UFMFRR-15-M Machine Vision
Group Report on Apple Counting in Orchards

Word count: 4761 words

Group: **WED1300 7**

Adithya Raghu 23080415
Chaojun Guo 23080457
Dexter Fernandes 23080425
Jingyu Han 23080449
Youssef Alboraei 23080413
Yulong Fang 23080410

Contribution Table

No	Name	Contribution to Project	Contribution to Report	Signature
1	Dexter Fernandes	Dataset exploration, Minneapple extracting bounding boxes and saving to XML/Yolo labels, Approach A algorithm research and tuning, Quantitative Analysis - Approach A, Qualitative Analysis	Data Acquisition and Datasets(100%), Approach A Experiment and Implementation (50%), Results and Evaluation Approach A(33%), Conclusion and Future Works (30%)	Dexter Fernandes
2	Youssef Alboraei	ML Literature Review, ML Data Preparation, ML Model Training, ML Model Testing, ML Results Quantitative and Qualitative Analysis, ML Results Preprocessing and Visualisation	Related Work (20%), Methodology Approach B (33%), Implementation Approach B (90%), Results and Evaluation Approach B (100%)	Youssef Alboraei
3	Adithya Raghu	Approach A algorithm - contouring of detected units using HSV and draw bounding boxes for the same	Introduction(50%), Related Work(20%), Methodology Approach A(50%), Experiment and Implementation Approach A(20%), Results and Evaluation Approach A(33%)	Adithya Raghu
4	Jingyu Han	ML Approach, ML Model Training, ML literature review	Related Work(30%), Approach B Methodology(37%), Approach B Implementation(10%)	Jingyu Han
5	Chaojun Guo	Approach A algorithm(hough circle detection) research and tuning	Approach A Implementation(50%), Approach A Methodology(50%), Related works(10%), Results and Evaluation A(33%)	Chaojun Guo
6	Yulong Fang	ML Approach, Dataset preparation, ML literature review, ML Results Postprocessing	Introduction(50%), Related Works(20%), Methodology B(30%), Conclusion and Future Works (20%)	Yulong Fang

Chapter 1

Introduction

In today's digital age, machine vision technology plays a crucial role in scientific and engineering research, especially in agriculture. Its use in orchard management has transformed traditional methods, offering innovative solutions for agricultural challenges.

Accurate fruit counting enables efficient resource management, allowing farmers to adjust practices based on actual fruit load, optimising irrigation, fertilisation, and other resources. Monitoring fruit counts at different growth stages provides insights into crop development, aiding in understanding growth patterns and addressing anomalies. It also helps determine the best harvest time, contributing to taste, shelf life, and nutritional value. Additionally, fruit counting data supports agricultural research, aiding scientists in developing new varieties and improving cultivation practices for better yields and quality.

However, estimating fruit counts from images poses several formidable challenges. Foremost among these is the identification of individual fruits within clusters, especially when they are partially obscured in images captured under varying lighting conditions. Additionally, the detection of overlapping fruits and those hidden behind leaves and branches presents a significant challenge (see Figure 1.1). It is equally crucial to prevent false detections by distinguishing leaves that may visually resemble fruits. These challenges collectively underscore the complexity of accurately counting fruits, particularly apples, and emphasise the pressing need for advanced solutions in image recognition and analysis (Jiménez, Ceres, and Pons 2000).

These challenges will be further discussed throughout this report by exploring the application of machine vision in apple counting within orchards, comparing a traditional image processing approach with a machine learning-based approach to assess their effectiveness and limitations.

The traditional image processing approach, grounded in established computer vision techniques, includes image pre-processing, feature extraction, and object detection. In this project, we initially employ these methods for apple counting, involving steps such as de-noising, segmentation, and shape analysis. While this approach benefits from simpler algorithms, less reliance on extensive training data, and faster computational speeds, its adaptability to varying lighting conditions and occlusions remains limited.

In contrast, machine learning methods, particularly deep learning models like Convolutional Neural Networks (CNNs), offer enhanced adaptability to complex scenarios. The strength of these models lies in their ability to autonomously learn features from images, eliminating the need for manual feature extraction. However, they require significant amounts of labeled data for training and are computationally more complex than traditional image processing methods. This report evaluates the differences between these approaches in terms of accuracy, generalisation capability, and computational efficiency. (Behera et al. 2018)

Figure 1.1: Samples from the MinneApple Dataset (Häni, Roy, and Isler 2020) Highlighting the Discussed Challenges

Chapter 2

Related Works

The study of machine vision for agricultural applications, specifically in apple detection and counting, has seen considerable contributions from various researchers. This chapter reviews significant studies in this area, highlighting their methodologies and findings.

Steven W. Chen et al. have developed a deep learning-based pipeline for apple detection and counting, addressing challenges such as varying illumination and occlusion from foliage and neighbouring fruits (Chen et al. 2017). Their approach uses a blob detection neural network and a count neural network, demonstrating effectiveness in datasets of oranges and apples, especially in handling heavily occluded fruits.

Guantao Xuan et al. utilise the YOLOv3 algorithm for apple detection, enhancing images to capture features efficiently under various environmental conditions (Xuan et al. 2020). Their supervised learning method involves manual annotations for detecting apples obscured by foliage, using YOLOv3 for detecting apples of varying sizes.

Häni et al. introduce the MinneApple dataset for advancing fruit detection, segmentation, and counting in orchards (Häni, Roy, and Isler 2018). This dataset includes detailed annotations using polygonal masks for precise object detection. They also conduct an algorithmic analysis using this dataset, evaluating models like Faster RCNN and Mask RCNN, providing benchmarks for object detection approaches.

In addition to detection and counting, yield estimation is critical in this domain. Häni et al. address the need to track fruits across image sequences and propose solutions for yield estimation (Häni, Roy, and Isler 2020). Their approach combines unsupervised GMM detection methods and CNN counting, achieving high accuracy in yield estimation.

Marzoa Tanco et al. employ classical computer vision techniques for apple detection, including steps like denoising, segmentation, and shape analysis (Marzoa Tanco, Tejera, and Di Martino 2018). These methods, while beneficial in simpler scenarios, face limitations in adaptability to complex environmental conditions.

These studies collectively contribute to the field of machine vision in agriculture, providing valuable insights and methodologies for apple detection, counting, and yield estimation, setting a foundation for future research and applications in agricultural automation.

Chapter 3

Data Acquisition and Datasets

As part of our methodology, a thorough exploration of various datasets was undertaken to identify the most suitable one for our project. Below is a summary of our findings:

Dataset Name	**Labeling Format**	**Observations and Considerations**
MinneApple	polygon masks (Bounding boxes extraction required)	Presents a challenging scenario but offers valuable insights for complex analysis.
ACFR Fruit Dataset	Circular labels (Stored in .csv format)	Requires conversion of circular labels to bounding boxes; more challenging than MinneApple.
Basic Fruit Detection Dataset	PASCAL VOC format	Simplistic for our project's assessment needs; suitable for benchmarking and initial experimentation.
OpenImages V7	Diverse object types (Subset download necessary for apples)	Contains a wide array of objects, many of which are irrelevant to our specific task.
Embrapa Apple Dataset (ADD 256)	JSON format	Represents an even more complex challenge compared to the ACFR Fruit Dataset.

Table 3.1: Comparative Analysis of Datasets for Project Evaluation

We opted for the Minneapple dataset as it met our learning objectives. The dataset is quite versatile, presenting the challenges of varying illumination conditions, occlusion scenarios, apples captured from different angles, and apples at different stages of ripeness from different orchards. The images were extracted from video footage shot across various orchard sections using a standard Samsung Galaxy S4 cell phone. The collection process involved moving the camera horizontally along a tree row at a pace of around 1 m/s, with every fifth image selected from these video sequences (Häni, Roy, and Isler 2020).

Data capture was spread across multiple days to introduce diverse illumination conditions. The ten training datasets were strategically sampled from six different tree rows, showcasing either the front (sunny) or back (shady) side. From this selection, 670 images of resolution 1280×720 pixels were randomly chosen and annotated. These images depict various apple varieties in different growth stages. Annotations for the detection and segmentation datasets were meticulously done using the VGG annotator tool. The labelling approach used polygons to identify fruits on foreground trees, leaving untagged those on the ground and background trees. (ibid.).

We focused on using the MinneApple detection subset, primarily because it offers ground-truth detection labels, an attribute not present in the counting and testing subsets. Such labels were necessary to apply

machine learning techniques, which tackle the counting task as a detection problem. In order to focus our analysis, we specifically selected images showcasing red apple orchards, thereby excluding images featuring green apples. This decision narrowed our dataset to 605 out of the original 670 images.

The rationale behind this choice was to mitigate the potential for confusion that could arise from the detection of green apples against a leafy background. This issue is particularly pertinent for the image processing algorithm, which heavily relies on colour-based thresholding. By simplifying the dataset in this manner, we aimed to create a more equitable basis for comparison between the two proposed approaches. Moreover, this simplification is anticipated to streamline the rapid prototyping and parameter tuning processes for both our image processing and machine learning algorithms, while also fostering a more comprehensive understanding of the methodologies involved.

To maintain result integrity, we randomly split the chosen images into an 80-20 train-test split—504 images for training/validation and 101 images for testing and comparison between both techniques.

Chapter 4

Methodology

4.1 Approach A – Image Processing

4.1.1 Initial Colour Analysis and HSV Thresholding

We begin with a thorough colour analysis of red apples, utilising an online HSV Detector tool to identify the precise HSV colour range. This preliminary step is crucial in setting the upper and lower bounds of the HSV threshold, thus allowing for accurate differentiation of red apples from the green foliage background. The colour threshold consists of values in the Red, Green, and Blue spectrum, which are pivotal for precisely identifying red apples in later processing stages.

4.1.2 Image Processing Techniques

Our image processing sequence involves several critical steps:

1. The original images are first processed using the colour threshold to generate images containing only objects within the specified red colour range. This step is vital for narrowing down the area of interest in each image.
2. These images are then converted to grayscale, followed by the application of Gaussian blurring (`cv2.GaussianBlur`) to reduce noise and smooth the image.
3. To further enhance contrast and highlight the features of apples, we use the `cv2.divide` function along with inversion operations. These techniques are particularly effective in emphasising the regions of interest.
4. A series of morphological operations, including closing, erosion, and dilation, are then performed. These steps are designed to fill small holes and gaps in the image, thereby rendering the target apples in a more complete state for effective counting.

4.1.3 Circular Hough Transform for Apple Detection

The Circular Hough Transform (CHT) is employed as a key step in isolating apple-like circular features from the rest of the image. This process involves fine-tuning parameters such as the canny edge threshold, the range of circle radius, and the distance between detected circles. By adjusting these parameters, the algorithm accurately locates the centres and radii of circular features, which correspond to apples.

4.2 Approach B – YOLOv8

In advancing machine vision for apple detection, we identified YOLOv8 as a highly capable algorithm, suited to the demands of this project. This section provides insights into our choice of YOLOv8, its structural intricacies, and the essential pre-processing and post-processing techniques we implemented.

4.2.1 Motivation behind YOLOv8 Selection

YOLOv8 stands out for its computational efficiency, a crucial consideration given the resource constraints within the scope of this coursework project. Its streamlined, single-shot detection mechanism notably reduces model complexity, offering a balance between performance and computational demand. This efficiency is particularly beneficial for processing large datasets or deploying models in resource-limited environments.

Additionally, the architectural simplicity of YOLOv8 aligns well with the educational aims of the project. It offers an accessible entry point into the complex world of object detection and counting, allowing for a deeper understanding of underlying concepts without the overhead of more intricate models. The active and supportive YOLO community further enhances this learning experience, providing extensive resources and guidance that have been instrumental in navigating the challenges of implementation.

Moreover, the MinneApple dataset, central to our investigation, presents unique characteristics that have not been extensively explored with YOLOv8, to the best of our knowledge. Applying this algorithm to such a dataset offers a novel avenue for academic exploration, potentially unveiling new insights into YOLOv8's adaptability and performance across diverse datasets.

4.2.2 YOLOv8 Structure

The structure of YOLOv8, as depicted in Figure 4.1, is composed of three primary sections: the backbone, neck, and head (detect). The backbone functions as the foundation, down-sampling the input to various scales (P1 to P5), each processed by a C2f module. This module is designed to expand the detail captured in feature extraction, a critical step for accurately identifying objects of different sizes and complexities.

The neck of YOLOv8 is tasked with fusing these down-sampled features, specifically aiming to enhance the semantic information in the higher-level features (P3 to P5). This fusion is essential to maintain the integrity and precision of object detection, especially when the input undergoes significant down-sampling.

In the head section of YOLOv8, the decoupled structure plays a pivotal role. It separates the tasks of object classification and bounding box regression. This division allows for more accurate and refined detection of objects, accommodating the varying sizes and complexities found within images (King 2023) (Wang et al. 2023).

4.2.3 Pre-Processing

Pre-processing in YOLOv8 includes a series of steps aimed at optimizing the model's performance (illustrated in Figures 4.2, 4.3, 4.4). Key among these is the modification of the HSV (Hue, Saturation, Value) properties of the images. Adjusting the hue, saturation, and value plays a critical role in achieving uniform lighting conditions across all images, which is essential for consistent and accurate object detection (Sural, Qian, and Pramanik 2002).

Other pre-processing techniques such as image rotation, translation, scaling, and shearing are employed to introduce variability and improve object alignment. This enhances the model's ability to recognize

and accurately detect objects in various orientations and sizes. The mosaic technique, which combines several images into one complex scenario, challenges the model further, improving its capacity to handle diverse and complicated detection tasks (Adjobo et al. 2023).

4.2.4 Post-Processing

In the post-processing stage, the focus is on refining the model outputs for practical application. This includes the decoding of features from the output tensor, which involves extracting detailed information like class confidence and object coordinates. Confidence filtering is applied to eliminate detections with low confidence scores, ensuring only the most probable detections are considered. Non-Maximum Suppression (NMS) is then used to remove any redundant bounding boxes that overlap significantly, retaining only the most accurate detection for each identified object (DeepLearningAI 2023).

The final stages involve transforming the coordinates of the predicted boxes to align with the original image's coordinate system and aesthetically adjusting the drawing of prediction boxes. These steps are crucial for visually representing the detection results on the original images in a clear and interpretable manner.

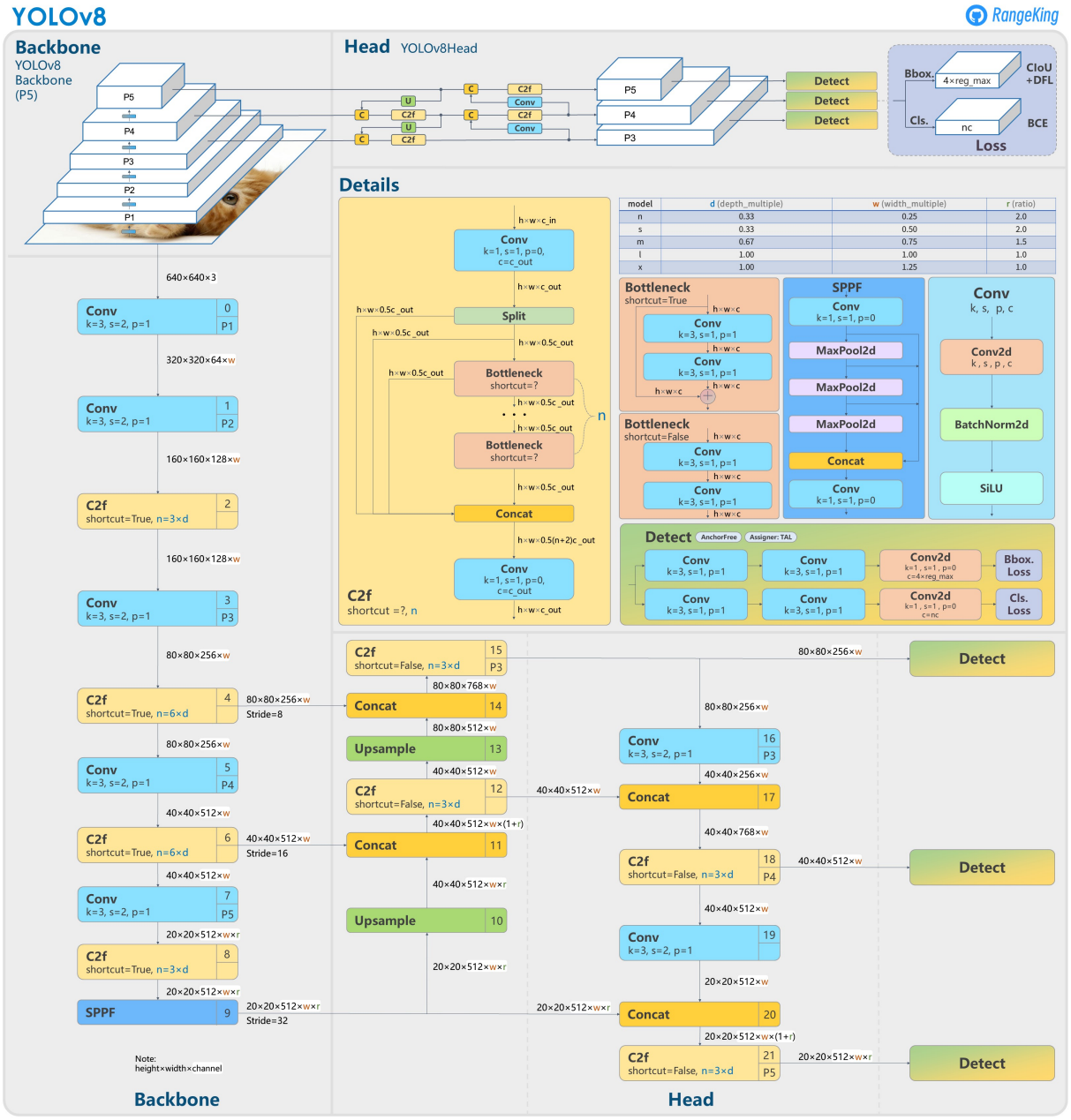

Figure 4.1: YOLOv8 Model Architecture (King 2023)

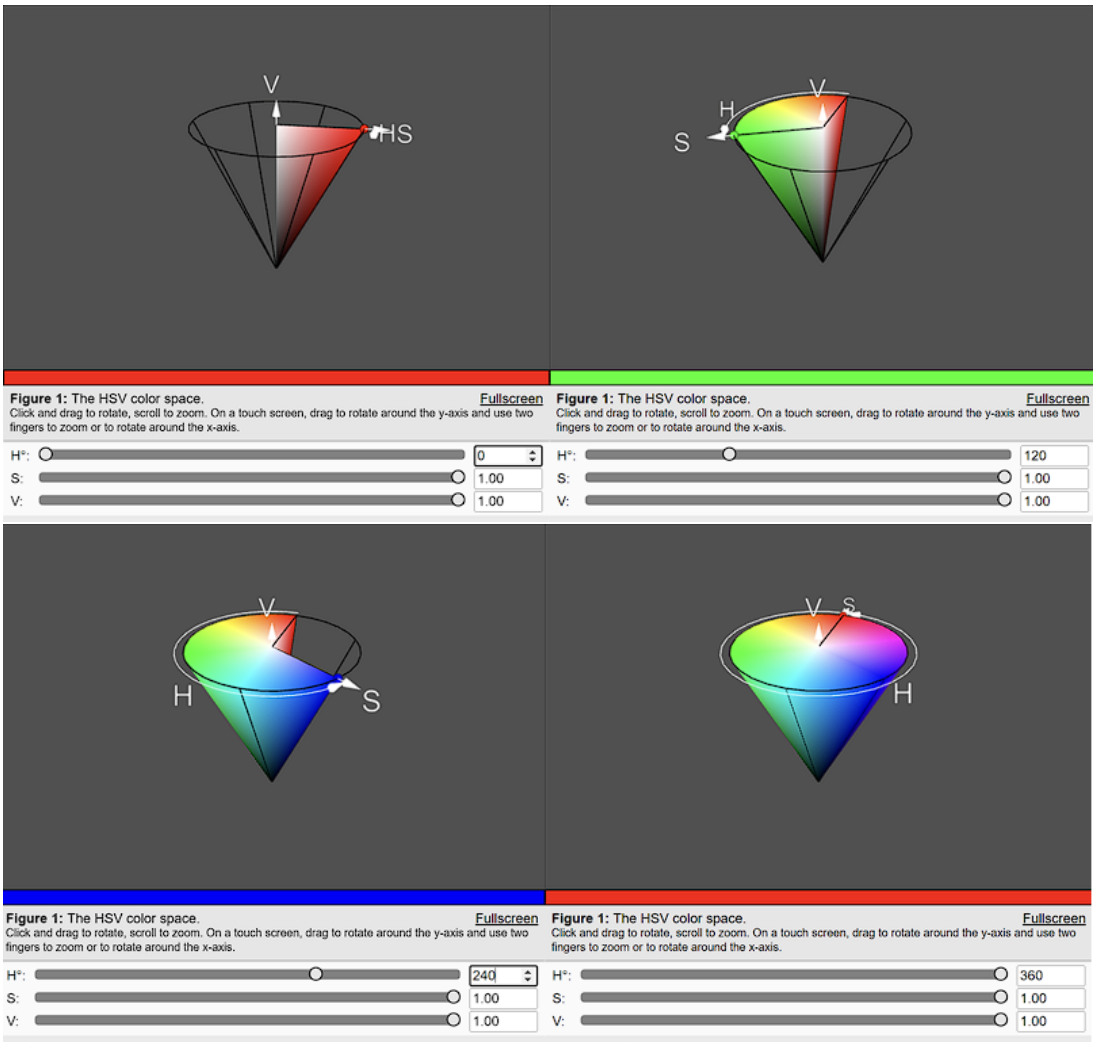

Figure 4.2: Hue (Top Left) degree 0: red axis (Top Right) degree 120: green axis (Bottom Left) degree 240: blue axis (Bottom Right) degree 360: red axis (Stratmann 2024)

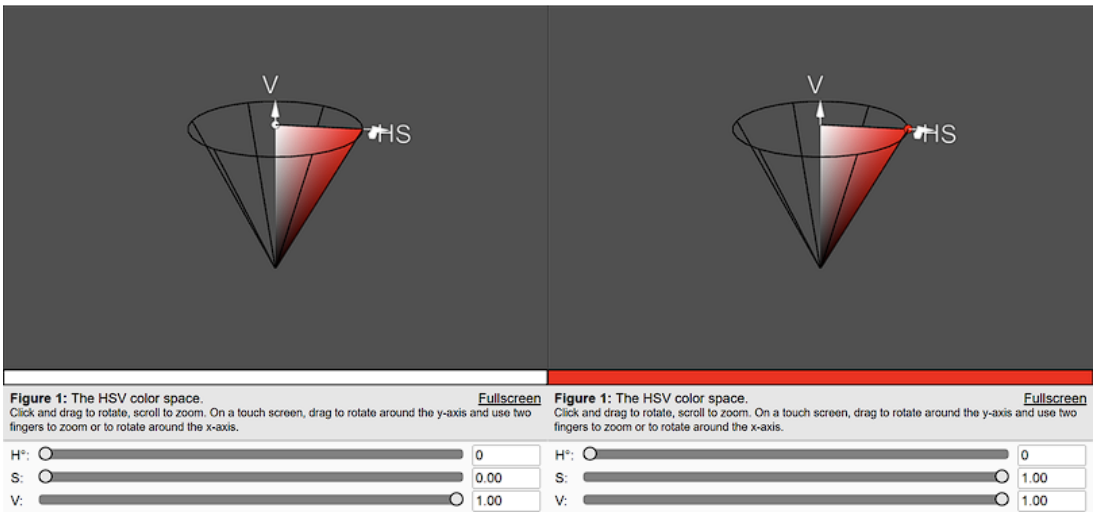

Figure 4.3: Saturation (Left) S = 0: white (Right) S = 1: red (Stratmann 2024)

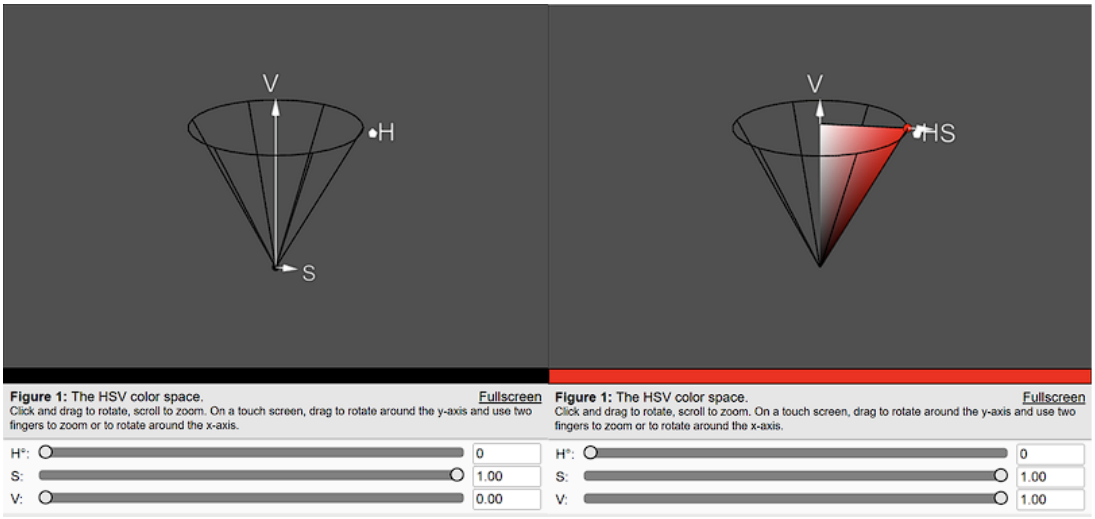

Figure 4.4: Value (Left) V = 0: black (Right) V = 1: red (Stratmann 2024)

Chapter 5

Experiment and Implementation

5.1 Image Processing Approach

5.1.1 Experimental Setup

In the process of developing the image processing algorithm, we utilised Visual Studio Code (VSCode) IDE and PyCharm IDE with Python version 3.10.12. Essential to the algorithm development were OpenCV version 4.8.1, NumPy version 1.25.1, and SciPy version 1.11.4.

5.1.2 Algorithm Tuning

The RGB images were transformed into the HSV colour space to better segregate colour information and diminish sensitivity to lighting variations. A series of iterative adjustments were made to the colour-based segmentation filters to enhance apple detection accuracy. Initially, a filter range of (0, 10, 10) - (10, 255, 255) was employed, capturing light-red tones; however, it included dark red shades and misidentified reddish soil as apples. Subsequently, the filter was refined to (0, 25, 25) - (10, 255, 255) to mitigate false positives associated with reddish soil. Another filter range of (160, 25, 25) - (180, 255, 255) was introduced to detect apples with dark-red tones. The cutoff values of saturation and value were set at 25-255 to adapt the algorithm to identify apples in varying lighting conditions and at different stages of ripeness, while discarding the reddish pixels belonging to the soil.

A bitwise OR operation was used to combine both masks, followed by single iterations of opening and closing operations using a 5x5 rectangular filter. Morphological opening and closing operations with a large kernel have a stronger impact on discarding noise, but it can also lead to the disappearance of smaller objects, such as apples far away from the camera. On the other hand, operation with a smaller kernel is more sensitive to edges.

Contour detection faced challenges in distinguishing closely placed apples. To address this, a minimum area threshold of 200 pixels was set for contour qualification. The initial approach yielded sub-par results prompting a shift to the Hough circle transform.

A 7x7 elliptical kernel replaced the 5x5 rectangular kernel, aligning better with the circular shape of most apples. Morphological opening and closing operations were conducted using an elliptical kernel for 2 and 1 iterations, respectively. The Hough circle transform was employed with carefully chosen parameters to enhance the ability to distinguish closely positioned apples. In this context, 'dp' represents the inverse ratio of accumulator resolution to image resolution, and it was specifically set to '2'. The choice of 'dp = 2' results in the accumulator resolution being half of the image resolution, causing near circles to accumulate votes in the same accumulator cell. This might lead to the detection of larger circles in the output. Notably, during testing, it was found that setting 'dp = 2' produced clearer displays

of apple contours. Additional specifications included a minimum distance of 10 pixels between circle centers, an upper bound of 100 for the Canny edge detector, an accumulator threshold of 8, and minimum and maximum radii of 2 and 50 pixels, respectively.

1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1
1	1	1	1	1

(a) 5x5 Rectangular Kernel

0	0	0	1	0	0	0
0	1	1	1	1	1	0
1	1	1	1	1	1	1
1	1	1	1	1	1	1
1	1	1	1	1	1	1
0	1	1	1	1	1	0
0	0	0	1	0	0	0

(b) 7x7 Elliptical Kernel

Figure 5.1: Structuring elements used in Morphological operations

5.2 Machine Learning Approach

5.2.1 Experimental Setup

Central to our approach was the use of a set of common software tools across both computational platforms:

- Key Software and Libraries:
  - YOLOv8 (ultralytics), PyTorch 2.1.2, and OpenCV 4.8.0 were utilised for their robust capabilities in machine learning model development and data processing.
  - Visual Studio Code with Python 3.11.5.

Specific computational platforms employed in the project included:

- **Google Colab**:
  - Provided cloud-based GPUs (Tesla T4), suitable for moderate computational tasks.
  - Offered a user-friendly interface and seamless integration with `Google Drive`, enhancing data accessibility and collaborative workflows.
  - Primarily used for pre-processing and initial results processing where extensive computational resources were not critical.
- **Local Computing Environment**:
  - Featured an NVIDIA GeForce RTX 4080 Laptop GPU, delivering advanced computational power and greater memory capacity.
  - Chosen for the more demanding tasks of model training and hyperparameter tuning, addressing the limitations of Google Colab in handling extensive computational workloads.

This dual-platform strategy was beneficial in the project's success, enabling a balanced use of collaborative, accessible tools for initial stages and high-powered, specialised resources for complex computational tasks. The consistent use of core software and libraries across both environments ensured a seamless workflow and integrity in the project's computational approach.

5.2.2 Data Preparation

The initial stage of data pre-processing, as discussed in Section 3, yielded ground truth labels formatted in `XML` format. To adhere to the specific requirements of the YOLOv8 model, we developed a Python script to convert these labels into the `YOLO` labelling format. Additionally, the training dataset was randomly partitioned into validation (50 images) and training (353 images) subsets. This division was structured following a specific folder arrangement for images and labels as detailed in Figure 5.2. These preparation steps were taken to ensure compatibility with YOLOv8, facilitating the subsequent phases of our implementation.

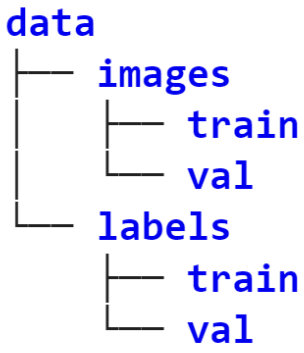

Figure 5.2: Training Dataset Folder Structure for YOLOv8

5.2.3 Model Training

For model training, we employed the `train()` method from the `ultralytics` library, specifically opting for the compact `YOLOv8n.pt` model. Being the smallest variant within the YOLOv8 architecture, this model demonstrated accelerated execution, enabling agile and iterative training sessions while also maintaining satisfactory performance for the apple counting task.

Hyperparameter Tuning

Hyperparameter tuning plays a pivotal role in the model training process as it directly influences the model's performance. To maximise the capabilities of YOLOv8, we performed hyperparameter tuning through the use of the `tune()` method. This method employs a genetic mutation algorithm to traverse the hyperparameter space, with the primary goal of enhancing the model's fitness using the `AdamW` optimiser.

Figure 5.3 illustrates the explored values for each hyperparameter against the corresponding model fitness. By dynamically exploring and exploiting the hyperparameter space, the tuning algorithm successfully identified an optimal combination of hyperparameters, ultimately refining the model training for enhanced performance.

In Table 5.1, the hyperparameter values associated with the highest fitness value of 0.37867 at iteration number 95 are presented. These selected hyperparameters will be employed in the upcoming training phase, as detailed in the subsequent discussion.

Training

During the training stage, the hyperparameters selected through the tuning process were utilised. The `train()` method was employed to conduct a training session consisting of 100 epochs. Throughout

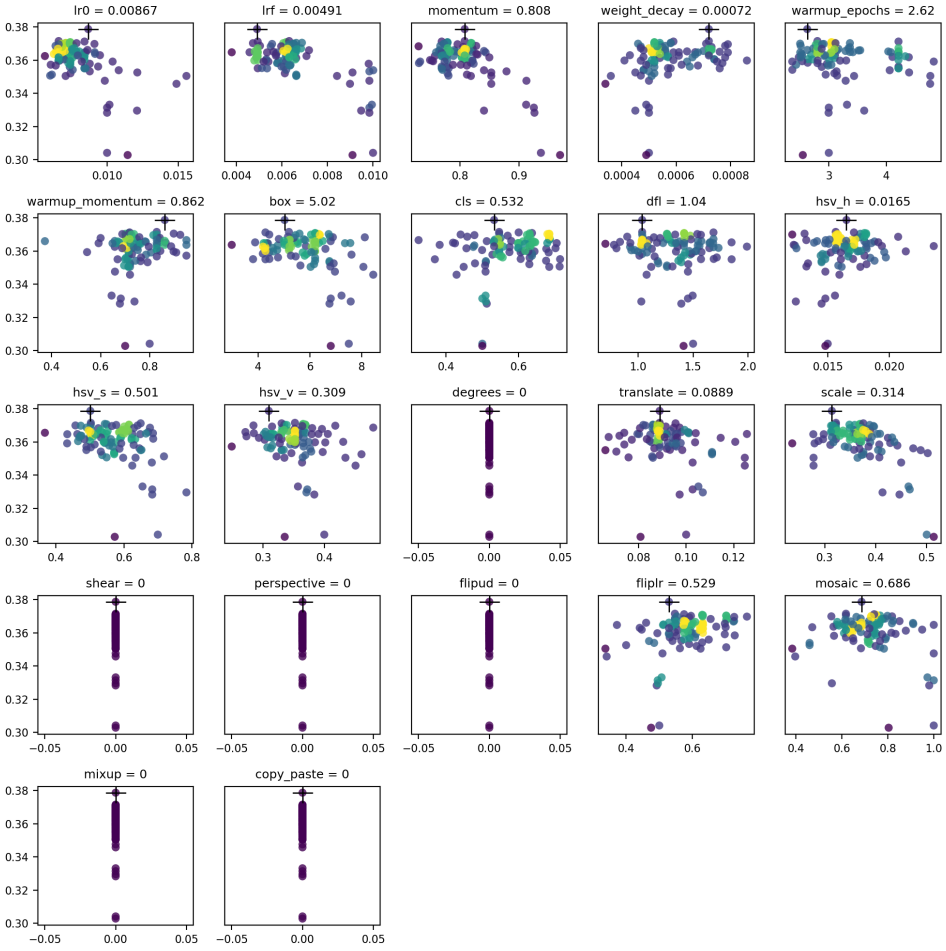

Figure 5.3: Model Fitness Over Tuning Search Iterations for Each Hyperparameter

training iterations, essential metrics such as training loss and validation metrics were closely monitored to ensure convergence and prevent overfitting.

The selection of using 100 epochs was deemed optimal through exploring different iteration counts, including 50, 100, 200, and 300 epochs. This comparative analysis allowed understanding the model's behaviour over time. The selected 100 epochs struck a delicate balance, enabling the model to attain peak performance by effectively capturing the complexity of the dataset while mitigating the risks associated with both underfitting and overfitting.

5.2.4 Model Testing

Following the training phase, model evaluation was conducted on the test set using the `val()` method on the best-performing model. This method required the test dataset in a similar folder structure for the training dataset, as illustrated in Figure 5.2.

The metrics, as discussed in Section 6, offered a comprehensive assessment of the YOLOv8n model's accuracy in counting apples. This evaluation not only contributed to optimising the machine learning model but also further facilitated a comparative analysis with the traditional method. The numerical results of the achieved metrics are detailed in Table 6.1, and further discussion of the results is outlined in Section 6.

Parameter	Value
lr0	0.00867
lrf	0.00491
Momentum	0.80801
Weight Decay	0.00072
Warmup Epochs	2.62482
Warmup Momentum	0.86167
Box	5.01667
Cls	0.5324
Dfl	1.03532
HSV_H	0.01652
HSV_S	0.50119
HSV_V	0.30947
Degrees	0
Translate	0.08886
Scale	0.31405
Shear	0
Perspective	0
Flipud	0
Fliplr	0.52908
Mosaic	0.68573
Mixup	0
Copy_Paste	0

Table 5.1: YOLOv8 Hyperparameters Resulting from the Tuning Process

Chapter 6

Results and Evaluation

In the task of counting apples, the choice of metrics plays a crucial role in the thorough evaluation of the algorithm's performance. **Precision** allows us to gauge the accuracy of the algorithm by assessing the proportion of correctly identified apples among all the identified instances. This is particularly relevant in ensuring that false positives, instances where the algorithm incorrectly identifies non-apple objects as apples, are minimised. **Recall** is employed to evaluate the algorithm's ability to locate and identify all apples present in the images. It addresses the concern of false negatives, where the algorithm might miss some apples. **F1-score** the harmonic mean of precision and recall, strikes a balance between these two metrics, offering a unified measure of the algorithm's overall effectiveness in counting apples.

$$Precision = \frac{TP}{TP + FP} = \frac{Correct\ detections}{All\ detections} \tag{6.1}$$

$$Recall = \frac{TP}{TP + FN} = \frac{Correct\ detections}{All\ labels} \tag{6.2}$$

$$F1 - score = 2 \times \frac{Precision \times Recall}{Precision + Recall} \tag{6.3}$$

6.1 Image Processing Approach

The quantitative evaluation of the traditional-based approach relies on precision, accuracy, and F-1 scores, while qualitative assessment involves scrutinising the resulting images. These results offer a comprehensive insight into the capabilities of the Python libraries employed and the impact of adjusting critical parameters on the overall outcome.

6.1.1 Results

In the evaluation of our image processing algorithm on a diverse and unbiased test dataset consisting of 101 images, we observed a mean precision of **0.5741**, indicating the measure of detected apples were indeed correct detections, a clear indication of the algorithm's accuracy. A mean recall of **0.5330** reflects the algorithm's ability to avoid missing relevant apples, representing its effectiveness in distinguishing between apples fallen on the ground and those still on the tree. The mean F1-score of **0.5286** represents a balanced trade-off between precision and recall, indicative of the algorithm's ability to accurately identify apples while maintaining a balance between false positives and false negatives. Overall, these results suggest that the algorithm exhibits a reasonably balanced performance in detecting apples within

Figure 6.1: Comparison between Original and Development of Image processing algorithm: (Middle) Apple counting with Contour-based approach (Right) Apple counting with Hough Circle transform approach, with Ground Truth boxes in Green, True Positives in Yellow, False Positives in Red, and False Negatives in Blue.

diverse and unbiased environmental conditions. The replacement of contour-based detection with circle Hough transform resulted in a notable enhancement, elevating the f1-score from **0.4905 to 0.5286**.

6.1.2 Evaluation

Our preliminary approach to the image processing problem was to make use of adaptive thresholds in order to deal with scenes of varying illumination. It quickly became apparent that this approach would not suffice as the mask generated contained lots of unwanted information. The pivot to **HSV-based thresholding** to segment out the reddish tones of the image was instrumental to the success of our approach. Another significant improvement was seen when the **Contour-based segmentation** was replaced with **Hough Circle transform**. The Hough circle transform utilised the circular feature of the apples to distinguish between closely packed apples.

6.2 Machine Learning Approach

The evaluation of the YOLOv8 model's performance in apple counting is presented similarly through a combination of quantitative and qualitative analyses, providing a thorough understanding of its capabilities and areas for improvement.

6.2.1 Results

The performance of the YOLOv8 model was quantitatively evaluated using precision, recall, and F1-score metrics, as detailed in Table 6.1. The model achieved a precision of **0.7553**, indicating high accuracy in correctly identifying apples, crucial for minimising false positives. The recall of **0.6976**,

Figure 6.2: Example Comparison of Original and Annotated Images Using YOLOv8: (Left) The original image from the data set. (Right) The annotated version showing object detection results, with Ground Truth boxes in Green, True Positives in Yellow, False Positives in Red, and False Negatives in Blue.

although substantial, suggests the need for improvement in detecting all apples, especially in complex scenarios with overlapping fruits or varying illumination. The F1-score of **0.7252** indicates a balanced performance, synthesizing precision and recall, and reflects the model's overall effectiveness in apple detection within the dataset.

Qualitative analysis, supported by visual inspection of images from the test set (see Figure 6.2), revealed the model's notable performance in clear conditions but highlighted challenges in more complex scenarios. These included difficulties in accurately detecting apples in scenes with closely clustered fruits or partial occlusion, leading to instances of missed detections (false negatives) and erroneous identifications (false positives).

6.2.2 Evaluation

The deep learning architecture of the YOLOv8 model and its rapid image processing capabilities significantly contributed to the performance metrics reported in Table 6.1. The choice of the compact YOLOv8n variant, while facilitating quick iterations and real-time processing, may limit the model's ability to discern fine details in complex scenes. This limitation is particularly noticeable in scenarios involving partial occlusion or dense clustering of apples.

Hyperparameter tuning with a genetic mutation algorithm played a vital role in optimising performance.

However, the tuning process itself could be further explored to enhance the model's adaptability to various orchard environments. Additionally, the model's performance under varied lighting and weather conditions, as observed in the test images, indicates a need for more robust data augmentation techniques to improve environmental resilience.

6.3 Comparison

6.3.1 Discussion on Capabilities and Limitations

Metric	Image Processing	Machine Learning
Precision	0.5741	0.7553
Recall	0.5330	0.6976
F1-Score	0.5286	0.7252

Table 6.1: Quantitative Comparison between Image Processing and Machine Learning Approaches

The YOLOv8 model exhibits advanced capabilities in apple detection and counting, significantly enhancing the precision of such tasks over traditional methods. Its proficiency in extracting complex features and recognising a variety of shapes and shades has been crucial in accurately identifying apples under diverse conditions. This ability is particularly beneficial for crucial tasks like yield estimation and orchard management, where it is essential to distinguish between apples on the tree and those that have fallen. The model's versatility in handling apples of different sizes and ripeness levels, and its capability to detect occluded apples, contribute significantly to accurate yield estimation. Moreover, the adaptability and rapid improvement potential of machine learning models like YOLOv8 are notable advantages. These models, equipped with robust training and testing pipelines, are well-suited for dynamic agricultural environments where accuracy and adaptability are key.

However, the model encounters challenges in scenarios with occlusion or low contrast, as indicated by the recall rate in the results. This limitation points to the need for enhancements in the model, particularly in its ability to process complex visual information. Possible improvements could include training with more diverse datasets (possibly by using the full Minneapple dataset) that encompass a wider range of environmental conditions. Additionally, incorporating additional data sources, such as multi-spectral imaging, can enrich the detection capabilities.

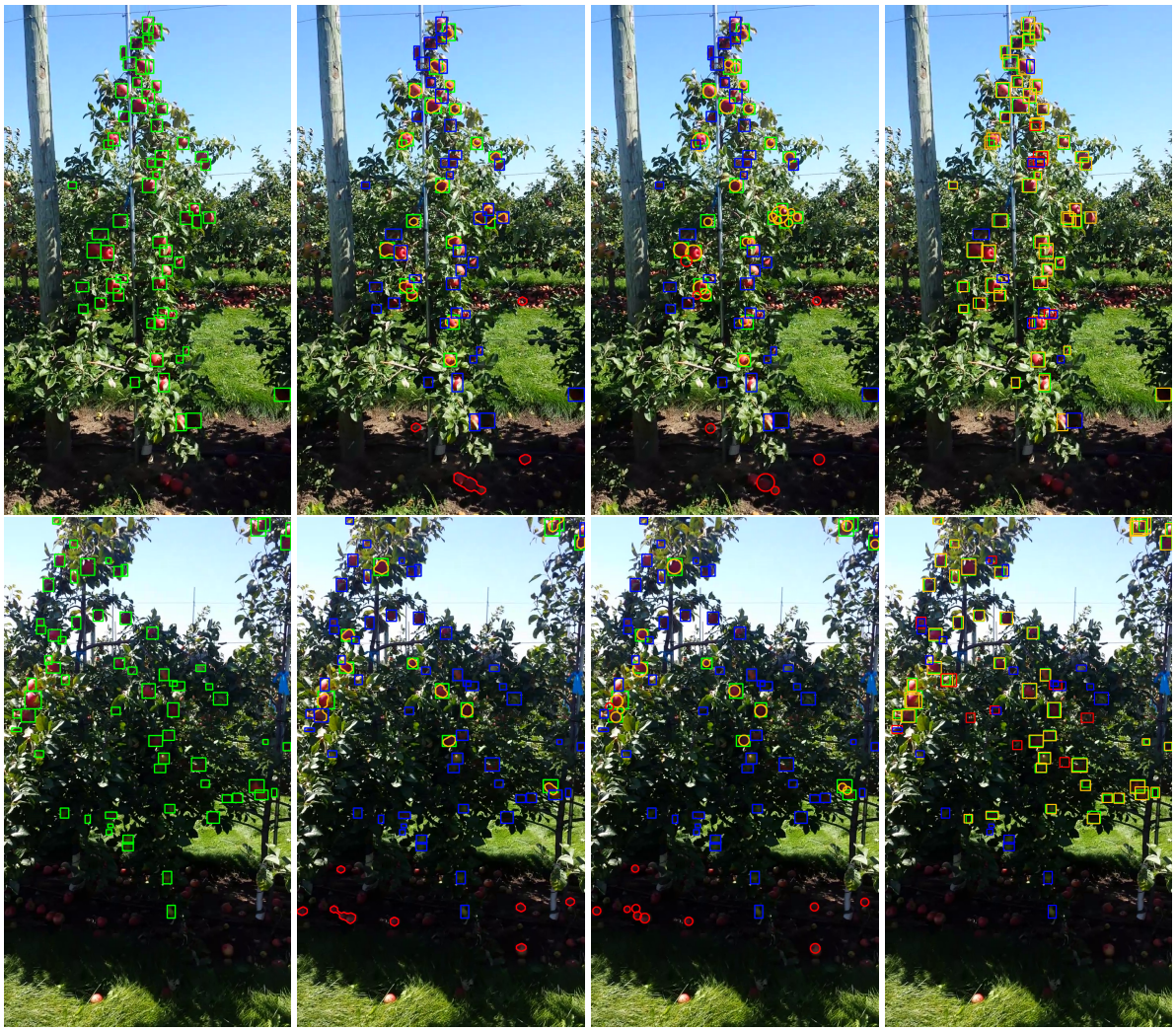

Figure 6.3: Qualitative Comparison of Machine Vision and Image Processing Methods on the Minneapple Dataset under Different Lighting Conditions. Top Row: Well-lit Conditions; Bottom Row: Shaded Conditions. Sequence from Left to Right: Ground-truth Labels from Minneapple, Apple Detection via Contour Detection, Apple Detection using Hough Circle Transform, Apple Detection with YOLOv8-. Colour Key: Green – Ground-truth Labels, Yellow – True Positives, Blue – False Negatives, Red – False Positives.

Chapter 7

Conclusion and Future Works

7.1 Conclusion

This study meticulously identified and addressed the multifaceted challenges associated with apple counting, detection, and localisation, including the impact of varying illumination conditions, the diverse stages of ripeness among apples, highly occluded scenarios, the presence of small apples, and the necessity to account for fallen apples. Our comprehensive exploration thoroughly delved into the nuances of both image processing and machine learning approaches, shedding light on their respective capabilities and limitations.

Notably, the findings underscored the superior performance of the machine learning approach in the tasks of apple detection, localisation, and subsequent counting when compared to the image processing approach. This emphasises the efficacy of machine learning techniques in navigating the complexities posed by diverse environmental conditions and varying apple characteristics.

7.2 Future Works

Future research in machine learning should address identified limitations by enhancing the YOLOv8 model. This involves expanding the training dataset, refining the model for occlusion management and varying lighting conditions, and integrating additional sensory data. Balancing model compactness with scene detail capture is crucial for accuracy in precision agriculture.

Enhancements in image processing include using Local Binary Patterns to differentiate fallen from harvest-ready apples. Improvements also involve detecting both green and red apples, accurate data annotations, and specialised algorithms like cv2.BackgroundSubtractorMOG2 for video-based detection. Exploring thermal cameras aims to reduce false positives by leveraging distinct thermal signatures of different apple colours. These efforts collectively aim to advance apple detection algorithms for robust real-world agricultural applications.

Bibliography

Adjobo, Esther Chabi et al. (2023). "Automatic Localization of Five Relevant Dermoscopic Structures Based on YOLOv8 for Diagnosis Improvement". In: *Journal of Imaging* 9.7.

Behera, Santi Kumari et al. (2018). "On-Tree Detection and Counting of Apple Using Color Thresholding and CHT". In: *2018 International Conference on Communication and Signal Processing (ICCSP)*, pp. 0224–0228. DOI: 10.1109/ICCSP.2018.8524363.

Chen, Steven W et al. (2017). "Counting apples and oranges with deep learning: A data-driven approach". In: *IEEE Robotics and Automation Letters* 2.2, pp. 781–788.

DeepLearningAI (2023). *NonmaxSuppression*. https://www.youtube.com/watch?v=VAo84c1hQX8. URL: https://www.youtube.com/watch?v=VAo84c1hQX8.

Häni, Nicolai, Pravakar Roy, and Volkan Isler (Oct. 2018). "Apple Counting using Convolutional Neural Networks". In: *2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*. arXiv:2208.11566 [cs], pp. 2559–2565. DOI: 10.1109/IROS.2018.8594304. URL: http://arxiv.org/abs/2208.11566 (visited on 11/22/2023).

— (Apr. 2020). "MinneApple: A Benchmark Dataset for Apple Detection and Segmentation". In: *IEEE Robotics and Automation Letters* 5.2. arXiv:1909.06441 [cs], pp. 852–858. ISSN: 2377-3766, 2377-3774. DOI: 10.1109/LRA.2020.2965061. URL: http://arxiv.org/abs/1909.06441 (visited on 11/27/2023).

Jiménez, Antonio, R. Ceres, and J.L. Pons (July 2000). "A Survey of Computer Vision Methods for Locating Fruit on Trees". In: *Trans. ASAE* 43. DOI: 10.13031/2013.3096.

King, Range (2023). *Brief summary of YOLOv8 model structure - Issue #189*. https://github.com/ultralytics/ultralytics/issues/189.

Marzoa Tanco, Mercedes, Gonzalo Tejera, and Matías Di Martino (2018). "Computer Vision based System for Apple Detection in Crops:" en. In: *Proceedings of the 13th International Joint Conference on Computer Vision, Imaging and Computer Graphics Theory and Applications*. Funchal, Madeira, Portugal: SCITEPRESS - Science and Technology Publications, pp. 239–249. ISBN: 978-989-758-290-5. DOI: 10.5220/0006535002390249. URL: http://www.scitepress.org/DigitalLibrary/Link.aspx?doi=10.5220/0006535002390249 (visited on 11/22/2023).

Stratmann, Lukas (2024). *HSV - ColorMaster*. https://web.cs.uni-paderborn.de/cgvb/colormaster/web/color-systems/hsv.html. Accessed: 2024-01-06.

Sural, Shamik, Gang Qian, and Sakti Pramanik (2002). "Segmentation and histogram generation using the HSV color space for image retrieval". In: *Proceedings. International Conference on Image Processing*. Vol. 2. IEEE, pp. II–II.

Wang, Gang et al. (2023). "UAV-YOLOv8: A Small-Object-Detection Model Based on Improved YOLOv8 for UAV Aerial Photography Scenarios". In: *Sensors* 23.16, p. 7190.

Xuan, Guantao et al. (2020). "Apple detection in natural environment using deep learning algorithms". In: *IEEE Access* 8, pp. 216772–216780.