Rapid tomato DUS trait analysis using an optimized mobile-based coarse-to-fine instance segmentation algorithm

Dan Jeric Arcega Rustia[1,*], Guido Alexander Jansen[1], Selwin Hageraats[1], Joseph Peller[1], Rick van de Zedde[1], Cécile Marchennay[2], Wim Sangster[2], and Gosia Blokker[2]

[1]*Wageningen Plant Research, Wageningen University and Research, Wageningen, The Netherlands*
[2]*Naktuinbouw, Roelofarendsveen, The Netherlands*
[*]*dan.rustia@wur.nl*

Abstract

As climate change continues to impact agriculture, there is a growing demand for the discovery of new crop varieties in order to address key goals such as accelerated production, disease resistance, and overall improved quality. One of the necessary procedures before a crop variety is accepted for production is distinctness, uniformity, and stability (DUS) testing. However, the current practice of DUS testing relies primarily on manual examination with limited technological assistance. This work aims to provide a solution to this challenge by developing an algorithm for rapid tomato DUS trait analysis using a mobile application. An image dataset comprised of tray and individual tomato images was compiled using multiple mobile devices. A coarse-to-fine instance segmentation algorithm was developed to analyze the tray images by detecting individual tomato images and detecting tomato and peduncle scar contours. In order to accommodate different mobile devices and achieve finer measurements, a conditional upscaling approach was applied on each individual tomato image, with the support of super-resolution. Android ARCore was utilized to obtain distances of each tomato from the mobile device camera, enabling fast morphological measurements without using reference scales. The proposed algorithm has a precision of 0.99 in detecting each tomato from each tray image, while having IoU_{seg} values of 0.97 and 0.83 in segmenting tomato and peduncle scars, respectively. Manual vs. automated trait analysis results also show that the mobile application was able to measure traits with an error from 1.66% to 7.19%. From the best of our knowledge, this work presents one of the first mobile phone applications for rapid tomato DUS trait analysis.

1. Introduction

Currently, Europe alone boasts a staggering number of more than 6,000 tomato varieties [2, 11]. Each tomato variety has been selectively bred to adapt according to diverse local preferences in terms of taste and morphology. Despite this, there is a persistent demand for discovering more tomato varieties due to the adverse affects of climate change that eventually led to varying plant disease and climactic susceptibility [11]. But before a variety is accepted for production, distinctness, uniformity, and stability (DUS) testing has to be performed.

The objective of DUS testing is to determine trait variations within a species, ensure uniformity of traits within a variety, and evaluate the consistency of phenotypic characteristics. Traditionally, DUS tests are conducted manually by examination officers [3, 9]. In the case of tomato DUS testing alone, about 61 phenotypic traits need to be evaluated, demanding significant time and effort. These phenotypic traits encompass attributes such as size, shape, color, resistance to disease, and more. For individuals without specialized training, quantifying size and shape DUS traits entails the measurement of at least three tomatoes (representing three replicates) from each variety using a caliper. However, for experienced DUS examiners, this process is expedited by assigning categorical size and shape values based on their existing knowledge and the variety database. After trait collection, the variability within a variety is scrutinized to ascertain its uniqueness and consistency before gaining acceptance as a new variety. This intricate procedure not only consumes significant time and effort, but also introduces subjectivity when examiners employ distinct strategies and perspectives. Consequently, there is a growing proposition to use automated methods for measuring DUS traits.

Image processing is one of the first automated methods

for automated DUS testing. Image processing involves finding hand-crafted image features based on edges, blobs, and more, which are eventually used for classification and regression applications. [1] collected okra stem, flower, and seed using a DLSR camera for automatically measuring morphological and color-based characteristics using image processing methods. [4] used a flat-bed scanner to acquire images of 20 peanut pod varieties while analyzing each image using morphological analysis. Similarly, [7] acquired *Sinningia speciosa* images using flatbed scanners and measured color and spot traits using image processing. Based on the above-mentioned research works, it is evident that most automated DUS testing methods utilizing image processing are performed on images acquired under controlled lighting conditions. To overcome this limitation and provide a more versatile and robust solution, the use of deep learning is proposed.

Unlike image processing, deep learning involves automated extraction of image features by feeding an image through a series of neural network layers. [17] used a VGG16 neural network model for identifying varieties from scanned peanut pods, achieving an accuracy of 0.97. Similarly, [12] also used VGG16 for discriminating between different chickpea seeds, with an accuracy of 0.94. [15] employed a You Only Look Once (YOLO) neural network model for detecting the number of leaves on rice. [10] worked on distinguishing between different peach varieties using deep learning for analyzing visual near-infrared (VIS-NIR) spectral data. The recent works in deep learning show that it is a viable solution for more reliable DUS testing.

Currently, there is a scarcity of efforts aimed at the development of portable or mobile applications for image-based DUS testing. This work aims to develop a novel and faster automated method for measuring tomato DUS traits in variety testing. The specific objectives are: (1) to develop a lightweight and accurate algorithm for tomato traits analysis; (2) to apply mobile augmented reality for more reliable morphological measurements; (3) to develop a reliable mobile application for DUS trait analysis; and (4) to assess the ergonomic benefit of using an automated DUS trait analysis method vs. manual traits analysis. This work shall prove that mobile computing and deep learning can be a solution to accelerate DUS trait testing.

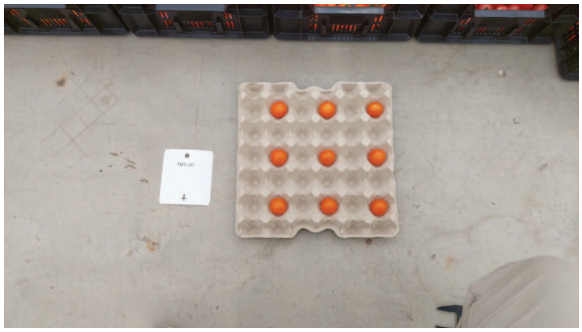
2. Methodology

2.1. Image acquisition and dataset preparation

An image dataset comprised of two subsets was collected in this work: tray and individual tomato. In this work, 63 different varieties of tomato were included in the dataset. The statistical information of the dataset is shown in Table 1.

Tray images The tray images are used as input to the coarse instance segmentation model which detects and estimates the size of each tomato. The tray images were obtained using three devices: Intel RealSense D415, Samsung Galaxy A52, and Google Pixel 3a, as shown in Figure 1. The Intel RealSense D415 was used for acquiring images in a closed environment. On the other hand, the Samsung Galaxy A52 and Google Pixel 3a were used for acquiring images in uncontrolled environments. In this context, uncontrolled environments refer to experimental greenhouses. Images were taken using device-to-tray distances, such as 60 cm, 80 cm, and 100 cm, to improve each model's scale invariability.

(a) RealSense D415



(b) Samsung Galaxy A52

(c) Google Pixel 3a

Figure 1: Tray images acquired using various imaging devices

Algorithm sub-routine	Model	Imaging device	Image resolution	Class	Instances
Coarse segmentation (trays)	YOLOv8-seg	Intel RealSense D415	1280x720	Tomato	1150
		Samsung Galaxy A52	640x480	Tomato	3861
		Google Pixel 3a	1920x1080	Tomato	606
Fine segmentation (tomato)	YOLOv8-seg	Intel RealSense D415	320x320	Tomato	382
				Peduncle scar	382
		Samsung Galaxy A52	320x320	Tomato	1287
				Peduncle scar	1287
		Google Pixel 3a	320x320	Tomato	202
				Peduncle scar	202

Table 1: Dataset statistical information

Tomato images The tomato images are used for training the fine instance segmentation model which aims to finely measure the size of each tomato and tomato peduncle scars. Each tomato image was collected by cropping from each tray image and resized into 320x320 pixels by conditional upscaling, as discussed later in Section 2.2.

The tray images were annotated with a single class: *tomato*, while the tomato images were annotated with two classes: *tomato* and *peduncle scar*. All annotations were performed using Darwin v7 annotation suite.

2.2. Coarse-to-fine tomato detection and segmentation algorithm

Each tray image was analyzed through a coarse-to-fine detection and segmentation approach, as shown in Figure 2. All algorithm routines were performed offline using the mobile phones without internet connection, for potential use in harsh and remote environments.

Coarse instance segmentation The purpose of coarse instance segmentation is to detect and get the contours of each tomato from the tray images. The deep learning model used to perform this task is YOLOv8. By the time of this writing, YOLOv8 is the latest version of the YOLO deep learning model, developed by Ultralytics [14]. Similar to its predecessor, YOLOv5, it is comprised of three components: backbone, neck, and head. Its backbone also uses cross-stage partial networks for image feature extraction but with minor modifications in its convolutional layers, while its neck combines the extracted features. Unlike YOLOv5, YOLOv8 uses an anchor-free detector, which means that it directly predicts bounding boxes and class probabilities for each object without considering offsets from predefined anchor box sizes [13]. By adding a fully convolutional network layer to the YOLOv8 head, YOLOv8 is also able to perform segmentation by producing mask coefficients; therefore called as YOLOv8-seg. Based on prior testing, it was found that the coarse instance segmentation routine was not sufficient for obtaining fine tomato contours and detect peduncle scars, due to the low resolution of the mobile device images. As a solution, each individual tomato image obtained through this step is used as input for conditional upscaling.

Conditional upscaling In this work, conditional upscaling refers to resizing an image to a larger resolution, using different methods based on the image size, to prepare the image for fine instance segmentation. As shown in Figure 2, if the pixel length (L) or pixel width (W) of an input image is smaller than a predefined threshold, a super-resolution model is used to upscale the image. Otherwise, the image is resized by cubic interpolation. The super-resolution model used in this work is Real Enhanced Super-Resolution Generative Adversarial Networks (Real-ESRGAN) [16]. Real-ESRGAN is a super-resolution model trained with pure synthetic data for generic restoration applications. Unlike other super-resolution models, it can be utilized even without fine-tuning and the availability of paired low and high resolution data, which are two major bottlenecks in super-resolution applications. Meanwhile, the predefined threshold was set to 64 since it was found to be the smallest resolution of the tomato images in which the peduncle scars were still visible.

Fine instance segmentation Fine instance segmentation aims to obtain finer contours of each tomato and detect peduncle scars. Fine contours for trait analysis are necessary since variations between different tomato contours should be easily distinguished for reliable DUS testing. Similar to coarse instance segmentation, YOLOv8-seg was used for fine instance segmentation but with two classes: tomato and peduncle scar. The tomato and peduncle scar contours obtained from this step are then retranslated based on the original image resolution for pixel to mm conversion in trait analysis.

2.3. Tomato DUS trait analysis

To obtain a complete shape representation of each tomato, each tray was prepared by placing tomatoes on three rows according to three placement orientations: top, side and bottom. The tomatoes in each row represent differ-

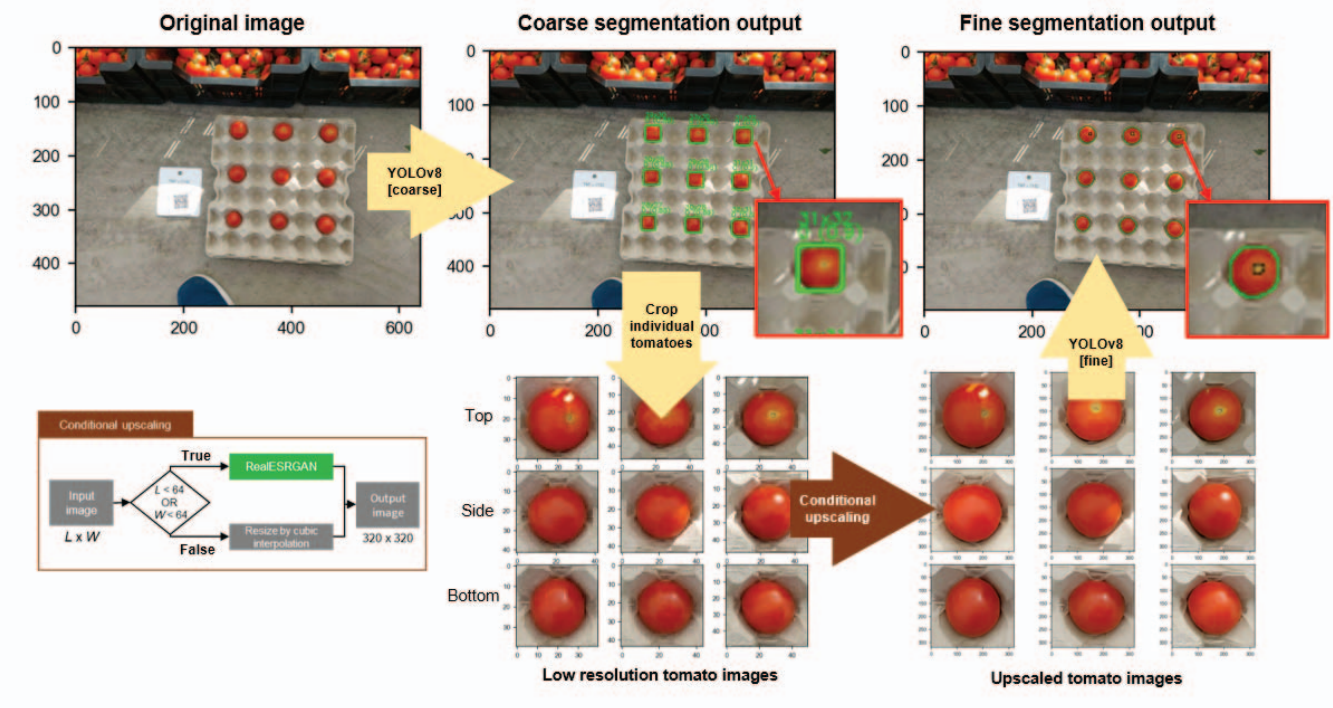


Figure 2: Coarse-to-fine tomato instance segmentation algorithm pipeline

ent sample replicates for obtaining statistically-backed measurements. The top and bottom tomatoes represent the *x* and *y* axes, while the side tomatoes represent the *x* and *z* axes, as shown in Figure 3. The lengths are represented as X_{ic}, Y_{ic}, and Z_{ic}, where *X*, *Y*, and *Z* are lengths in mm corresponding to axes *x*, *y*, and *z*, with *i* as index and *c* as class, such as tomato (t) and peduncle scar (ps).

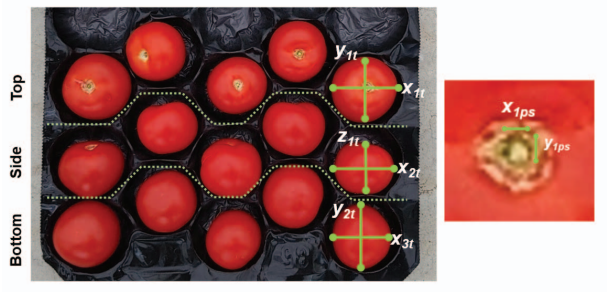


Figure 3: Measurement points for tomato DUS trait analysis

Depth measurement Depth data was collected using an Android mobile device, with Augmented Reality Core (ARCore) compatibility [8]. ARCore implements Simultaneous Localization and Mapping (SLAM) to track the orientation and position of a camera, relative to the world over time, which is referred to as a pose. By keeping track of changes of the position and orientation, SLAM can be used to determine the distance to an object that is within view of the camera.

In every recorded frame, a new pose is calculated and new depth information was acquired based on the depth image, as provided by the ARCore depth API. The depth API provides depth data in two ways: through a depth map or raw depth. By default, the depth API provides the depth map, which has an equal resolution to the camera RGB image, with matching distance estimations. However, the depth map is actually interpolated based on raw depth, making it less accurate. Therefore, the raw depth data was used in this work. Raw depth data is provided by the depth API as a 16-bit image equal to the size of the depth sensor, as recommended in the ARCore documentation for more accurate measurements [6].

Distance After obtaining individual tomato masks from the proposed algorithm, each mask centroid *x* and *y* pixel value pair was used to get a distance value. The individual distance values are stored in a byte buffer. In order to get the index corresponding to an *x*, *y* coordinate of the RGB image, the equation below was used:

$$Distance = \frac{RGB_x}{RGB_{width}/depth_{width}} * pixel\ stride + \frac{RGB_y}{RGB_{height}/depth_{height}} * row\ stride \quad (1)$$

where *x* and *y* are the coordinates of a pixel of the RGB image, pixel stride is the number of bytes used to store a value, and row stride is equal to the width of the depth map. Since

a 16-bit value uses 2 bytes, the pixel stride was equal to 2. Through this method, each tomato is assigned an individual distance measurement.

Pixel to millimeter conversion After determining the distance from the camera to the tomato, the image pixels were converted to mm. This can be done by measuring the Field of View (*FOV*), as calculated based on the focal length provided by the Android camera intrinsics. The conversion factors for *x* and *y* pixel values were calculated as below:

$$Pixels\,per\,mm_x = \frac{2 * \tan(0.5 * FoV_x) * distance}{RGB\ image\ width} \quad (2)$$

$$Pixels\,per\,mm_y = \frac{2 * \tan(0.5 * FoV_y) * distance}{RGB\ image\ height} \quad (3)$$

Trait extraction The algorithm is able to extract four tomato DUS traits including: fruit size, fruit shape ratio, peduncle scar size, and fruit volume, as calculated using the following equations:

$$Fruit\ size\ (mm^2) = X_t * Y_t \quad (4)$$

$$Fruit\ shape\ ratio = X_t / Y_t \quad (5)$$

$$Peduncle\ scar\ size\ (mm^2) = X_{ps} * Y_{ps} \quad (6)$$

$$Fruit\ volume\ (mm^3) = 4/3\pi * X_t * Y_t * Z_t \quad (7)$$

In this work, fruit size is the tomato area based on a 2-dimensional view, fruit shape ratio is the ratio between the tomato length (X_t) and width (Y_t) for determining tomato squareness, peduncle scar size is the peduncle scar size area based on a 2-dimensional view, and fruit volume is calculated based on ellipsoid volume.

2.4. Model training, optimization and deployment

YOLOv8-seg models were trained using a desktop computer, equipped with an Intel Xeon ES-1650 CPU and an NVIDIA GeForce GTX Titan X GPU, operating under Ubuntu OS 22.04. The primary training goal was to optimize model inference speed for mobile application deployment, with accuracy optimization as secondary goal. Different model sizes were used in training, namely YOLOv8-nano (YOLOv8n-seg), YOLOv8-small (YOLOv8s-seg), and YOLOv8-medium (YOLOv8m-seg), while learning rate was also iteratively tested with values of 0.05, 0.01, and 0.005, using stochastic gradient descent as optimizer. A train-validation-test split ratio of 70:20:10 were used in all trials while using a batch size of 2, learning rate of 0.01, and 50 iterations per trial. Each model was exported to ONNX format and deployed in the mobile phones using ONNX runtime [5]. All algorithm testing results were obtained from inferences using the exported ONNX models, deployed in a Samsung Galaxy A52 mobile phone.

2.5. Algorithm performance evaluation

The detection performance of the coarse and fine instance segmentation models were measured using precision, as measured using the following equation:

$$Precision = \frac{TP_{det}}{TP_{det} + FP_{det}} \quad (8)$$

where TP_{det} is a true positive detection and FP_{det} is a false positive detection. Each detection was automatically evaluated by measuring the detection Intersection-over-Union (IoU_{det}) between each set of ground truth bounding box coordinates with each predicted box coordinates, as computed below:

$$IoU_{det} = \frac{area(B_1 \cap B_2 \cap ...B_i)}{area(B_1 \cup B_2 \cup ...B_i)} \quad (9)$$

where B_i is the bounding box coordinates of each detected object, with *i* as the object index. B_i includes four coordinates: x_1, y_1, x_2, and y_2, where x_1 and y_1 belong to the object's *x* and *y* vertex box coordinates, and x_2 and y_2 belong to the vertex opposite to x_1 and y_1. IoU_{det} values closer to 1 indicate higher overlap and 0 otherwise. If the TP_{det} between two paired coordinates was higher than 0.5, it was considered as a TP_{det}.

Meanwhile, segmentation performance was measured using segmentation Intersection-over-Union (IoU_{seg}), which is a slight variation of IoU_{det}. Instead of matching box coordinates, TP_{seg} measures the overlap between individual pixels of the ground truth masks and predicted masks, as shown below:

$$IoU_{seg} = \frac{TP_{seg}}{TP_{seg} + FP_{seg} + FN_{seg}} \quad (10)$$

where TP_{seg} is the number of correctly predicted pixels, FP_{seg} is the number of incorrectly predicted pixels, and FN_{seg} is the number of predicted pixels that are outside of the ground truth mask.

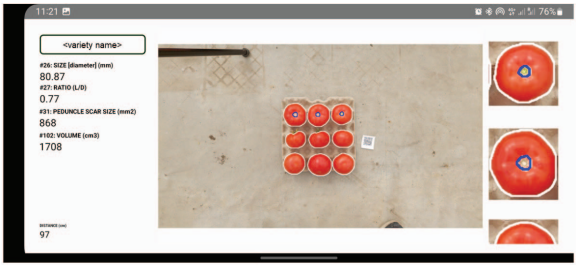
2.6. Trait quality evaluation experimental setup

Manual reference measurements were taken to evaluate the accuracy of the ARCore distance measurements and the automatically measured traits.

Experimental setup A Pixel 3a phone was used for testing with a fixed setup. The phone was clipped to a holder attached to a tripod, resulting in a fixed height of 954 mm from camera to the ground. The distance was determined to be a reasonable height in which the examiner can hold the phone and see the samples on-screen.

There were 3 varieties used for testing, consisting of 9 tomatoes each. Tomatoes were spaced out evenly on the black tray resulting in a 3 x 3 grid representing the 3 orientations. The center of the camera was focused on the tomato

(a) Main menu



(b) Sample results

Figure 4: Mobile application screenshots

in the middle of the grid, referenced as *Side 2*. For each tomato, relevant X_t, Y_t and Z_t values were manually determined, as well as the distance from the center of the tomato to the camera as measured using a tape measure.

ARCore distance measurement evaluation A total of 5 replicate images per variety were obtained for evaluating the distance measurement accuracy of ARCore, resulting into 114 individual tomato measurements, while excluding missed detections. For each measurement, the error was determined in reference to manual measurement.

2.7. Mobile application user interface

For practical use, a mobile application was developed. Upon opening the mobile application, as shown in Figure 4, the user can press *Open camera* to start taking images. While attempting to acquire an image, the user is notified to take the images 50 cm to 120 cm away from the target samples, which is found to be the best distances for acquiring high quality images. To simplify the process of logging the tomato variety information, the user can enable the QR code scanner; otherwise, the tested tomato variety will be labelled as *unknown*. The user can view the individual analysis results by pressing the image of a tested variety in the main menu, as shown in Figure 4b. Finally, the user can export all the results into a .csv file by pressing *Export*.

3. Results and discussion

3.1. Algorithm optimization

3.1.1 Coarse instance segmentation

The coarse instance segmentation model optimization results are summarized in Table 2. In terms of detection performance, it can be clearly seen that the precision of the model slightly decreased after shrinking the model size and input size. But in terms of segmentation performance, the IoU_{seg} of all the models were relatively too low for accurate measurements, which proves that the coarse instance segmentation step was not yet enough to obtain reliable DUS traits. But based on the optimization goal mentioned in Section 2.4, The YOLOv8n-seg, with input size of 640 x 640, was selected as the coarse instance segmentation model for deployment to accurately detect all tomatoes with minimal inference time.

Model	Input size	Precision	IoU_{seg}	Ave. inference time
YOLOv8n-seg	320x320	0.96	0.86	335 ms
YOLOv8s-seg	320x320	0.97	0.86	515 ms
YOLOv8m-seg	320x320	0.96	0.86	834 ms
YOLOv8n-seg	640x640	0.99	0.86	820 ms
YOLOv8s-seg	640x640	0.99	0.87	1491 ms
YOLOv8m-seg	640x640	0.99	0.86	2795 ms

Table 2: Coarse instance segmentation model optimization summary

Model	Input size	Precision	IoU_{seg}	Ave. inference time
YOLOv8n-seg	320x320	0.94	0.91	292 ms
YOLOv8s-seg	320x320	0.94	0.91	330 ms
YOLOv8m-seg	320x320	0.95	0.91	650 ms
YOLOv8n-seg	640x640	0.94	0.90	437 ms
YOLOv8s-seg	640x640	0.94	0.92	1113 ms
YOLOv8m-seg	640x640	0.93	0.91	2532 ms

(a) Tomato class

Model	Input size	Precision	IoU_{seg}	Ave. inference time
YOLOv8n-seg	320x320	0.72	0.65	292 ms
YOLOv8s-seg	320x320	0.72	0.65	330 ms
YOLOv8m-seg	320x320	0.72	0.65	650 ms
YOLOv8n-seg	640x640	0.73	0.70	437 ms
YOLOv8s-seg	640x640	0.73	0.70	1113 ms
YOLOv8m-seg	640x640	0.73	0.71	2532 ms

(b) Peduncle scar class

Table 3: Fine instance segmentation model testing summary, with cubic interpolation for upscaling

Model	Input size	Precision	IoU_{seg}	Ave. inference time
YOLOv8n-seg	320x320	0.94	0.97	504 ms
YOLOv8s-seg	320x320	0.94	0.97	542 ms
YOLOv8m-seg	320x320	0.98	0.97	862 ms
YOLOv8n-seg	640x640	0.96	0.97	649 ms
YOLOv8s-seg	640x640	0.96	0.97	1325 ms
YOLOv8m-seg	640x640	0.97	0.97	2744 ms

(a) Tomato class

Model	Input size	Precision	IoU_{seg}	Ave. inference time
YOLOv8n-seg	320x320	0.84	0.84	504 ms
YOLOv8s-seg	320x320	0.84	0.84	542 ms
YOLOv8m-seg	320x320	0.84	0.83	862 ms
YOLOv8n-seg	640x640	0.85	0.83	649 ms
YOLOv8s-seg	640x640	0.85	0.82	1325 ms
YOLOv8m-seg	640x640	0.86	0.82	2744 ms

(b) Peduncle scar class

Table 4: Fine instance segmentation model testing summary, with super-resolution for upscaling

3.1.2 Fine instance segmentation

The results of testing the inferences of the fine instance segmentation model are shown in Tables 3 and 4. Among the models tested, the YOLOv8n-seg with an input size of 320 x 320 was selected for deployment since it had a reasonable IoU_{seg} and best inference time for both classes. The lowest inference time had to be considered for fine instance segmentation since there were a minimum of 9 tomatoes to analyze per image.

It clearly shows that using super-resolution for upscaling allows the model to detect the peduncles more reliably compared to using cubic interpolation alone. It can also be noticed that the segmentation performance was boosted from 0.91 to 0.97 for the tomato class and from 0.70 to 0.83 for the peduncle scar class, if considering the YOLOv8n-seg with input size of 320 x 320.

In total, the average inference time of the algorithm was about 5.4 s, if considering 9 tomatoes per tray. Despite the slight increase in inference time, the results prove that applying super-resolution for mobile computing was beneficial since some mobile phone images have poor quality. As summary, it was shown that the coarse-to-fine instance segmentation approach generated better results compared to using only a single instance segmentation model.

3.2. Trait analysis quality evaluation

Sample manual and automatic measurements taken for a single variety are shown in Tables 5 and 6, respectively. The testing summary for all the varieties is also shown in 7.

Based on the reference measurements in Tables 5 and 6, the automated measurements show that the distance from the camera to each tomato can be reliably estimated, with an average error of 2.44%, as shown in Table 7. Among the three axes of measurement, it was observed that the X_t values had the largest errors due to variations within the variety in which some were more round and square.

Tomato	Distance	X_t	Y_t	Z_t
Top 1	909	81	72	-
Top 2	901	87	81	-
Top 3	902	90	81	-
Side 1	882	81	-	54
Side 2	874	82	-	55
Side 3	891	84	-	59
Down 1	906	81	76	-
Down 2	894	82	76	-
Down 3	914	74	74	-
Average	897	82.44	76.67	56

Table 5: Manual measurements (mm) for variety 1 over the X, Y and Z axis as defined in Figure 3

One limitation of ARCore is that the depth map resolution cannot be exactly the same as the corresponding RGB image without interpolation. Thus, distances per pixel cannot be easily obtained, even though it may improve the measurements. There were also few cases in which a tomato was not properly in place, causing measurement discrepancies. This causes an issue in terms of image perspective since there are unavoidable variations in how the images were acquired. Therefore, it is recommended to apply some correction based on the shooting angle and other related factors. But even though the results needed some improvement, the average absolute measurement error for the tested varieties were close to 5%, which was the allowable error percentage by the examiners.

3.3. Ergonomic benefit assessment

Based from the algorithm optimization results, the proposed algorithm has a total average processing time of 5.4 s. By including the average image acquisition time of 6.6 s, which involves camera focusing and finding the right distance from camera to tray, the total average measurement time was about 12 s. On the other hand, manually measuring the traits and writing up the results take an average time of 57 s. It clearly shows that the mobile application was able to accelerate DUS trait analysis and provide support for DUS examiners. As future consideration, the models can also be selected based on the phone model to optimize over-all performance.

Tomato	Distance	% error	X_t	% error	Y_t	% error	Z_t	% error
Top 1	895	-1.54	85.5	5.56	77.2	7.22	-	-
Top 2	888	-1.44	88.3	1.49	84.2	3.95	-	-
Top 3	879	-2.55	90.3	0.33	81.1	0.12	-	-
Side 1	882	0.00	83.1	2.59	-	-	56.4	4.44
Side 2	882	-0.92	80.2	-2.20	-	-	59.3	7.82
Side 3	877	-1.57	86.2	2.62	-	-	61.8	4.75
Bottom 1	885	-2.32	82.6	1.98	78.1	2.76	-	-
Bottom 2	882	-1.34	81.3	-0.85	78.4	3.16	-	-
Bottom 3	900	-1.53	76.5	3.38	75.9	2.57	-	-
Average	886	-1.26	83.78	1.66	79.15	3.24	59.17	5.67

Table 6: ARCore measurements (mm) for variety 1 over the X, Y and Z axis as defined in Figure 3

Variety	% error (distance)	% error (X_t)	% error (Y_t)	% error (Z_t)
1	2.57	4.49	3.67	2.79
2	2.66	7.09	4.49	2.12
3	2.10	9.98	8.37	0.07
Average	2.44	7.19	5.51	1.66

Table 7: Average absolute measurement error for all tested varieties

4. Conclusion

This work has proven that deep learning and augmented reality were useful techniques for building a reliable mobile application for rapid DUS trait analysis. The coarse-to-fine instance segmentation algorithm, coupled with super-resolution, was useful for finding hard to measure tomato traits from various image resolutions. ARCore was proven to be capable of measuring object distances with an acceptable error, compared to manual measurement. The biggest challenges lie in distortion due to perspective but can be mitigated as future improvement. As future work, the mobile application shall also be developed for other agricultural produce in order to promote the use of faster DUS trait analsis methods.

5. Acknowledgements

This work was funded by the Horizon 2020 Framework Programme (Grant No. 817970) under the project name Innovations in Plant Variety Testing in Europe (INVITE).

References

[1] Gopinath Bej, Abhra Pal, Tamal Dey, Sabyasachi Majumdar, Amitava Akuli, Alokesh Ghosh, and Nabarun Bhattacharyya. Extraction of appearance-based dus characteristics of okra stem, flower, and seed using image processing. In *Proceedings of International Conference on Computational Intelligence, Data Science and Cloud Computing*, pages 209–223. Springer Nature Singapore, 2022.

[2] Jose Blanca, Clara Pons, Javier Montero-Pau, David Sanchez-Matarredona, Peio Ziarsolo, Lilian Fontanet, Josef Fisher, Mariola Plazas, Joan Casals, Jose Luis Rambla, Alessandro Riccini, Samuela Palombieri, Alessandra Ruggiero, Maria Sulli, Stephania Grillo, Angelos Kanellis, Giovanni Giuliano, Richard Finkers, Maria Cammareri, Silvana Grandillo, Andrea Mazzucato, Mathilde Causse, Maria José Díez, Jaime Prohens, Dani Zamir, Joaquin Cañizares, Antonio Jose Monforte, and Antonio Granell. European traditional tomatoes galore: a result of farmers' selection of a few diversity-rich loci. *Journal of Experimental Botany*, 73(11):3431–3445, 2022.

[3] J. Borys, B. Kowalczyk, and J. Waszak. Distinctness, uniformity and stability testing of tomato varieties in poland. *Acta Physiologiae Plantarum*, 22(3):225–229, 2000.

[4] Limiao Deng and Zhongzhi Han. Image features and dus testing traits for peanut pod variety identification and pedigree analysis. *Journal of the Science of Food and Agriculture*, 99(5):2572–2578, 2019.

[5] ONNX Runtime developers. Onnx runtime. `https://onnxruntime.ai/`, 2021.

[6] Google. Use raw depth in your android app. `https://developers.google.com/ar/develop/java/depth/raw-depth`, 2023.

[7] Hao-Chun Hsu, Kung-Ling Hsu, Chuan-Yi Chan, Chun-Neng Wang, and Yan-Fu Kuo. Quantifying colour and spot characteristics for the ventral petals in sinningia speciosa. *Biosystems Engineering*, 167:40–50, 2018.

[8] Micheal Lanham. *Learn ARCore-Fundamentals of Google ARCore: Learn to build augmented reality apps for Android,*

Unity, and the web with Google ARCore 1.0. Packt Publishing Ltd, 2018.

[9] Sameena Lone, Khursheed Hussain, Khalid Masoodi, Dr Narayan, Hussain Mazahir, Majid Rashid, and Harish Kumar. Distinctness, uniformity and stability testing of various cherry tomato accessions. *Journal of Pharmacognosy and Phytochemistry*, 10:Journal of Pharmacognosy and Phytochemistry, 2021.

[10] Dian Rong, Haiyan Wang, Yibin Ying, Zhengyong Zhang, and Yinsheng Zhang. Peach variety detection using vis-nir spectroscopy and deep learning. *Computers and Electronics in Agriculture*, 175:105553, 2020.

[11] Henk J. Schouten, Yury Tikunov, Wouter Verkerke, Richard Finkers, Arnaud Bovy, Yuling Bai, and Richard G. F. Visser. Breeding has increased the diversity of cultivated tomato in the netherlands. *Frontiers in Plant Science*, 10, 2019.

[12] Amin Taheri-Garavand, Amin Nasiri, Dimitrios Fanourakis, Soodabeh Fatahi, Mahmoud Omid, and Nikolaos Nikoloudakis. Automated in situ seed variety identification via deep learning: A case study in chickpea. *Plants*, 10(7):1406, 2021.

[13] Juan Terven and Diana Cordova-Esparza. A comprehensive review of yolo: From yolov1 to yolov8 and beyond. *arXiv preprint arXiv:2304.00501*, 2023.

[14] Ultralytics. Yolov8, 2023.

[15] Mukesh Kumar Vishal, Biplab Banerjee, Rohit Saluja, Dhandapani Raju, Viswanathan Chinnusamy, Sudhir Kumar, Rabi Narayan Sahoo, and Jagarlapud Adinarayana. Leaf counting in rice (oryza sativa l.) using object detection: A deep learning approach. In *IEEE International Geoscience and Remote Sensing Symposium*, pages 5286–5289.

[16] Xintao Wang, Liangbin Xie, Chao Dong, and Ying Shan. Real-esrgan: Training real-world blind super-resolution with pure synthetic data. In *2021 IEEE/CVF International Conference on Computer Vision Workshops (ICCVW)*, pages 1905–1914.

[17] Haoyan Yang, Jiangong Ni, Jiyue Gao, Zhongzhi Han, and Tao Luan. A novel method for peanut variety identification and classification by improved vgg16. *Scientific Reports*, 11(1):15756, 2021.