IEEE Access
Multidisciplinary ⋮ Rapid Review ⋮ Open Access Journal

Received 7 January 2025, accepted 27 January 2025, date of publication 30 January 2025, date of current version 5 February 2025.

Digital Object Identifier 10.1109/ACCESS.2025.3536701

RESEARCH ARTICLE

Novel Fusion Technique for High-Performance Automated Crop Edge Detection in Smart Agriculture

F. MARTÍNEZ[1], JAMES B. ROMAINE[1], P. JOHNSON[2], (Senior Member, IEEE), A. CARDONA RUIZ[1], AND PABLO MILLÁN GATA[1]

[1]Departamento de Ingeniería, Universidad Loyola Andalucía, 41704 Seville, Spain
[2]School of Engineering, Liverpool John Moores University, L2 2QP Liverpool, U.K.

Corresponding author: F. Martínez (fbmartinez@al.uloyola.es)

This work was supported in part by the Junta de Andalucía (Projects oliVAr) under Grant GOPO-SE-23-0001.

ABSTRACT Optimising vegetable production systems is crucial for maintaining and enhancing agricultural productivity, particularly for crops like lettuce. Separating the crop from the background poses a significant challenge when using automated tools. To address this, a novel technique has been developed to automatically detect the vegetative area of lettuces, optimising time and eliminating subjectivity during crop inspections. The proposed deep learning model integrates the YOLOv10 object detector, the K-means classifier, and a segmentation method known as superpixel. This combination enables lettuce area identification using bounding box labels instead of contour labels during training, improving efficiency compared to other methods like YOLOv8 and Detectron2. Additionally, the combination of the YKMS method with YOLOv8 (YKMSV8) is evaluated, where YKMS serves as a label assistant. These methods are also used as benchmarks to compare the proposed approach. For the training of each methods, a custom database has been created using a low-cost, low-power custom IoT node deployed on a real farm to provide the most accurate data. Throughout the comparison, a custom metric is used to evaluate performance both in training and inference, balancing computational cost and area error, making it applicable in agriculture. Performance metric is associated with computational cost factor and accuracy factor whose value are respectively 65 % and 35 %, ensuring applicability for autonomous agricultural devices. Computational cost is prioritised to maintain battery life during extended campaigns. The results of the custom metric during inference indicated that the YKMSV8 method achieved the highest performance, followed by Detectron2, YOLOv8, and, lastly, YKMS. Regarding area error, YOLOv8 exhibited the lowest mean error, followed by Detectron2, while YKMSV8 and YKMS produced similar values. In terms of inference time, YKMSV8 was the most computationally efficient, followed by YOLOv8, YKMS, and, finally, Detectron2.

INDEX TERMS Computer vision, object detection, YOLOv8 segmentation, YOLOv10, superpixel, K-means, threshold, detectron2, smart agriculture.

I. INTRODUCTION

A. MOTIVATION

With the growing global demand for food and the effects of climate change limiting the availability of resources such as water and arable land, the optimisation of vegetable production systems has become more critical than ever. In this context, the precise measurement of physiological parameters, such as leaf area, is considered essential for improving productivity and agricultural sustainability, particularly in crops like lettuce. Leaf area is regarded as a vital indicator in plant physiology, providing detailed information about the plant's health and growth conditions. Measurements of leaf area are used to assess the plant's capacity for photosynthesis,

The associate editor coordinating the review of this manuscript and approving it for publication was Muhammad Khalil Afzal.

the process by which sunlight is converted into chemical energy by plants [1], [2].

Generally, an increase in leaf area is taken to indicate that the plant is healthy and developing well. Nitrogen, a key nutrient for plant growth, is a primary component of chlorophyll, the pigment used by plants for photosynthesis. If adequate nitrogen levels are present in lettuce, the leaves tend to be larger and greener, resulting in an increased leaf area [3].

Leaf area can also serve as a broader indicator of the plant's overall health. By knowing the exact growth status of plants, fertilisers can be applied more precisely and in the necessary amounts, reducing excessive use and lowering costs [4]. Reduced leaf area may indicate that the plant is experiencing stress, lacks essential nutrients, receives insufficient water, or is affected by disease. Therefore, frequent monitoring of leaf area size is considered valuable for preventing unnecessary fertiliser application in lettuce cultivation [5]. Although manual methods for measuring leaf area are accurate, they are impractical for large-scale farming systems. In contrast, automated methods based on artificial intelligence are recognised for their ability to reduce time and costs, enabling continuous and accurate monitoring. Such methods provide farmers with real-time information to improve decision-making and crop management. For instance, as presented in [6], an automated device integrated with an algorithm for the identification of weeds and insects operates autonomously, combining real-time data acquisition with advanced image processing techniques to detect and classify various types of weeds and insects accurately. This solution helps farmers optimise pest management and improve crop yields by reducing the need for manual inspection and enabling more targeted application of treatments.

Although accurate, manual methods for measuring leaf area are impractical on a large scale. Automated techniques based on artificial intelligence are acknowledged not only for reducing time and costs but also for minimising destructive intervention, allowing continuous, real-time monitoring of plant health. Several methods are traditionally relied upon by farmers to inspect leaf area. Among the most common are manual measurements, where each leaf is measured individually. While highly accurate, this method is laborious, time-consuming, and often destructive to the plants. The planimetric method, which employs planimeters to measure the perimeter of the leaves and calculate the total area, is also labour-intensive and can be destructive. Similarly, the gravimetric method estimates leaf area index based on the relationship between biomass and leaf area by measuring the dry weight of the leaves [7].

Today, the adoption of artificial intelligence methods is beginning to alleviate these tedious processes [8]. The YOLOv8 and Detectron2 methods have been applied to identify the leaf area of lettuce. While these methods have proven effective for object detection, they require a preliminary step of detailed labelling, which consumes significant time and resources, as it involves coordinates representing the object's contour. In fact, dedicated labelling tasks are often assigned to handle this process due to its complexity and labour intensity. To overcome the limitations of current methods, the YKMS method is proposed, which simplifies the labelling process by predicting contours from bounding boxes, significantly reducing labour time and accelerating the implementation of the process in lettuce crops.

B. STATE OF THE ART

Crop detection and identification through contour detection from an image involves three specific challenges: 1. The separation of a specific crop from the background, 2. The separation of the specific crop from other crops, and 3. The detection of the exact leaf area in order to assess the growth and health of the leaf area. The separation of a specific crop from the background has been identified as a significant challenge for the application of artificial intelligence techniques. Various approaches have been proposed by researchers to address this problem, each with its advantages and limitations. One such method is the use of the Colour Index of Vegetation Excess Green Index (CIVE), as mentioned in [9]. This method has been shown to be particularly useful in green-coloured crops, as its main function is the identification of the colour green. However, its applicability is limited to cases where the crop is green, and its effectiveness may be compromised if the crop colour changes due to diseases or nutritional deficiencies, preventing precise detection. In addition, this article explores common computer vision methods for pixel classification, such as k-means and Support Vector Machines (SVM). The k-means method is used to cluster similar pixels and has been combined with semantic segmentation techniques to improve results in images with complex weed presence, as described in [10]. This approach has been compared with the superpixel method, which groups neighbouring pixels with similar characteristics to simplify the image and reduce computational load while maintaining important features such as edges and textures [11], [12]. While traditional methods such as those mentioned have proven to be effective in specific cases, limitations in accuracy and generalisability to variable field conditions, such as changes in crop colour or complex backgrounds, have been noted. To address these challenges, convolutional neural network-based models have emerged as a more robust solution.

A widely used method for segmentation today is based on deep convolutional neural networks (CNNs). Among the most employed are Mask R-CNN [13], [14], YOLOv8 [15], and Detectron2 [16], [17]. YOLOv8 is notable for being the first official YOLO model capable of performing segmentations, requiring a robust database to train the model to detect the contours of the objects of interest.

The applicability of these models in agriculture has been demonstrated, as shown in [18], where the contour of canola

pods has been identified using YOLOv8, along with a comparison to Detectron2 using Mask R-CNN. This study has highlighted how these advanced models can significantly improve the accuracy and efficiency of segmentation in agricultural images.

When discussing CNNs, their performance is typically evaluated based on the accuracy with which the object or the contour of the mask is identified. However, an often-overlooked aspect is the computational cost required during the inference of the images. In [19] and [20], a metric has been proposed that involves not only accuracy but also computational cost.

Other methods widely applied in agriculture belong to the YOLO (You Only Look Once) family of object detection [21]. Although the primary aim of this article is contour identification, the relevance of object detection to the proposed YKMS method makes it important to describe the evolution of this family of techniques. An object detector identifies bounding boxes around objects of interest. The first model developed in PyTorch was YOLOv5, and since then, numerous versions of YOLO have been developed. The most recent iteration is YOLOv10, an object detector designed to introduce substantial improvements in computational cost and prediction efficiency.

In [22], YOLOv5 and YOLOv7 were applied for insect detection in crops. While promising results have been obtained in terms of precision and the reduction of computational costs, the evolution of YOLO has continued with YOLOv10 [23], achieving improvements not only in accuracy but also in computational efficiency. This improvement is primarily due to enhancements in the prediction of the bounding boxes that enclose the objects. YOLOv10 demonstrates superior accuracy, particularly in detecting small objects and in complex scenarios, without compromising inference speed [24].

C. CONTRIBUTION

This work evaluates three techniques to calculate the area of lettuce, considering both computational cost and area error. Among the techniques considered, YOLOv8 and Detectron2 employ a labelling method that requires the coordinates enclosing the contour of the object in question to be known. On the other hand, the technique proposed by the authors is based on the use of bounding boxes, which facilitates the labelling process. This is considered a crucial part of the process in preparing the input data of the neural network for further learning.

The YKMS technique proposed by the authors combines YOLOv10, k-means, and superpixels. Labelling training images in YOLOv5 format with bounding boxes is regarded as requiring less time and effort than the contour labelling needed by other methods.

YKMSV8, which combines YKMS and YOLOv8 techniques, was also evaluated. In this method, predictions from the YKMS method are used as input labels for the YOLOv8 process. This approach leverages the strength of the YKMS technique in creating accurate labels along with the ability of YOLOv8 to require less time for making predictions. In effect, the YKMSV8 technique uses YKMS as a label assistant for YOLOv8.

Images were captured using a low-cost, autonomous device at BioAlverde farm in Seville, Spain. The goal is to provide farmers with a remote tool to monitor crop health, specifically by evaluating lettuce leaf area and growth percentage, while balancing computational cost and precision.

The remainder of this article is structured as follows: Section II explains the image acquisition process, algorithms, and metrics used for evaluation. Section III presents the training of each of the methods along with the segmentation results and error metrics. The work concludes with a brief conclusion and future directions in Section IV.

II. METHODOLOGY AND EXPERIMENTATION

A. EXPERIMENTAL SETUP

The images used for training were collected by an autonomous device called a vision node deployed at a farm in Spain named BIOAlverde. The node is composed of a Raspberry Pi 3, on which the algorithms were loaded, and it utilises a pair of USB cameras with OV2710 chips (HBV-1716WA model) to capture the images. Additionally, a Pycom microcontroller (Lopy) was employed to send the processed data via LoRa technology to a gateway, which then transmitted the data to a server named TTS (via WiFi or 4G), where it was later stored in a database from which the website retrieves the information to enable visualisation of the processed data. The database consists of 324 images, although 76 images were sufficient to train the neural networks and achieve the desired results. In Fig. 1, the vision node at the farm is shown, which was first presented in [25].

FIGURE 1. Vision node deployed on the farm.

One of the objectives was to use low-cost resources, which is why the selected camera does not have the capacity to adapt to changes in light. To solve this problem of possible excessive light, two time bands were chosen for image

capture, specifically at sunrise and sunset, to create the dataset.

On the other hand, the trainings were conducted using resources available in Google Colab. This platform provides access to a high-end CPU, which is usually sufficient to run many types of code, including moderate-sized machine learning models. Additionally, the available resources include free access to an Nvidia Tesla K80 or T4 GPU, which are useful for accelerating the training of deep learning models. Although these GPUs are beneficial, their performance may be lower compared to more powerful GPUs available on other platforms. Google Colab also offers 12 GB of RAM and a limited amount of storage space in the Colab environment, which is reset every time the session is closed. With the free version of Colab, access to a maximum runtime of 12 hours is granted; after this period, the session is automatically disconnected, and all unsaved data is lost.

B. YKMS

The method proposed by the authors, YKMS, is shown in Fig. 2. The first step evolved ROI (Region of interest) identification involved the application of the YOLOv10 method is used to identify objects and plays a pivotal role in the identification of the ROI.

In addition to the parameters and metrics used in YOLO for object detection, YKMS employs complementary techniques such as superpixel segmentation, and k-means clustering to improve the accuracy of segmentation and analysis of detected areas whose process is shown in Fig. 2.

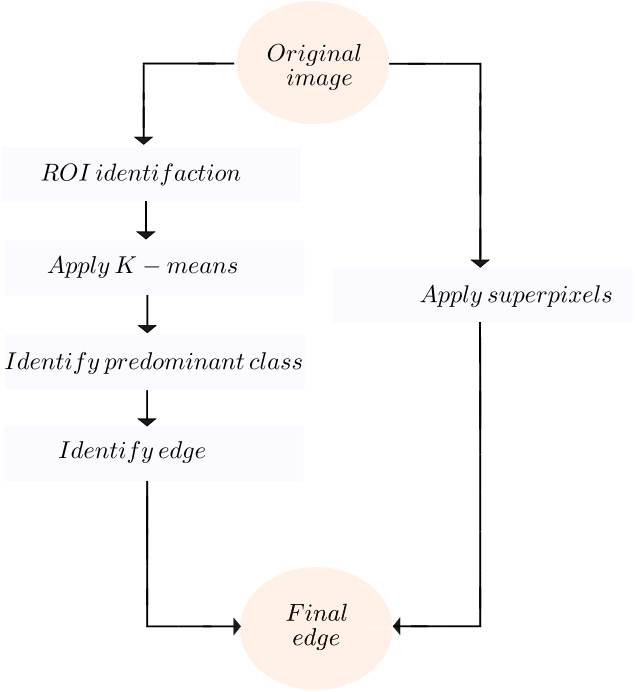


FIGURE 2. YKMS method process.

YOLOv10, manual labelling was required. In this case, software called Roboflow was used for labelling.[1] The labelling system is shown in Fig. 3, where a bounding box is drawn around the object and repeated for all images to complete the database. To use this database, the label information was exported in YOLOv5 format, where these labels were normalised, meaning each pixel value was scaled between 0 and 1 to facilitate learning.

FIGURE 3. Label with Roboflow to YOLOv10.

The next step after the label obtained in the specific format was training, for which a pre-trained network was used. A pre-trained network is a neural network model that has been trained on a large dataset and is then used as a starting point to identify characteristics of a specific object, achieved by training the last layer of this pre-trained model. Therefore, training only the last layer is sufficient to adapt it to the identification of a specific object, such as lettuce. To train, the dataset he pre-trained YOLOv10s weights were chosen because they offer the highest training speed with moderate accuracy available in the Ultralytics repository,[2] along with everything necessary to implement this method and ultimately obtain the desired object identification.

After the implementation of the YOLOv10 object detector, which identifies the bounding boxes, the proportion of the image contained within the boxes is isolated, as this is the area that will be used from this point onwards. The new area of interest changes from the RGB (red, green, blue) colour space to the CIELUV workspace, which uses the **L**, $\boldsymbol{a}^*$, and $\boldsymbol{b}^*$ channels, where **L** represents illumination, $\boldsymbol{a}^*$ is the chromaticity on the purple-green axis, and $\boldsymbol{b}^*$ is the chromaticity on the blue-yellow axis. The **L** channel is used for further processing, as the luminosity reflected by a surface is similar across it. Thus, any area of the surface belonging to the lettuce will have similar luminosity values. The **L** channel of this image fragment is referred to as the Region of Interest (ROI).

In parallel to ROI identification, the superpixel method is applied to the original image. This method, based on the k-means method and known as SLIC (Simple Linear Iterative Clustering), is one of the most widely used methods for superpixel image segmentation due to its simplicity and efficiency. Seeds are evenly distributed in the image, serving as the initial centres of the superpixels to perform

[1] https://roboflow.com/

[2] https://github.com/THU-MIG/yolov10

the clustering of similar pixels. Each pixel in the image is assigned to the nearest superpixel centre. The distance considered is a combination of the spatial distance and the colour distance in CIELAB space [12]. The superpixel method groups nearby pixels with similar characteristics into areas. Among all the areas corresponding to the superpixels obtained from the original image, only those contained within the bounding boxes are considered.

To determine the area considered part of the lettuce, an intersection between the results obtained from the k-means method and the superpixel method is performed. The k-means method alone cannot perfectly identify the outline of the lettuce due to variations in luminance, making it necessary to combine it with the superpixel method. The superpixel method groups similar pixels in the image, helping to identify areas belonging to the lettuce more accurately. While the k-means method effectively separates the lettuce from the background, it does not achieve good segmentation on its own. By combining it with the superpixel method, the segmentation accuracy is improved. Finally, the outline of the preserved areas is identified as the final outline of the lettuce, with any areas that do not have at least 20 % of their area within the contour being eliminated.

1) YKMS PARAMETERS, METRICS

In this first part, the usage of YOLO within the YKMS algorithm to detect objects is explained, alongside how YOLO parameters and metrics contribute to improving the accuracy of object detection in the RGB-based workspace.

To utilise YOLOv10, certain basic parameters must be considered, which are described below:

epochs: An epoch refers to a complete iteration over the training dataset.

batch_size: The batch size defines how much data is processed before the model parameters are updated.

learning_rate: The learning rate controls the step size at each iteration as the model moves toward minimising the loss function.

Intersection over Union (IoU): This metric not only considers *IoU*, which measures the overlap between the labelled object box and the predicted box, but also accounts for the alignment of their centres and the aspect ratio of the boxes. This provides a more comprehensive measure of how well the predicted bounding box matches the ground truth [26].

CIoU is defined as follows:

$$\text{CIoU} = \text{IoU} - \frac{\rho^2(\mathbf{b}, \mathbf{b}^g)}{c^2} - \alpha v \tag{1}$$

where:

$$\rho(\mathbf{b}, \mathbf{b}^g) = \sqrt{(x - x^g)^2 + (y - y^g)^2} \tag{2}$$

$$c = \sqrt{(w + w^g)^2 + (h + h^g)^2} \tag{3}$$

$$v = \frac{4}{\pi^2}\left(\arctan\left(\frac{w^g}{h^g}\right) - \arctan\left(\frac{w}{h}\right)\right)^2 \tag{4}$$

$$\alpha = \frac{v}{(1 - \text{IoU}) + v} \tag{5}$$

where:

$\mathbf{b}$ and $\mathbf{b}^g$ are the predicted and ground truth bounding boxes, respectively. $\rho(\mathbf{b}, \mathbf{b}^g)$ is the Euclidean distance between the centres of the predicted and ground truth bounding boxes. c represents the diagonal length of the smallest enclosing box that contains both $\mathbf{b}$ and $\mathbf{b}^g$. v measures the aspect ratio consistency between the predicted and ground truth bounding boxes. α is a weight factor that balances the influence of v.

A higher *CIoU* value, with a maximum of 1, represents greater precision in predicting the object's position and size. It is a key parameter for evaluating the accuracy of an object detection model's predictions. A value of 1 indicates perfect alignment between the predicted and actual bounding boxes, and approaching this value reflects higher detection precision [23]. However, when annotations cover significantly different areas, achieving a high CIoU is not necessarily critical for YOLO's performance. YOLO is designed to detect and localise objects accurately, even if the bounding boxes are not perfectly drawn. In such cases, CIoU may not fully represent detection quality, and other factors, such as the accuracy of object identification, become more relevant in evaluating YOLO's effectiveness.

Moreover, the CIoU score is often used to evaluate how well the predicted bounding box matches the actual location and size of the object. Although CIoU is not directly involved in calculating the confidence score, it is essential for training the model and improving the accuracy of its predictions.

Multiple object categories: Object detection models must identify and localise multiple object categories within an image. Therefore, Average Precision (AP) calculates the accuracy for each category, and the average of these is then taken.

The metric corresponding to YOLOv10 involves the confidence score, which in object detection models like YOLOv10 is calculated as the product of two factors: the probability that a bounding box contains an object (P_{obj}) and the probability of the most likely class (P_{class}). The first factor, object probability (P_{obj}), is determined during the training process by evaluating the likelihood that a bounding box encloses a relevant object. This probability is obtained by applying a Sigmoid function to the model's predicted logits.

The second factor, class probability (P_{class}), evaluates how likely it is that the object inside the bounding box belongs to a particular class. This probability is computed by applying a Softmax function to the class logits and selecting the highest probability.

Consequently, the confidence score serves two purposes: it quantifies the model's certainty in the presence of an object within a specific bounding box, and it indicates the expected accuracy of that box's positioning. A higher confidence score suggests a stronger belief in both the object's presence and the correctness of the bounding box's boundaries.

F1-Score: The F1-Score is computed separately for each category and then combined to form an overall curve. This combined curve helps visualise the trade-off between precision and recall at different confidence levels. It is particularly useful for selecting the best confidence threshold for the entire model, especially where the overall F1-Score is highest. This method helps optimise the balance between precision and recall, ensuring the model performs well across all detected categories. Additionally, the F1-Score is calculated as:

$$F1 = 2 \times \frac{Precision \times Recall}{Precision + Recall}, \tag{6}$$

In order to evaluate the model, the following parameters must be taken into account:

Precision and recall: Precision measures the accuracy of the model's positive predictions, while recall measures the proportion of actual positive cases that are correctly identified by the model. In both scenarios, precision and recall must be calculated. These metrics provide a balanced assessment by considering the area under the precision-recall curve [27], and they are defined as follows:

$$\text{Precision} = \frac{TP}{TP + FP} \quad \text{Recall} = \frac{TP}{TP + FN}$$

where TP represents true positives, FP stands for false positives, and FN indicates false negatives [28].

True Positives (TP) occur when a positive instance is accurately identified by the model. In object detection, a TP happens when the model correctly detects and classifies an object within an image, including precisely placing a bounding box around it. The criteria for a TP can vary, often involving a threshold for classification confidence and, in detection tasks, a minimum Intersection over Union (IoU) with a ground truth bounding box.

False Positives (FP) arise when an incorrect positive result is predicted by the model. In classification, this means assigning an instance to an incorrect class. In object detection, an FP can result from either misclassifying an object or correctly identifying the class but placing the bounding box inaccurately (e.g., with no significant overlap with any ground truth object or overlapping with an object of a different class).

False Negatives (FN) occur when the model fails to predict a positive result that should have been identified. In classification, this happens when a positive instance is incorrectly classified as negative or not recognised as belonging to the relevant class at all. In object detection, an FN might occur when the model overlooks an object entirely or fails to meet the criteria for a TP, such as not detecting the object with sufficient confidence or the bounding box not meeting the IoU threshold with the ground truth.

The Average Precision (AP) is used to evaluate the performance of the model, measuring precision across all categories. It is based on the precision-recall metric, seeking the intersection over union [29]. In the evaluation of classification and object detection models, performance metrics such as precision, recall, and Average Precision (AP) are crucial. These metrics are derived from the concepts of True Positives (TP), False Positives (FP), and False Negatives (FN), each representing a specific type of prediction outcome.

Non-maximum suppression (NMS): This technique is used for filtering overlapping bounding boxes, since the detector can identify several bounding boxes around the same object. NMS selects the one with the highest confidence score [26]. The threshold for determining whether the detection is sufficiently confident is referred to as the confidence score, a value assigned by the model that indicates the probability that a detection is correct. A higher score signifies greater certainty in the detection and accuracy of the bounding box.

YOLOv10 does not use predefined anchor boxes. Instead of starting with preset sizes and aspect ratios, it directly predicts the coordinates and dimensions of the bounding boxes for objects in the image. For each cell in the grid covering the image, the model predicts the centre coordinates, width, and height of the bounding boxes, along with a confidence score that reflects the likelihood that the box contains an object.

During the training of object detection models like YOLO, several performance curves are tracked to assess how well the model is learning. The two primary curves are the boxes loss curve (**box_loss**) and the classification loss curve (**cls_loss**). The object loss curve (**box_loss**) measures the difference between the predicted coordinates and the actual coordinates of the bounding boxes, while the classification loss curve (**cls_loss**) reflects how the model's capacity to accurately classify objects evolves during training. Monitoring these curves is crucial for fine-tuning the model's training process and ensuring effective learning [30].

C. YOLOv8

This subsection describes the process for employing the YOLOv8 method. First, when labelling images in Roboflow for segmentation, it is necessary for the labels to contain points corresponding to the contour. It should be noted that for the model to predict correctly, these contours must be as accurate as possible. The aforementioned labelling process is illustrated in Fig. 4, and the labels are normalised to values between 0 and 1. Once all the images had been labelled, the training proceeded using the Ultralytics repository for segmentation.[3] The yolov8n-seg weights, a pre-trained model available in the mentioned repository, were fine-tuned with our own database, thus obtaining a new model capable of identifying lettuce. After this process, it was tested to determine whether the new model could indeed detect lettuce.

YOLOv8 uses the same parameters as YOLOv10, as mentioned in Section II-B. However, it includes additional parameters related to segmentation, such as *object_loss* and variables associated with mask model evaluation, including *mask_precision*, which measures improvements in the accuracy of segmentation masks, and *dfl_loss*, which relates to

[3]https://ultralytics.com/assets/coco2017val.zip

FIGURE 4. Label with Roboflow for YOLOv8 and Detectron2.

the improvement of anchor point accuracy in bounding box predictions. A lower loss indicates an improvement in anchor point accuracy. Additionally, *seg_loss* evaluates the model's accuracy in segmenting areas of interest in the image [31].

A variation in the parameters compared to YOLOv10 occurs in the **Complete Intersection over Union (CIoU)** parameter. In previous models, the IoU (Intersection over Union) parameter was used to measure the overlap between two bounding boxes, calculated as the intersection area divided by the union area of the boxes. However, in YOLOv8, the CIoU parameter not only considers the intersection of areas between the labelled object box and the predicted box but also takes into account the alignment of their centres, as well as the aspect ratio of the boxes. This provides a more comprehensive measure of how well the predicted bounding box matches the ground truth.

YOLOv8 predicts bounding boxes differently. In YOLOv8, a list of anchor box proposals is generated across the image. These predefined boxes, with specific sizes and aspect ratios, are used as reference points during object detection, arranged in a grid across the image. During detection, the model predicts bounding boxes within these anchor boxes and adjusts them to better match the objects in the image. Once the bounding box proposals have been generated and adjusted based on the anchor boxes, a technique called Non-Maximum Suppression (NMS) is applied to eliminate duplicates. NMS reviews the predictions for the same object and selects the bounding box with the highest confidence score, thus preventing multiple predictions for the same object [32].

D. DETECTRON2

This section describes the process for employing the Detectron2 method, a library developed by Facebook AI Research (FAIR) for object detection and segmentation, built on the PyTorch software library. It supports various detection and segmentation models such as Mask R-CNN, Faster R-CNN, Cascade R-CNN, and DensePose. In this work, Mask R-CNN was employed for segmentation [33]. Detectron2 is an open-source platform, and the implementation used the repository from Facebook Research,[4] utilising the pre-trained weights from the COCO dataset. Detectron2 was selected because it can learn with a limited number of images, which is a significant advantage in agriculture, as obtaining the training dataset requires substantial time and effort [34].

The process to implement Detectron2 beginning with image labelling, the labels created in Section II-C were used. However, in this case, the labels had to be un-normalised before training. For this application, both normalised and un-normalised values were tested, and it was found that normalised values caused the detector to malfunction. Therefore, un-normalised values were used. Finally, after the new model had been applied to the lettuce, the images were inferred to observe the segmentation detection.

Detectron2 selects areas of the image through the Region Proposal Network (RPN), which evaluates different parts of the image to identify potential objects of interest. This process is iterative and optimised during model training [35].

The adjustable parameters of Detectron2 include: the number of classes (in this case, 1, for lettuce), the learning rate (depending on the complexity of the features to be learned), and the number of iterations (chosen based on the model's behaviour). The model reaches its learning limit when the evaluation values stop decreasing. The following parameters are crucial for identifying the model's progress:

Total_loss: A consistent decrease in total loss indicates that the model is learning and improving. If it stabilises, it suggests that the model has reached its maximum learning capacity for the current data or that the learning rate parameter needs adjustment.

RPN_localization_loss: Measures how well the model predicts the object locations.

RPN_classification_loss: Evaluates how accurately the model identifies regions likely to contain a specific class, compared to the actual labels of the proposed regions.

Classification_loss: Measures the model's accuracy in classifying objects, refining the RPN classification loss.

Box_regression_loss: Indicates the error in predicting the coordinates of the bounding boxes around objects.

mask_loss: A consistent decrease in this value indicates that the model is learning the mask and improving.

Detectron2 uses several key metrics to evaluate model performance, similar to YOLO. These include:

Precision: The ratio of true positives to the total number of positive predictions, indicating the accuracy of the detections.

Recall: The ratio of true positives to the total number of actual objects in the image, indicating how many real objects were correctly detected.

AP (Average Precision): The mean of precision scores calculated at various recall levels, used to evaluate the overall performance of the model [17].

E. YKMSV8

It is proposed to combine the YKMS bounding box labelling method to predict labels that is used as input in the YOLOv8 method, with the aim of leveraging the strengths of each

[4] https://github.com/facebookresearch/detectron2

method and optimising not only the prediction results but also the robustness and stability of the system.

F. EXPERIMENTAL EVALUATION

When making evaluations in computer vision methods that employ deep learning, precision is typically considered. However, when applying these methods to an automated system in agricultural settings, such as on a farm, it is important to consider using devices with processors that might not have high computational capacity. This can reduce the cost of the equipment, consume less battery during operation, and make it more feasible to acquire them in larger quantities. It is an aspect to be considered in understanding the computational cost employed.

Therefore, it is proposed to use the equation presented in [19], which was originally used in RNN; however, in this case, it will be applied with a CNN approach. This equation reflects the balance between computational cost and error, where high values indicate that the model is performing effectively within its own limits, and low values indicate that the model does not demonstrate robustness and stability against the various values it encounters. This equation prevents a metric with larger magnitude values from dominating the combined metric, which is the case with accuracy.

$$\begin{aligned} \varepsilon_{error} &= \left(\frac{w_a}{\alpha_a - \beta_a} \times (\alpha_a - x_a) \right), \\ \varepsilon_{time} &= \left(\frac{w_t}{\alpha_t - \beta_t} \times (\alpha_t - x_t) \right) \end{aligned} \tag{7}$$

where ε_{error} is the performance metric from the error area, ε_{time} is the performance metric from the computational cost, α_a is the largest RMS, β_a is the smallest RMS, α_t is the longest time taken, β_t is the shortest time taken, w_a is the error weighting factor, w_t is the computational cost weighting factor, and $w_a + w_t = 100$. x_a is the actual RMSE, and x_t is the actual time taken.

Finally, the performance metric is ε:

$$\varepsilon = \varepsilon_{error} + \varepsilon_{time} \tag{8}$$

To evaluate the method proposed by the authors (YKMS), as well as the Detectron2 and YOLOv8 methods, a metric that involves both area error and computational cost is used. The YKMS method uses neural networks only for bounding box detection and not for edge detection, thus having only one possible prediction. As a result, the accuracy metric, which measures the confidence of the network in its predictions by considering discarded predictions, does not apply to YKMS. Therefore, we will use the adjusted precision metric error between the contour predictions of each method and the manually created labels. Additionally, we will evaluate the time required to perform these contour predictions. The area error involved by the contour is calculated according to the RMSE of the enveloping area and is calculated as follows:

$$Error_{\%} = \frac{A_1 - A_2}{A_1} 100\%$$

where $RMSE$ is the error A_1 is the area resulting from the manual label.A_2 is the area calculated with the result of the YKMS segmentation.

The mean square error is calculated as follows:

$$RMSE = \sum_{1}^{n} \frac{Error_{\%}}{n}$$

where $RMSE$ is the mean error absolute and n is the number of images involved in the process.

III. RESULTS AND DISCUSSION

A. TRAINING AND PREDICTION RESULTS

This subsection initially describes common aspects for all implemented methods and then presents the results of training and area identification.

The database images used for all implemented methods were distributed into test, train, and validation sets in proportions of approximately 11 %, 70 %, and 19 %, respectively.

In the case of the YKMS method, the ROI results were first obtained by applying the YOLOv10 method to identify bounding boxes. The training was conducted over 200 epochs with a batch size of 16, and it was completed in approximately 410.22 s, using 589.5 MB of RAM and 4178.5 MB of GPU memory.

In Figures 5a and 5b, the *box_loss* curves show how well the model predicted the position and dimensions of the bounding boxes. Likewise, the *cls_loss* illustrated in Figures 5c and 5d evaluated how accurately the model classified objects within those boxes.

In Fig. 6, the F1-Confidence curve illustrates the appropriate value for the confidence parameter. The confidence value derived from this curve is 0.9, as it provides the best balance between recall and precision. The recall, precision, and mAP values with a confidence value of 0.9 were 100 %, 99.6 %, and 99.5 %, respectively, as shown in Fig. 7.

In Fig. 8a, the labels created in Roboflow are shown, while in Fig. 8b, the prediction is displayed. It can be noted that the YOLOv10 object detector was able to perfectly identify the lettuces and correctly plot the bounding boxes. Once the bounding boxes of the lettuce were successfully identified, a workspace channel change was performed, transitioning from RGB to CIELUV. The L channel corresponding to illumination was then used, resulting in Fig. 9a, which shows the ROI.

After obtaining the ROI, the k-means method was applied, whose result is seen in Fig. 9b, from which the class with the most predominant area, corresponding to the green colour, was extracted. In Fig. 9c, the result of applying an area contour detector to the predominant area extracted from the k-means method is presented. In parallel, the superpixel method was applied to the original image, whose result is shown in Fig. 17d. Both methods were intersected to determine which areas identified by the superpixel method corresponded to the lettuce. The result of the retained areas is seen in Fig. 9e. Finally, a contour detector was applied, and the result is shown in Fig. 17f.

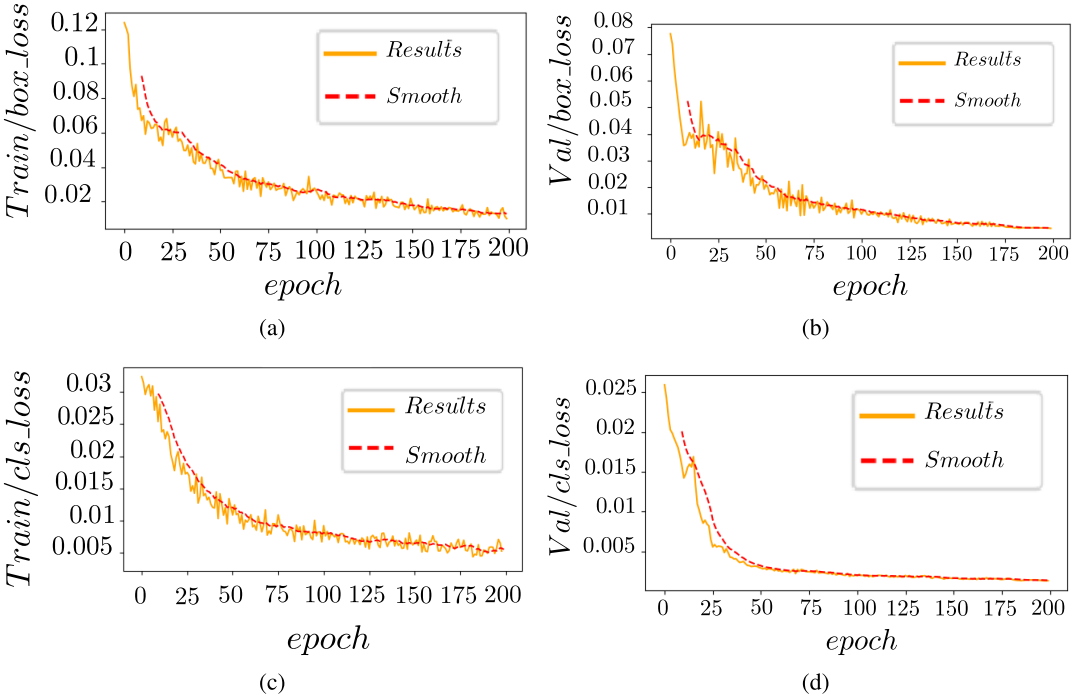


FIGURE 5. Training data YOLOv10: (a) Training the location and size of the boxes, (b) Predicting the location and size of the boxes, (c) Training for classifying objects within the bounding boxes, (d) Prediction of classifying objects within the bounding boxes.

TABLE 1. Performance parameters.

Parameters	YOLOv8	YKMS	YKMSV8	Detectron2
w_a	35 %	35 %	35 %	35 %
w_t	65 %	65 %	65 %	65 %
α_a	63.3	10.8	14.62	48.35
β_a	0.055	0.09	0.099	0.0099
α_t	3.85	14.7	8.37	8.13
β_t	0.059	1.37	4.52	3.11

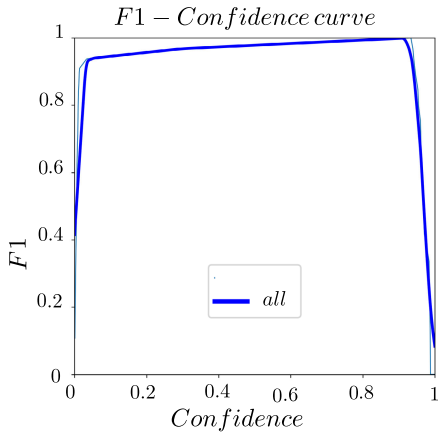


FIGURE 6. F1 vs confidence curve.

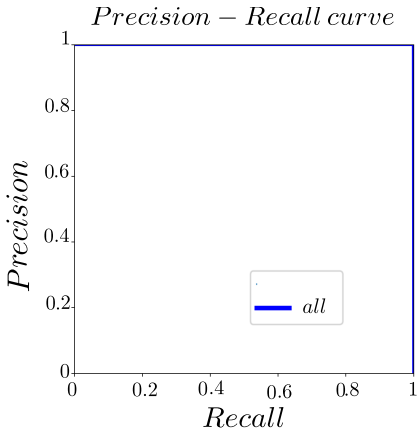
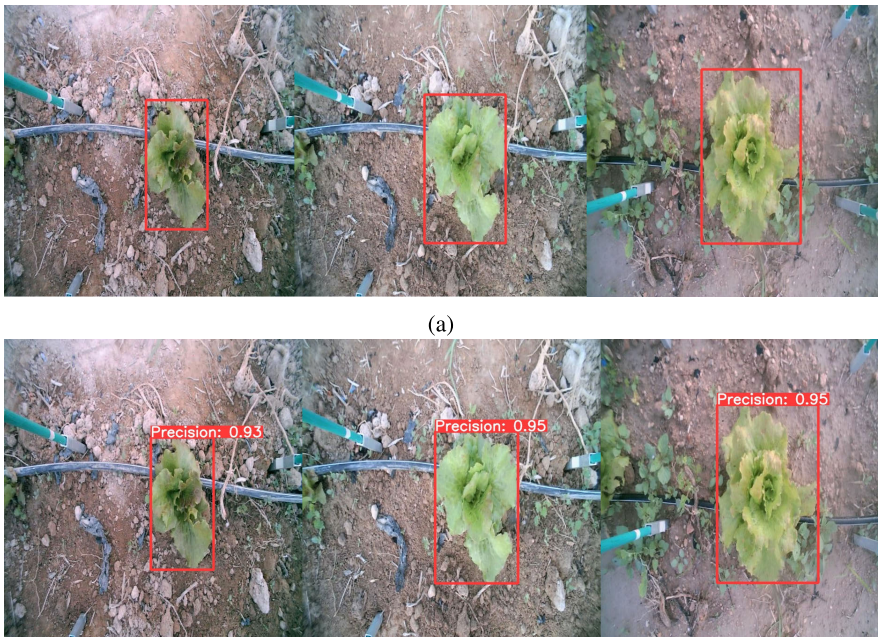


FIGURE 7. Precision vs recall curve.

In Fig. 10, the results of applying methods 3: YOLOv8 and method 4: Detectron2 are shown, with the annotation labels displayed in Fig. 10a. Meanwhile, Figs. 10b and 10c show the results of each method, respectively. The results of training for each of these cases will be described below.

By applying the YOLOv8 method, 500 epochs were used. The training time was 1308.18 s, using 114.79 MB of RAM and 114.79 MB of GPU memory. In Figs. 11a and 11b, the *box_loss* curves depict how well the model predicted the position and dimensions of the bounding boxes. The *dfl_loss* shown in Figs. 11c and 11d is related to the improvement in the accuracy of anchor points in the prediction of bounding boxes. The loss of segmentation during training *seg_loss* is shown in Figs. 11f and 11e.

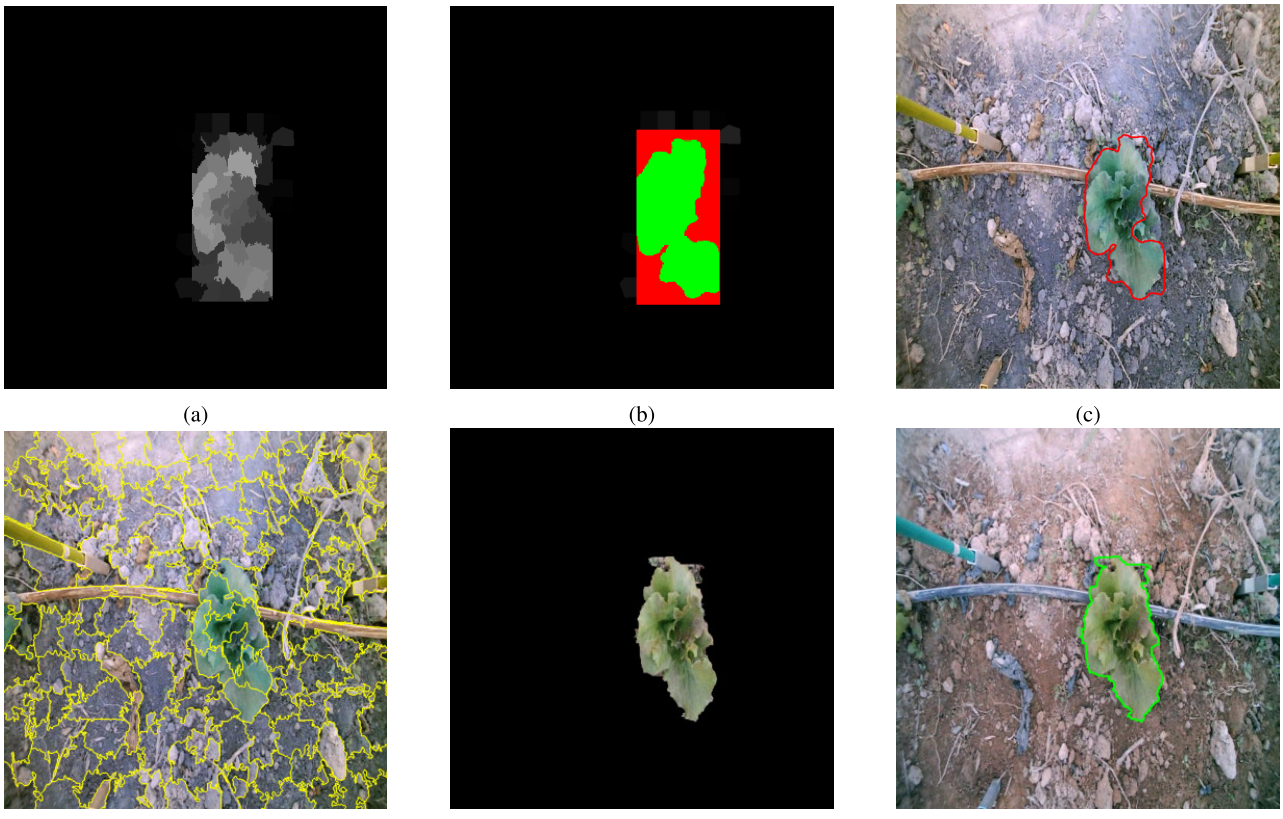
In Fig. 12, the F1-confidence curve is observed, from which the confidence value of 0.882 is obtained. In Fig. 13,



(a)

(b)

FIGURE 8. ROI: (a) Lettuce label in Roboflow, (b) Lettuce predicted from YOLOv10 results.



(a) (b) (c)

(d) (e) (f)

FIGURE 9. Method 2 results: k-means-superpixel (a) ROI: Channel L^* from CIELUV, (b) K-means results, (c) Class edge, (d) Superpixel areas, (e) Superpixel areas and class edge intersection, (f) Final edges.

the Precision vs Recall curve obtained using the confidence value of 0.882 is observed. The recall, precision, mAP, and *mask_precision* values with a confidence value of 0.882 were 100 %, 93.1 %, 99.5 %, and 99.7 %, respectively.

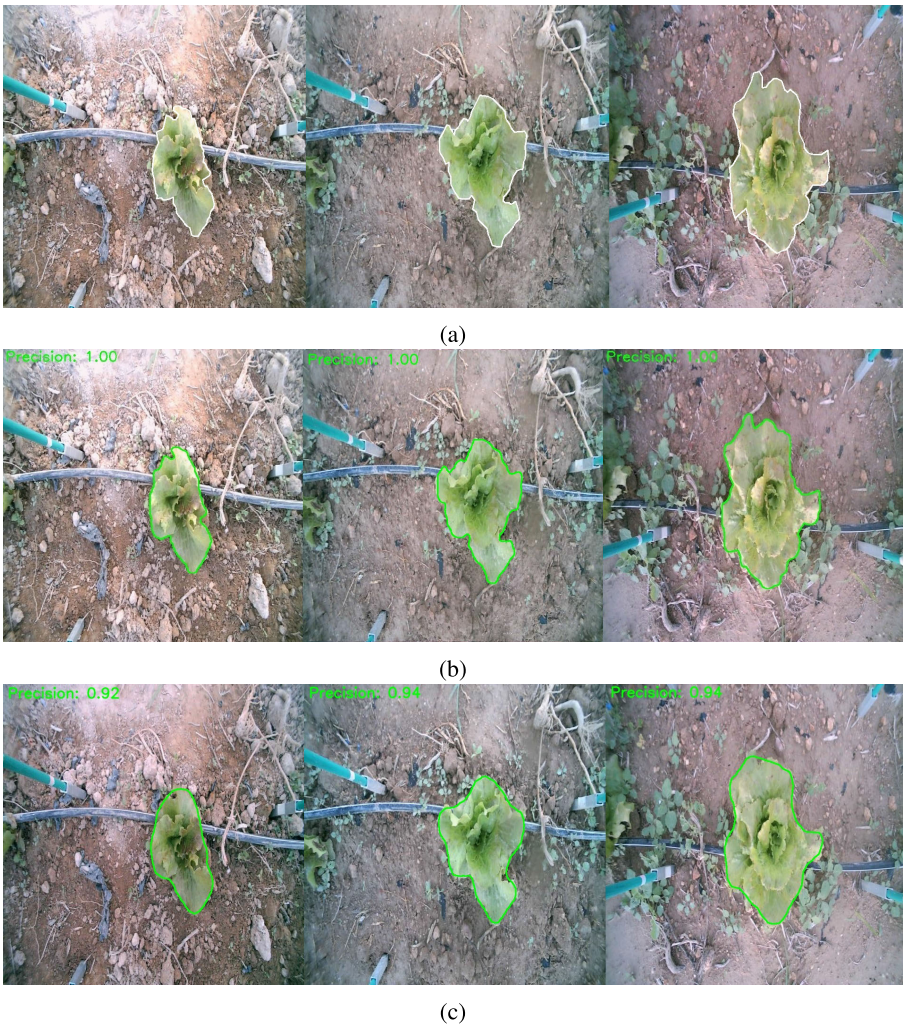


FIGURE 10. **(a) Lettuce label in Roboflow, (b) Lettuce predicted from YOLOv8 segmentation results, (c) Lettuce predicted from Detectron2 results.**

Detectron2 was employed for 1000 epochs, and the training time was approximately 364.5860 s, using 1770 MB of RAM, with a learning rate of 0.0001.

In Fig. 14a, the general behaviour of the model, represented by the *Total_loss* curve, is shown. Figs. 14b and 14c display the curves associated with the proposed regions for objects, namely *RPN_classification_loss* and *RPN_localization_loss*. In Fig. 14d, the *Classification_loss* curve shows how the model classifies the classes, with only one class considered in this case. In Fig. 14e, *Box_regression_loss* relates to the improvement of the accuracy of anchor points in the prediction of bounding boxes. The loss of mask accuracy during training, represented as *mask_loss*, is shown in Fig. 14f.

By applying the YOLOv8 method with YKMS labels, 500 epochs were used, with a training time of 1190 s, utilising 3252.48 MB of RAM and 128 MB of GPU memory. The Precision vs Recall curve, obtained using a confidence value of 0.995, is shown. The recall, precision, mAP, and *mask_precision* values at a confidence value of 0.898 were 100 %, 99 %, 99.5 %, and 99 %, respectively.

In Fig. 15, a graph shows the inference time for each method. A notable observation is that the proposed YKMS method requires more time to be inferred, followed by Detectron2, YOLOv8, and finally, YKMSY8, which has the shortest inference time. This method combines the strengths of two approaches: the ability to infer quickly from the YOLOv8 method and the robustness in labelling provided by the YKMS method.

In Table 1, the parameters used in the performance metric ε are shown. A value of w_a of 35 % and w_t of 65 % were considered, prioritising execution time to reduce battery consumption, which is crucial considering that the algorithm is intended for implementation in an autonomous field device.

Table 2 shows the percentage root mean squared error (RMSE), the performance metric ε, and the average time spent per model. The ε results obtained were 70.26 % for YKMS, 81.9 % for YOLOv8, 84 % for Detectron2,

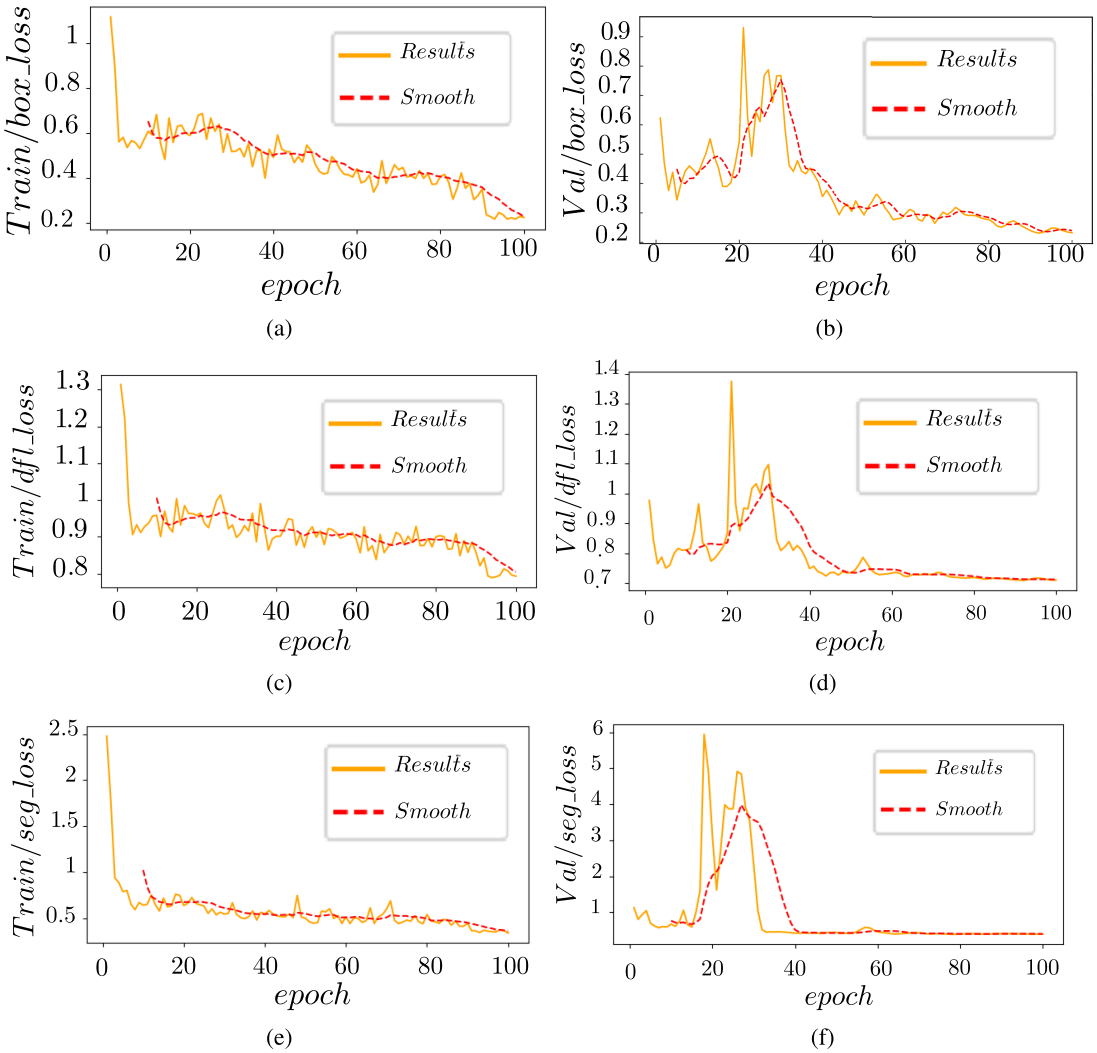


FIGURE 11. Training data YOLOv8: (a) Training the location and size of the boxes, (b) Predicting the location and size of the boxes, (c) Training for classifying objects within the bounding boxes, (d) Prediction of classifying objects within the bounding boxes.

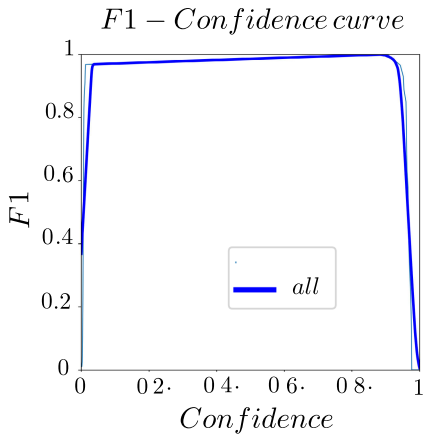


FIGURE 12. F1 vs confidence curve YOLOv8.

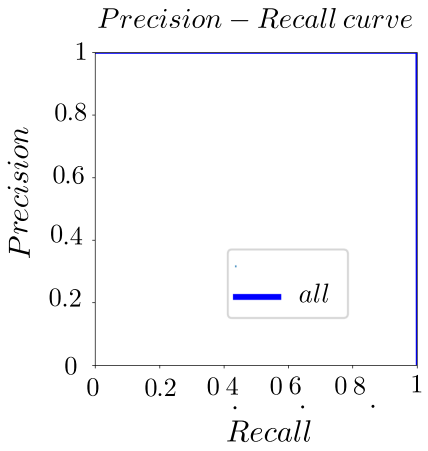


FIGURE 13. Precision vs recall curve YOLOv8.

and 87.3 % for YKMSV8. The RMSE values are as follows: the YKMS method achieved 5.2 %, YOLOv8 achieved 3.3 %, Detectron2 achieved 3.9 %, and YKMSV8 achieved 5.2 %. The average time spent per model was as follows: the

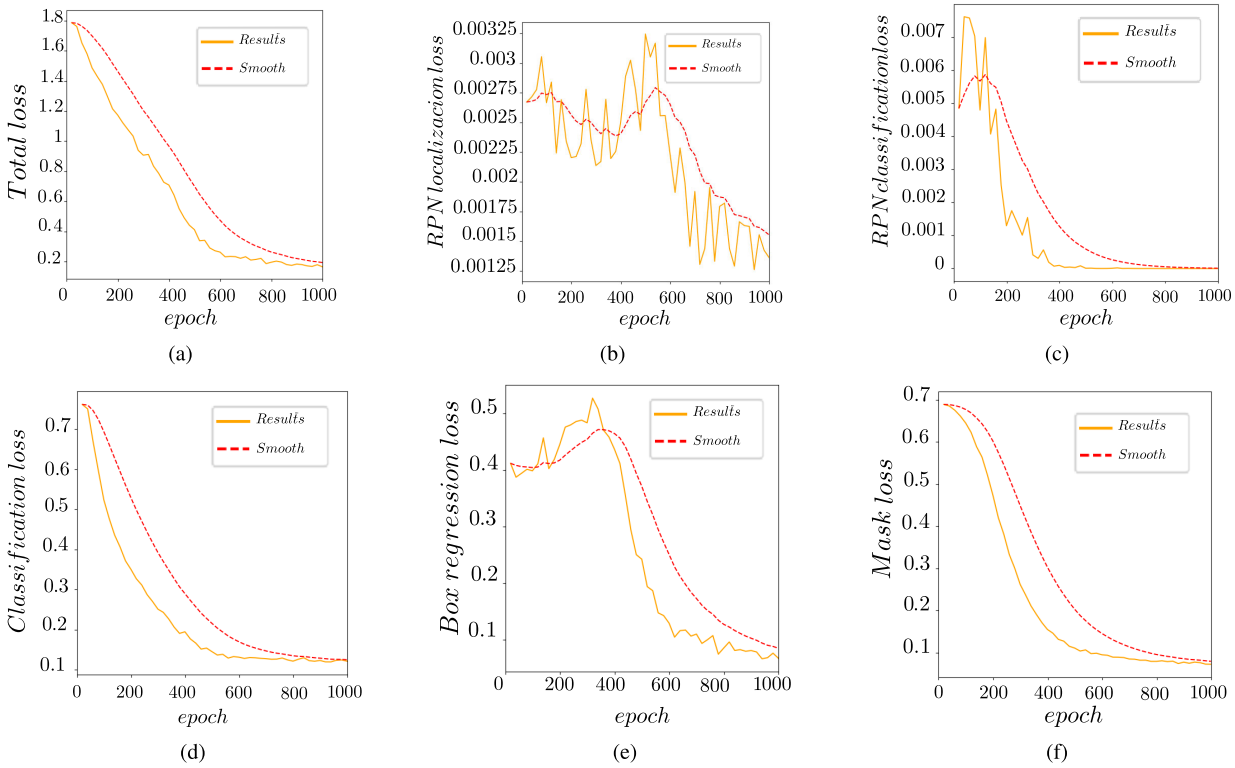


FIGURE 14. Training data Detectron2: (a) Model learning, (b) Predicting the object location, (c) Training model identifies regions likely to contain a specific class, (d) Measures the model accuracy in classifying objects, (e) Error in predicting the coordinates of the bounding boxes around objects, (f) Predicting the mask.

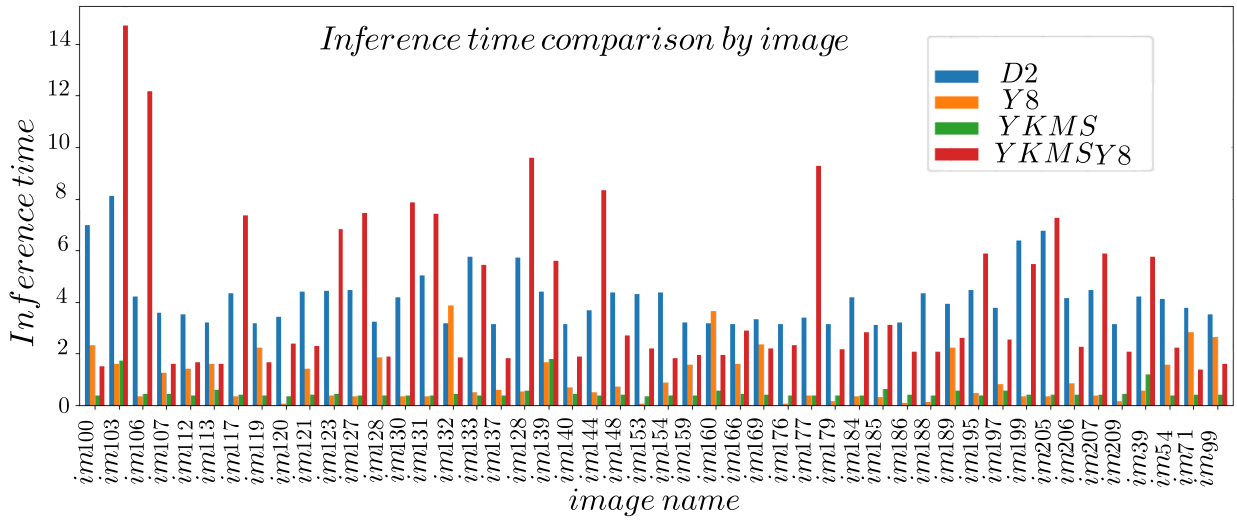


FIGURE 15. Inference time of each method.

TABLE 2. CNN evaluation models.

Model	ε	RMSE (%)	Mean Time (s)
YKMS	70.26 %	5,2	4.07
YOLOv8	81.9 %	3,3	1,01
Detectron2	84 %	3.9	4,12
YKMSY8	87.3 %	5,2	0,46

YKMS method took 4.07 s, YOLOv8 took 1.01 s, Detectron2 took 4.12 s, and YKMSV8 took 0.46 s.

In Table 3, the values of ε_{error} and ε_{time} for each method are shown. In terms of ε_{error}, the YOLOv8, Detectron2, and YKMSY8 methods have similar values, while in terms of time, the best performance is achieved by the YKMSY8 method. Although the values of the other methods are not far off, the aim is to find the method with the best performance, with a focus on time. This is because the analysis of the algorithms is conducted with the goal of making them feasible for use in a battery-operated device that needs to

TABLE 3. Comparison of ε_{error} and ε_{time} for each method.

Parameters	YOLOv8	YKMS	Detectron2	YKMSY8
ε_{error}	33.2 %	18.46 %	32.14 %	27.96 %
ε_{time}	48.7 %	51.8 %	51.9 %	60.1 %

(a) (b) (c)

FIGURE 16. YOLOv8 combined with YKMS: (a) Manual label, (b) Predicted label with YKMS, (c) Predicted label with YKMSY8.

be autonomous for as long as possible, designed for use in isolated locations such as farms.

B. DISCUSSION

In the case of lettuce, achieving a very low area error is not deemed necessary, as the area is usually estimated through visual inspections when performing non-destructive measurements. Time remains a significant factor, as these methods are analysed for use in a battery-operated device.

In Fig. 16a, a manually created label is shown. In Fig. 16b, a prediction from the YKMS method is displayed. In Fig. 16c, the prediction from the YKMSV8 method is presented. Upon comparing the three images, there is no significant difference in the area enclosed. However, the YKMSV8 method exhibits a more refined result than the YKMS method, adapting better to the contour of the lettuce, even more so than the manual label. This is because YOLOv8 learns certain common characteristics of the pixels enclosed, making the prediction sometimes more accurate than the manual label, which is subject to human error due to its manual creation.

An interesting aspect, depending on the application, is that the YKMS method proposes a reduction in the time required for labelling, as it identifies the lettuce area using bounding boxes, thereby saving considerable time during labelling. In contrast, the YOLOv8 and Detectron2 methods require labels with multiple coordinates corresponding to the object's contour. While the time required for labelling is somewhat subjective, the accuracy of the labels is crucial for training. This subjectivity is eliminated by using the YKMS method, which only labels the bounding boxes. Therefore, the combination of these methods leverages their strengths. The YKMSV8 method takes advantage of the versatility in labelling from YKMS and the speed in inference from YOLOv8.

Since in the YKMS method, the contours identified by the Python `findContours` command are stored as prediction contour coordinates, a high number of points belonging to the contour is obtained. When compared with manual annotations, it is observed that the YKMS method generates a higher number of coordinates.

Therefore, by using YKMS predictions as labels, the YKMSV8 method has more annotations of points belonging to the contour than the manual annotations. This results in the YKMSV8 method having more input annotations for training, implying that during training, the model generalises better and becomes more efficient during inference. For this reason, the YKMSV8 method performs better than the YKMS method, even though the mean squared error (MSE) remains the same when compared to the latter method. However, better performance implies less dispersion, demonstrating greater stability in the method.

The YOLOv8 method, during inference, internally proposes multiple prediction options, among which it selects the one with the highest accuracy. However, being a more efficient model due to the number of coordinates in its annotations, it exhibits greater proximity between these proposed predictions, facilitating the calculation of **CIoU** to identify redundant predictions. This proximity allows for easier identification and suppression of significantly overlapping predictions using the **NMS** parameter, optimising the time needed to obtain the final predictions.

The importance of having well-detailed information in the annotations for training the neural network plays an essential role. However, in some crops, achieving a perfect contour during labelling is even more challenging, either due to their irregular shape or the lack of agility of the person labelling, introducing subjectivity. An example is chard, which, due to its many leaves, shows irregularity in its shape. Labelling it would be an easier task if the YKMS method were applied as a labeller. In Fig. 17, the preliminary result process for obtaining labels with the YKMS method is shown, which could be used by another method such as YOLOv8.

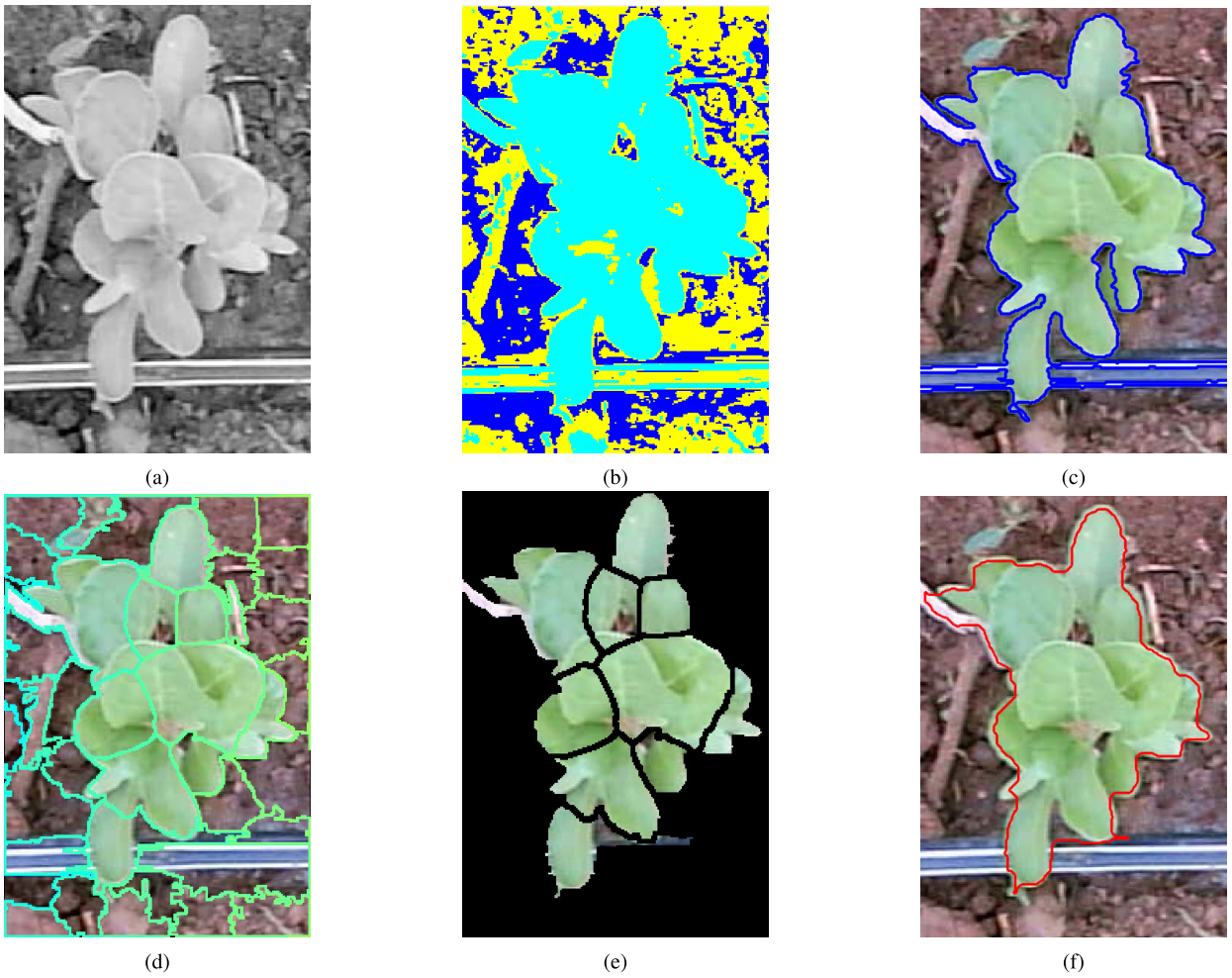


FIGURE 17. YKMS results: k-means-superpixel (a) ROI: Channel L^* from CIELUV, (b) K-means results, (c) Class edge, (d) Superpixel areas, (e) Superpixel areas and class edge intersection, (f) Final edges.

IV. CONCLUSION AND FUTURE WORK

Methods to standardise leaf area measurement have been introduced, mitigating the subjectivity present in visual assessments and making them applicable to low computational cost devices. These methods aim to automate activities, thereby reducing the time farmers spend on periodic inspections. One key method, YKMS, employs bounding box labels around objects and predicts contours.

Different contour identification methods were trained in Google Colab, using the following resources during inference: YOLOv10: 589.5 MB of RAM, 4178.5 MB of GPU memory, and 410.22 seconds; YOLOv8: 3330.71 MB of RAM, 114.79 MB of GPU memory, and 1210.21 seconds; Detectron2: 1770 MB of RAM, 2510.8 MB of GPU memory, and 364.586 seconds; YKMSY8: 3252.48 MB of RAM, 128 MB of GPU memory, and 1190 seconds.

A performance metric was proposed to evaluate the balance between precision and computational cost. This metric assesses stability by analysing the variation of values within their extremes, with weighting factors of 65% for computational cost and 35% for accuracy. It evaluates functionality and stability, ultimately determining the real inference time, which directly impacts battery consumption. For this application, the results were: YOLOv8: 81.9 %, Detectron2: 84 %, YKMS: 70.26 %, and YKMSV8: 87.3 %.

Future work will focus on expanding the database to include other crops and utilising the YKMS method as a labelling assistant to generate object contours from bounding boxes, particularly useful in cases with irregular contours.

Additionally, it is proposed to modify YKMS to label multiple classes instead of a single one and to combine it with other techniques to enhance its precision and evaluate its performance.

ACKNOWLEDGMENT

During the preparation of this work the authors used ChatGPT in order to improve readability and language. After using this tool, they reviewed and edited the content as needed and take full responsibility for the content of the publication.

REFERENCES

[1] L. Wang, S. Ning, W. Zheng, J. Guo, Y. Li, Y. Li, X. Chen, A. Ben-Gal, and X. Wei, "Performance analysis of two typical greenhouse lettuce production systems: Commercial hydroponic production and traditional soil cultivation," *Frontiers Plant Sci.*, vol. 14, Jul. 2023, Art. no. 1165856.

[2] M. A. Sharaf-Eldin, M. B. Elsawy, M. Y. Eisa, H. El-Ramady, M. Usman, and M. Zia-ur Rehman, "Application of nano-nitrogen fertilizers to enhance nitrogen efficiency for lettuce growth under different irrigation regimes," *Pakistan J. Agricult. Sci.*, vol. 59, no. 3, 2022.
[3] Z. Razmjooei, M. Etemadi, S. Eshghi, A. Ramezanian, F. M. Abarghuei, and J. Alizargar, "Potential role of foliar application of azotobacter on growth, nutritional value and quality of lettuce under different nitrogen levels," *Plants*, vol. 11, no. 3, p. 406, Feb. 2022.
[4] J. J. Murray, G. Basset, and G. Sandoya, "Nutritional benefits of lettuce consumed at recommended portion sizes," *EDIS*, vol. 2021, no. 3, Jun. 2021.
[5] G. G. Parker, "Tamm review: Leaf area index (LAI) is both a determinant and a consequence of important processes in vegetation canopies," *Forest Ecol. Manage.*, vol. 477, Dec. 2020, Art. no. 118496.
[6] N. Chamara, G. Bai, and Y. Ge, "AICropCAM: Deploying classification, segmentation, detection, and counting deep-learning models for crop monitoring on the edge," *Comput. Electron. Agricult.*, vol. 215, Dec. 2023, Art. no. 108420.
[7] O. S. Castillo, E. M. Zaragoza, C. J. Alvarado, M. G. Barrera, and N. Dasgupta-Schubert, "Foliar area measurement by a new technique that utilizes the conservative nature of fresh leaf surface density," 2012, *arXiv:1212.5761*.
[8] R. Nandan, V. Bandaru, J. He, C. Daughtry, P. Gowda, and A. E. Suyker, "Evaluating optical remote sensing methods for estimating leaf area index for corn and soybean," *Remote Sens.*, vol. 14, no. 21, p. 5301, Oct. 2022.
[9] M. P. Rico-Fernández, R. Rios-Cabrera, M. Castelán, H.-I. Guerrero-Reyes, and A. Juarez-Maldonado, "A contextualized approach for segmentation of foliage in different crop species," *Comput. Electron. Agricult.*, vol. 156, pp. 378–386, Jan. 2019.
[10] S. G. Sodjinou, V. Mohammadi, A. T. S. Mahama, and P. Gouton, "A deep semantic segmentation-based algorithm to segment crops and weeds in agronomic color images," *Inf. Process. Agricult.*, vol. 9, no. 3, pp. 355–364, Sep. 2022.
[11] H. Yu, H. Jiang, Z. Liu, S. Zhou, and X. Yin, "EDTRS: A superpixel generation method for SAR images segmentation based on edge detection and texture region selection," *Remote Sens.*, vol. 14, no. 21, p. 5589, Nov. 2022.
[12] B. V. Kumar, "An extensive survey on superpixel segmentation: A research perspective," *Arch. Comput. Methods Eng.*, vol. 30, no. 6, pp. 3749–3767, Jul. 2023.
[13] Z. U. Rehman, M. A. Khan, F. Ahmed, R. Damaševičius, S. R. Naqvi, W. Nisar, and K. Javed, "Recognizing apple leaf diseases using a novel parallel real-time processing framework based on MASK RCNN and transfer learning: An application for smart agriculture," *IET Image Process.*, vol. 15, no. 10, pp. 2157–2168, Aug. 2021.
[14] K. He, G. Gkioxari, P. Dollár, and R. Girshick, "Mask R-CNN," in *Proc. IEEE Int. Conf. Comput. Vis. (ICCV)*, Oct. 2017, pp. 2980–2988.
[15] M. Hussain, "YOLO-v1 to YOLO-v8, the rise of YOLO and its complementary nature toward digital manufacturing and industrial defect detection," *Machines*, vol. 11, no. 7, p. 677, Jun. 2023.
[16] M. Wadhwa, T. Choudhury, G. Raj, and J. C. Patni, "Comparison of YOLOv8 and Detectron2 on crowd counting techniques," in *Proc. 7th Int. Symp. Innov. Approaches Smart Technol. (ISAS)*, Nov. 2023, pp. 1–6.
[17] F. Chincholi and H. Koestler, "Detectron2 for lesion detection in diabetic retinopathy," *Algorithms*, vol. 16, no. 3, p. 147, Mar. 2023.
[18] N. Wang, H. Liu, Y. Li, W. Zhou, and M. Ding, "Segmentation and phenotype calculation of rapeseed pods based on YOLO v8 and mask R-convolution neural networks," *Plants*, vol. 12, no. 18, p. 3328, Sep. 2023.
[19] Z. Bartlett, L. Han, T. T. Nguyen, and P. Johnson, "A novel online dynamic temporal context neural network framework for the prediction of road traffic flow," *IEEE Access*, vol. 7, pp. 153533–153541, 2019.
[20] Z. Bartlett, L. Han, T. T. Nguyen, and P. Johnson, "Prediction of road traffic flow based on deep recurrent neural networks," *IEEE Access*, vol. 7, pp. 153533–153541, 2019.
[21] M. A. R. Alif and M. Hussain, "YOLOv1 to YOLOv10: A comprehensive review of YOLO variants and their application in the agricultural domain," 2024, *arXiv:2406.10139*.
[22] X. Yue, K. Qi, X. Na, Y. Zhang, Y. Liu, and C. Liu, "Improved YOLOv8-seg network for instance segmentation of healthy and diseased tomato plants in the growth stage," *Agriculture*, vol. 13, no. 8, p. 1643, Aug. 2023.
[23] R. Sapkota, Z. Meng, M. Churuvija, X. Du, Z. Ma, and M. Karkee, "Comprehensive performance evaluation of YOLO11, YOLOv10, YOLOv9 and YOLOv8 on detecting and counting fruitlet in complex orchard environments," 2024, *arXiv:2407.12040*.
[24] M. Hussain, "YOLOv5, YOLOv8 and YOLOv10: The go-to detectors for real-time vision," 2024, *arXiv:2407.02988*.
[25] F. Martinez, J. B. Romaine, J. M. Manzano, C. Ierardi, and P. M. Gata, "Deployment and verification of custom autonomous low-budget IoT devices for image feature extraction in wheat," *IEEE Access*, vol. 12, pp. 124636–124657, 2024.
[26] J. Terven, D.-M. Córdova-Esparza, and J.-A. Romero-González, "A comprehensive review of YOLO architectures in computer vision: From YOLOv1 to YOLOv8 and YOLO-NAS," *Mach. Learn. Knowl. Extraction*, vol. 5, no. 4, pp. 1680–1716, Nov. 2023.
[27] Ultralytics. (2024). *YOLOv5 Model Structure—Ultralytics Documentation*. Accedido el 23 de enero de 2024. [Online]. Available: https://docs.ultralytics.com
[28] P. Jiang, D. Ergu, F. Liu, Y. Cai, and B. Ma, "A review of YOLO algorithm developments," *Proc. Comput. Sci.*, vol. 199, pp. 1066–1073, Jan. 2022.
[29] M. J. Jhatial, D. R. A. Shaikh, N. A. Shaikh, S. Rajper, R. H. Arain, G. H. Chandio, A. Q. Bhangwar, H. Shaikh, and K. H. Shaikh, "Deep learning-based Rice leaf diseases detection using YOLOv5," *Sukkur IBA J. Comput. Math. Sci.*, vol. 6, no. 1, pp. 49–61, Jul. 2022.
[30] A. S. Geetha and M. Hussain, "A comparative analysis of YOLOv5, YOLOv8, and YOLOv10 in kitchen safety," 2024, *arXiv:2407.20872*.
[31] J. N. Opara, R. Moriwaki, and P.-J. Chun, "Delineating landslide and debris flow detection in Japan through aerial photography: A YOLO v8 approach to disaster management," *Intell., Inform. Infrastruct.*, vol. 5, no. 1, pp. 111–123, 2024.
[32] H. Lou, X. Duan, J. Guo, H. Liu, J. Gu, L. Bi, and H. Chen, "DC-YOLOv8: Small-size object detection algorithm based on camera sensor," *Electronics*, vol. 12, no. 10, p. 2323, May 2023.
[33] A. B. Abdallah, A. Kallel, M. Dammak, and A. B. Ali, "Olive tree and shadow instance segmentation based on Detectron2," in *Proc. 6th Int. Conf. Adv. Technol. Signal Image Process. (ATSIP)*, May 2022, pp. 1–5.
[34] A. B. Abdusalomov, B. M. S. Islam, R. Nasimov, M. Mukhiddinov, and T. K. Whangbo, "An improved forest fire detection method based on the Detectron2 model and a deep learning approach," *Sensors*, vol. 23, no. 3, p. 1512, Jan. 2023.
[35] R. Valdarnini, "A multifiltering study of turbulence in a large sample of simulated galaxy clusters," *Astrophys. J.*, vol. 874, no. 1, p. 42, Mar. 2019.

F. MARTÍNEZ was born in Paraguay, in 1993. She received the Electrical Engineering degree from the National University of Asunción, in 2018, and the M.Sc. degree in electronic engineering with a focus on renewable energy and energy efficiency from the University of the Southern Cone of the Americas, in 2020. This was achieved through the Prociencia I Program, which is financed by CONACYT. She gained research experience in energy quality, focusing on active power filters and model predictive control. She completed a research stay at Liverpool John Moores University, U.K. Her current work is centered on computer vision, utilizing the IoT-based devices for capturing images to monitor crop health through machine learning to detect key parameters. Her aim is to propel significant advancements in the intersection of technology and agriculture. She taught at the National University of Asunción, from 2019 to 2020, as a Teaching Assistant, and has been a Professor with the University of the Southern Cone of the Americas, since 2020. In 2021, she began working as a Research Assistant with the Universidad Loyola Andalucía (ULA).

JAMES B. ROMAINE received the B.Eng. degree in electronic engineering from the University of York, and the Ph.D. degree in physical sciences from the University of Seville. He is an Associate Professor at Loyola University, where he has taught over 120 credits, consistently earning a student satisfaction rating above nine. His courses, including digital electronics, microcontrollers, and digital electronic systems, are offered in both Spanish and English to a diverse student body. In addition to his teaching, James coordinates the Action Tutorial Plan (PAT) and leads the Erasmus program for outgoing students. His research spans EEG signal processing, epileptic seizure prediction, and neurotechnology, with a focus on developing real-time brain activity analysis systems. He is also involved in cutting-edge projects related to the Internet of Things (IoT) and sustainable agriculture, such as developing IoT-based platforms for real-time monitoring and process optimization. He actively participates in European research projects, including Horizon Europe's DoCBox and MSCA-RISE-H2020, and his work has been published in top journals like IEEE Transactions on Biomedical Circuits and Systems and IEEE Access.

P. JOHNSON (Senior Member, IEEE) is currently a Senior Lecturer with the Department of Electronics and Electrical Engineering, Liverpool John Moores University. Her research interests include wireless sensor networks and intelligent algorithms for health-related applications. She is also developing international collaborative research on leadership skills.

A. CARDONA RUIZ was born in Seville, Spain, in April 1993. He received the degree in electronics, robotics, and mechatronics engineering, specializing in robotics and automation from the University of Seville, in 2017. From 2008 to 2009, he was a Research Assistant with the Institute of Physics, Academia Sinica, Tapei, Taiwan. His professional career includes experience as a Programmer in the RETAIL Project, from 2016 to 2017, where he developed software for humanoid robots, such as NAO and worked with systems, such as Turtlebot and ROS. Later, he was a Researcher with the Estimation, Prediction, Optimization, and Control Group (GEPOC), University of Seville, from 2018 to 2019, contributing to the development of models and controllers for building climate control systems. Since 2021, he has been with Universidad Loyola Andalucía, where he participates in the development of smart agriculture projects, such as CELEGAND, INNORIEGA, and DENORI, aiming for efficient water use. His research interests include the development of surface processing and biological/medical treatment techniques using nonthermal atmospheric pressure plasmas, fundamental study of plasma sources, and fabrication of micro- or nano-structured surfaces.

PABLO MILLÁN GATA received the M.Sc. degree in environmental engineering and the M.Sc. and Ph.D. degrees in control engineering from the Universidad de Sevilla, USA, and the Dr.-Eng. degree. He was a Visiting Researcher with the Department of Electrical Engineering, KTH, Sweden, in 2011, and the University of Newcastle University, Australia, in 2012. In 2013, he joined Universidad Loyola Andalucía (ULA). Since 2014, he has been maintaining a regular research collaboration with the Laboratoire d'Analyse et d'Architecture des Systemes (LAAS), CNRS, France. In 2015, he was a Visiting Researcher with the Department of Electrical Engineering, Marquette University, and Loyola Chicago University, USA. In 2022, he was a Visiting Professor with the Universitat di Bergamo, Italy. He is currently an Associate Professor and the Director of the School of Engineering, Universidad Loyola Andalucía. He has authored or co-authored more than 60 technical papers published in ISI technical journals, books, and international conferences. He has worked on 17 national and international research projects, being a PI in seven of them. His current research interests include distributed estimation and control for smart agriculture, autonomous surface vehicles, and cyber-physical systems. His contributions to the Ph.D. dissertation were recognized with the best thesis awards of the University of Sevilla and the Best Thesis Award of Sevilla City Council.