Implementation and Comparison of Machine Vision Algorithms for Detecting Apples at an Orchard

Frank S, Heath F, Nair RRP, Rolland E, Rowland T, Sharma M
(Dated: January 7, 2023)

ABSTRACT

Calculating and predetermining the yield of crops is vital for estimating the successful seasonal turnover in the agricultural industry. To make this process more efficient and accurate, machine vision algorithms are incorporated. This report is a comprehensive analytical investigation for implementing and comparing machine vision algorithms in the detection of apples at an orchard. Two different computer vision approaches were used for the detection: a conventional method, which entailed a direct manipulation of the images placed in the algorithms, and a machine learning method, whereby a convolutional neural network is trained to recognise the data. The data set applied to both methods consisted solely of red apples on trees, for the ease of the algorithm being able to differ between the surrounding elements and the apples. It was determined through the analysis of the results that the conventional method with no split consists of the lowest rate of error in the detection. Similarly, the YOLO framework for the machine learning approach constitutes of the lowest percentage rate of error in the detection stages. Overall it is concluded that the machine learning based algorithms with the YOLO framework is the most efficient and accurate approach for the purpose of this investigation.

CONTENTS

I. INTRODUCTION

The efficiency and accuracy of the statistics within the agricultural industry are deemed volatile as they are directly affected by the population of the data, as well as the control variables in place of the environments. To establish a solid set of results, computer vision has been incorporated increasingly within this industry for a plethora of factors. These consist of, but are not limited to, yield estimation, monitoring, disease prevention and detection. Hence, the purpose of this report is to implement machine vision algorithms and compare different methods, to find the most effective approach, specifically in the detection of apples at an orchard.

To ensure that the models tested produce reliable statistics, a large data set was acquired and sorted into types of images with their corresponding labels. For this investigation, only red apples in the trees were examined and accounted for in the detection algorithms. The initial approach was a conventional method, whereby the images from the data set were manipulated by their colour properties and the erosion conducted on them. Understanding the detection capabilities of the algorithms is crucial, as knowing which elements to alter is crucial. The second approach is applying machine learning algorithms within computer vision. This process consists of a convolutional neural network being trained to recognise and detect set patterns provided by the input data set. Within this method, two separate architectures are examined: YOLO and Mask-RCNN.

The reason behind investigating these methods for detecting the apples is because the conventional method is useful for quick out-of-the-box functionality and provide potentially higher accuracy. Certain scenarios lend themselves better to this method, such as perceiving changes in colour for viewing if apples are rotting or not with polarised light. However, this source of accuracy is dependant on the ways the data is manipulated. On the other hand, machine learning is deemed a very useful approach due to the depth of its machine vision ability and higher accuracy levels. Both approaches have valuable qualities that are discussed later in this report.

The assumptions within this report are mainly focused around the environment involving the data set. Specifically features such as the lighting where the image has been captured and the localised background. It becomes increasingly difficult to train the algorithms is certain patterns are not perceivable due to immense changes between the data set images.

In order to successfully determine the best approach for detecting the apples at an orchard, specific aims and objects were set as checkpoints. The overall aim was to find the method which has the highest level of accuracy at detecting red apples on tree, this values detected were compared to their corresponding labels to measure the percentage of error. To achieve this aim, the objectives provided a chronological order of tasks to complete. The objectives are listed as the following:

Objectives:

- Sort and select the images and labels from the data set
- Complete data acquisition to know what features must be taught/manipulated for the approaches
- Understand elements in images to manipulate to apply to the conventional method
- Decide on the thresholds for the method and the combination of operations required for manipulation
- Create code which applies chosen conventional approaches and test on data set and compare results
- create code for machine learning algorithms and train to detect patterns in images
- Test Machine learning algorithms with apples to see if detection is occurring
- compare two different machine learning architectures to find highest accuracy method
- Compare results to highlight the best approach

Successfully completing these objectives requires overcoming certain challenges. A focal challenge being finding the optimum threshold for the green filter for the conventional approach, as well as deciding on the best combination of the morphological operations. This is further discussed and explored in the latter part of this report.

II. RELATED WORKS

Fruit counting by detection has been extensively researched in this literature. This section provides a comprehensive discussion of classification schemes used to sort fruits. The flaws of present methods used are discussed. According to this thorough literature assessment conducted, fruit classification is still an unsolved issue.

Segmentation and localization of fruit detection for robotic harvesting has gained significant attention from the research community. Older approaches used colour thresholds that required manual setup. This required manual extraction of colour, shape and texture features, followed by classification and clustering techniques, until deep learning techniques automated these thresholds became more prevalent [8]. The various approaches used differ in terms of the sensing techniques used - [1],[4] used RGB, [17] used LIDAR, [19] used YOLO and [16] used stereoscopic vision to detect fruits.

To classify the pixels of the apple images from the background, back propagation was used in [15]. After employing segmentation, edge detection was thereby used to sort out individual apple samples, after which they were summed to yield total count result. Over 89.5 % of the fruits were successfully identified from 160 testing photos by [16], a system that employs stereoscopic vision to find apples in trees. [17] employs a cutting-edge multi-sensor architecture that effectively maps and identifies individual fruits in a mango orchard. With only a 1.36 % error rate for individual trees, this method requires little calibration.

Faster R-CNN [5] which combines both region proposal and classification phase into a single network has been adapted for detecting fruits and vegetables. This method was employed by [17] to count apples, mangoes and almonds, which resulted in F1 scores greater than 90 %. Hence, this was the preferred method in this experiment.

The majority of articles reviewed thus far have aimed towards precise fruit counts. After post-processing the output from the neural networks, they acquire the counting results. While the detection portion of the processing pipeline has experienced fast advancements, culminating in the deployment of Faster R-CNN in orchards, the counting portion has been mostly overlooked.

In a typical image classification procedure, the goal is to identify if an object is present or absent, while in a counting problem, the goal is to determine how many instances of the object are visible in the scene. The process of classifying fruits includes a number of tasks, including: Grading and sorting of fruit which is a crucial step in the agriculture industry. Fruits must be categorised to ensure they fulfil all quality requirements, whether they are intended for consumption or export. For this experiment, the fruits are classified based on the following criteria: form, size, colour, intensity, and texture. Based on the literature review conducted, several researchers grade and sort fruits and vegetables using multiple techniques. A strategy that works well for one fruit or vegetable might not be the best choice for classifying another. So far, classification techniques have relied primarily on predetermined classification models, with little stage-by-stage classification of fruits.

A conventional strategy, which involved directly manipulating the photos entered into the algorithms, and a machine learning approach, in which a convolutional neural network is taught to recognise the input, were both employed for the detection. The dataset was used with a Mask R-CNN and a Faster R-CNN and converted to COCO format to be used in YOLO. To make it easier for the algorithm to distinguish between the surrounding elements and the apples, the data set applied to both approaches was limited to red apples on trees. Approaches such as Faster R-CNN and YOLO that can provide minimum error were chosen for detecting apples in this experiment. The analysis of the results revealed that the conventional method with no split had the lowest rate of detecting error. In the detecting stages, the YOLO framework for the machine learning technique has the lowest percentage rate of error.

III. DATA ACQUISITION

We approach the challenge of precisely counting apples by noting the following observations: The apples in the dataset are (1) sporadically scattered throughout the entire image, (2) frequently grouped together, and (3) uneven distribution of cluster sizes.

The system analysed red apple identification and classification from a dataset of real images [10] containing complicated disturbance information (background was similar to the surface of the apples and trees).The image contains apple clusters that are identified by segmentation method, and a counting method computes per-image counts. Cluster of apples will be apparent in multiple images. The contour images were created using pretreatments such as noise removal, area filling, and so on. A model that identifies a set of general but distinguishing image characteristics of the apple was used to extract the features. The location of each cluster must be monitored, and counts must be avoided to prevent double counting. The feature extractor's output was mapped to labels using training data for the model. This method is prone to false positives and therefore requires the counting algorithm to reject any false positives.

Computer vision has been increasingly used within agriculture as computing and imaging technologies have improved. Applications within agriculture include but are not limited to disease prevention, yield estimation, ripeness estimation, weed detection and livestock monitoring which has lead to the curation of many datasets of images for computer vision within agriculture. To make a system that can count and detect apples we require a dataset containing many images of apples. We envision however that out system for counting apples may be used to estimate the yield within an orchard, we therefore ideally require a dataset of apples with context of their surrounding, i.e still attached to the branches of a tree.

The most suitable dataset for the task was deemed to the be MinneApple dataset [10]. The MinneApple dataset was created to provide a large dataset of high-resolution images of apples in an orchard environment. Images were taken using a standard mobile phone camera and have been annotated by a human. The objects are annotated as polygon masks for each apple instance. In total the dataset contains 40,000 apple instances from 1000 images. Our needs dictated that we would not use the full dataset, a large subsection within the dataset contains small bounding-box style images that may be useful for classification problems over detection and counting.

FIG. 1. An example image from the MinneApple dataset [10]

In order to compare conventional methods with machine learning, the same MinneApple subset must be used against both for validation. The chosen subset is 32 images of full apple trees in the orchard, which have the number of apples labelled in a separate .txt file corresponding to each image. Results of this comparison are detailed in section VI. It should be noted that labels for these images count only those still on the tree as apples that are desirable to harvest; fallen apples may have been there a long time and begun to rot. Hence the algorithms aim to

avoid counting apples which have fallen to the floor.

The conventional algorithm of course requires no training data, as it manipulates each image directly to separate and count instances of apples. Hence, the only input is the validation set of 32 images and their labels. The images are then run through the algorithm, the number of detected apples is counted and compared to the labelled number as a percentage of labelled apples detected.

In contrast, the machine learning approach requires a collection of images for which the neural network weights can be optimised for. These images must have associated annotations whereby the images object information can be used by the model.

The dataset had been originally used with a Mask R-CNN and a Faster R-CNN and the annotations where incorrect for use with the YOLO model as they were in the COCO format. For the subset of the dataset that we intended to use the annotations must be converted. A python script using the Pylabel package had to be employed to read the annotations from a .JSON file, create a text file for each image and write the instance details in the correct format.

IV. METHODOLOGY

A. Approach 1: Conventional

Conventional approaches to machine vision entail direct manipulation of images input to the algorithm. Our method has two distinct processing stages, applied after the image set is recorded in an array along with a corresponding labels array.

For our first attempts, we tried using the thresholding methods from the tutorial and the lecture [3][2] which are the adpative, manual and Otsu. We set the parameters for these to have the highest possible accuracy.The best method among those was Otsu, but when we looked at the resulting binary image, the difference between the apples and the remaining is not marked enough (FIG.2). Therefore, we decided to employ a tailormade thresholding method.

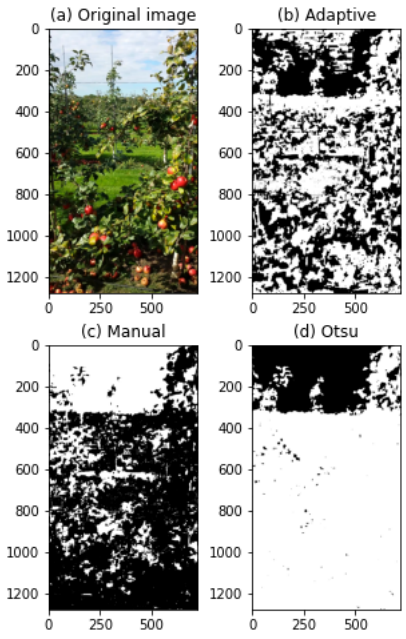


FIG. 2. Comparison between the three considered thresholding methods at first. Even if Otsu gives the best results, many apples are not detected, but the sky and some parts of the tree are.

We made the decision to limit our scope to red apples that are on the tree only. They are more easily distinguished against the green of the tree, and the labelled dataset used for the machine learning approach considered only those, so we can compare the approaches. The first stage of our method is therefore the application of a custom colour filter to remove “non-red” pixels. This choice and the process will be detailed in section V A. Two algorithms have been

developed which differ only slightly. While both filter out "non-red" pixels, one splits the image into ten segments before setting the threshold independently for each segment. The intention was to increase accuracy, section VI, details why the non-split algorithm was kept in use as well.

The colour filter outputs in white the original red regions on a black background. At this point, we are already able to spot in one look the apples and have a brief idea of the amount (as we can see in FIG. 6 original images after the threshold is applied). Binary images are much more easily processed by machines than the original complexly coloured image. Morphological operations can then be applied to great effect, adding smoothing and separating each of the detection regions.

Finally, the white detection regions remaining are labelled with circles which are counted, layered over the original image and compared to the labelled number of apples. The combined combination of visual and numerical results allows the user to discern if false-positives are causing significant inaccuracy.

B. Approach 2: Machine Learning

Machine learning within computer vision is where a convolutional neural network gets 'trained' to recognise patterns within the input data. A lot of data is required to achieve this successfully. During the training process the network gets exposed to many training examples within the dataset which contain the ground truth bounding boxes and ground truths. The model then makes a prediction for that input image which is compared to the ground truth with a loss function. The loss is then backpropagated back through each layer in the network from the output layer through each hidden layer to the input layer. All layers with the exception of the input layer (which has no weights) will have their weights updated as to minimise the loss. This process is then repeated through many iterations to converge the network to a point with minimal loss. It is important to have a large set of data to prevent overfitting which is where the model gets too optimised for the limited training data, this then impedes its ability to generalise to other object examples that it has not previously seen.

Some important hyperparameters that are involved include learning rate, weight decay and batch size. Learning rate determines the amount at which each weight can be updated in each correction, having a faster learning rate will mean that the loss can converge quicker, but it could also lead to oscillations or divergence. Weight decay can be altered to try counter overfitting, it is implemented with the loss function and penalises the CNN for having weights that are too large. Instead by having smaller weights the network can learn more generalisable patterns. Batch size is commonly altered and is often dictated by the computing resources when training. It is the number of images that the network makes predictions on before there is an update to the weights. It is generally set to 32, 64 and 128 but could go anywhere from 16 - 512.

There are many different convolutional neural network architectures and adaptations. We have decided to explore the use of 2 popular architectures, YOLO and the Mask R-CNN.

1. YOLO

The YOLO architecture has become one of the most widely used object detection algorithms in recent years. The initial release with v1 was in 2015 with a paper titled "You Only Look Once: Unified, Real-Time, Object Detection" [13]. It has since been re-released with many improvements, the first 3 versions from the original author with the remainder being other organisations. For our application it was deemed YOLOv5 was a suitable iteration, the large amount of accompanying documentation meant that it could be more readily implemented than other versions. YOLOv5 unlike previous versions was not released with an accompanying paper, instead it undergoes ongoing development through the Ultralytics GitHub page [11].

The YOLO object detection algorithm uses bounding box annotations and produces a bounding box with a class label as an output as shown in FIG.9. The bounding box is drawn over the desired object and the whole area within is attributed to that class label. This has some advantages over semantic segmentation annotation such as it is quicker to annotate, it will however be subject to noise from the areas within the bounding box that are not actually part of the object in question. Training could therefore require more examples as the neural network weights get optimised.

2. Masked region based convolution neural networks (Mask-RCNN)

a. General principle

This architecture of neural networks called masked region based convolution neural networks (Mask-RCCN) allows instantaneous segmentation problems to be solved: it makes it possible to extract an object in an image and identify it. This method makes it possible to carry out semantic segmentation where each pixel of an image is associated with an class label. [9]

FIG. 3. Example of segmentation in the context of apple detection prior to training using weights trained on MS-COCO

b. Working principle

This machine learning method is based on the R-CNN architecture. This method first extracts interesting regions from the image and then uses these regions as input data for a convolutional neural network. This separation into regions makes it possible to detect several objects of several different classes in the same image.

This architecture is then improved to become the Faster R-CNN architecture. The addition of convolution layers allows the generation of regions of interest and facilitates their classification. Then the region proposal network uses sliding windows of different sizes and ratios to analyse the feature map in depth. These changes improve the accuracy and speed of the architecture compared to R-CNN.

The Mask R-CNN is the next improvement after the Faster R-CNN. This method changes the output of the final model. Indeed, the Faster R-CNN architecture allows to locate distinct objects with a bounding box. The Mask R-CNN architecture allows to segment each instance of an object with a semantic mask. The Mask R-CNN architecture is similar to Faster R CNN but adds layers in parallel to the classification. [9] [18]

c. Application to apple counting

The Pytorch Library was used to utilise the R-CNN for object detection. In order to apprehend the various notions associated with the use of the latter, the tutorial allows to refine a pre-trained R-CNN mask model to detect pedestrians.

Initially the Mask R-CNN has pre-trained weights from the MS-COCO dataset, we then trained the network further with the use of a tutorial [12] on the dataset of apples to optimise it for our application.

The code provided is adapted to meet the needs of our study: the dataset used presents detection masks and associated images.

V. IMPLEMENTATION & EXPERIMENT

A. Conventional

We are using Google Colab for coding, as it enables us to work simultaneously on the same file, perform cloud computing executions, using a GPU. All our scripts, datasets and documents related to this project were stored in a shared Google Drive for the same reasons.

We are using matplotlib for image reading and plotting. It reads images as 3D-arrays of size width*height*3. For each pixel, it has the RGB value as 3 floats numbers. Thus, we handle their manipulation with numpy.

The bottom 17% of each image is cropped as, across the MinneApple dataset, this is the average portion of the image taken up by the floor. With the dataset uploaded, Roboflow [14] allows the visualisation of the annotation locations . FIG. 4 provides the heat map of label placement used to discern this floor percentage value. We can see that no high density label areas are excluded, while we avoid potential detection of many apples on the floor. To avoid spending time working with the floor, we are cropping it out in the very beginning of our process.

The thresholding operation consists in calculating a weight value, w, that is actually our threshold. After a quick look at the dataset, we could postulate that the tree will always take the biggest part of the image and is green. From this, we can design our w as in Algorithm 1. The objective was to separate the apples, that are considered as the only red parts, from the rest (see FIG.5). Thus apples' pixels would have a low green component, which is the reason we turn to white everything that is below the threshold. The risk is low for the sky to be whited too (actually it did not happened, or only very small amounts, which is not a problem thanks to the morphological operations described below), since it is is generally light blue so its green component is big enough (as it gets close to the white). The exact formula for w was found empirically to give the best results with the whole detection process. In this way, all detected apples (at this stage, apples are just red zones from the original image) are rendered as white silhouettes on a black background.

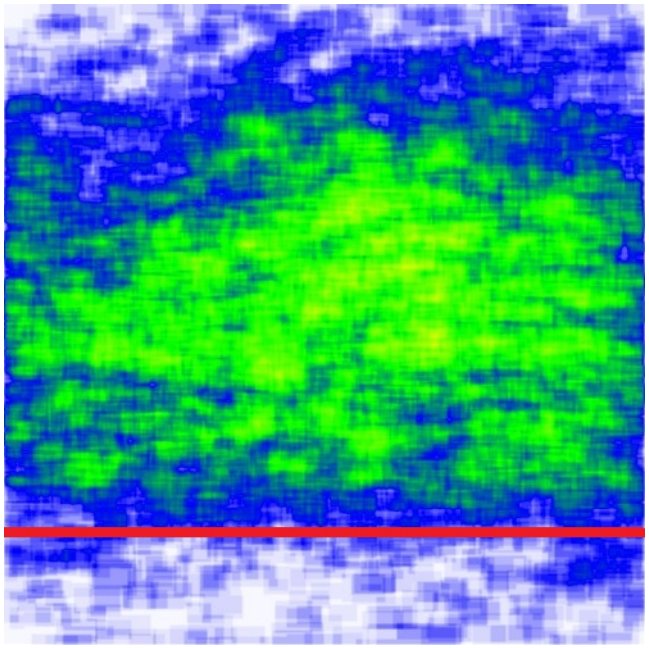


FIG. 4. A heat map of where apples are labelled on images in the MinneApple dataset. Green indicates high density, blue low density, white shows no labels. The red line marks the point at which each image is cropped (the lower 17%).

Algorithm 1 Thresholding method to separate "red" and "non-red" pixels

Input: im: image to filter and crop out the floor
Output: filtered and cropped image

$im2$ ←black image which size is the size of the cropped im
$weight$ ←average[green/(red +2*green+blue)]
#green refers to the green component of a pixel. The average is for all the pixels of im.

for each pixel **in** im **do**
  if proportion of the green component $< w$ **then**
    the pixel in $im2$ at the same position becomes white
  end if
end for

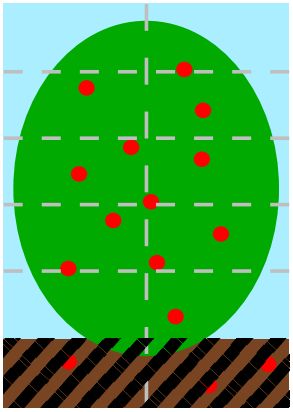


FIG. 5. Schematic of a typical image from the dataset we use. The tree takes most of the space, we also have the sky, the floor and of course the red apples (some are on the floor, thus purposely ignored). The dashed lines represent the segmentation used in the split algorithm.

For the split-image algorithm, the image is cropped, then split into ten segments before a threshold is calculated and applied to each of them. The segments are then merged back together to give a similar output as the non-split method. (Algorithm 2)

Algorithm 1.1 Thresholding method to separate "red" and "non-red" pixels (without cropping the floor)

Input: *im*: image to filter
Output: filtered image

*im*2 ←black image which size is the size of *im*
weight ←average[green/(red +2*green+blue)]
#green refers to the green component of a pixel. The average is for all the pixels of *im*.

for each pixel **in** *im* **do**
  if proportion of the green component $< w$ **then**
    The pixel in *im*2 at the same position becomes white
  end if
end for

Algorithm 2 Splitting-thresholding-merging method

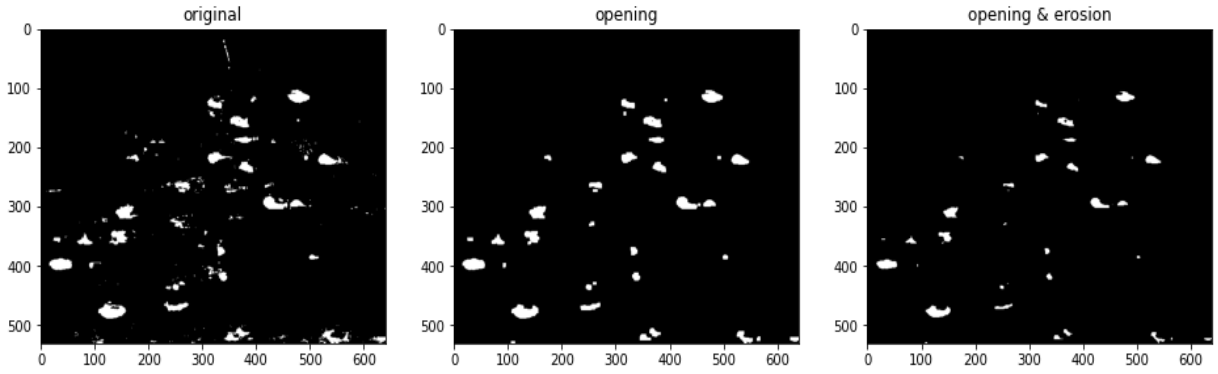
Input: *img*: image to filter
Output: filtered and cropped image

img ← *img* without the bottom 17% #cropping the floor
Split *img* into 10 parts

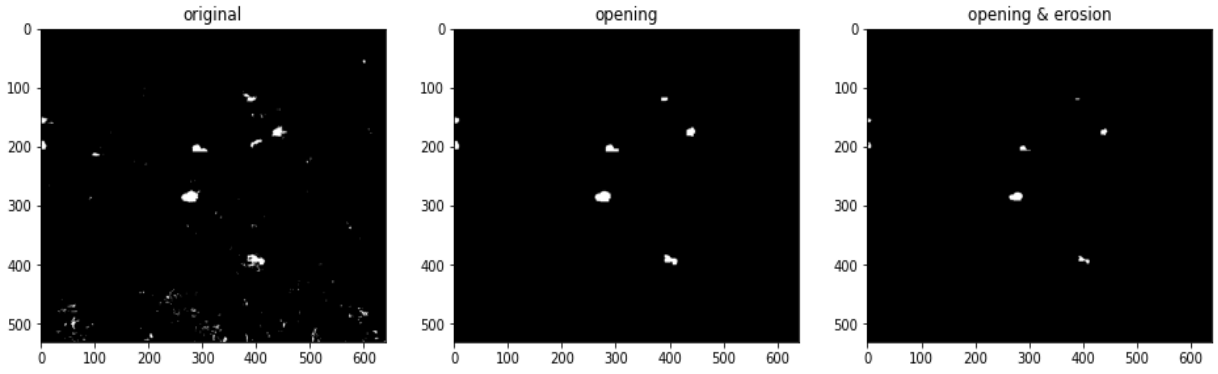
for each part **do**
  Perform Algorithm 1.1
end for
Merge the 10 parts #to get the full filtered and cropped version of *img*

Once the above has been applied, both conventional algorithms perform one round of opening, then closing on each image. Opening allows better separation of white points of interest on a black background compared to closing [7]. Applying erosion after this better clarifies each point to minimise the number of closely bunched apples detected as a single apple. These morphological operations are displayed in FIG. 6. They have been achieved using the python cv2 package and google.colab.patches to adapt it.

It is clear from FIG. 6 that applying a different threshold to ten segments in the image results in much harsher designation of green overall, with comparatively few apple detections remaining after morphological operations. This due to the difficulty in finding an adapted w formula here. Our previous solution worked because the tree-sky surface distribution was very similar from one image to another. With the splitting algorithm, this distribution varies a



(a) No Image Segmentation



(b) Image Segmentation Used

FIG. 6. Here we can see the image output from the green filter function, labelled as "original" on both (a) and (b). Opening and erosion are then applied in sequence as shown.

lot from a segment to another, as shows FIG.5. This leads the w values to be way lower than previously, so less apples are detected (see Algorithm 1.1). Due to the diversity of the segments, finding a universal w expression does not seem to be possible and so is trying to find specific thresholds for each segments (i.e. w_1 for segment 1, w_1 for segment 2,...). Even if the images look a lot like each other, when zooming to a segment scale, they happen to be very different. However, this splitting method still has interesting advantages that will be expanded in the next section.

Each of the remaining white spots are then converted to a white circle and layered over the original image to show where the algorithm has detected apples. Circles are counted and compared to the number of labels for the image to generate a detection percentage.Obviously this percentage is inaccurate if the algorithm has any false-positive detections. The ndimage library from the scipy package was used here.

B. Machine Learning

1. YOLO

The GPU required for training was gained by using Google Colab. The converted dataset could be easily loaded from the Roboflow server for use within Google Colab without the need upload images to the current runtime. The dataset contained 331 images which collectively contained 12,285 apple instances for the network to train on This was split into 75% training images, 15% validation and 10% remaining for the comparitive test. YOLO requires the input images from the dataset to be accompanied with their own annotation file, this comes in the form of a .txt file, this must be the same name as the associated image aside from the .txt suffix instead of .jpg/png. Within this text file, each row represents one object instance annotation within the image, the row starts with a class identifier and is followed with 4 numbers in the range *0 - 1*. These four numbers represent the centre x and y coordinates and the

width and height of the box with the image dimensions normalised to 1. The image dataset and annotations must be accompanied with a .YAML file. This is a configuration file that contains the training setting and hyperparameter options for training the model and information about the dataset. The path to the training, validation and test datasets are contained within in this file as well as the class label map. As our requirement is to only train for and detect one class, nc = 1 and names = [”apple”]. The class identifier in the annotation file will correspond the the element number in the names array, hence every row is prefixed with a *0*.

Prior to training the images are resized to 640x640 pixels and training was carried out for 85 epochs with a batch size of 16. This was enough training iterations for the loss to decrease sufficiently without causing the model to overfit to the dataset. Learning with the default hyperparameters was sufficient with the learning rate = 0.01 and a weight decay of 0.0005.

Key metrics when evaluating the success of a machine learning system include, precision, recall - which can be used to calculate mean average precision (mAP) and box loss. Precision is a measure of the number of true positive predictions divided by the number of total number of predictions. This shows that the model is good at identifying an object but it wont take into account any false negatives where the model failed to identify an apple. Recall on the other hand would be measure of how many of the apples the model correctly managed to identify, but it doesn't take into account the false positives. mAP is a metric that combines precision and recall to judge the overall accuracy of the system. Box loss is measure of how well the model predicted the bounding boxes, this is then used to make corrections to the weights. As shown in FIG.7, after 85 epochs the model had a mAP0.5 of 0.814 which equated to an accuracy of 81.4% which is calculated from a precision 0.845, a recall 0.733. The model has a box loss which started at approximately 0.13 was reduced to 0.0392 indicating that the loss had converged. The graphs in FIG.7 show that further training may have diminishing return as all recall, precision, mAP and box loss had plateaued indicating no further improvements could be gained with the current dataset and algorithm.

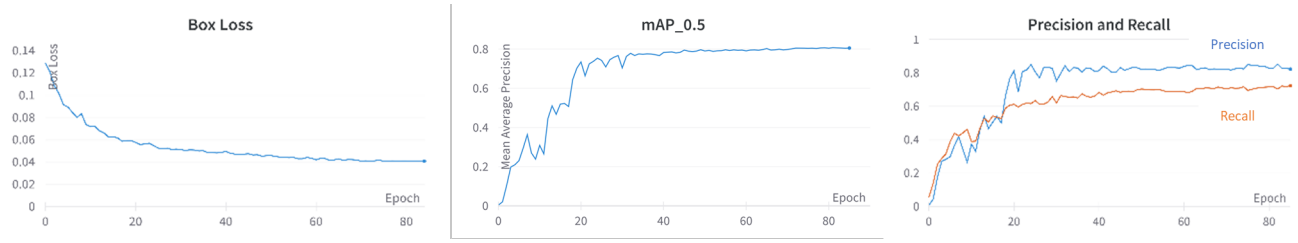


FIG. 7. Training graphs for YOLOv5 after 85 epochs. From left to right the graphs show box loss, mean average precision (mAP) and, precision and recall.

2. Mask-RCNN

The Mask R-CNN was also trained with Google Colab. To train the Mask R-CNN deep learning model, we used a different dataset than with YOLO as it requires two distinct sets of images: the apple tree images and the associated masks as presented in figure 8

We will refine a pre-trained R-CNN mask model. The weights of the neurons associated with the pre-training are provided by the Pytorch tutorial and come from a training performed with the Coco library (a large-scale object detection, segmentation, and captioning dataset [6]).

The training carried out in this study aims at refining the provided model by taking into account the MinneApple database [10] which contains 670 images and its 670 associated masks. Training was carried out over 50 epochs.

FIG. 8. Image of the database with the associated mask

VI. RESULTS & EVALUATION

Regarding the two conventional methods, one can see in FIG. 9 (e) and (f), using segmentation leads to far fewer apples detected overall; many prominent apples in the image are not detected. However the segmentation approach leads to far fewer false-positives or instances of multiple detection points on a single apples. While avoiding segmentation, leads to fewer apples missed, the high number of multiple detections on single apples with this method not only results in cases where more apples are detected than are in the image, it invalidates a purely numerical output as you cannot be sure that the apples detected are separate apples before the annotated image is viewed. For example, FIG. 9 (e) contains 32 points of detection, matching the number labelled. While on the surface this looks to be a perfect success, the image shows that some apples are missed and some are detected multiple times. Distinguishing multiple bunched apples is where the machine learning methods truly outmatch our conventional algorithms.

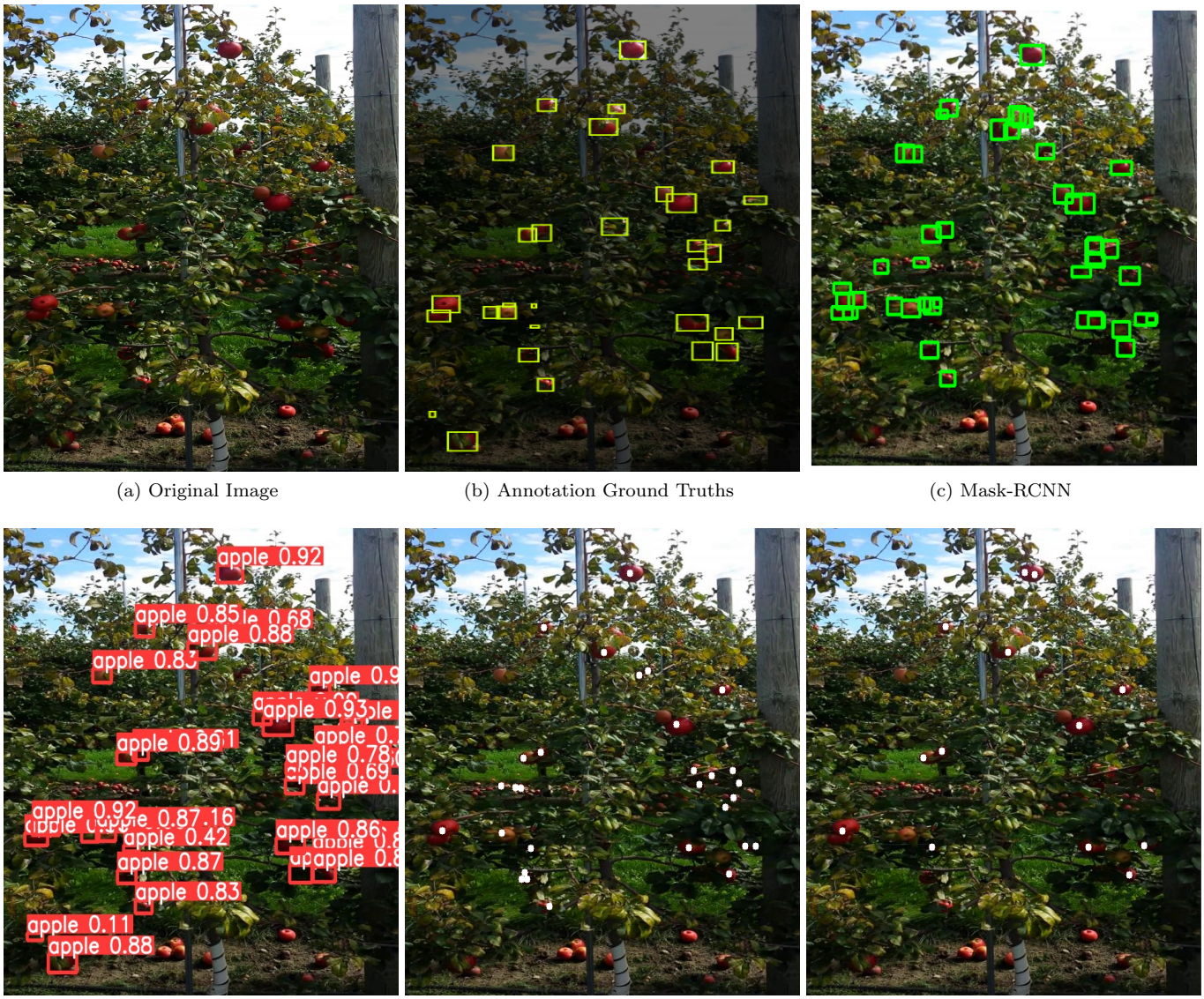
Utilising YOLO as a CNN model to detect apples in an orchard has proven itself to be a good method. The training dataset although relatively small with the number of images contains a sizeable number of object instances which has allowed the network weights to optimise well. FIG.9 shows that both the YOLO and Mask R-CNN have successfully identified most of the apples annotated. The bounding boxes in the image for YOLO inference (d) are accompanied by the class label and an associated confidence score that the models prediction is correct.

A comparison was made between all of the methods tested in this paper. A dataset of 32 images was created and each method used these images to make a prediction for the number of apples in the images. All of the images in this dataset were new images that both machine learning models had not been exposed to in their respective training processes.

Labelled Number of Apples	**Conventional: No Split**	**Conventional: Split**	**ML: YOLO**	**ML: Mask R-CNN**
1246	811	295	1279	1143

TABLE I. Overall comparison of the number of apples labelled across the 32 image dataset and how many were detected by each algorithm.

Table I shows that the machine learning approached has a significantly improved accuracy when compared to conventional computer vision methods. The YOLO model performed best having only overestimated the number of apples in the dataset by 2.6%, the Mask-RCNN performed slightly worse having underestimated by 8.2%. Comparatively, both conventional methods performed poorly with the segmentation detecting only 23.7% of apples. The non-segmentation method theoretically detected 65.1% of apples but looking at the direct results per image there are multiple instances of overestimation. This implies less credence should be placed on the overall result for this method. FIG. 10 shows that the error rate per image was much higher for conventional methods, with YOLO outperforming Mask R-CNN in terms of accuracy.



(a) Original Image (b) Annotation Ground Truths (c) Mask-RCNN

(d) YOLO Detection (e) Conventional: No Segmentation (f) Conventional: Segmentation

FIG. 9. Here we see the output from each of the four algorithms with same input image. (a) shows the unprocessed image, (b) shows the 32 labels on the image. (c) and (d) display the two machine learning algorithm results, while (e) and (f) show the conventional algorithm results for non-segmentation and segmentation respectively.

VII. CONCLUSION & FUTURE WORK

Having investigated the various machine learning algorithms for detecting apples at an orchard, it can be implied that machine learning based approaches produce more accurate results compared to conventional methods. The data set was organised into various types of apples, whereby this report solely focused on red apples which were in the trees. This was to avoid confusion by the algorithms if the green apples were also considered, as the algorithm perhaps would have found it more difficult to differ between the green of the trees and the fruit. Therefore, sorting the data images out by preferred styled ensured this challenge to be mitigated.

The conventional method was analysed with both split and no split manipulation and it was found that the method with no split may have produced higher frequency of error at lower rate of error, however the rate of error with the split was much higher, making that method less desirable. similarly, with the machine learning approach, two architecture styles were delved into: YOLO and Mask R-CNN. From the results of these two styles, it was found that the YOLO algorithms produced much lower levels of error compared to the MASK R-CNN, making it a more suitable approach for the purpose of this task.

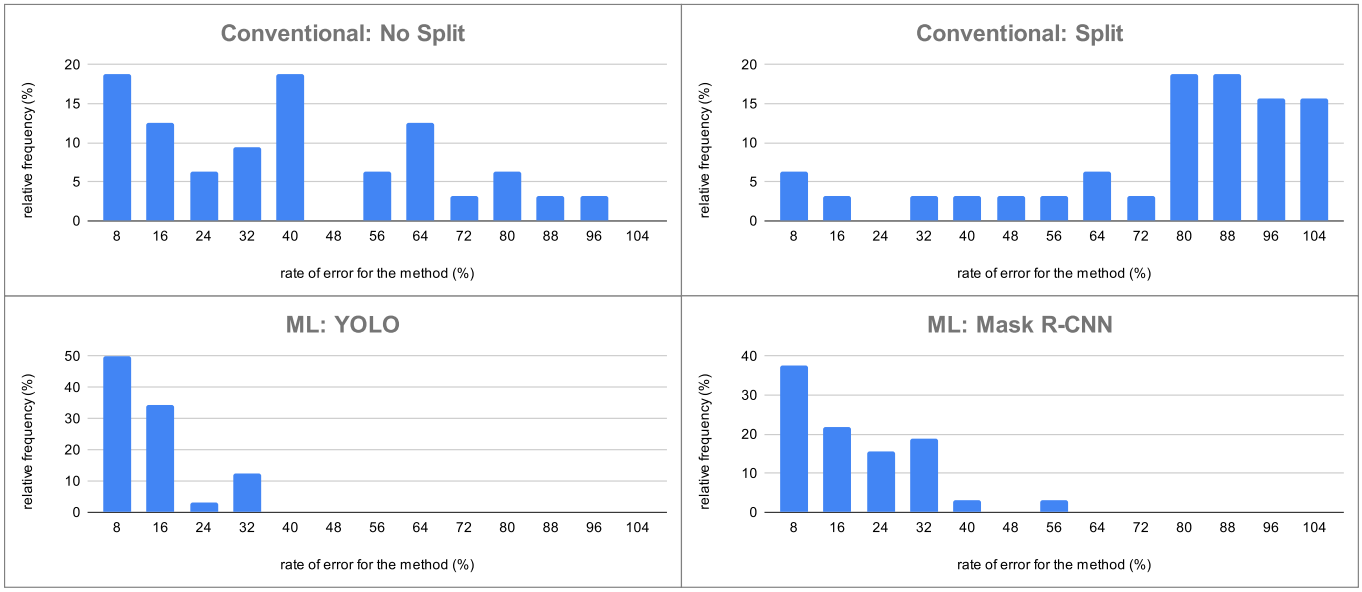


FIG. 10. Relative frequency histograms for each algorithm. These display the percentage detection error for an image on the horizontal axis, the vertical axis shows the percentage of images where that error rate occurred.

For comparisons between the conventional method and machine learning method, it can be stated that the machine learning approach is more suited as the accuracy of the depth of the machine vision is much higher than the capabilities of manipulating the images. This investigation may have been concluded, however the essence of this work merely touches the surface at which the machine vision algorithms are capable of doing. If given the time and opportunity, this analysis could be explored and elaborated for increasingly new tasks within the agricultural industry. The future works section below demonstrates these potential avenues.

Future work: Crop yield estimation to enable precision agriculture is an important task in apple orchard management. There are difficulties in designing dependable, automated counting systems that are particular to the task at hand. Uneven lighting, noise, obscured objects, and clumped objects are a few challenges in counting encountered during this experiment. A robot equipped with a camera programmed to move around the farm and predict farm yield can be used to increase yield estimation. The most common source of error is difficulty counting fruit clusters and counting apples on the ground. To count clustered apples precisely and prevent counting of fallen apples, a method to generate training data synthetically can be used.

[1] Bargoti, S. and Underwood, J. [2017], Deep fruit detection in orchards, *in* '2017 IEEE international conference on robotics and automation (ICRA)', IEEE, pp. 3626–3633.

[2] Chen, D. [2022*a*], 'Tutorial_four.ipynb', Blackboard.

[3] Chen, D. [2022*b*], 'Week 4: Image thresholding and binary morphology', Blackboard.

[4] Chen, S. W., Shivakumar, S. S., Dcunha, S., Das, J., Okon, E., Qu, C., Taylor, C. J. and Kumar, V. [2017], 'Counting apples and oranges with deep learning: A data-driven approach', *IEEE Robotics and Automation Letters* **2**(2), 781–788.

[5] Chen, Y., Li, W., Sakaridis, C., Dai, D. and Van Gool, L. [2018], Domain adaptive faster r-cnn for object detection in the wild, *in* 'Proceedings of the IEEE conference on computer vision and pattern recognition', pp. 3339–3348.

[6] *Coco : Common objects in context* [n.d.], Accessed Jan 02, 2023 [Online].
URL: *https://cocodataset.org/*

[7] *Computer Vision: Opening and Closing* [n.d.], Accessed Jan 01, 2023 [Online].
URL: *https://cvexplained.wordpress.com/2020/05/18/opening-closing/*

[8] Gongal, A., Amatya, S., Karkee, M., Zhang, Q. and Lewis, K. [2015], 'Sensors and systems for fruit detection and localization: A review', *Computers and Electronics in Agriculture* **116**, 8–19.

[9] He, K., Gkioxari, G., Dollár, P. and Girshick, R. [n.d.], 'Mask r-cnn'.
URL: *https://openaccess.thecvf.com/content$_I$CCV$_2$017/papers/He$_M$ask$_R$ − CNN$_I$CCV$_2$017$_p$aper.pdf*

[10] Häni, N., Roy, P. and Isler, V. [2019], 'Minneapple: A benchmark dataset for apple detection and segmentation'.

[11] Jocher, G. [2020], 'YOLOv5 by Ultralytics'.
URL: *https://github.com/ultralytics/yolov5*

[12] PyTorch [n.d.], 'Torchvision object detection finetuning tutorial'. Accessed: 2022-11-20.
URL: *://pytorch.org/tutorials/intermediate/torchvision$_t$utorial.html*

[13] Redmon, J., Divvala, S., Girshick, R. and Farhadi, A. [2015], 'You only look once: Unified, real-time object detection'.
URL: *https://arxiv.org/abs/1506.02640*

[14] *Roboflow* [n.d.], Accessed Jan 02, 2023 [Online].
URL: *https://roboflow.com/*

[15] Shahin, M., Tollner, E., McClendon, R. and Arabnia, H. [2002], 'Apple classification based on surface bruises using image processing and neural networks', *Transactions of the ASAE* **45**(5), 1619.

[16] Si, Y., Liu, G. and Feng, J. [2015], 'Location of apples in trees using stereoscopic vision', *Computers and Electronics in Agriculture* **112**, 68–74.

[17] Stein, M., Bargoti, S. and Underwood, J. [2016], 'Image based mango fruit detection, localisation and yield estimation using multiple view geometry', *Sensors* **16**(11), 1915.

[18] *viso.ai Deep Learning : Mask R-CNN* [n.d.].

[19] Wu, D., Lv, S., Jiang, M. and Song, H. [2020], 'Using channel pruning-based yolo v4 deep learning algorithm for the real-time and accurate detection of apple flowers in natural environments', *Computers and Electronics in Agriculture* **178**, 105742.