithm was written in OpenCV C++, which needed to work for both iOS and Android. In order to get this algorithm to be compatible with Android, communication between Java and C++ had to be made, which brought about a set of challenges. The Java Native Interface (JNI) was a viable solution because it allows Java to talk back and forth with other languages, such as C++. Configuring JNI required adjusting the Android applications build process to get these two languages to cooperate correctly. The Android Native Module had to be able to interact with C++, too. Configuring the Android Native Module to interact with the C++ code meant writing more Java code to get the native module to recognize that C++ code was available. This configuration allowed the native module to establish a communication channel with C++ for sending the original apple tree image and receiving the processed image. In the C++ image processing file, a JNI function was implemented to get the communication channel working on the C++ side. The C++ could then accept the sent-over image data, process it, and relay it back to the Java side. Once the processed image reached Java, Java passed the image over to React Native to display it on the UI.

To finish setting up the Android back end, OpenCV had to be installed and integrated into the current setup. Installing OpenCV was straightforward since this involved downloading OpenCV version 4.6.0 for Android. The researchers downloaded and used this version for iOS, too. However, once OpenCV was installed, the Android build process had to be configured again so Android devices could recognize the OpenCV library. When

OpenCV started working within the setup, the apple-detection algorithm was copied from the iOS C++ file and pasted into the Android C++ file. Although this meant the code for the algorithm was used twice, this solution saved time and was able to process images just fine. One drawback of this entire solution for image processing on Android was that JNI worsened processing times.

3. Results

This research study had two main objectives. These objectives were to create a mobile application for iOS and Android devices to detect Pink Lady apples in images of apple trees and to verify the detection algorithm's efficacy by testing it and deriving a correlation. While testing the color ratio-based image-segmentation algorithm's efficacy was significant, developing the Fruit Harvest Helper mobile application was the highest priority of this research. In other words, creating the app and making it completely functional is where most of the attention was put, leaving improvements to the algorithm and thorough testing as a task for future research. However, some testing was still conducted to verify the detection algorithm's ability, yielding results for detection ability and performance. As discussed earlier, the algorithm's hyperparameters were tweaked and the results were observed repeatedly in order to fine-tune the app's algorithm and get it to detect Pink Lady apples to the best of its ability.

The evaluation method for assessing the algorithm's detection abilities involved calculating the Mean Absolute Percentage Error (MAPE). While the MAPE is usually used to evaluate regression models, it allowed for a relatively easy assessment of the algorithm's apple-detection capabilities. As stated in an article that uses MAPE for economic forecasting, MAPE penalizes underprediction and overprediction relative to the actual outcome [20]. In the case of using MAPE to asses the Pink Lady apple-detection algorithm, this metric evaluates how close the detected apple values were to the actual number of apples on a specific tree. The researchers calculated three MAPE values: one for the east apple tree photos, one for the west apple tree photos, and one combining both images. In obtaining these values, it was found that the lowest MAPE value came from using both images, yielding an error of 8.52%. Since the lowest MAPE value was associated with using both photos, this indicated that farmers would receive the best results with this algorithm when capturing an image from both sides of a Pink Lady apple tree. To see the best results with this app, a farmer must take a picture from both sides of an apple tree and combine the number of detected apples, not apple clusters, from each image. The correlation between detected and actual apples found from experimentation could then be applied to this number to estimate the apple yield of a single tree.

As discussed, the MAPE equation was used to calculate three distinct MAPE values for the apple tree image dataset by using the elaborate method discussed previously. However, the researchers focused on only the most significant of these values. The MAPE equation is shown as Equation (3). In this equation, n represents the dataset's total number of observations or trees analyzed. The variable y_i denotes the ground truth number of apples observed by humans, obtained by counting the apples on both images, one from each apple tree, and combining these values. Similarly, the variable $\hat{y}_i$ represents the detected apples found by counting the apples within the green bounding boxes on both images of an apple tree and combining the values. This MAPE equation works by subtracting the detected number of apples from the manually counted number of apples and dividing by the manually counted apples to find the error rate for a single tree as a decimal value. This algorithm's decimal error rate is then converted to a percentage. A sum is then used to repeat this process for every tree, and the average is then found by dividing the sum of percentages by the total number of trees observed n. As a result, the MAPE of plugging in all of the manual and detected apple counts into this equation, the MAPE of 8.52%, was found to signify how far away the algorithm's predictions tended to be from an actual number of apples within both images of an apple tree.

$$\text{MAPE} = \frac{1}{n}\sum_{i=1}^{n}\left|\frac{y_i - \hat{y}_i}{y_i}\right| \times 100\% \quad (3)$$

The researchers also tested the algorithm's performance or speed within this study. A timer had to be created within the React Native code to test for this. The timer started right before an apple tree image was processed and ended after the image was processed, and the results were displayed to the user on the UI. Since the Android and iOS mobile apps had different back ends for image processing, each platform had to be tested separately as the performance would vary. A total of 5 apple tree images were chosen to test the algorithm's performance. After processing all five images on an Android and iOS device, the average performance for each platform was calculated. The image processing algorithm's performance on Android was 1.16 s, and on iOS, it was 0.14 s. The Android platform saw a significantly worse average performance because JNI was used for the back end, which is known to be slow. However, the performance on both devices was reasonable for a fruit yield-estimation app that helps farmers, as 1 s is not too long to wait at all.

Along with calculating the MAPE metric and performance for the algorithm, three statistically significant correlations were also found during testing. Unlike the MAPE calculations, which used the true number of apples on the tree of interest in an image, the number of apples physically on the Pink Lady apple trees was needed to calculate these correlations. The type of correlation coefficient used when finding the correlation between detected apples and actual apples was a Pearson Correlation Coefficient, denoted as r in Equation (4). The Pearson Correlation Coefficient typically ranges from -1 to 1, where -1 is the strongest negative linear relationship, 0 means no correlation, and 1 is the strongest positive linear relationship between the two variables being analyzed [21]. Like in the MAPE equation, the Pearson Correlation Coefficient equation n represents the number of observations or apple trees used. In this equation, the two other variables, x and y, represent the number of apples detected by the algorithm and the number of apples physically on the trees, respectively. From using Equation (4), a significant Pearson Correlation Coefficient of around 0.6 was the result when using the combined number of detections from both sides of an apple tree. This value indicates a moderately strong correlation between Pink Lady apples detected by the algorithm developed in this study and the apples physically on the Pink Lady apple trees. Refer to Figure 5 to visualize this relationship between the two variables, in which each red dot represents a Pink Lady apple tree observed. Out of the three correlations, the one found using detections from both images had the strongest correlation. It would be best for farmers to apply to the detected apple count to help them get a fruit yield estimation.

$$r = \frac{n\sum_{i=1}^{n} xy - \sum_{i=1}^{n} x \sum_{i=1}^{n} y}{\sqrt{n\sum_{i=1}^{n} x^2 - (\sum_{i=1}^{n} x)^2} \cdot \sqrt{n\sum_{i=1}^{n} y^2 - (\sum_{i=1}^{n} y)^2}} \quad (4)$$

As mentioned, each of the correlations calculated was statistically significant, meaning that the correlations likely were not an accident and indicated a real relationship between the variables. The correlations were tested for statistical significance by conducting a null hypothesis test, which is a statistical method of providing evidence for an effect [22]. The first step of this test was calculating a t-statistic denoted as t in Equation (5). In the equation, t is the t-statistic, r is the Pearson Correlation Coefficient, and n is the number of Pink Lady apple trees observed. This t-statistic value was used to assess the strength and direction of the linear relationship between the variables.

Each of these correlations was tested, and they were all statistically significant. A significant Pearson correlation coefficient of around 0.6 was found using images from both sides of the apple trees. Refer to Figure 5 to see this relationship between actual and detected apples, which was the strongest of the three correlations calculated. As mentioned, this correlation was tested and was statistically significant, with a p-value of 0.000197, signifying its relevance.

$$t = r\sqrt{\frac{n-2}{1-r^2}} \tag{5}$$

Once this t-statistic value was calculated, a p-value was obtained by plugging in the t-statistic, t, into Equation (6). The equation to get the p-value represents the probability of obtaining a t-statistic as extreme as the observed value under the null hypothesis. This study compared The calculated p-value to a predetermined significance level of 0.05. Since all of the correlations between detected and actual apples had associated p-values less than 0.05, the null hypothesis was rejected, and therefore, the Pearson Correlation Coefficients were statistically significant. The most crucial correlation obtained from using two images had a p-value of 0.000197, strongly suggesting the correlation was not a byproduct of chance.

$$p\text{-value} = P(|T| > |t|) \tag{6}$$

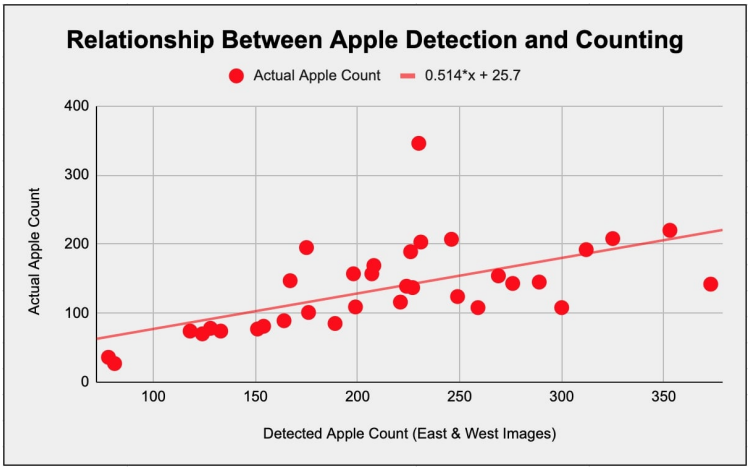


Figure 5. A scatter plot of the relationship between the apples detected by the algorithm and those counted on the physical apple trees. Each red dot represents 1 of the 34 apple trees the algorithm was tested on. Since two images were used for each apple tree, the detected apple count combines the number of apples detected within the east and west images of an apple tree.

4. Discussion

The color ratio-based algorithm's strengths and limitations were discovered by evaluating its detection capabilities. The algorithm's strengths were that it detected Pink Lady apples well and even spotted some apples in shadows that were harder to find. On the downside, there were recurring instances where the algorithm falsely detected things as apples that were not. The false detections are mainly attributed to this algorithm being color-based, which can only perform so well on detection tasks. These misidentifications included classifying orange tape, sunlight, dark brown leaves, and a red car as apple clusters. To reduce the number of false detections, the researchers added a minimum contour size setting to the algorithm, which prevented it from detecting a lot of small clusters that were not apples but also caused it to miss some smaller apples. With this setting in place, if an apple tree were huge and required the user to zoom out to capture an image, the apples in the image would be too small for the algorithm to detect. Due to this, the researchers recommend that farmers take pictures with the app from about 2.5 m away from a tree and not use the app on giant trees.

Similarly, the algorithm also did not detect apples that were significantly more green than red since the color ratio-based classification equations classify pixels as apples if they contain more red than other colors. This means that the algorithm experienced false negatives or did not detect apples that were entirely green. Once again, this is due to the limitations of a color-based detection algorithm. Although the detection algorithm missed all of the completely green apples, the researchers were surprised to observe that the apple-detection algorithm detected apples that were pinkish green or a mixture of red

and green. The algorithm was able to detect these because if it detected a small amount of pink or green on an apple, it would "think" that it found an entire apple. In reality, it only partially detected these kinds of apples as it did not spot the green parts of them. The researchers tried to fine-tune the algorithm to detect the green on apples. Still, when they adjusted the classification equations further, the algorithm detected lots of leaves as apples. However, it is essential to remember that the primary goal of this research study was the completion of the application and that the current algorithm is not the final algorithm. Another algorithm will be made in the future to help make the app effective for agronomic purposes. In conclusion, even though the researchers fine-tuned the algorithm, these limitations could not be fixed due to the nature of the color ratio-based classification equations used. Various other machine-learning algorithms could be used to improve Pink Lady apple detection with the Fruit Harvest Helper app.

Although the algorithm faced various challenges, it still successfully detected most apples in the dataset. However, the approach used to evaluate the algorithm's abilities impacted its overall success. More specifically, intentionally evaluating the apples only within the boundaries of the tree of interest in the images of apple trees helped out. This helped because extra apples on the surrounding trees and the environment were not counted as apples detected. Only the main apple tree in the image was focused on for testing the algorithm. This approach was acceptable since the trees in the photos from Symm's Fruit Ranch are so close together and often overlap, making it impossible to capture an image of only a single tree.

Furthermore, part of the scope of this study was to develop a Pink Lady apple-detection algorithm that farmers could easily use within a mobile application. Creating or implementing techniques to isolate apple trees when given an image was outside the scope of this research. Saying this, one reason the mobile app is still being prepared to be used as an apple yield-estimation tool is that humans have to identify the trees manually. Further research could overcome this challenge, improving the app's usefulness in Pink Lady apple orchards.

While the dataset was helpful in testing and deriving a correlation, it also had limitations. The dataset NNU's Robotics Vision Lab collected only captures some apple tree environments across different Pink Lady apple orchards. Considering factors like lighting conditions, image quality, and apple tree shapes would help determine whether or not the algorithm would be effective in all orchards. In future studies, using datasets containing trees from more than one orchard created with these factors in mind would help better validate how well the algorithm works. This testing would aid in improving the algorithm so more farmers would be willing to adopt this technology. That said, the image dataset contained two different lighting conditions, which the algorithm appeared to perform consistently across. While more lighting conditions need to be checked before making generalizations, this does indicate that the algorithm's capabilities might not be significantly affected by lighting conditions.

The results of this study closely match the objectives of creating a mobile application that detects apples of the Pink Lady variety, validating the algorithm's effectiveness, and deriving a correlation. The MAPE value of 8.52% suggests that the algorithm could detect Pink Lady apples in images from Symm's Fruit Ranch. While the MAPE is a valid evaluation method, a more common method is finding F1 scores. The F1 score is defined as the harmonic mean of the precision and recall of a classification algorithm. Another study for apple detection using a deep learning approach by Xuan et al. calculated F1 scores for their four different algorithms. The best of the four detection algorithms was the YOLOv3 model, reporting an F1 score of 95.0%. Most other studies for apple detection used an F1 score to assess accuracy as well since that is the most suitable metric [23]. Although the MAPE value calculated in this study does not directly compare to these F1 scores, it would have a lower F1 score than 95.0% but would still be pretty accurate. An F1 score should be calculated for this algorithm in the future. Another research study from 2021 claimed that their Convolutional Neural Network (CNN) detects apples, some of which were Pink Lady

apples, with an accuracy of 99.97% [24]. While the apple-detection abilities were relatively high in the researcher's study, one could guess it was less accurate than that proposed CNN. However, these were standalone apple-detection algorithms and were not put into a mobile application for precision agriculture. In saying this, the main advantage of the color ratio-based algorithm proposed is that it is in the Fruit Harvest Helper mobile application, which will help farmers estimate fruit yield using either an iOS or Android device. Considering that no mobile application exists to help farmers estimate fruit yield in Pink Lady apple orchards, this study benefits farmers needing this precision agriculture tool. It should also be mentioned that the moderately strong Pearson Correlation Coefficient supports the algorithm's effectiveness at estimating apple yield based on the number of Pink Lady apples detected. While the correlation could be more robust, it will likely improve upon the creation of a more advanced machine-learning algorithm.

The performance of the algorithm's processing time for Android and iOS should also be discussed. The average performance on the iOS device was high-speed, sitting at 0.14 s. The iOS device does not need to see any boosts in performance as this is quick enough. On the other hand, the average performance on the Android device was 1.16 s due to using JNI for the Android back end. Considering the processing time of the algorithm is over 1 s, improving the performance on the Android platform could be beneficial. However, it is not a direct priority. Considering that the Fruit Harvest Helper will save farmers hours of labor by automating part of the fruit yield-estimation task, waiting 1 s is fine.

The Fruit Harvest Helper mobile application shows the potential to transform the current fruit yield-estimation task in Pink Lady apple orchards. In the future, it would offer an alternative to manual apple yield estimation, a laborious and time-consuming task for farmers. Using technology to simplify the current process would save farmers time and money and could even become more accurate than the current agricultural practice due to human error.

5. Future Work

Although the Fruit Harvest Helper is a fully functional mobile application, many improvements could still be made. The current image-segmentation algorithm is effective. Still, it needs to be better at detecting apples for the app to be viable for helping farmers with the fruit yield-estimation task in Pink Lady apple orchards. Creating a more advanced algorithm would be required to achieve high levels of accuracy. Python could be used in the future since it has more options and capabilities for machine learning. A machine-learning model may be made with YOLOv8, Mask R-CNN, or another computer vision architecture. If a developed model results in high accuracy, the Fruit Harvest Helper's back end may need to be remade for Python support. The back end could be done with Django, a Python-based web framework that would execute the Python algorithm on a server. Another option would be to export the algorithm using TensorFlow Light, which is compatible with mobile applications. In doing this, the algorithm could be easily embedded into the JavaScript portion of the mobile application. Upon adding a more accurate algorithm, it would need to be assessed to see how well it estimates fruit yield within Pink Lady apple orchards. The app's fruit yield-estimation value would need to be compared with the manually found fruit yield-estimation value and the actual fruit yield of the orchard. In comparing the app's estimation with these two values, insight into how the app does compared to the current practice and the actual yield would be obtained.

Aside from apple detection and testing, other algorithms could be developed and inserted into the Fruit Harvest Helper now that it is fully functional. Slight modifications to the UI would need to be made to support multiple algorithms, but necessary changes would be minimal. Some changes to the UI could be adding other buttons to run new algorithms. An exciting algorithm that could be added is a better apple blossom-detection algorithm, which would help farmers receive a very early fruit yield estimation for Pink Lady apple orchards. The mobile application could also expand to fruit yield estimation for peach orchards. Peach and peach blossom-detection algorithms could be developed to

estimate peach yield, another area that needs further research. The Fruit Harvest Helper could be improved in many ways to help farmers with the fruit yield-estimation task within different types of orchards.

6. Conclusions

In this research study, the Fruit Harvest Helper mobile application was successfully developed for iOS and Android devices. The proposed mobile application detects Pink Lady apples in images of apple trees in order to help farmers by automating the fruit yield-estimation task. The research explained the development of the mobile application, describing the process of making the front end, iOS back end, Android back end, and the Pink Lady apple-detection algorithm. Also, the efficacy of the color ratio-based image-segmentation algorithm was demonstrated, achieving a Mean Absolute Percentage Error (MAPE) of 8.52% when using images from both sides of the apple trees. This low error rate indicates that the algorithm is reasonably practical in detecting Pink Lady apples. However, as discussed, the algorithm detected a lot of false positives because the researchers took a color-based approach to classification. Despite these limitations, the Fruit Harvest Helper is still the only app that could help farmers with fruit yield estimation in Pink Lady apple orchards. The moderately strong Pearson Correlation Coefficient of approximately 0.6 between detected and actual apples supports the algorithm's potential use as a precision agriculture tool that could help farmers.

Compared to more advanced apple-detection algorithms, such as those utilizing Convolutional Neural Networks (CNNs) with accuracies reported as high as 99.97%, the color ratio-based algorithm proposed in this study is not as capable of successfully detecting apples [24]. However, what makes the Fruit Harvest Helper unique is that the Pink Lady-detection algorithm was developed and integrated into an iOS and Android mobile app, providing farmers with a convenient and accessible tool for estimating the apple yield in Pink Lady orchards. Furthermore, these other studies did not find a correlation that farmers could use for fruit yield estimation. The algorithm's performance or processing speed was also tested, yielding an average processing time of 0.14 s on iOS devices and 1.16 s on Android devices. While Android performance is slow, the current processing times are adequate for farmers to use, especially considering the significant amount of labor the app saves.

The Fruit Harvest Helper mobile application demonstrates significant promise in helping to automate fruit yield estimation in Pink Lady apple orchards. While future improvements are necessary before it is viable for farmers to use, such as developing a new detection algorithm, the current version is near-ready. Further research should focus on developing a more advanced Pink Lady apple-detection algorithm, validating the app's effectiveness across diverse orchard environments to make it more reliable, and implementing a technique to isolate apple trees from surrounding elements. Overall, the development of this mobile application marks a substantial step towards having an accessible precision agriculture tool that farmers can use for Pink Lady apple yield estimation.

Author Contributions: Conceptualization, D.M.B., J.I.B. and B.D.; methodology, D.M.B., J.I.B., B.D. and J.N.; software, B.D., J.I.B. and J.N.; validation, D.M.B., B.D. and J.N.; formal analysis, D.M.B. and B.D.; investigation, D.M.B., J.I.B., B.D. and J.N.; resources, D.M.B. and B.D.; data curation, B.D. and J.N.; writing—original draft preparation, D.M.B., B.D., J.I.B. and J.N.; writing—review and editing, D.M.B., B.D., J.I.B. and J.N.; visualization, B.D.; supervision, D.M.B.; project administration, D.M.B. and J.I.B.; funding acquisition, D.M.B. All authors have read and agreed to the published version of the manuscript.

Funding: This research was funded by the Idaho State Department of Agriculture through the Specialty Crop Block Grant 2021.

Data Availability Statement: Data will be made available upon request.

Acknowledgments: We would like to thank and acknowledge Symms Fruit Ranch and Williamson Orchard for allowing us to use their orchards for research, and the Idaho State Department of Agriculture for the Specialty Crop Block Grant.

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

Abbreviations

The following abbreviations are used in this manuscript:

NNU	Northwest Nazarene University
MAPE	Mean Absolute Percentage Error
UI	User Interface
MVC	Model-View-Controller
JNI	Java Native Interface
CNN	Convolutional Neural Network

References

1. Ahmad, T.I.; Tahir, A.; Bhatti, M.A.; Hussain, A. Yield and Profitability Comparisons of Modern Agricultural Production Technologies Adoption: Evidence From Cotton-Wheat Punjab (Pakistan). *Rev. Educ. Adm. Law* **2022**, *5*, 203–216. [CrossRef]
2. A Survey on Mobile Applications for Smart Agriculture | Semantic Scholar. Available online: https://www.semanticscholar.org/paper/A-Survey-on-Mobile-Applications-for-Smart-Oteyo-Marra/3e1f0489c2ccba86fa0fc5936b07e40c8a067972 (accessed on 1 April 2024).
3. Fulton, J.; Hawkins, E.; Taylor, R.; Franzen, A.; Shannon, D.K.; Clay, D.; Kitchen, N.R. Yield Monitoring and Mapping. In *Precision Agriculture Basics*; Wiley Online Library: Hoboken, NJ, USA, 2018; ISBN 978-0-89118-367-9.
4. He, L.; Fang, W.; Zhao, G.; Wu, Z.; Fu, L.; Li, R.; Majeed, Y.; Dhupia, J. Fruit Yield Prediction and Estimation in Orchards: A State-of-the-Art Comprehensive Review for Both Direct and Indirect Methods. *Comput. Electron. Agric.* **2022**, *195*, 106812. [CrossRef]
5. Gong, A.; Yu, J.; He, Y.; Qiu, Z. Citrus Yield Estimation Based on Images Processed by an Android Mobile Phone. *Biosyst. Eng.* **2013**, *115*, 162–170. [CrossRef]
6. A Smartphone-Based Apple Yield Estimation Application Using Imaging Features and the ANN Method in Mature Period. Available online: https://www.researchgate.net/publication/324164204_A_smartphone-based_apple_yield_estimation_application_using_imaging_features_and_the_ANN_method_in_mature_period (accessed on 1 April 2024)
7. Fruit Harvest-Estimating Apple Yield and Fruit Size. Available online: https://extension.psu.edu/fruit-harvest-estimating-apple-yield-and-fruit-size (accessed on 26 March 2024).
8. Braun, B.; Bulanon, D.; Colwell, J.; Stutz, A.; Stutz, J.; Nogales, C.; Hestand, T.; Verhage, P.; Tracht, T. A Fruit Yield Prediction Method Using Blossom Detection. In Proceedings of the 2018 ASABE Annual International Meeting. American Society of Agricultural and Biological Engineers, Detroit, MI, USA, 29 July–1 August 2018.
9. Image Segmentation for Fruit Detection and Yield Estimation in Apple Orchards-Bargoti-2017-Journal of Field Robotics-Wiley Online Library. Available online: https://onlinelibrary.wiley.com/doi/10.1002/rob.21699 (accessed on 1 April 2024).
10. Karkhile, S.; Ghuge, S. Modern Farming Techniques Using Android Application. *Int. J. Innov. Res. Sci. Eng. Technol.* **2015**, *4*, 10499–10506.
11. Zhou, Z.; Song, Z.; Fu, L.; Gao, F.; Li, R.; Cui, Y. Real-Time Kiwifruit Detection in Orchard Using Deep Learning on Android™ Smartphones for Yield Estimation. *Comput. Electron. Agric.* **2020**, *179*, 105856. [CrossRef]
12. Available online: https://www.researchgate.net/profile/Sotiris-Karetsos/publication/313868513_Studying_Mobile_Apps_for_Agriculture/links/58ad4a2e4585155ae77aef24/Studying-Mobile-Apps-for-Agriculture.pdf (accessed on 3 April 2024).
13. Available online: https://web.p.ebscohost.com/ehost/pdfviewer/pdfviewer?vid=1&sid=dd87b161-2893-4084-aac9-298a27059ace (accessed on 3 April 2024).
14. Zou, K.; Ge, L.; Zhou, H.; Zhang, C.; Li, W. An Apple Image Segmentation Method Based on a Color Index Obtained by a Genetic Algorithm. *Multimed. Tools Appl.* **2022**, *81*, 8139–8153. [CrossRef]
15. Li, H.; Chaudhari, P.; Yang, H.; Lam, M.; Ravichandran, A.; Bhotika, R.; Soatto, S. Rethinking the Hyperparameters for Fine-Tuning. *arXiv* **2020**, arXiv:2002.11770.
16. Gonzalez, R.C.; Woods, R.E. *Digital Image Processing*, 3rd ed.; Prentice-Hall, Inc.: Wilmington, DE, USA, 2006; ISBN 978-0-13-168728-8.
17. Mat Said, K.A.; Jambek, A.; Sulaiman, N. A Study of Image Processing Using Morphological Opening and Closing Processes. *Int. J. Control Theory Appl.* **2016**, *9*, 15–21.
18. Brambilla, M.; Mauri, A.; Umuhoza, E. Extending the Interaction Flow Modeling Language (IFML) for Model Driven Development of Mobile Applications Front End. In Proceedings of the 11th International Conference, MobiWIS 2014, Mobile Web Information Systems, Barcelona, Spain, 27–29 August 2014; Awan, I., Younas, M., Franch, X., Quer, C., Eds.; Springer International Publishing: Cham, Switzerland, 2014; pp. 176–191.

19. Thakur, R.N.; Pandey, U.S. The Role of Model-View Controller in Object Oriented Software Development. *Nepal J. Multidiscip. Res.* **2019**, *2*, 1–6. [CrossRef]
20. McKenzie, J. Mean Absolute Percentage Error and Bias in Economic Forecasting. *Econ. Lett.* **2011**, *113*, 259–262. [CrossRef]
21. Pearson's Correlation Coefficient | The BMJ. Available online: https://www.bmj.com/content/345/bmj.e4483.full.pdf+html (accessed on 13 May 2024).
22. Null Hypothesis Significance Testing: A Short Tutorial—PMC. Available online: https://www.ncbi.nlm.nih.gov/pmc/articles/PMC5635437/ (accessed on 13 May 2024).
23. Apple Detection in Natural Environment Using Deep Learning Algorithms | IEEE Journals & Magazine | IEEE Xplore. Available online: https://ieeexplore.ieee.org/abstract/document/9269995 (accessed on 1 April 2024).
24. Robustness of Convolutional Neural Network in Classifying Apple Images | IEEE Conference Publication | IEEE Xplore. Available online: https://ieeexplore.ieee.org/abstract/document/9502258 (accessed on 1 April 2024).

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.