Slicing aided large scale tomato fruit detection and counting in 360-degree video data from a greenhouse

Citation

TUREČKOVÁ, Alžběta, Tomáš TUREČEK, Peter JANKŮ, Pavel VAŘACHA, Roman ŠENKEŘÍK, Roman JAŠEK, Václav PSOTA, Vít ŠTĚPÁNEK, and Zuzana KOMÍNKOVÁ OPLATKOVÁ. Slicing aided large scale tomato fruit detection and counting in 360-degree video data from a greenhouse. *Measurement: Journal of the International Measurement Confederation* [online]. vol. 204, Elsevier, 2022, [cit. 2024-02-01]. ISSN 0263-2241. Available at https://www.sciencedirect.com/science/article/pii/S0263224122011733

DOI

https://doi.org/10.1016/j.measurement.2022.111977

Permanent link

https://publikace.k.utb.cz/handle/10563/1011191

Slicing aided large scale tomato fruit detection and counting in 360-degree video data from a greenhouse

Alžběta Turečková[a], Tomáš Tureček[a], Peter Janků[a], Pavel Vařacha[a], Roman Šenkeřík[a], Roman Jašek[a], Václav Psota[c], Vit Štěpánek [b], Zuzana Komínková Oplatková[a,*]

[a] *Tomas Bata University in Zlin, Faculty of Applied Informatics, Nam. T. G. Masaryka 5555, Zlín, 760 01, Czech Republic*

[b] *NWT a.s., Trida Tomase Bati 269, Zlin, 760 01, Czech Republic*

[c] *Farma Bezdinek, s.r.o., K Bezdinku 1515, Dolni Lutyne, 735 53, Czech Republic*

ABSTRACT

This paper proposes an automated tomato fruit detection and counting process without a need for any human intervention. First of all, wide images of whole tomato plant rows were extracted from a 360-degree video taken in a greenhouse. These images were utilized to create a new object detection dataset. The original tomato detection methodology uses a deep CNN model with slicing-aided inference. The process encompasses two stages: first, the images are cut into patches for object detection, and consequently, the predictions are stitched back together. The paper also presents an extensive study of post-processing parameters needed to stitch object detections correctly, especially on the patch's borders. Final results reach 83.09% F1 score value on a test set, proving the suitability of the proposed methodology for robotic farming.

Keywords: Tomato fruit detection, Tomato fruit counting, 360-degree video, Image processing, Computer vision, Deep CNN, Slicing aided inference, Robotic farming

1. Introduction and motivation

The concept of precise agriculture is aimed to provide an effective crop management system [1]. By precisely monitoring the plant's health and crop physiological status, precise agriculture aims to use cost-effective fertilizer application, pesticide application, plant management, and more. The mandatory part of this concept is a detailed set of information about a representative amount of plants or field sections that are necessary to gain as often as possible. For example, the Ohio Agricultural Research and Development Center [2] operates a web-based decision support system for decision support in greenhouse climate control. The author of [3] extends the methodology by studying the optimality degrees of microclimate parameters such as air temperature and relative humidity.

Even if various sets of sensors are used for monitoring the plant, every new piece of information can increase its accuracy. This research is aimed to provide a new method of tomato fruit detection and counting. Knowing the fruit's size and location in the greenhouse can be utilized for precise harvest prediction. Any difference between accurate harvest prediction and actual harvest can point out some new plant stress situations in the greenhouses.

Moreover, the harvest prediction itself is an essential business aspect of commercial tomato growing in greenhouses. The tomato delivery contracts are based on tomato harvest predictions in the following week. The greenhouse management is usually committed to delivering a specific volume of tomatoes on a particular week or day. Therefore, every difference between harvest prediction and actual harvest can cause significant commercial losses and logistic problems. The further described research was performed on a dataset created in cooperation with a specific tomato greenhouse company. Based on information from various greenhouse agronomists, an estimation of a new harvest is mainly based on the agronomist's experiences. Nowadays, the greenhouse agronomist estimates a new harvest based on information about crop growth factors, pest and disease status, weather forecasts, planned tasks in the greenhouse, and previous harvest. An average error of estimation accuracy usually is around 10-20 percent but can fluctuate distinctly. Even if the visual greenhouse inspection can provide sufficient information, the exact tomato count can significantly improve the harvest prediction.

The precision of a tomato counting solution is not the only requirement for the desired system. Due to the large area of commercial tomato greenhouses, any possibility of counting automatization is very beneficial, too, since it simplifies and speeds up the prediction processes. Our proposed approach uses a simple 360-degree camera mounted on an autonomous or semiautonomous robotic system. The processing phase is detached from the data acquisition; therefore, the capturing process is straightforward and quick.

Here, we summarize main contributions of a presented work:

• We present a unique automated process of detecting and counting tomato fruit in a greenhouse without human intervention. ith our methodology, we can not only estimate but count the exact numbers of fruits present in one whole plant row of a greenhouse. The process consists of three main phases:

1. First, a wide image is extracted from a 360-degree video of an entire tomato plant row in a greenhouse.
2. Second, the wide image is cut into overlapping patches, and a deep CNN model is applied for tomato detection.
3. Third, the model predictions were stitched together to form a comprehensive output detecting all tomato fruits in the whole tomato row of a green house.

• we provide an extensive study of post-processing parameters applied during the image stitching phase (i.e., step three) to ensure the best quality of the final output.

The following parts of the paper describe specific steps of our developed algorithm acquiring the source data, image preprocessing, tomato detection, and tomato counting. Furthermore, the crucial aspects of each used method are described. Finally, the evaluation results are provided at the end of the paper.

2. State of the art

Since the harvest prediction is essential for further commercial activities, the principal emphasis of this research was accurate tomato fruit detection and amount counting. Nowadays, the machine learning techniques, including deep learning convolutional neural networks (CNN) applied also by authors, dominate a machine vision field. The following subsections deal with the detailed inspection of current technologies for successful predicting.

2.1. *Fruit detection and counting*

Many researchers have investigated fruit detection over the past several decades. Authors of [4,5] brought a valuable overview and significant improvements in the field. Traditional image processing techniques have many constraints, making applying the algorithms to other fruits or environments hard. They often struggle with uneven illumination, and incorrect leaf detection [6]. The growth and development of artificial intelligence techniques enabled the application of machine learning to computer vision tasks in agriculture, and it has practically dominated the field.

Authors of [7] present a multi-class fruit detection system utilizing an adjusted Faster RCNN model. The paper focuses mainly on model adjustments and training process automation. For this purpose, it utilizes the artificially created dataset [8] where the fruits were filmed while rotating around a fixed axis on a white background. On this dataset, they achieved a mean average precision of 88.94%. Unfortunately, the results for the tomato class are not stated in the paper; even though it is present in the original dataset, the authors present results only for apple, mango, and orange.

The paper [9] proposes an improved YOLOv3-tiny architecture for tomato detection in real-time. The experimental results show that the F1 score of the model is 91.92%, while the detection speed on a CPU can reach 25 frames/s. Another paper also utilizes the YOLO-based architecture, presenting a YOLO-tomato detector [10]. It modifies the well-known YOLOv3 architecture for object detection by adding the dense connections between the layers for feature extraction and by applying the circular bounding box instead of the classic rectangle. YOLO-tomato model achieves an excellent 93.91% F1-score.

The above referred research only focused on tomato detection in images directly capturing a tomato or a tomato cluster. Compared to that, the paper [11] analyzes the images of the whole tomato plant from a greenhouse, similarly as we do. The Faster RCNN with 101-Resnet backbone model was utilized and obtained an F1-score of 83.67%. Compared to the previously stated research, the performance drop can be accounted to a more compilated scenario, including more small immature tomatoes and frequent obstruction by leaves. Although the authors provide the yield mapping of the whole image row by stitching the images together, the counting results were estimated only on single images, which should, by principle, bring additional errors into the final count: the tomatoes on the overlapped borders would be counted twice.

A few research papers deal with fruit counting on a large scale, i.e., accessing the whole crop counts. The authors of [12] attempt to produce a real-time pear fruit counter using a video captured by a mobile phone capturing the bottom side of the joint-tree pear orchard. It utilizes real-time object detection by the YOLOv4-tiny model and consequently applies the unique ID deep sort method for pear counting, achieving an F1 score of 87.85%. Compared to our scenario, pears do not form trusses/clusters, and the sky forms a relatively uniform background, making the detection process more straightforward.

Finally, the paper [13] presents a similar goal as our work, delivering an approach for detection, counting, and maturity assessment of cherry tomatoes using multi-spectral images. Authors train deep CNN networks on images captured in a tomato greenhouse. Consequently, similar to the paper [12], they apply the DeepSORT algorithm to track the tomatoes in a video. The value F1-score is not stated; the IDF1-score achieved by the best solution is 51.4%. The results show that detecting and tracking objects in complicated and obscured scenes of tomato growth is very challenging. On the contrary, our approach eliminates the tricky part of object tracking, making the whole process easier.

In our paper, we present a different solution. The objects are not detected in video frames, but rather a extra wide image capturing the whole tomato plant row was produced first. This big image is cut into overlapping patches, which are processed by an object detector, and the output predictions are then stitched together to form the final output. The extended study of the post-processing parameters was performed to ensure the best possible outcome of the stitching procedure. The study of common techniques utilized after bounding box regression to eliminate redundant bounding boxes is provided in the following Section 2.2.

2.2. *Object detection post-processing methods*

Non-maximum suppression (NMS) has been an integral part of many detection algorithms in computer vision for almost 50 years. It was first employed in edge detection techniques [14]. For human detection, Dalal and Triggs [15] demonstrated that a greedy NMS algorithm, where a bounding box with the maximum detection score is selected and its neighboring boxes are suppressed using a predefined overlap threshold improves performance over the approach used for face detection [16]. Greedy NMS still obtains the best performance when average precision (AP) is used as an evaluation metric and is therefore considered the gold standard and is employed in state-of-the-art detector Faster R-CNN [17].

Over the years, some new approaches have been presented. Authors of [18] propose Soft-NMS, which attempts to address the problem of classic NMS that it suppresses even a correctly detected object if the object lies within the predefined overlap threshold. The algorithm decays the detection scores of all other objects as a continuous function of their overlap with the first most confident one. Hence, no object is eliminated, just the confidence score is lowered. Another approach is presented in [19], the algorithm judges the neighboring bounding boxes of each bounding box and combines the neighboring boxes that are strongly correlated with the corresponding bounding boxes. This approach was designed for instance segmentation, showing a steady increase of accuracy.

Our scenario deals with a slightly different problem of multiple detections. The original wide image is split into overlapping patches; consequently, there are multiple detections in the overlapping areas of the final output. Therefore, aside from a classical suppression after model inference, we also need to apply post-processing suppression to the final stitched output. e apply an extensive study of different post-processing suppression algorithms to achieve the best possible output.

3. **Methods**

This section describes the methodology from automatic data acquisition to the final number of tomato fruits detected.

3.1. *Field data collection*

As was mentioned in Section 2.1, the well-known approach in already published research about fruit detection uses object detection algorithms applied on a video stream or simple images of plants. The practical usage of this approach faces several limitations. The alleys between each tomato row are narrow, while the vertical range of tomato fruits on the plant is wide. Therefore the vertical field of view (FOV) of the used camera needs to be more than 90 degrees to capture the full plant height. Another limitation is based on the requirement for maximal counting accuracy. hen applying image detection in single video frames or overlapping photos, some indexing or tracking algorithm must be used to prevent multiple counting of single tomato fruit. We have chosen a different approach and used 360 degrees camera (Ricoh Theta Z1 with high resolution 4 K UHD (3840 X 2160) video) for image acquisition to cover the necessary wide field of view. Moreover, our algorithm works only with the source image from the camera. This approach has two significant consequences. First, the process of data acquisition and data processing can be separated. Second, the automation of the data acquisition can be done by simple autonomous or semiautonomous devices and by mounting the camera devices on already existing technical equipment in greenhouses such as for instance commonly used trolleys. The source videos for experiments described in this paper were acquired during several visits of authors in the hydroponic greenhouse of the company Farma Bezdínek (Dolní Lutyně, Czech Republic) in 2020 and 2021. The videos were taken during the harvest period of the fully developed crops cherry tomato (10-12 g/fruit) and cocktail (35-45 g/fruit). The camera was mounted on a greenhouse trolley (Berkvens - Control Lift), allowing a maximum 5 km/ha speed.

3.1.1. Video converting

The 360-degree camera produces a video stream with two 180 degree views captured by a fisheye camera lens. In the first preprocess phase, the two views are stitched into one panorama equirectangular projection. This projection captures an almost 180 degrees vertical view of the tomato row. Although the upper and bottom details of the picture might be distorted, the middle part of the image can be successfully used for tomato detection. Moreover, we can capture both sides of the alley both tomato plant rows at once.

The process of image production from a video follows. Firstly, the video is decomposed into individual frames. Only the part directly capturing the row of tomatoes is extracted from the 360-degree equirectangular frame. A small vertical region directly in front of the camera is clipped out of each frame. Consequently, these clipped regions are stitched together, forming one wide picture of the whole row. Deformities on the bottom and the top of the picture caused by the 180-degree view are mathematically compensated. This approach allows us to work with a wide picture rather than a video or video frames. In the context of this paper, wide image can be referred as a panorama image too. In the picture, every fruit is visible only once, and there are no boundaries that complicate the counting process.

3.1.2. Data annotation

To select a proper annotation tool, the following requirements for the annotation tool were defined based on the obtained data by the 360-degree video and its processing:

- the ability to process high image resolution of the composed images (image size over 350MB and 100 000 pixels wide),
- the tool’s stability for dealing with the high frequency of tomato fruit occurrence in images,
- the possibility to annotate tomato fruits with polygons (to be used both for object detection and instance segmentation models),
- dividing the annotation task between multiple persons (working on different platforms).

The multi-platform desktop annotation tool LabelMe [20] was selected as the annotation software considering all those factors. After completing the initial annotation process (by multiple persons), the segmentation masks were cross-validated by leading person of the group of annotators. In the end, the annotations were exported to COCO format.

3.2. Dataset preparation

This section describes the dataset preparation required for model training. The dataset is available on demand. The composed panorama images are 2448 pixels high and even 100 000 pixels wide. To process such sized images at one pass far exceeds the ability of available hardware. Concurrently, the image cannot be downsampled because the tomato fruit objects are tiny and would not be distinguishable in lower resolution. To overcome this issue, the images are sliced into several smaller patches, which are evaluated separately by the model. The model predictions are then stitched together to form the final output. The Section 3.4 describes the stitching process in detail.

The patch size was selected according to the used hardware and pre-trained model (described in Section 3.3) to be 1333 X 735 pixels. We tested three different settings:

1. Slice the patches of the size 2448 X 4078 pixels and downsample them (approximately 9 times to match the desired image size 1333 X 735 pixels) before model processing. This way, the image slices encapture the entire image height, and the patches are created only along a horizontal axis. Therefore, the model can see the whole tomato plant in one patch, but the tomato fruits' resolution is significantly reduced. e refer to this settings as low resolution.
2. Slice the patches of the size 1469 X 2448 pixels and downsample them (approximately 4 times to match the desired image size 1333 X 735 pixels) before model processing. This settings should allow the model to see the greater context while keeping the objects big enough to be detected. e refer to this settings as medium resolution.
3. Slice the patches of the size 1333 X 735 pixels and allow the model to process the image in original resolution at the cost of losing the greater context of the image. We refer to this settings as high resolution.

While the computation speed is not our primary concern, please note that the prediction time grows linearly with the number of patches cut from the input image. Therefore, processing an input image in the low-resolution settings is approximately 13 times quicker than in the high-resolution settings.

Please also note the fact that all the objects in our dataset are small or medium size according to the standard COCO detection cat-egorization.[1] The mean object size corresponds to a square of size 42 pixels.

<u>3.2.1. Data splitting</u>

The gathered image data were divided into training, validation, and test split. There are 50 gathered images: training split contains 35 of them, validation split has 7, and the last 8 are in the test set.

The training and the validation sets were cut beforehand. Training and evaluation within the training process were made on the cuts, not the whole image. The number of images in the training and the validation set after slicing were 95+20, 311+67, and 1240+264 for low, medium, and high-resolution settings, respectively.

[1] The COCO detection challenge (https://cocodataset.org/) divides the object into three categories according to the bbox size: small < 32 X 32 pixels, medium > 32 X 32 pixels and < 96 X 96 pixels and large > 96 X 96 pixels.

The original panorama images from the validation set were used to assess the best post-processing parameters. e performed an extensive study to choose the best model confidence threshold, the postprocessing algorithm, the match metric, and its threshold value. The results are presented in Section 4.2. Finally, the test set results were also assessed in the original panorama image to judge the methodology performance on unseen data.

3.3. Model and training settings

A Faster R-CNN model [21] with a ResNet-50 backbone [22] was utilized in the experiments. The model was initialized with weights trained on the COCO dataset and modified for a single class present in our dataset. The model used Cross-Entropy loss for class optimization and L1 loss (Least Absolute Deviations) for bounding box optimization.

The model had been trained for twenty epochs, considering one epoch as a single pass through all training data. SGD (Stochastic gradient descent) with a learning rate of 0.02, momentum 0.9, and weight decay of 0.0001 were utilized to optimize the model weights. On epochs 16 and 19, we lowered the learning rate ten times.

We trained on the Titan XP graphic card with a batch size two. Before feeding into a model, images were always resized to image scale 1333 X 800 pixels and normalized with mean and standard deviation values available from the original COCO dataset. The random flip augmentation with the 0.5 probability was utilized to increase the training data variability and help to prevent overfitting.

3.4. Image stitching

The main target of this work is the localization and counting of all tomatoes present in the big panorama picture produced from the 360-degree video. However, the model process only the cut patches; hence, the individual outputs need to be stitched back together in the next step. The patches overlap by a 0.2 ratio of their size to minimize the risk of missing the object cut off on the border.

Several possible strategies exist to merge or suppress multiple predictions in neighboring patches. Generally speaking, the process has two main parameters: the match metrics and the post-processing algorithm.

The match metrics determine the candidate detections for merg-ing/suppression. The value of the match metrics threshold is crucial; the higher value enables only more similar predictions to be merged. The post-processing algorithm determines the order in which the candidate detections are processed and how the final detection instance is constructed.

We tested two different match metrics:

1. Intersection over union (IoU) is a term used to describe the extent of overlap of two bounding boxes. The greater the region of overlap, the greater the IoU. The metric is defined as:

$$IoU = \frac{Area\ of\ intersection}{Area\ of\ union} \tag{1}$$

2. Intersection over a smaller area (IOS) is very similar to the IoU metric; only the area of the intersection is divided by the area of the smaller of the two boxes:

$$IOS = \frac{Area\ of\ intersection}{Area\ of\ smaller\ box} \quad (2)$$

We experimented with those variants of the post-processing algorithms:

1. Greedy non-maximum supression (NMS),
2. Non-maximum merging (NMM),
3. Greedy non-maximum merging (GREEDYNMM).

Each of the post-processing algorithms is described in one of the corresponding Section 3.4.1, 3.4.2, 3.4.3 below. Section 4.2 then shows the influence of the parameters mentioned above on the final output.

3.4.1. Greedy non-maximum suppression (NMS)

We employ only the greedy variant of non-maximum suppression because it is the standard in object detection. The algorithm chooses the predictions with the maximum confidence and suppresses all the other predictions overlapping with the selected predictions greater than a threshold of the match metrics.

In the beginning, we got a list P of prediction bounding boxes (bboxes) coordinates and the corresponding predicted confidence score of the model. e also got the overlap match metrics threshold. The algorithm proceeds as follows:

Step 1:. Select the prediction S with the highest confidence score, remove it from P, and add it to the final prediction list keep, which is initially empty.

Step 2:. Compare the prediction S with all the predictions present in P. Calculate the overlap of this prediction S with every other prediction in P, using the IoU or IOS metrics. If the overlap exceeds the match metrics threshold for any prediction T present in P, remove prediction T from P.

Step 3:. Repeat the process until there are still predictions left in P, then return the list containing the filtered predictions.

As we can observe from the algorithm above, the whole filtering process depends on the match metrics threshold value. Therefore, a suitable threshold value selection is vital for the final performance. Although the NMS is one of the commonly adopted algorithms, it has several drawbacks. It is not ideal for object clusters because it leads to a misdetection if an object lies within the predefined overlap threshold. Several other algorithms try to fix the NMS's weaknesses.

3.4.2. Non-maximum merging (NMM)

From the nature of our problem: splitting the image, predicting the patches, and then stitching the resulting predictions; It is not unusual that the bounding boxes contain the cutoff objects. In this scenario, the NMS can only choose the best cut of the object, dumping the others. The non-maximum merging algorithm targets to solve this problem. It takes traditional NMS as the first step and matches all the detected boxes between themselves. Instead of keeping only the box with the higher confidence threshold and throwing away all the overlapping boxes, it merges them to form the new output box.

3.4.3. Greedy non-maximum merging (GREEDYNMM)

The greedy variant of non-maximum merging sorts the bboxes according to their confidence and removes the processed boxes from the list, similar to greedy non-maximum suppression, making the algorithm more efficient and effective.

The implementation details of all described post-processing algorithms can be found in the open-source library SAHI: Slicing Aided Hyper Inference [23], which we utilized in our execution. It is a lightweight vision library for performing large-scale object detection and instance segmentation.

3.5. Validation metrics

We utilize several metrics to interpret and understand the outputs correctly. e offer their definitions in Sections 3.5.1 and 3.5.2 below.

3.5.1. Metrics operating with the confusion matrix values

Thanks to the simplicity of one class, we can easily calculate the confusion matrix (sometimes named as error matrix) to analyze the model performance in greater detail. Confusion matrix in the context of object detection compares the results of the classifier under test with trusted external judgments (dataset ground truth) using the terms true positives (TP), true negatives (TN), false positives (FP), and false negatives (FN). The terms positive and negative refer to the classifier's prediction, and the terms true and false refer to whether that prediction corresponds to the ground truth.

There are several metrics associated with the confusion matrix from which we use the following ones:

$$Precision = \frac{TP}{numDet} * 100\%, \tag{3}$$

$$Recall = \frac{TP}{numGT} * 100\%, \tag{4}$$

$$F1\ score = \frac{2 * Precision * Recall}{Precision + Recall}, \tag{5}$$

$$False\ discovery\ rate = \frac{FP}{numDet} * 100\%, \tag{6}$$

$$False\ negative\ rate = \frac{FN}{numGT} * 100\%, \tag{7}$$

where numGT and numDet refers to the number of the ground true and detected objects, respectively.

The precision or positive predictive value gives the percentage of true positive samples from all the samples returned by the model. The recall value, sometimes named as sensitivity or true positive rate, informs us about the percentage of the objects correctly retrieved by the model from all the objects present. The F1 score is the harmonic mean of precision and recall. False discovery rate gives the fraction of incorrectly detected objects, while the false negative rate gives the fraction of objects left out by the model.

The authors of the recent review dealing with the fruit detection and counting [24] have made a recommendation for the use of F1 score as an overall performance measure, allowing models and methods to be compared and benchmarked with a single metric.

Aside from assessing the values stated above, we use the precision-recall curve as a diagnostic tool to judge model performance. Precision-Recall curve summarizes the trade-off between the true positive rate and the positive predictive value for a model using different probability thresholds. A high area under the curve represents both high recall and high precision, where high precision relates to a low false-positive rate, and high recall relates to a low false-negative rate. An average precision metrics described in the following section calculates just that area under a precision-recall curve, although the precise calculation methodology may differ.

3.5.2. Object detection evaluation according to COCO challenge

We also conducted the object detection evaluation according to COCO challenge[2] [25] because those are the main metrics currently used in research papers dealing with object detection. A 101-point interpolated mean average precision (AP) definition is used in the evaluation of this challenge. Its primary metric is then an AP averaged over multiple intersection over union (IoU) values; specifically, it uses 10 IoU thresholds ranging from 0.5 to 0.95 with step 0.05. This metric emphasizes the precision of the bbox localization. Please refer to the original challenge evaluation for the details about these metrics.

In addition to the calculation of the AP values, we also provide a detailed breakdown of false positives in Derek's PR curve inspired by [26]. This plot is a series of precision-recall curves where each PR curve is guaranteed to be strictly higher than the previous as the evaluation setting becomes more permissive. The curves are as follows:

• C75: AP at IoU=0.75 (AP at strict IoU), area under curve corresponds to AP at IoU=0.75 metric.

• C50: AP at IoU=0.50 (AP at PASCAL IoU), area under curve corresponds to AP at IoU=0.50 metric.

• Loc: AP at IoU=0.10 (localization errors ignored, but not duplicate detections). All remaining settings use IoU=0.1.

• Sim: AP after supercategory false positives are removed. Specifically, any matches to objects with a different class label but that belong to the same supercategory do not count as either a FP (or a TP). Sim is computed by setting all objects in the same supercategory to have the same class label as the class in question and setting their ignore flag to 1. Note that when we use a single category, Sim result is identical to Loc.

• Oth: AP after all class confusions are removed. Like the Sim, except now if a detection matches any other object, it is no longer a FP (or a TP). Oth is computed by setting all other objects to have the same class label as the class in question and setting their ignore flag to 1. Note that when we use a single category, Oth result is identical to Loc.

• BG: AP after all background (and class confusion) FP samples are removed. BG is a step function for a single category that is 1 until max recall is reached then drops to 0 (the curve is smoother after averaging across categories).

• FN: PR after all remaining errors are removed (trivially AP=1).

[2] The coco detection challenge evaluation is described here: https:// cocodataset.org/#detection-eval

4. Results

The results section is divided into three subsections. The first Section 4.1 deals with the validation and test set results evaluated on the cut images. The second one 4.2 shows the extensive analysis of the postprocessing parameters needed in the stitching procedure and evaluates it on the validation split data. The last Section 4.3 shows the outputs of the whole proposed methodology on the unseen data from the test set split.

4.1. Results before stitching

In this section, the training process and the preliminary results of the model are described. Fig. 1 illustrates how the loss value and results of the validation average precision evolve during the training. As an example, the medium resolution set-up was selected since the other resolutions performed in a similar way.

Table 1 presents average precision and recall values retrieved on the cut images of the test set (before stitching). e compare the results for all three image resolution set-ups (for more information about the patch sizes, refer to Section 3.2). e can see that both medium and high-resolution set-ups give promising results. On the other hand, the low-resolution plainly loses too many details by the image downsampling. Consequently, the smaller objects are not detected; both the average precision and the average recall for small objects were basically equal to zero. The high-resolution set-up shows slightly better performance than a medium resolution. However, there is a higher risk of miss detection on the patches boundaries, and the model needs to process five times more patches which enlarges the processing time. Therefore the medium resolution setting is used in our following post-processing parameter analysis.

4.2. Post-processing parameters analysis

This section deals with the analysis of post-processing parameters. These parameters determine the stitching process and, therefore, significantly impact the quality of the final output.

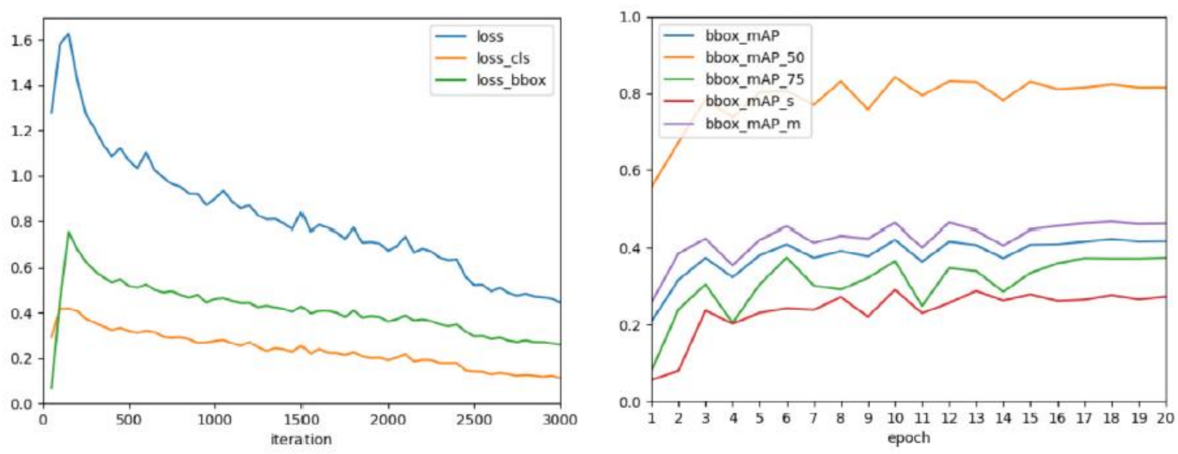

Fig. 1. Loss value and validation average precision development during the training process with the medium resolution set-up.

It is probably closely dependent on the dataset. e tested all combinations of the values listed below to see the parameters' influence on the output performance and discover the best settings for our application.

Table 1 COCO challenge style average precision results on cut images from test set (before stitching).

Metric	Low-res.	Medium-res.	High-res.
AP	0.309	0.433	0.450
AP@50IoU	0.620	0.819	0.845
AP@75IoU	0.269	0.407	0.427
AP@50IoU-small	0.049	0.573	0.672
AP@50IoU-medium	0.736	0.870	0.884

• model confidence thresholds:

- (0, 0.1, 0.2, 0.3, 0.4, 0.5, 0.6, 0.7, 0.8, 0.9, 0.99),

• match metrics:

- intersection over union (IoU),
- intersection over a smaller area (IOS),

• match metrics thresholds:

- (0.3, 0.5, 0.7, 0.9),

• post-processing algorithms:

- greedy non-maximum suppression (NMS),
- non-maximum merging (NMM),
- greedy non-maximum merging (GREEDYNMM).

For more information about the individual parameters, refer to Section 3.4. The analysis was conducted on the images from the validation split, using the medium resolution set-up.

Fig. 2 shows the precision-recall curve for each combination of post-processing parameters. To improve the readability, we clipped both axes to (0.4,1), displayed the results for each metric in a single graph, and then compared just the best solution for each of the metrics in the last graph. As we can see, the best match metrics and match threshold were interestingly the same for all the post-processing algorithms, i.e., IOS and 0.5 threshold value. The best results achieved by the three post-processing algorithms: NMS, NMM, and greedy NMM, are compared in a bottom-right graph of Fig. 2. e can see that both the non-maximum merging algorithms surpass the non-maximum suppression one. In our consequent analysis, we use the combination of the greedy NMM algorithm with the 0.5 threshold of the IOS match metric.

The bar plots in Fig. 3 demonstrate the model performance depending on the applied model confidence threshold. The smallest error was obtained while using the 0.4 confidence threshold. This threshold value also corresponds to the point where the precision value slightly surpasses the recall value. Thus we have chosen it for our final solution.

We applied the above-chosen post-processing parameters to evaluate all three resolution settings. Table 2 summarize validation results.

Table 2 Validation results for each patch resolution settings.

Metrics	Low-res.	Medium-res.	High-res.
num. Ground Truth	1343	1343	1343
num. Detected	840	1281	1327
True Positive	733	1069	1081
False Positive	107	212	246
False Negative	610	274	262
Precision	87.26%	83.45%	81.46%
Recall	54.58%	79.59%	80.49%
F1-score	67.16%	81.47%	80.97%
False-Discovery-Rate	12.74%	16.55%	18.54%
False-Negative-Rate	45.42%	20.40%	19.51%
Number of slices	4	12	60
Evaluation time* [ms]	1353	2422	7785
Image slicing* [ms]	199	211	240
Prediction* [ms]	1140	2192	7529
Post-processing* [ms]	14	19	16

**time needed for slicing, patches inference, and post-processing of one image of size 2448 x 10000 pixels, measured on GeForce GTX 1080 Ti GPU and AMD Ryzen 7 2700 Eight-Core Processor.*

As in the preliminary results before the stitching, the low-resolution setting is clearly the poorest. On the other hand, the medium and the high-resolution setting still show very similar performance. However, considering the computation demand and the F1-score as the main metric, the medium-resolution setting seems to be the better choice.

The final post-processing parameters were applied in time analysis, which is presented in Table 2. The timing was assessed as a mean from 50 runs over the same image of size 2448 x 10000 pixels. We excluded the time needed for model loading because it is constant for all settings. We can see that the slicing and post-processing times are similar in all cases. The number of patches implies the processing time, although the dependence is less than linear thanks to the GPU implementation and parallelization.

As a recap of the post-processing parameters analysis, the following set-up was chosen for our final evaluation:

- patch size: medium-resolution,
- model confidence threshold: 0.4,
- match metric: intersection over a smaller area,
- match metric threshold: 0.5,
- post-processing algorithm: greedy non-maximum merging.

4.3. Final test set results

Table 3 shows the final results on the test set split computed by the model with post-processing chosen in post-processing parameter analysis, i.e., medium resolution patches, 0.4 model confidence threshold, The absolute difference between the number of ground truth and detected objects is negligible, accounting only for 2.32% of the ground-true tomatoes.

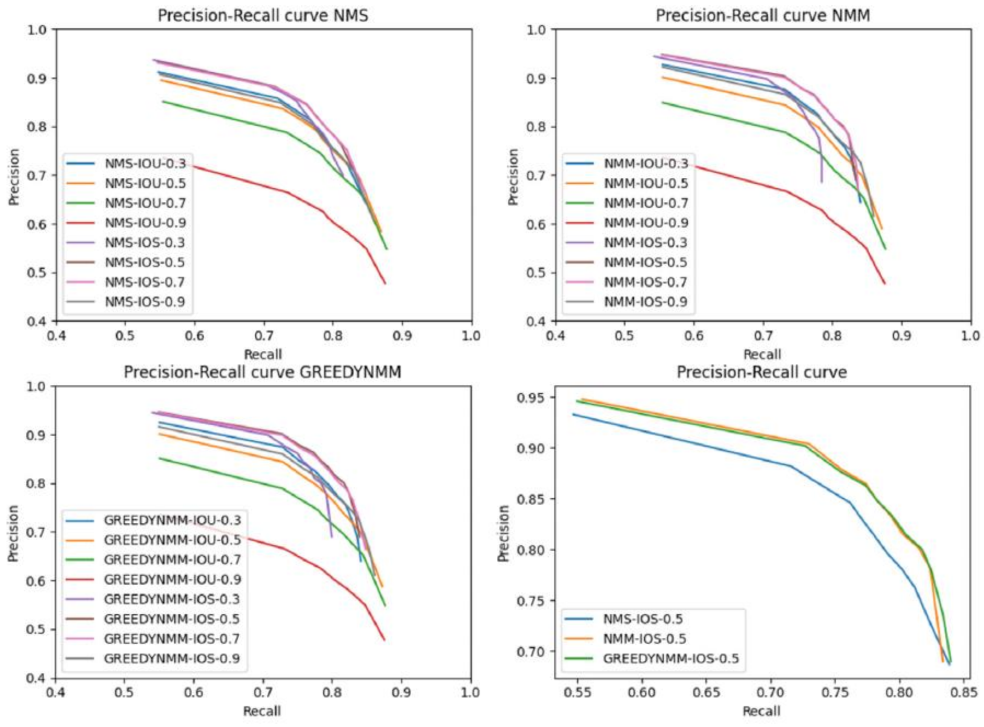

Fig. 2. Precision-recall curves for all combinations of the post-processing parameters, the bottom-right graph shows the comparison of the best precision-recall curve for each of the post-processing algorithms.

Table 3 Test results for the final proposed methodology settings: medium resolution cuts, 0.4 model confidence threshold, 0.5 IOS match metrics threshold, and greedy NMM post-processing algorithm.

num. Ground Truth	1382
num. Detected	1414
True Positive	1162
False Positive	253
False Negative	220
Difference	32
Precision	82.12%
Recall	84.08%
F1-score	83.09%
False-Discovery-rate	17.89%
False-Negative-rate	15.92%

The absolute difference between the number of ground truth and detected objects is negligible, accounting only for 2.32% of the ground-true tomatoes. However, this was caused by diminishing the false positive and false negative objects. The false-discovery and falsenegative rates are 17.89% and 15.56%, respectively. The precision and recall values 82.12% and 84.08% correspond to that. At last, the final F1 score value of our methodology is 83.09%. We can observe that the validation and the test set performance is comparable or even slightly better, indicating the model's ability to generalize.

COCO challenge style precision-recall curve and bar plot, which divides the results into the categories according to the object size (viz Section 3.2), are in Fig. 4. The area under each curve is shown in brackets in the legend. Our Faster R-CNN detector achieved the following precision values: the overall AP at I_oU>= 0.75 is 0.396, and liberating the localization to use I_oU>= 0.5 would increase AP to 0.811. Since we have only one class, there are no class confusions (both within supercategory and across supercategories). Removing background false positives would boost performance to 0.871 AP, and the rest of the errors are missing detections (although presumably, if more detections were added, this would also add lots of false positives).

In summary, our model’s errors are dominated by imperfect localization and background confusion. On top of that, the bar plot shows the drop in AP for small objects compared to medium objects (large objects are not present in the dataset at all). Fortunately, since the mediumsized objects dominate the dataset, it does not significantly affect the overall performance.

4.4. Visual analysis of test set results

Fig. 5 displays cut-out examples of tomato fruit detection showing both correctly detected objects and also some errors we deal with in our dataset. The leaf occlusion is one of problems. The fruits are sometimes entirely hidden from the view by the leaves. This error is not even included in our performance study since those tomatoes would not be annotated in the ground-truth image either. Sometimes the occlusion is partial, and we can assume the fruit presence but cannot be sure, and therefore the tomato cannot be included in the annotations. Such an example is in Fig. 5(c). Interestingly enough, the model predicted the fruit presence although with lower confidence.

Especially tiny green tomatoes present in high parts of a plant are very challenging to detect. It can be easily confused for leaves and vice versa. An example of such a misdetection is in Fig. 5(d). On the other hand, those fruits will need substantial time to grow and ripe before they can be harvested and therefore they are not the leading interest for our actual application. Nevertheless, it is one of the most common errors of our model. Therefore, we would seek to dissolve it in our future work by modifying the model itself, changing a loss function to accommodate training towards small object detection, or modifying the capturing methodology not to include the very high parts of the plant.

Another trouble with our dataset is that tomatoes/whole plants from a rear plant row are sometimes visible and mistakenly detected. This was the most common error that needed repair in the annotation process’s second (control) stage. So it can be very misleading even for a human annotator. An example of such false detection is depicted in Fig. 5(e). Although the capturing process was designed to minimize this problem, it is still present and challenging. One possible approach tosolve this problem would be foreground segmentation and removal of the background objects before the detection.

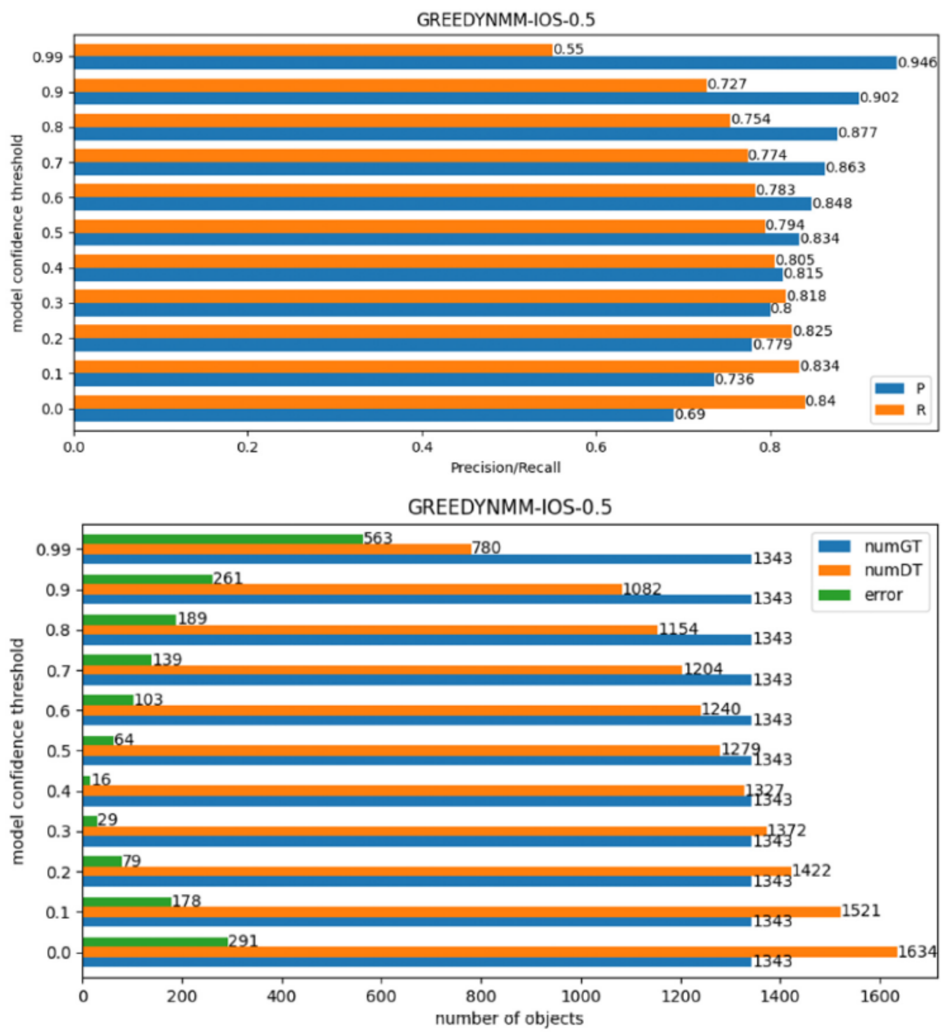

Fig. 3. Bar plots representing the performance for different model confidence threshold values, top: Precision (P) and Recall (R) values; bottom: Number of ground truth objects (numGT), detected objects (numDT), and the difference between them (error).

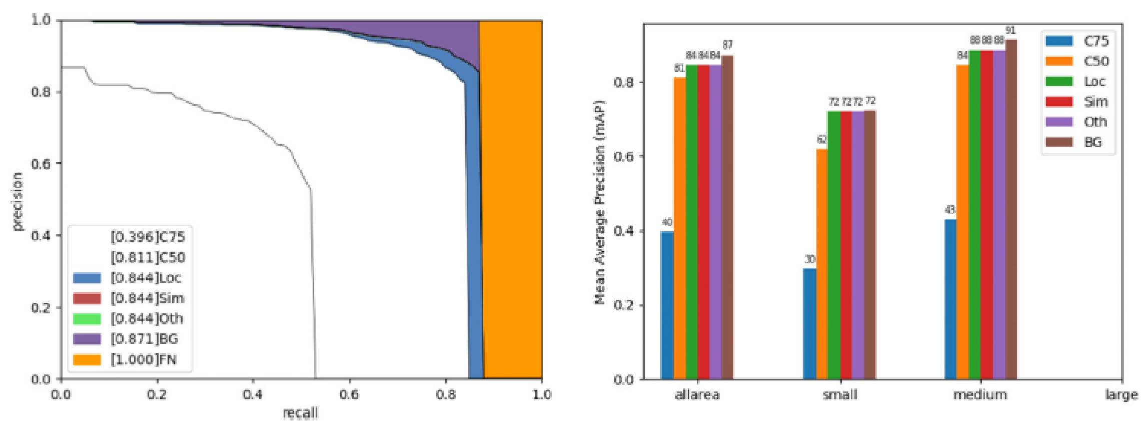

Fig. 4. COCO challenge style precision-recall curve (left side) and bar plot (right side) on the test set data for the final proposed methodology settings: medium resolution cuts, 0.4 model confidence threshold, 0.5 IOS match metrics threshold, and greedy NMM post-processing algorithm. For more information about COCO evaluation see Section 3.5.2.

Finally, some image defects might be caused by an uneven speed of the robot during the capturing process; for reference, see Fig. 5(f). Our study shows that while the deep convolutional neural network is able to deal with a certain level of distortion, there are limitations that should be controlled during the capturing process to obtain satisfactory results. Those limits will be studied concretely and employed in the subsequent development.

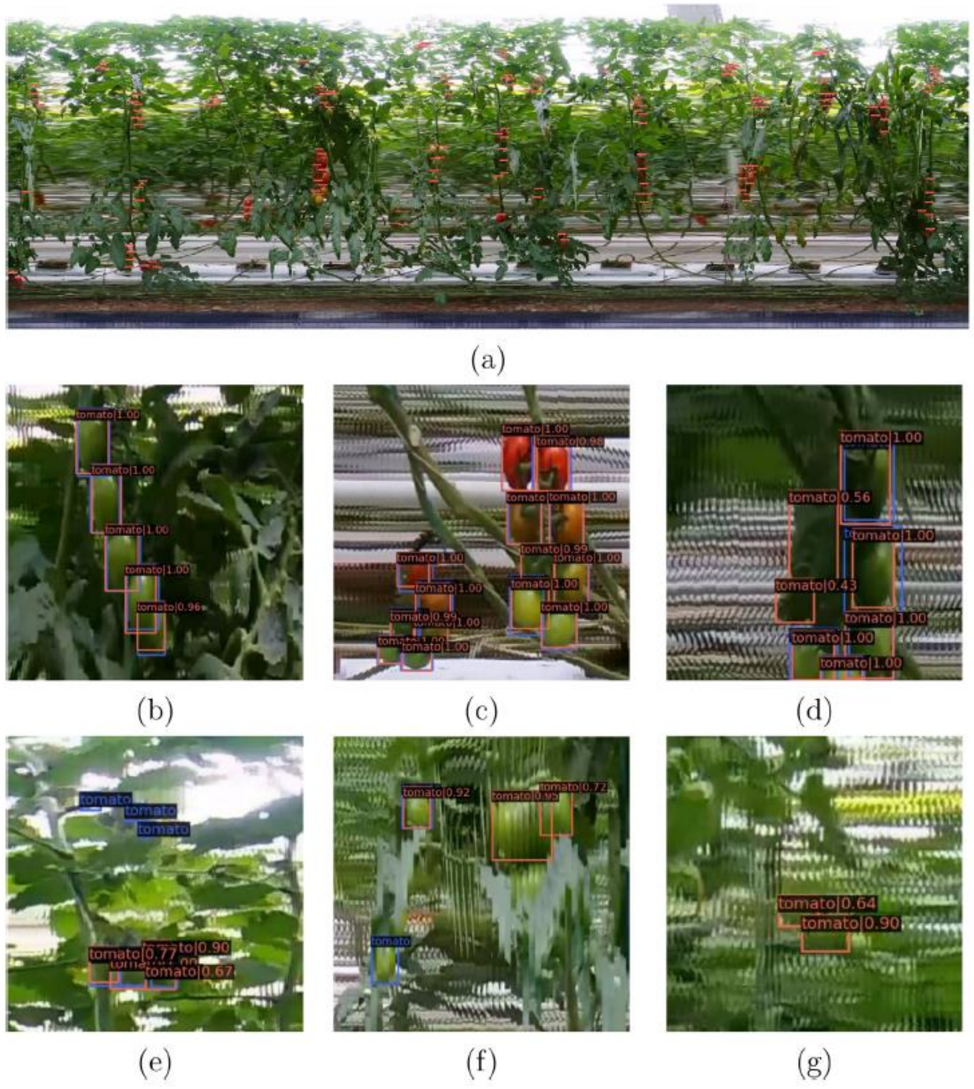

Fig. 5. An example of tomato fruit detection output, ground truth boxes are marked blue, detected boxes are labeled orange (or red) with a corresponding model confidence value; (a) shows a part of the whole tomato row panorama image with only model detections, (b) and (c) are correct detections, (d) leaf mistaken for fruit, (e) omission of small green tomatoes, (f) detection of tomatoes from a rear row, (g) impaired detection due to poor image quality.

5. Discussion

Our methodology for counting tomatoes shows relatively high precision on the test dataset. This section compares our solution with the recent tomato detection method's performance. Following the recommendation of [24], the F1-score of the test set detection in each experiment was calculated for comparison. The method and the best result of each paper are listed in Table 4. Please bear in mind that each paper uses a different dataset, and therefore, the comparison is just for a general idea of the method's performance.

Authors of YOLO-tomato detector [10] modify the well-known YOLOv3 architecture for object detection by adding the dense connections between the layers for feature extraction and by applying the circular bounding box instead of the classic rectangle. YOLO-tomato model achieves an excellent 93.91% F1-score, although the dataset used in this study consists of single images containing the tomato or tomato cluster captured manually. Moreover, this research only focused on detection and did not address the possibility of counting the objects in a greater context, which is an entirely different matter. Interestingly, in their stated comparison of different model architectures, the Faster R-CNN architecture (the one we apply in our study) achieves the second-best results, losing only 1.58% on F1-score. Those results indicate that choosing the Faster R-CNN network architecture in our study of the overall methodology for processing the whole tomato row images was a good option.

Table 4 The recent tomato detection method's performance comparison. Please note that the methods were evaluated on different datasets.

Method	Dataset	F1-score
Yolo v2 [10]	tomato truss	86.71%
Yolo v3 [10]	tomato truss	91.24%
Faster R-CNN r50 [10]	tomato truss	92.33%
Yolo-tomato [10]	tomato truss	93.91%
Faster R-CNN r101 [11]	part of tomato row	83.67%
Faster R-CNN r50 + sliced inference	whole tomato row	83,09%

An approach most similar to our methodology was presented in [11]. The authors manually capture the whole row of plants in a greenhouse. Still, the counting evaluation is unfortunately done only on single images before stitching. Just images from one example row are stitched to create a wide photograph to provide an example of yield mapping. Authors of [11] use a simple approach of merging the border boxes according to their distance and shape similarity; they do not provide any study of post-processing parameters nor a very detailed description of the stated merging method. hile both our and [11] paper uses the Faster R-CNN model, we use Resnet-50 while Resnet-101 is utilized in [11]. Aside from the fact that [11] does not evaluate the results on stitched images, our methodology deals with noisier data and applies shallower model architecture, yet it practically equalizes the final performance, losing only 0.56% on F1-score. Some performance gain is probably acquired during the post-processing phase of stitching, which supports the extended analysis presented in our work.ý

Finally, the paper [13] presents the same goal as our work, delivering an approach for detection, counting, and maturity assessment of cherry tomatoes using multi-spectral images and machine learning techniques. Authors train deep CNN networks on images captured in a tomato greenhouse. From a comparison of YOLOv3 and FasterRCNN architecture, the authors choose the FasterRCNN model for further analysis. Consequently, they use those models in combination with the DeepSORT algorithm to track the tomatoes in a video. Unfortunately, the results of final counts in the analyzed growth are not precisely stated nor commented on. The closest metric to the F1-score recommended for fruit counting evaluation is the IDF1-score, which estimates the total number of unique objects correctly identified in a scene. The value of the IDF1-score achieved is 51.4%. The results show that detecting and tracking objects in complicated and obscured scenes of tomato growth is very challenging. This favors our approach, which eliminates the tricky part of object tracking altogether.

6. Conclusion

This study presents a fully automated process of tomato fruit detection and counting. e utilize a unique data collection methodology employing the 360-degree camera to capture the whole tomato plant row in one pass. Consequently, a wide image representing the whole row is produced from a video. This image is used for tomato fruit counting, overcoming the need for object tracking or other processes to count the objects in a sequence of images/video frames.

On the other hand, the extra-wide image size exceeds any reasonable limits for processing at one pass, which we solve by applying the slicing-aided inference. We present an extensive examination of the post-processing parameters needed to stitch the predictions correctly. The results are visualized in Figs. 2 and 3 and create the core of this study.

The final test set results achieve the F1-score value of 83.09%, which is an acceptable outcome considering both the dataset's difficulty and the actual need for predictions in a greenhouse. Still, while we prove the suitability of our concept in this study, our future work will consider several improvements from different network architectures to enlarging/repairing the dataset.

Aside from the tomato counting, the output of the proposed methodology can be utilized to estimate the fruit's location and size. Therefore this method shows great potential for ripeness and yield prediction and might be successfully applied in robotic farming.

References

[1] K. Fuglie, The growing role of the private sector in agricultural research and development world-wide, Glob. Food Secur. 10 (2016) 29-38, http://dx.doi.org/ 10.1016/j.gfs.2016.07.005, URL https://www.sciencedirect.com/science/article/ pii/S2211912416300190.

[2] T. Short, C. Draper, M. Donnell, Web-based decision support system for hydroponic vegetable production, in: International Conference on Sustainable Greenhouse Systems-Greensys2004 691, 2004, pp. 867-870.

[3] R. Shamshiri, Measuring optimality degrees of microclimate parameters in protected cultivation of tomato under tropical climate condition, Measurement 106 (2017) 236-244, http://dx.doi.org/10.1016/j.measurement.2017.02.028, URL https://www.sciencedirect.com/science/article/pii/S0263224117301276.

[4] Y. Zhao, L. Gong, Y. Huang, C. Liu, A review of key techniques of vision-based control for harvesting robot, Comput. Electron. Agric. 127 (2016) 311-323, http://dx.doi.org/10.1016/j.compag.2016.06.022, URL https://www. sciencedirect.com/science/article/pii/S0168169916304227.

[5] A. Gongal, S. Amatya, M. Karkee, Q. Zhang, K. Lewis, Sensors and systems for fruit detection and localization: A review, Comput. Electron. Agric. 116 (2015) 8-19.

[6] X. Wei, K. Jia, J. Lan, Y. Li, Y. Zeng, C. Wang, Automatic method of fruit object extraction under complex agricultural background for vision system of fruit picking robot, Optik 125 (19) (2014) 5684-5689.

[7] S. Wan, S. Goudos, Faster R-CNN for multi-class fruit detection using a robotic vision system, Comput. Netw. 168 (2020) 107036, http://dx.doi.org/10.1016/ j.comnet.2019.107036, URL https://www.sciencedirect.com/science/article/pii/ S1389128619306978.

[8] H. Mure§an, M. Oltean, Fruit recognition from images using deep learning, 2017, arXiv preprint arXiv:1712.00580.

[9] Z.-F. Xu, R.-S. Jia, Y.-B. Liu, C.-Y. Zhao, H.-M. Sun, Fast method of detecting tomatoes in a complex scene for picking robots, IEEE Access 8 (2020) 55289-55299, http://dx.doi.org/10.1109/ACCESS.2020.2981823.

[10] G. Liu, J.C. Nouaze, P.L. Touko Mbouembe, J.H. Kim, YOLO-tomato: A robust algorithm for tomato detection based on YOLOv3, Sensors 20 (7) (2020) http: //dx.doi.org/10.3390/s20072145, URL https://www.mdpi.com/1424-8220/20/ 7/2145.

[11] Y. Mu, T.-S. Chen, S. Ninomiya, W. Guo, Intact detection of highly occluded immature tomatoes on plants using deep learning techniques, Sensors 20 (10) (2020) http://dx.doi.org/10.3390/s20102984, URL https://www.mdpi.com/ 1424-8220/20/10/2984.

[12] A.I.B. Parico, T. Ahamed, Real time pear fruit detection and counting using YOLOv4 models and deep SORT, Sensors 21 (14) (2021) http://dx.doi.org/10. 3390/s21144803, URL https://www.mdpi.com/1424-8220/21/14/4803.

[13] I.-T. Chen, H.-Y. Lin, Detection, counting and maturity assessment of cherry tomatoes using multi-spectral images and machine learning techniques, in: VISIGRAPP, 5: VISAPP, 2020, pp. 759-766.

[14] A. Rosenfeld, M. Thurston, Edge and curve detection for visual scene analysis, IEEE Trans. Comput. 100 (5) (1971) 562-569.

[15] N. Dalal, B. Triggs, Histograms of oriented gradients for human detection, in: 2005 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 1, CVPR'05, IEEE, 2005, pp. 886-893.

[16] P. Viola, M. Jones, Rapid object detection using a boosted cascade of simple features, in: Proceedings of the 2001 IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Vol. 1, CVPR 2001, IEEE, 2001, p. I.

[17] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2016) 1137-1149.

[18] N. Bodla, B. Singh, R. Chellappa, L.S. Davis, Soft-NMS — Improving object detection with one line of code, in: 2017 IEEE International Conference on Computer Vision, ICCV, 2017, pp. 5562-5570, http://dx.doi.org/10.1109/ICCV. 2017.593.

[19] J. Chu, Y. Zhang, S. Li, L. Leng, J. Miao, Syncretic-NMS: A merging nonmaximum suppression algorithm for instance segmentation, IEEE Access 8 (2020) 114705-114714, http://dx.doi.org/10.1109/ACCESS.2020.3003917.

[20] B.C. Russell, A. Torralba, K.P. Murphy, W.T. Freeman, LabelMe: a database and web-based tool for image annotation, Int. J. Comput. Vis. 77 (1-3) (2008) 157-173.

[21] S. Ren, K. He, R. Girshick, J. Sun, Faster R-CNN: Towards real-time object detection with region proposal networks, IEEE Trans. Pattern Anal. Mach. Intell. 39 (6) (2017) 1137-1149.

[22] K. He, X. Zhang, S. Ren, J. Sun, Deep residual learning for image recognition, in: 2016 IEEE Conference on Computer Vision and Pattern Recognition, CVPR, IEEE, Las Vegas, NV, USA, 2016, pp. 770-778, http://dx.doi.org/10.1109/CVPR. 2016.90, URL http://ieeexplore.ieee.org/document/7780459/.

[23] F.C. Akyon, C. Cengiz, S.O. Altinuc, D. Cavusoglu, K. Sahin, O. Eryuksel, SAHI: A Lightweight Vision Library for Performing Large Scale Object Detection and Instance Segmentation, Zenodo, 2021, http://dx.doi.org/10.5281/zenodo. 5718950.

[24] A. Koirala, K.B. Walsh, Z. Wang, C. McCarthy, Deep learning - method overview and review of use for fruit detection and yield estimation, Comput. Electron. Agric. 162 (2019) 219-234, http://dx.doi.org/10.1016/j.compag.2019.04.017, URL https://www.sciencedirect.com/science/article/pii/S0168169919301164.

[25] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, C.L. Zitnick, Microsoft coco: Common objects in context, in: European Conference on Computer Vision, Springer, 2014, pp. 740-755.

[26] D. Hoiem, Y. Chodpathumwan, Q. Dai, Diagnosing error in object detectors, in: European Conference on Computer Vision, Springer, 2012, pp. 340-353.