NORTHWEST NAZARENE UNIVERSITY

Android Fruit Harvest Helper Apple Detection App Development with React Native

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF SCIENCE

Brandon Thomas Duncan
2024

THESIS
Submitted to the Department of Mathematics and Computer Science
in partial fulfillment of the requirements
for the degree of
BACHELOR OF SCIENCE

Brandon Thomas Duncan
2024

Android Fruit Harvest Helper Apple Detection App Development with React Native

Author: Brandon Thomas Duncan
Brandon Thomas Duncan

Approved:
Dale Hamilton, Ph.D., Chair, Department of
Mathematics and Computer Science, Faculty Advisor

Approved:
Duke Bulanon, Ph.D., Chair, Department of
Engineering and Physics, Second Reader

Approved:
Dale Hamilton, Ph.D., Chair, Department of
Mathematics and Computer Science

ABSTRACT

Android Fruit Harvest Helper Apple Detection App Development with React Native.
DUNCAN, BRANDON (Department of Mathematics and Computer Science), BULANON, DUKE (Department of Engineering and Physics), HAMILTON, DALE (Department of Mathematics and Computer Science).

The Android Fruit Harvest Helper is an innovative application developed using React Native, aimed at assisting farmers in estimating fruit yields, particularly in apple orchards. This project bridges the gap between iOS and Android platforms by recreating an existing iOS app for blossom detection in a cross-platform environment. The app's intuitive user interface (UI), designed with farmers in mind, simplifies the process of estimating fruit yields by providing a frictionless experience for image selection and processing. The Fruit Harvest Helper's UI features two boxes for the selected and processed image, along with a button enabling users to select or take a photo. For the backend, the app employs a color-based feature extraction machine learning algorithm, implemented in OpenCV C++, to detect apples in images of apple trees. Results show a Mean Absolute Percentage Error (MAPE) of 8.52% in apple detection when evaluating pictures from both sides of an apple tree. A correlation coefficient of 0.6 between detected and actual apples was found as well. While the app has promising functionality, opportunities for improvement exist, including the development of an object-based machine learning apple detection algorithm and methods to detect various types of fruits. Fruit Harvest Helper represents a significant step towards precision agriculture, nearing real-world utility for farmers in yield monitoring and farm management.

Acknowledgement

I am grateful for all of those who have helped me both get to college and get through college in order to have better opportunities in my life. I would first like to thank all of my family members including my Mom and Dad who listened to me talk about this project and supported me throughout the ups and downs. Also, my brother Kyle who listened to me talk about the project too and gave me ideas to use and build upon. I would like to thank my friends here at Northwest Nazarene University that I went through everything with, especially Carter Katzenberger. Lastly, I would like to thank my professors for building up my knowledge in computer science and teaching me important life lessons both in and out of this field: Drs. McCarty, Hamilton, Myers, and Bulanon.

Table of Contents

List of Figures

Introduction

As the population increases, farmers will have to collectively meet the world's increasing demands for food. To produce more food and keep up with their competitors, farmers must use precision agriculture to adapt quickly. Precision agriculture refers to using any technology or analysis tool to optimize a current farming practice. According to a study conducted in Pakistan comparing cotton farms that adopted precision agriculture technologies with those that did not, the precision agriculture users experienced up to a 22% higher crop yield and 27% higher profits than non-adopters (Ahmad et al. 2022). While this study was only for cotton farms, these findings suggest that precision agriculture can help farmers, making their farms significantly more sustainable. If farmers want to outperform their competitors and achieve agricultural success, they will have to start utilizing various kinds of precision agriculture tools.

A current task in farming that still needs to be simplified by technology is fruit yield estimation. The process of fruit yield estimation is tedious and time-consuming for farmers, requiring them to count all of the fruits on a tree manually, repeat this task for a group of trees, find the average number of fruits on this subset of trees, and finally apply the average to the whole orchard to get a fruit yield estimation (Marini 2024). Once a fruit yield estimation is obtained, farmers use this value to determine how many laborers they need to hire for fruit picking. Thus, the estimation needs to be accurate. In these times, due to the increasing amount of crops farms produce, there is a shortage of laborers, making hiring the correct number of fruit pickers a problematic task. This is why a fruit yield estimation is essential, as it tells farmers at least two weeks before the harvesting season how many laborers they will need to find and hire. Using technology to

come up with an accurate estimation would make finding the fruit yield for an orchard quick and easy.

As a solution to the task of estimating fruit yield, Dr.Bulanon, the agricultural engineering professor at Northwest Nazarene University (NNU), decided that a mobile application could be a practical precision agriculture technology. Since just about everyone has a smartphone, a downloadable mobile application available to the public would be a convenient way to get an estimation tool into the hands of farmers. With a mobile application, the process of counting the fruits on a tree would be automated. Farmers would be able to open the app and take a picture from either one or both sides of a fruit tree, though there may be inconsistent lighting when taking pictures from both sides. The application would process the image, providing the farmer with a quantitative estimate for the fruit on that tree. They could then repeat this process for a subset of fruit trees within the orchard. By finding the mean fruit yield for these trees, they could then use this mean to come up with a fruit yield estimation for their entire orchard. While fruit yield estimation mobile applications for kiwis and citrus already exist, no mobile application to aid in orchards with apples has been made, which is what originally caused Dr.Bulanon to research this topic (Zhou et al. 2020; Gong et al. 2013).

Initially, Dr.Bulanon assigned an NNU Robotics Vision Lab research team to develop an iOS mobile application that helped farmers come up with an apple yield estimation. This former research team created the iOS app with Swift and OpenCV C++ to help estimate fruit yield. OpenCV is an Open Source Computer Vision library for image and video analysis (Culjak et al. 2012). The mobile application tool used image processing to detect blossoms on an image of a blossoming apple tree, displaying the

number of blossoms detected to the farmer. For this approach to help with coming up with a fruit yield, a correlation also had to be found by these researchers, providing a relationship between the number of blossoms and the number of apples on Pink Lady apple trees from Symms Furit Ranch Apple Orchard (Braun 2018). Although this application saw a lot of promise as the blossom detection approach would provide farmers with a fruit yield prediction before harvesting season, the blossom detection algorithm still needed improvements to spot blossoms with a high enough accuracy for this method to be viable in orchards (Bulanon 2021).

Considering this, Dr.Bulanon thought it would be best to get an apple detection approach working first, which is less complex than the blossom detection task. Creating and implementing an apple detection algorithm into a mobile application was one of the goals of this project. The primary goal of this project, also decided by Dr.Bulanon, was to make an Android mobile app and an iOS app to reach and help a wider audience of farmers (Ortega et al. 2015). View Figure 1 to see the Android mobile application being used to capture an image of an apple tree. The React Native framework was used to develop this cross-platform app for both iOS and Android. While there was already an iOS app developed by researchers in the past, React Native allows for the simultaneous building of the front end of iOS and Android apps, which enables the creation of both platforms instead of just the Android platform. In other words, React Native made it logical to recreate the iOS app since this framework would put both the iOS and Android apps into the same codebase or program without putting in too much extra effort. Having both apps in one codebase would make it easier for future developers to maintain the code.

Given that the existing blossom detection app's user interface (UI) worked great, the idea was to remake this UI for the React Native iOS and Android apps. However, the back end for the iOS and Android apps had to be set up independently since React Native only allows simultaneous front-end development. Finally, once everything was set up and the iOS and Android platforms were ready for an image-processing algorithm that detects Pink Lady apples, an algorithm was created, tested, and implemented into the app.

Figure 1: The Fruit Harvest Helper Android mobile app being used to capture an image of an apple tree of the Pink Lady variety.

Methods

Data Collection

Since a requirement of this project was to create an apple detection algorithm and integrate it into the mobile application, data had to be collected before the development

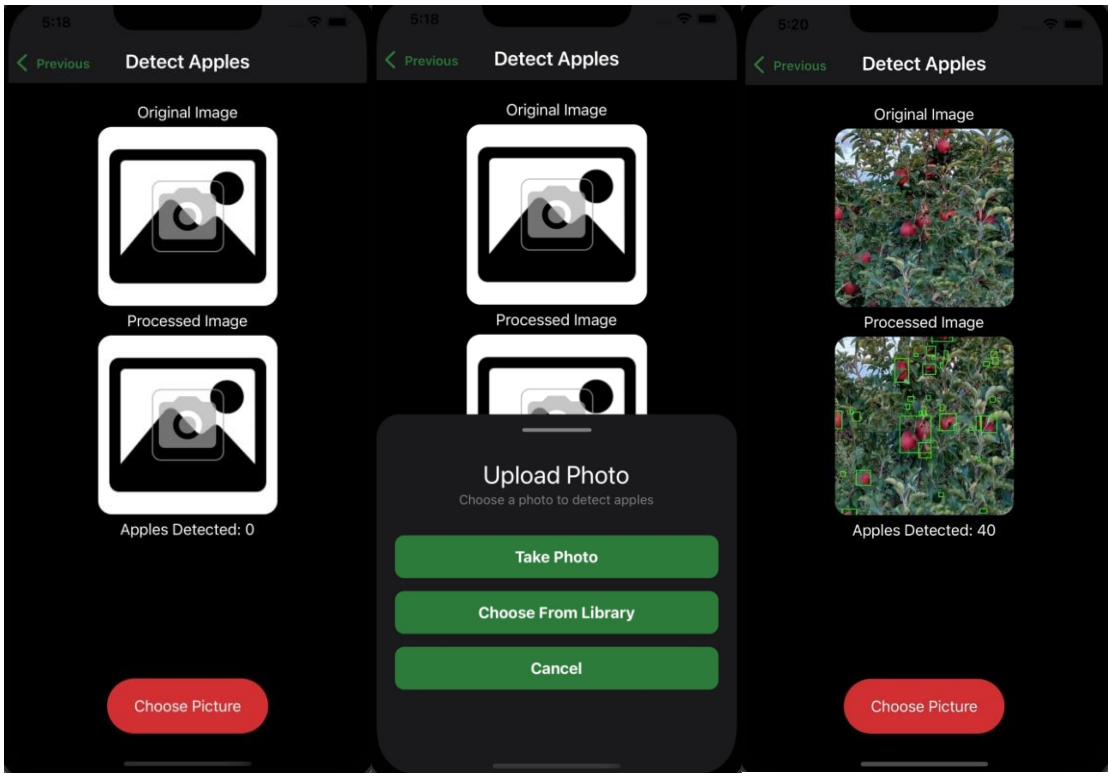
of the algorithm began. The algorithm's purpose would be to detect Pink Lady apples in images of apple trees. This meant that images of apple trees had to be taken for the algorithm to use. When the apples were ripe enough, in October of 2023, pictures were taken of 40 selected apple trees in Symm's Fruit Ranch Apple Orchard, which is near Marsing, Idaho. These were the trees that NNU's Robotics Vision Lab research team was permitted to use. It should also be mentioned that the number of apples on each tree was also manually counted. The researchers used an Android S7 tablet to capture images of these apple trees from the west-facing and east-facing sides, providing information on as many of the trees as possible. Unfortunately, one of the trees fell over, and the researchers could not get pictures of 5 other apple trees from the west-facing side since the team used those trees for harvesting apples with a robot, which would have skewed the data. Given these factors, the images collected consisted of 39 east-facing and 34 west-facing images.

React Native iOS and Android UI

When designing an application with model driven mobile development, developers should consider two things: the front end and the back end. When the user interface (UI) is being designed, it should be well thought out to meet the user's needs and ensure a positive experience (Brambilla et al. 2014). When creating a mobile application for farmers, a UI that satisfies their needs would likely focus on simplicity since they would not want unnecessary clutter or extra features. Due to this, it was settled that a straightforward UI should be made for the Fruit Harvest Helper, keeping the layout the same as the previous iOS front end created with Swift.

Beginning the development of the new mobile application for the iOS and Android platforms, the React Native framework was used along with JavaScript. As mentioned earlier, concurrently building out the front ends for both apps was efficient. The only difference between iOS and Android was setting the camera and photo library permissions. The UI for these applications consisted of four main components that were recreated from the existing iOS app. The following components and their use can be visualized below in Figures 2 , 3, and 4:

- A display for the original image.
- A display for the processed image.
- The number of apples detected by the algorithm.
- A button to choose an image to process.



Figures 2 - 4: The leftmost image shows the display of the Fruit Harvest Helper mobile app with the four main components clearly visible. In the middle image, the pop-up is shown. The rightmost image shows the app after an apple tree image is processed.

The original image and processed image displays were vertically aligned to take up most of the space a mobile device offers. Farmers needed to see these images to validate their image selection and the app's functionality. The number of apples detected by the algorithm was the most crucial piece of the UI, giving farmers helpful information that could be used in calculating an apple yield estimate. As for the button, once clicked, an easy-to-use sliding panel popped up that gave them the choice of selecting an image from their library, capturing it, or canceling out of the pop-up panel. Upon choosing an image for processing, the image would be sent to the back end for apple detection. Then, the UI was updated to display the original image, the processed image, and the number of detected apples.

Setting Up Android Back End

React Native allows developers to create the front ends of iOS and Android apps together but cannot handle more complicated tasks like image processing. When using React Native, some aspects of the back end for the iOS and Android applications must be created separately since these platforms are most compatible with specific languages. Josh Nelson developed the iOS back end that used C++ and, therefore, will not be discussed. As for the Android back end, its most compatible language was Java, but it also included using Java Native Interface (JNI) and C++ code. As pointed out by book

about JNI, calling C++ methods from within Java programs is exactly what JNI makes possible (Wang 1970). The back end had to use JNI and C++ because the apple detection algorithm was going to be developed in C++. While the algorithm could have been translated to Java, having an algorithm in two different languages would create problems. These problems include issues with algorithm maintenance, trying to produce identical results in both languages, and undergoing bad programming practices. Although the following setup may seem convoluted, this is why communication between Java and C++ was added.

For the Android back end, an Android native module was implemented and configured, which allows for communication between JavaScript on the front end and Java code on the back end. A communication channel between Java and C++ was also necessary so that the Android back end could utilize the Apple detection algorithm. Fortunately, getting Java code to interact with C++ is what JNI was created for. So, a JNI function was created in the C++ file with the algorithm, and some Java was written to ensure Java could recognize and use the C++ code to process the image. Once this was done, the build settings needed to be modified to account for the Android native module and JNI. Now that everything was communicating, an image of an apple tree a farmer selected for processing could get from JavaScript to C++ and back but could not be processed yet. The algorithm still had to be implemented into the C++ file. Since the algorithm used OpenCV, a popular image processing library, this was downloaded and integrated into the app by modifying the build settings so Android devices could use it. At this point, the apple detection algorithm with OpenCV C++ was added to the application, successfully detecting apples in images of Pink Lady apple trees.

The Apple detection algorithm

The Pink Lady apple detection algorithm created for the Fruit Harvest Helper utilized a color ratio-based image segmentation algorithm to detect apple clusters. Dr.Bulanon first created this algorithm in MatLab in order to generate the color-ratio based equations that were the algorithm's core. MatLab is a high-performance programming language for technical computing that is primarily math-based (The Math Works, Inc. 1997). Next, this MatLab algorithm was translated into OpenCV C++ to be compatible with the mobile application. This conversion process was complicated since MatLab code has much more concise tools built into it to handle certain operations. Thus, more code had to be written in OpenCV, essentially building around the color ratio-based equations.

The image segmentation algorithm followed a series of steps to detect apple clusters in images. First, some preprocessing needed to be done; this began when the original image that was captured or chosen by a farmer was copied to ensure that it was not altered in any way. It is essential to address that OpenCV stores images as a multi-dimensional matrix of pixels, providing one dimension for each of the rows and columns of pixels and the color channels. Commonly, images are stored in the red, green, blue (RGB) format. However, OpenCV stores an image's color channels in the blue, green, red (BGR) format. The next step of the algorithm splits the input image into three different color channels, BGR. Splitting the image into its color channels allowed each color channel to be evaluated independently, providing more information for image processing. Once the image was divided, each color channel and the copied image were converted to grayscale. The color intensity of each of these grayscale color channels was then

compared to the intensity of the entire grayscale image, concluding the preprocessing phase of the algorithm. These color ratios would be weights in the color ratio-based apple detection equations.

After this, the apple detection equations with the weighted color ratios were used to classify each pixel as an apple or non-apple pixel. Each of the pixels of the image was iterated one at a time to do this, applying the color ratio-based equations to a given pixel and classifying it as an apple pixel if it had significantly more red than blue and green in it. After the pixels of the image were classified, a binary image was left out. A binary image is an image that contains only two colors, usually black and white, where black indicates negative and white indicates positive when referring to the presence of a feature. In this case, white pixels represented apples, and black pixels represented anything that was not an apple. However, there was a lot of noise in this binary image that needed to be reduced through post-processing to improve the classification results. First, the blobs of white pixels or contours considered by the algorithm to be apple clusters were analyzed, and the contours that fell under the minimum contour size were deleted. Deleting these contours was useful since most apples are above a specific size, meaning this mainly eliminated apple clusters that were false positives or falsely detected as apples.

Furthermore, a morphological opening operation that used a structuring element was performed on the binary image. According to “Digital Image Processing”, a morphological opening is a common image processing technique that works by performing an erosion operation on a region followed by a dilation operation on that result. Erosion and dilation refer to shrinking and expanding of a region or cluster of

pixels. Morphological openings are often called a filtration operation as they filter out small white regions surrounding identified clusters (apple clusters), smooth the edges of clusters, and remove noise (Gonzalez and Woods 2007). Finally, the remaining contours on the binary image better represented actual apple clusters, so these contours could then be drawn onto the colored version of the copied apple tree image. The significant apple clusters, which were the clusters of pixels above a certain size threshold, were drawn onto the image as green bounding boxes. Highlighting these clusters with green bounding boxes helped indicate which regions were identified as apple clusters for visualization purposes. The processed image was then returned for display.

Results

Testing Method

Testing the apple detection algorithm involved finding two values for the apple tree of interest in the dataset of images: the number of apples manually spotted in the image and the number of apples detected by the algorithm. Considering that only the apple tree is of interest, none of the surrounding trees or other aspects of the environment were looked at since the scope of this research was only to create a Pink Lady apple detection algorithm rather than an algorithm that could isolate apple trees. A line was drawn using Microsoft Paint to delineate the tree of interest. This program also drew circles around apples when spotting the actual and detected apples on each tree of interest. Counting these two values for each tree of interest could have been better. However, considerable effort was put into ensuring the results accurately represented the algorithm's capabilities. When analyzing the trees of interest in the processed images, it

was assumed that the algorithm detected all apples within the green bounding boxes. While these green bounding boxes detected apple clusters, given the algorithm's ability to detect hard-to-detect apples such as ones covered by shadows, it seemed reasonable to provide the algorithm with this benefit of the doubt. For transparency, an example of the process conducted on both an original and processed image is shown in Figures 5 and 6.

Figures 5 and 6: The process used to evaluate the image segmentation algorithm is displayed. Microsoft Paint was used to delineate the trees of interest in the dataset.

After these values were found for the entire dataset of images, a metric could be calculated to evaluate the algorithm’s success in apple detection. A correlation could also be derived using the detected apple values and the actual number of apples physically counted by the researchers.

Findings

The metric used to evaluate the Pink Lady apple detection algorithm was the Mean Absolute Percentage Error (MAPE). While the MAPE is usually used to evaluate regression models not tasks like apple detection, it still allows for the assessment of how well the algorithm detected apples. Addressed in an article discussing the use of MAPE in economic forecasting, MAPE penalizes both underprediction and overprediction relative to the true outcome, which in this case refers to the actual number of apples on the tree of interest (McKenzie 2011). Three MAPE values were calculated in total: one for the east, one for the west, and both photos combined. Interestingly, the lowest error was found on the combined image dataset, yielding an error of 8.52%. This result indicated that having a farmer capture images of a Pink Lady apple tree from both sides and combining the detected number of apples, not apple clusters, would give them the best results with this algorithm.

Given that taking images from both sides of the apple trees was the best approach, it is worth discussing the correlation between the detected number of apples on both sides and the physical apple count on each tree. When finding a correlation, the Pearson correlation coefficient was used, which measures how strong the linear relationship between two variables is (Sedgwick 2012). The Pearson correlation coefficient for this relationship was determined to be 0.6, suggesting a moderately strong correlation between detected and actual apples when using images from both sides. Figure 7 contains a graph of this relationship. A null hypothesis test was also conducted, which is a statistical method of providing evidence for an effect (Pernet 2016). This null hypothesis

resulted in a p-value of 0.000197, which proves that the correlation found is statistically significant and likely not an accident.

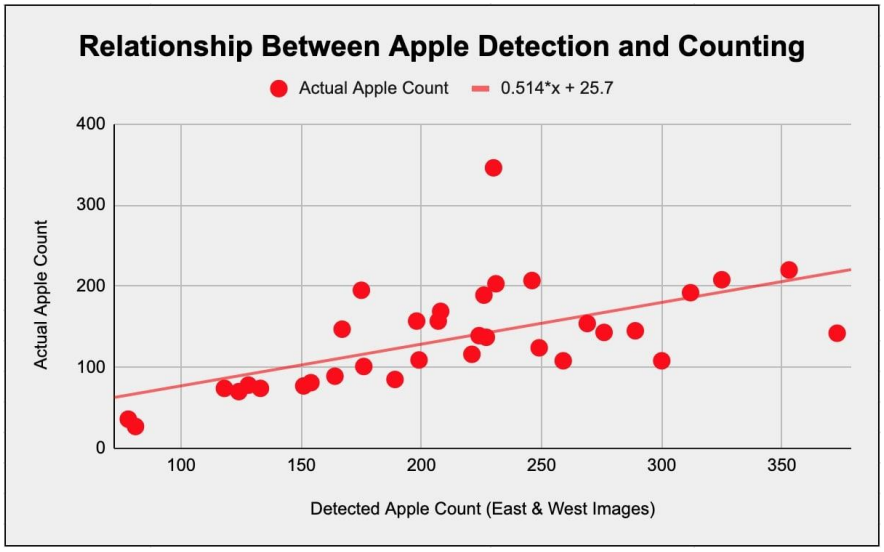


Figure 7: A relationship between the detected apples on the trees of interest and the apples physically on the trees is illustrated.

Discussion

In testing the apple detection algorithm, it was found that the detection algorithm has promise as the MAPE was low and a moderately strong correlation existed. However, the color ratio-based image segmentation algorithm had many limitations which need to be noted. The most obvious of these is that the algorithm cannot detect individual apples and can only detect apple clusters since the algorithm uses a less advanced color-based approach. An object-based detection algorithm could be used to detect individual Pink Lady apples. Another drawback of the color ratio-based algorithm was that it detected a lot of false positives, classifying orange tape, sunlight, dark brown leaves, and even a red car as apple clusters. While the algorithm was fine-tuned, these issues remained and are

once again attributed to the limitations of color-based approaches. However, this could be improved without the use of object detection. On the other hand, the algorithm detected Pink Lady apples well. The algorithm spotted most of the Pink Lady apples, except some that were green or too small since they were hiding in the leaves.

As for the Fruit Harvest Helper mobile application, the development was primarily successful. A completed iOS and Android app was created due to this project with a "farmer-friendly" front end and a back end that allowed for an image of a Pink Lady apple tree to be processed with an OpenCV C++ apple detection algorithm. However, currently there is no way to isolate apple trees, which is an issue since they often overlap with neighboring trees and form rows resembling large hedges like in Symm's Fruit Ranch Apple Orchard. This makes it difficult for farmers to consistently detect all apples that belong exclusively to a tree of interest. While methods can be created to creatively isolate apple trees, given the current state of the Fruit Harvest Helper application, farmers still need to wait before they can use it to easily obtain a fruit yield estimation.

Future Work

The main focus in the future should be to improve the Pink Lady apple detection approach to make the method usable for farmers in a real orchard. Once the Fruit Harvest Helper mobile application is good enough to assist farmers with estimating a fruit yield, it could be published in both the Apple Store and the Google Play Store for them to download and use for free. Along with improving the detection of Pink Lady apples, the challenge of blossom detection could be pursued once again. Accurately detecting apple blossoms would give farmers a fruit yield prediction earlier in the year before harvesting

season, which would be valuable. Since the user interface already exists and the back end for both the iOS and Android devices is already set up, new algorithms could easily be added to the app. Considering this, future researchers or computer science students who decide to continue with this project would add detection algorithms for several types of fruits, such as peaches.

References

Ahmad, Tusawar Iftikhar, et al. “Yield and profitability comparisons of modern agricultural production technologies adoption: Evidence from cotton-Wheat Punjab (Pakistan).” *Review of Education, Administration & Law*, vol. 5, no. 2, 30 June 2022, pp. 203–216, https://doi.org/10.47067/real.v5i2.232.

Brambilla, Marco, et al. “Extending the interaction flow modeling language (IFML) for model driven development of mobile applications front end.” *Mobile Web Information Systems*, 2014, pp. 176–191, https://doi.org/10.1007/978-3-319-10359-4_15.

Braun, B, et al. “≪I>a fruit yield prediction method using blossom detection</i>” *2018 Detroit, Michigan July 29 - August 1, 2018*, 2018, https://doi.org/10.13031/aim.201801542.

Bulanon, Joseph. *WHDL*, 2022, www.whdl.org/sites/default/files/resource/10741816/ENG_Detecting_stock_market_patterns.pdf.

Culjak, Ivan, et al. “A Brief Introduction to Opencv | IEEE Conference Publication | IEEE Xplore.” *A Brief Introduction To OpenCV*, 2012, ieeexplore.ieee.org/abstract/document/6240859.

“Fruit Harvest - Estimating Apple Yield and Fruit Size.” *Penn State Extension*, extension.psu.edu/fruit-harvest-estimating-apple-yield-and-fruit-size. Accessed 5 Apr. 2024.

Gong, Aiping, et al. “Citrus yield estimation based on images processed by an Android mobile phone.” *Biosystems Engineering*, vol. 115, no. 2, June 2013, pp. 162–170, https://doi.org/10.1016/j.biosystemseng.2013.03.009.

Matlab, 1997, itb.biologie.hu-berlin.de/~kempter/Teaching/2003_SS/gettingstarted.pdf.

McKenzie, Jordi. “Mean Absolute Percentage Error and Bias in Economic Forecasting.” *Economics Letters*, North-Holland, 25 Aug. 2011, www.sciencedirect.com/science/article/abs/pii/S0165176511003119.

Ortega. *EBSCO Publishing Service Selection Page - EHOST2*, 2015, web.ebscohost.com/ehost/pdfviewer/pdfviewer.

Pernet, Cyril. “Null Hypothesis Significance Testing: A Short Tutorial.” *F1000Research*, U.S. National Library of Medicine, 25 Aug. 2015, www.ncbi.nlm.nih.gov/pmc/articles/PMC5635437/.

Sedgwick, Philip. “Pearson’s Correlation Coefficient.” *The BMJ*, British Medical Journal Publishing Group, 4 July 2012, www.bmj.com/content/345/bmj.e4483.full.pdf+html.

Wang, Sun-Chong. “JNI Technology.” *SpringerLink*, Springer US, 1 Jan. 1970, link.springer.com/chapter/10.1007/978-1-4615-0377-4_14.

Zhou, Zhongxian, et al. “Real-time kiwifruit detection in orchard using Deep Learning on AndroidTM Smartphones for yield estimation.” *Computers and Electronics in Agriculture*, vol. 179, Dec. 2020, p. 105856, https://doi.org/10.1016/j.compag.2020.105856.