Synthetic Dataset Generation for Tomato Ripening Stage Detection in Different Scenes

Gerardo Antonio Alvarez-Hernandez, Juan Irving-Vasquez, Abril Valeria Uriarte-Arcia, and Luis Alberto Tovar-Ortiz

***Abstract*—The development of intelligent robotic systems for agriculture depends on large and representative datasets, which are essential for training computer vision models. However, the availability of public datasets in this area is limited, hindering the implementation and improvement of these technologies. To address this problem, we propose a methodology for synthetic dataset generation. This methodology includes the automated creation of datasets optimized through evolutionary algorithms, thereby improving the quality and diversity of the generated data.To validate the method, we tested it in a case study: the detection of tomato ripening stages in greenhouses. The experiments showed that training a detector (YOLOv5m model) with this synthetic data significantly improves its performance in real scenarios, increasing detection from null to acceptable performance. These results validate the effectiveness of synthetic data generation as a viable and affordable alternative to compensate for the shortage of agricultural datasets.**

Link to graphical and video abstracts, and to code:
https://latamt.ieeer9.org/index.php/transactions/article/view/10390

***Index Terms*—Synthetic data, YOLO, Tomato ripening stages, Genetic algorithm, Optimization**

Fig. 1. Testing scene. The robot detects the ripening stage of the tomatoes in a real scenario. Even though the available dataset was captured in laboratory, our proposed method generates datasets that can be used for training a model that can detect tomatoes in the real scene.

I. Introduction

CURRENT agricultural production faces increasing pressure to optimize yields, which has driven the strategic adoption of automated solutions. In this scenario, automated object recognition is a fundamental capability for mobile robots, allowing them to interact accurately with their environment. Specifically, detecting the stages of tomato ripeness is a critical task, as it directly impacts harvest efficiency and the quality of the final product [1], [2].

However, the development of these perception systems is severely limited by the difficulty of collecting and annotating large-scale datasets within the agricultural domain [3]. This scarcity of labeled data generates a domain mismatch, where models trained in specific environments fail to generalize their performance in the real operational context [4]. Although approaches based on generative artificial intelligence and data standardization have been explored to mitigate this shortcoming [5], [6], these often lack the granular control and explainability required for specialized agricultural tasks. Furthermore, existing public resources, such as the LaboroTomato dataset [7], are insufficient for this purpose, as they do not align with the color ripeness criteria required by Mexican regulations (NMX-FF-031-1997) and present morphological variations that accentuate the domain mismatch. To address these limitations, this study proposes a reusable methodology for generating explainable synthetic datasets, which is our main contribution. Unlike conventional synthesis techniques or classic data augmentation, our approach is based on the reuse of specialized post-harvest dataset, created by experts, where object information is explicit and rigorous. The core of our proposal is the supervised synthetic transfer of these objects from their original context to a pre-harvest environment (greenhouse plant) through image processing and an optimization stage. This context transfer not only avoids the bias of purely generative models, but also allows the detection system to achieve acceptable functional performance under real operating conditions, increasing the mAP@50 (mean Average Precision with an IoU threshold of 0.50) from a null value of 0.00381 to 0.217. With this, we demonstrate that our methodology is a superior and scalable solution for tasks where the original data is unable to solve the detection problem.

The associate editor coordinating the review of this manuscript and approving it for publication was Martín Pedemonte (*Corresponding author: Gerardo Antonio Alvarez Hernandez*).

The authors thank SECIHTI. IPN-SIP science and innovation projects (grants 20250144 and 20250342).

Gerardo Antonio Alvarez Hernandez, J. I. Vásquez-Gómez, A. V. Uriarte-Arcia, and L. A. Tovar-Ortiz are with the Centro de Investigación en Computación, Instituto Politécnico Nacional, Mexico City, Mexico (e-mails: galvarezh1400@alumno.ipn.mx, jvasquezg@ipn.mx, auriartea@ipn.mx, and ltovaro2100@alumno.ipn.mx).

The main contributions of this work are summarized as follows:

- A reusable and explainable synthetic dataset generation methodology structured in two phases: parameter optimization and dataset generation.
- A supervised context-transfer mechanism that reuses expert-labeled post-harvest data (D_c) for pre-harvest greenhouse scenarios.
- An optimization framework based on a Genetic Algorithm to estimate the optimal transformation parameters θ^* using mAP@50 as the objective function.

The rest of this article is structured as follows: Section II reviews related work; Section III details the proposed model; Section IV describes the datasets and computational environment; Section V presents the experimental results and evaluation of the approach; finally, Section VI presents the conclusions and future research directions.

II. Related Work

Data augmentation has become a key strategy in object detection and classification in the agricultural domain, allowing to improve the accuracy and robustness of deep learning applications. Several approaches have been proposed in the literature to address data limitation through image generation and transformation.

Data augmentation is a widely adopted technique to enhance the performance of deep learning models by artificially increasing the diversity of training data. Various strategies have been explored in agricultural and plant-related applications, leveraging augmentation techniques to improve classification and detection tasks. The following studies can be grouped into three main categories based on their augmentation approach: (1) classical image transformations, (2) generative adversarial networks (GANs) and synthetic image generation, and (3) specialized augmentation techniques tailored to specific challenges.

A. Classical Image Transformations

Several studies have relied on traditional augmentation techniques such as rotation, brightness adjustments, and noise addition to improve model performance. Wu et al. [8] applied basic transformations, including doubling, brightness changes, rotations, and Gaussian noise, achieving an increase of 0.29 in the mAP metric for camellia detection. Similarly, Enkvetchakul et al. [9] used transformations like rotations, displacements, and zooming, improving disease classification accuracy by 1.97%. Nithya et al. [10] also leveraged common transformations, such as rotation, flipping, and cropping, to detect defective mangoes, reaching 98.5% accuracy with only 100 images.

B. Generative Adversarial Networks (GANs) and Synthetic Image Generation

Some researchers have explored GAN-based techniques to generate synthetic data and enhance classification tasks. Jordan et al. [11] employed conditional generative adversarial networks (cGANs) for healthy and unhealthy fruit classification, achieving a 31% performance improvement. Similarly, Divyantha et al. [12] used adversarial networks to generate synthetic images for plant germination classification, leading to a 1.37% increase in F1-score. Min et al. [13] implemented CycleGAN for plant leaf classification, achieving a remarkable improvement in F1-score from 0.9969 to 0.9998. Tan et al. [14] applied DCCGAN networks to augment melon spectral data for pesticide residue classification, resulting in a 13.13% accuracy boost.

C. Specialized Augmentation Techniques

Beyond traditional and GAN-based approaches, some studies have introduced specialized augmentation strategies tailored to specific problems. Momeny et al. [15] implemented a Bayesian optimization-based noise introduction followed by an encoder reconstruction process to generate more representative data. Dai et al. [16] combined classical transformations with luminosity modifications to simulate meteorological changes, improving diseased leaf classification by 3%. Rahman et al. [17] developed an augmentation method incorporating 11 transformations, including contrast, saturation, blur, and Gaussian noise, leading to a 1.6% increase in mAP@50 for weed detection in cotton. Li et al. [18] introduced an occlusion-based augmentation algorithm for apple detection, improving the AP metric by 0.109. Fu et al. [19] proposed a copy-paste strategy combined with the SAM model, mixup, and mosaic, improving the mAP metric in orange detection by 7.3%. Finally, Gao et al. [20] developed an offline-online augmentation scheme focusing on illumination and color variations, achieving a 1.1% increase in mAP.

Despite advances in augmentation techniques, the selection of the base dataset remains a determining factor for the success of deep learning models. In the current literature, significant resources such as the Laboro Tomato dataset [7] or various repositories [21]–[23] are reported; however, these are often insufficient when specific classification criteria are required. In this study, the choice of the original dataset was not arbitrary, but rather because it is the only one that strictly complies with the tomato type and ripeness grades by color stipulated in the Mexican Standard NMX-FF-031-1997. Furthermore, having been generated under the supervision of specialists in the field, this dataset guarantees technical accuracy in classification, avoiding the common biases found in general repositories.

However, the main challenge lies in the fact that this dataset was originally designed for a post-harvest environment. Our research proposes a methodology that allows this expert information to be reused and transferred in a synthetic way to a pre-harvest environment, i.e., placing the fruits directly on the plant within the context of a greenhouse.

III. Preliminaries

A. Dataset

Two datasets were used in this study. The first one was used as a basis for the creation of synthetic images to train the detector of the different ripening stages of tomatoes. This set contains a total of 3,000 images of tomatoes with a resolution

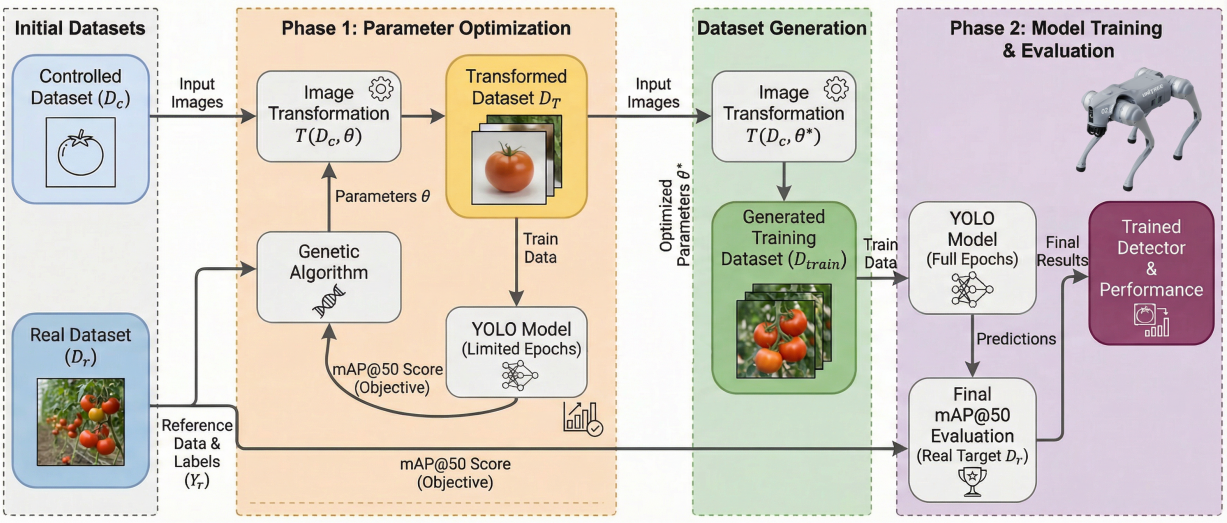
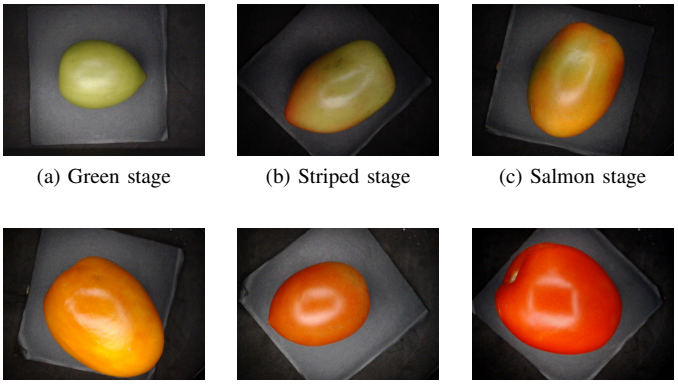


Fig. 2. General flow diagram of the proposed explainable method for generating datasets.



(a) Green stage (b) Striped stage (c) Salmon stage

(d) Orange stage (e) Orange-red stage (f) Red stage

Fig. 3. Example images from the dataset illustrating the six ripening stages according to NMX-FF-031-1997. (a) Green stage. (b) Breaker stage. (c) Salmon stage. (d) Orange stage. (e) Orange-red stage. (f) Red stage.

of 800×600 pixels in JPG format and in RGB channels. Six classification labels are handled: orange, striped, red, red-orange, salmon and green. This dataset was published in [24]. In that work, the dataset contains only the classes of each of the images (without bounding boxes). We manually labeled the bounding boxes enclosing the tomato found in each image, as well as the label indicating the ripening class of the tomato. This labeling has been done using the tool "labelImg" [25]. Representative examples of each of these labels can be seen in Fig. 3.

The second dataset was used in the validation and testing stage during the optimization and deep training of the network for the detection of ripening stages. This set is composed of images of tomatoes taken under greenhouse conditions at CIITA Veracruz. It contains 132 images with a resolution of 640×640 pixels in RGB channels. Some examples of these images can be seen in Fig. 4.

(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Fig. 4. Examples of tomato images from the dataset acquired under real greenhouse conditions. (a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

IV. Synthetic Dataset Generation

This study proposes an explainable methodology for generating datasets to train detection models. Since a dataset created from images generated under controlled conditions is used, a synthetic dataset generator optimized by genetic algorithms is proposed to create synthetic images that simulate tomatoes in the target environment.

The general diagram of the system is shown in Fig. 2. In the following subsection, the problem to be solved will be formally stated and the solution is presented. The generated data is available at https://www.kaggle.com/datasets/gerardoantony/dataset-for-tomatoes-to-use-for-augmentation.

A. Problem Formalization

***Definition** 1:* Let D be a dataset which consists of n ordered pairs:

$$D = \{(x_1, y_1), (x_2, y_2), ..., (x_n, y_n))\}, \quad (1)$$

where x_i is an image and y_i its label.

In our particular case, we have the following sets:

- D_c is a dataset containing examples of images of tomatoes in controlled conditions (see section III-A).
- D_r is the dataset containing images again in real conditions (see section III-A), where X_r are the images it

contains and Y_r are the labels that correspond to each one.

***Definition** 2:* Let D_i and D_j be two image datasets. Let $\phi(\cdot)$ denote a feature extractor that maps images into a d-dimensional embedding space. Let $\mathcal{P}_i$ and $\mathcal{P}_j$ denote the empirical distributions induced by ϕ over D_i and D_j, respectively. We define the distributional discrepancy S between datasets as:

$$S(D_i, D_j) := \mathcal{D}(\mathcal{P}_i, \mathcal{P}_j), \tag{2}$$

where $\mathcal{D}$ is a statistical divergence or distance between the empirical feature distributions $\mathcal{P}_i$ and $\mathcal{P}_j$. This discrepancy satisfies the identity property:

$$S(D_i, D_j) = 0 \quad \text{if } \mathcal{P}_i = \mathcal{P}_j, \tag{3}$$

$$S(D_i, D_j) > 0 \quad \text{if } \mathcal{P}_i \neq \mathcal{P}_j. \tag{4}$$

A training of a Φ network adjusts the w parameters using a D dataset. The trained network is written as $\Phi_{w(D)}$.

Suppose a network trained with the dataset D_{train} whose predictions for a set X are written $\hat{Y}$, i.e.

$$\hat{Y} = \Phi_{w(D_{train})}(X) \tag{5}$$

The following premise is proposed:

***Premise** 1:* Let D_r denote the real target dataset, D_c the controlled source dataset, and D_{train} a training dataset used to train the detector. If

$$S(D_{train}, D_r) < S(D_c, D_r), \tag{6}$$

then the performance obtained by training on D_{train} is expected to be greater than that obtained by training on D_c:

$$\text{mAP@50}(\hat{Y}_{train}, Y_r) > \text{mAP@50}(\hat{Y}_c, Y_r). \tag{7}$$

We will assume the above premise to be true. Previous premise is supported with statistical evidence [26], [27] shows us that when two sets are similar it is possible to make predictions for examples of them.

For our particular case: we have a dataset D_c, whose distance S with D_r is greater than 0. When training Φ_w with D_c its performance is poor, i.e. $mAP@50 \approx 0$. So, one problem to solve in this paper is:

***Problem** 1:* Find a dataset D_g such that the following is satisfied: $S(D_g, D_r) < S(D_c, D_r)$ and $mAP@50(\hat{Y_g}, Y_r) \gg mAP@50(\hat{Y_c}, Y_r)$, where:

- $\hat{Y_g}$ are the predictions obtained by using D_g in training, formally: $\hat{Y_g} = \Phi_{w(D_g)}(D_r)$
- $\hat{Y_c}$ are the predictions obtained by using D_c in training, formally: $\hat{Y_c} = \Phi_{w(D_c)}(D_r)$
- Y_r are the real labels of D_r

To obtain the dataset D_g from the original one, a transformation function is defined. It consists of a series of explainable image processing steps to generate synthetic images. To ensure that the parameters of this transformation are as objective as possible, an optimization stage is carried out, with the objective of finding the values that best suit the task to be solved. This is formally described as follows:

***Definition** 3:* Let T be a function that performs a transformation to a dataset such that:

$$T(D, \theta) : \mathcal{D} \times \Theta \to \mathcal{D}, \tag{8}$$

where:

- D is a dataset consisting of a set of images X and their respective labels Y.
- θ is a vector of parameters that controls the transformations on D.

In consequence, a second problem is stablished:

***Problem** 2:* Find the values of θ^* that maximizes mAP@50 over the validation dataset, i.e:

$$\theta^* = \arg\max_{\theta} mAP@50(\hat{Y_T}, Y_r) \tag{9}$$

where

$$\hat{Y_T} = \Phi_{w(T(D_c,\theta))}(D_r) \tag{10}$$

B. Proposed Solution

To address these challenges, we propose a two-phase strategy. In the first phase, a genetic algorithm optimizes the parameters θ of an explainable image transform, $T(D, \theta)$. The objective is to maximize the mAP@50 of a YOLO model trained for a limited number of epochs on the transformed dataset, with performance measured against a small validation set. This first phase specifically addresses Problem 2. The explainable transform incorporates segmented tomatoes and leaves subjected to various geometric transformations. In the second phase, we utilize the optimized parameters θ to generate a significantly larger training dataset, D_{train}. This expanded dataset is used to train the YOLO model over more epochs to ensure better generalization. Finally, the fully trained model is evaluated on the real target dataset.

C. Synthetic Data Transformation

The dataset transformation function (T), as formalized in Equation (8), aims to create a generated dataset (D_g) using only a dataset outside the real context (D_c). We use an explainable approach to generate examples, which consists of a series of basic image processing transformations to build the synthetic dataset function. First, the background is changed using real greenhouse images and generative-IA backgrounds. These backgrounds were obtained through an iterative prompting process using DALL-E 3 (via ChatGPT), specifically engineered to generate hyper-realistic scenes of tomato plants within greenhouse settings. Then, multiple tomatoes are placed at random positions within these new backgrounds. To simulate occlusion, leaves are added over the tomatoes, controlling the percentage of the fruit that is covered. In summary, this process defines the algorithm developed to generate synthetic images with realistic variations in the arrangement and visibility of tomatoes. Four different operations are proposed that can be applied to the images: changing the background, adding multiple tomatoes, introducing occlusion, and adjusting the size of both tomatoes and leaves in case of occlusion. In general, all these transformations are grouped

Algorithm 1 Background change

Require: $I \in \mathbb{R}^{H\times W\times C}$, $M \in \{0,1\}^{H\times W}$, $N_{\text{t}} \in \mathbb{N}$, $s \in \mathbb{R}^{+}$ $B_{\text{c}} = \{F_k\}$, $F_k \in \mathbb{R}^{H_F\times W_F\times C}$
Ensure: $I_{\text{trans}} \in \mathbb{R}^{H_o\times W_o\times C}$, $Coord = \{\mathcal{C}_j\}$
1: $ISF \leftarrow Resize(I, s)$
2: $MSF \leftarrow Resize(M, s)$
3: $Ib \leftarrow Zeros(H_F, W_F, C)$
4: $K \leftarrow RandomInt(0, |B_c| - 1)$
5: $Fn \leftarrow B_c[K]$
6: **for** $i = 0$ **to** $N_{\text{t}} - 1$ **do**
7: $\quad x \leftarrow RandomInt(0, W)$
8: $\quad y \leftarrow RandomInt(0, H)$
9: $\quad Ib \leftarrow Paste(Ib, ISF, (x, y), mask = MSF)$
10: $\quad Fn \leftarrow Paste(Fn, MSF \cdot 0, (x, y))$
11: **end for**
12: $I_{\text{trans}} \leftarrow AlphaBlend(Fn, Ib)$
13: $Coord \leftarrow Contours(Ib)$
14: **Output** $I_{\text{trans}}, Coord$

(a) Sample 1 (b) Sample 2 (c) Sample 3

Fig. 5. Examples of synthetic images generated using the background change function (Algorithm 1). (a) Sample 1 (b) Sample 2 (c) Sample 3

into two functions: background change and occlusion, which are described below.

The function *Background change* starts by receiving as input an image (I), its mask (M), the number of tomatoes to process (N_t), a scale factor (s) and the set of backgrounds images (B_c). It then resizes both the image and the mask using this factor and stores the results in ISF and IM, respectively (Lines 1-2). Next, a black background image (Ib) is created (Line 3) and a background is randomly selected from a list of available backgrounds (B_c), choosing its index at random (K) and obtaining the corresponding background (Fn) (Lines 4-5).

The function then enters a loop that repeats $(N_t - 1)$ times (Line 6). At each iteration, random (x, y) coordinates are generated within the range (0, 640) (Line 7-8). Subsequently, the resized image (ISF) is placed on the black background image (Ib) at position (x, y) (Line 9) and the mask (IM) is inserted into the selected background (Fn) at the same position (Line 9-10).

At the end of the loop, Ib and Fn are combined to obtain the transformed image (I_{trans}) (Line 12). Then, the contours of Ib are extracted and stored in $Coord$ (Line 13). Finally, the algorithm returns as output the transformed image (I_{trans}) and the extracted coordinates ($Coord$) (Line 14). This can be seen in a more simplified way in Algorithm 1, and Fig. 5 shows some examples of the images generated by this function.

The function *Occlusion* receives as input an image (I), the number of tomatoes to process (N_t), a list of coordinates ($Coord$) and a scale factor (s) (Require). Then, it loads a single, predefined leaf image (H_{sf}) and its mask (H_m) (Lines 1-2), which serves as a constant reference with fixed orientation, shape, and color properties during the generation process. Both images are resized according to the scale factor (s), storing the results in H_{sf} and H_m (Lines 3-4). Subsequently, a black background image (Ib) is created (Line 6).

Algorithm 2 Occlusion

Require: $I \in \mathbb{R}^{H\times W\times C}$, $N_{\text{t}} \in \mathbb{N}$, $Coord = \{(x_i, y_i)\}_{i=1}^{N_{\text{t}}}$, $s \in \mathbb{R}^{+}$
Ensure: $I_{\text{oc}} \in \mathbb{R}^{H\times W\times C}$
1: $H_{sf} \leftarrow LeafImage$
2: $H_m \leftarrow LeafMask$
3: $H_{sf} \leftarrow Resize(H_{sf}, s)$
4: $H_m \leftarrow Resize(H_m, s)$
5: $Ib \leftarrow Zeros(H, W, C)$
6: **for** $i = 0$ **to** $N_{\text{t}} - 1$ **do**
7: $\quad Ib \leftarrow Paste(Ib, H_{sf}, Coord[i])$
8: $\quad I \leftarrow Paste(I, H_m, Coord[i])$
9: **end for**
10: $I_{\text{oc}} \leftarrow Ib + I$
11: **Output** I_{oc}

(a) Sample 1 (b) Sample 2 (c) Sample 3

Fig. 6. Examples of synthetic images generated using the leaf occlusion function (Algorithm 2). (a) Sample 1 (b) Sample 2 (c) Sample 3

Next, the function enters a loop that is repeated $N_t - 1$ times (Line 6). In each iteration, the leaf image (H_{sf}) is inserted into IB at the position specified by $Coord[i]$ (Line 7) and, similarly, the leaf mask (H_m) is pasted into the original image (I) at the same position ($Coord[i]$) (Line 8). After finishing the loop, Ib and I are combined to generate the image with occlusion (I_{oc}) (Line 10). Finally, the algorithm returns as output the generated image (I_{oc}) (Line 12). This can be seen in a more simplified way in Algorithm 2, and Fig. 6 shows some examples of the images generated by this function.

D. Optimization with genetic algorithms

To solve problem 2, which consists of finding the θ^* values that satisfy the Equation (9), a metaheuristic algorithm known as the genetic algorithm [28] is used as an optimizer to determine these values.

A genetic algorithm is composed of several key elements, the most fundamental being the gene, which represents the main unit with which the algorithm operates. This gene contains the possible values of the variables to be optimized to solve the objective function, known as its genotype. The actual representation of these values within the algorithm, used for manipulation and evaluation, is called the phenotype, which in this case corresponds to the set of images that must be created to solve problem 1.

The function T (Equation (8)) is in charge of generating D_g.The values that make up the set of the θ parameter of the T function are the following:

$$\theta = \{p_f, n_t, p_o, o, l\}, \tag{11}$$

where:

- p_f is the probability of background change
- n_t is the number of tomatoes
- p_o is the probability of occlusion
- o is the occlusion percentage
- l is the tomato size

Since these values are in charge of controlling the transformations of each image for the creation of D_g. The encoding of this genotype to obtain its corresponding phenotype is detailed in Algorithm 3, which receives as input a genotype (θ) and a dataset (D_c). Then, it selects an image ($Imgs$) from the dataset (D_c) (Line 1) and segments the tomatoes in the image to obtain their masks ($Masks$) (Line 2). From these masks, it extracts the regions of interest ($Tomatoes$) corresponding to the segmented tomatoes (Line 3). Subsequently, it starts a loop that runs through all the images in $Imgs$ (Line 4). At each iteration, it generates a random value ($random1$) between 0 and 1 (Line 5). If this value is less than $\theta[0]$ (Line 6), it obtains the mask and the corresponding tomato region of interest (Lines 7-8). Then, it changes the background of the image using the function $Background$, which receives the mask, the tomato image and certain genotype parameters ($\theta[1]$ and $\theta[4]$), returning the transformed image (Img_t) and its coordinates ($coords$) (Line 9).

Then, another random number ($random2$) is generated (Line 10). If this value is less than $\theta[2]$ (Line 11), occlusion is applied to the image (Img_t) using the function $Occlusion$, which receives the transformed image, the coordinates and other parameters ($\theta[1]$, $\theta[3]$ and $\theta[4]$), generating a new version of the image (Line 12). Both the final image (Img_t) and its coordinates ($coord$) are saved (Lines 13). If the $random2$ condition is not satisfied, the transformed image is saved without occlusion together with its coordinates (Lines 15). If $random1$ is not less than $\theta[0]$ (Line 17), the image is resized to 640×640 pixels (Line 18), and its coordinates are calculated by $calcCoord$ (Line 19). Then, the image and its coordinates are stored (Lines 20). Finally, the loop continues until all images are completed (Line 23).

Within the genetic algorithm there are several operations in charge of searching for the best candidate. These operations are the fitness function, parent selection, crossover, mutation and individual selection. In this study, these operations were implemented as follows: for the fitness function, the mAP@50 metric, obtained by evaluating the detection performance of the YOLO model using the D_g dataset with the values of each individual; the selection of parents was performed using the roulette method; the crossover operation was implemented using single-point crossover; mutation was executed using cross mutation; and the selection of the best individuals was carried out using the tournament selection method. With these elements, the algorithm was implemented, which is presented systematically in the Algorithm 4.

The Algorithm 4 implements a genetic algorithm and receives as input the number of generations (g), the crossover probability (p_c) and the mutation probability (p_m)

Algorithm 3 Genotype decoding

Require: Genotype θ, dataset D_c
Ensure: Synthetic dataset D_g
1: $Imgs \leftarrow D_c[X]$
2: $Masks \leftarrow \texttt{SegmentTomatoes}(Imgs)$
3: $Tomatoes \leftarrow \texttt{ROI}(Imgs, Masks)$
4: **for** $i = 0$ **to** $|Imgs| - 1$ **do**
5: $random1 \leftarrow \mathcal{U}(0, 1)$
6: **if** $random1 < \theta[0]$ **then**
7: $msk \leftarrow Masks[i]$
8: $imsf \leftarrow Tomatoes[i]$
9: $Img_t, coord \leftarrow \texttt{Background}(msk, imsf, \theta[1], \theta[4])$
10: $random2 \leftarrow \mathcal{U}(0, 1)$
11: **if** $random2 < \theta[2]$ **then**
12: $Img_t \leftarrow \texttt{Occlusion}(Img_t, coord, \theta[1], \theta[3], \theta[4])$
13: Save Img_t, $coord$
14: **else**
15: Save Img_t, $coord$
16: **end if**
17: **else**
18: $Img_t \leftarrow \texttt{resize}(Imgs[i], 640, 640)$
19: $coord \leftarrow \texttt{calcCoord}(Img_t)$
20: Save Img_t, $coord$
21: **end if**
22: **end for**
23: **Output** D_g

Algorithm 4 Genetic algorithm

Require: Number of generations g, crossover probability p_c, mutation probability p_m
Ensure: Final population *popu*
1: $popu \leftarrow \texttt{populationGeneration}()$
2: $ng \leftarrow \emptyset$
3: **for** $j = 0$ **to** $g - 1$ **do**
4: $f \leftarrow \texttt{Fitness}(popu)$
5: **for** $k = 0$ **to** $|popu|/2 - 1$ **do**
6: $p1, p2 \leftarrow \texttt{Selection}(popu, f)$
7: $random \leftarrow \mathcal{U}(0, 1)$
8: **if** $random < p_c$ **then**
9: $h1, h2 \leftarrow \texttt{Cross}(popu[p1], popu[p2])$
10: $h1, h2 \leftarrow \texttt{Mutation}(h1, h2, p_m)$
11: $m1, m2 \leftarrow \texttt{Tournament}(p1, h1, p2, h2)$
12: **else**
13: $m1 \leftarrow popu[p1]$
14: $m2 \leftarrow popu[p2]$
15: **end if**
16: $ng \leftarrow ng \cup \{m1\}$
17: $ng \leftarrow ng \cup \{m2\}$
18: **end for**
19: $popu \leftarrow ng$
20: $ng \leftarrow \emptyset$
21: **end for**
22: **Output** $popu$

(Require). Initially, it generates a population using the $populationGeneration()$ function (Line 2) and defines an empty list ng to store the new generation (Line 2). The evolutionary process is repeated for $g-1$ generations (Line 3). At each iteration, the fitness (f) of the current population is calculated using the function $Fitness(popu)$ (Line 4). Then, a loop is started that runs through half of the population (Line 5), as each iteration will generate two new individuals. In each iteration, two parents ($p1$ and $p2$) are selected using the $Selection$ function, which takes into account the fitness f (Line 6).

Then, a random number ($random$) in the range (0,1) is generated (Line 7). If this value is less than p_c (Line 8), crossover ($Cross$) is applied between the parents to generate

two children ($h1$ and $h2$) (Line 9). Next, these offspring can undergo mutation via the $Mutation$ function, which uses the p_m mutation probability (Line 10). Then, a tournament ($Tournament$) is applied between parents and offspring to select the two best individuals ($m1$ and $m2$) (Line 11).

If $random$ is not smaller than p_c, the new individuals are simply copied from the parents ($m1 \leftarrow popu[p1]$ and $m2 \leftarrow popu[p2]$) without applying crossover or mutation (Lines 13-14). In both cases, the selected individuals ($m1$ and $m2$) are added to the new generation (ng) (Lines 16-17).

Once enough individuals have been generated to complete the new population, $popu$ is updated with ng (Line 19), and ng is emptied for the next iteration (Line 20). Finally, after completing all generations, the algorithm returns the final population as the result (Line 22).

V. Experimental Evaluation

A. Implementation

The algorithm was implemented in Python 3.8.15, using open source libraries such as NumPy, OpenCV, Rembeg and PIL; the source code is publicly available at https://github.com/GerardoAAlvarezHernandez/tomato-ripening-synthetic-dataset. To calculate the mAP@50 metric, used as a measure of the fitness of individuals, 190 images of tomatoes captured in the real environment where detection is desired, which has a total of 3,626 examples, were evaluated. The execution of the algorithm was carried out on a computer running Ubuntu 22.04.2 LTS, an AMD Ryzen 7 5700G processor, 32 GB of RAM and an Nvidia GeForce RTX 3080 Lite Hash Rate GPU.

The training process of the YOLO object detector in the two stages of the proposed system was carried out as follows: for the optimization stage and the calculation of the mAP@50 metric, YOLOv5n was used with the following hyperparameters: 50 epochs, a learning rate of 1×10^{-3}, the ADAM optimizer and a batch size of 16. In the final stage of the system, YOLOv5m was used with a learning rate of 1×10^{-3}, a batch size of 16, the ADAM optimizer and a total of 200 epochs.

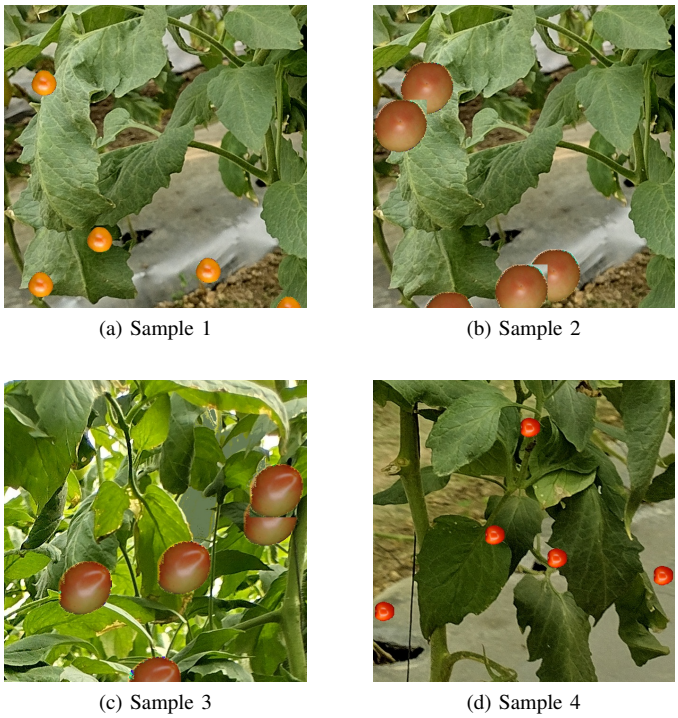
B. Optimization Stage

In the optimization stage the genetic algorithm for the search of the best parameters of θ was run with the following parameters: 10 generations, a population of 10 individuals, a crossover probability of 0.85, mutation probability of 0.1. The algorithm took approximately 2.1 days to finish the search for the best solution.

At the end of the execution of the algorithm the values of the best population were the following (see Table I). As it could be observed the best individual that resulted was the one whose values of θ are: $[0.5039, 5, 0.7947, 0.1662, 0.0266]$. Which can be interpreted in the following way: the 1° value indicates has a probability of almost 50.39% that an image with a new background is created, the 2° indicates that the new images will have 5 objects (tomatoes) in it, the 3° indicates the new image will have a probability of 79. 47% that the objects will be covered by a leaf, the 4° indicates that when there is occlusion the leaf will cover 16.62% of the object that is present in the image and finally the 5° indicates that the variation of the tomato sizes will be between 2.66% to 100% with respect to the final size of the image. Some examples images of the dataset generated by using values of the best individual in this experiment can be seen in Fig. 7.

TABLE I
Values of Individuals Belonging to the Best Population Found by the Genetic Algorithm

Individual	*Values*	*Fitness*
1	**[0.5039, 5, 0.7947, 0.1662, 0.0266]**	**0.11103**
2	[0.6414, 2, 0.5392, 0.3604, 0.1574]	1.240×10^{-2}
3	[0.5039, 2, 0.5392, 0.3604, 0.1574]	0.0
4	[0.6414, 2, 0.5392, 0.36041, 0.1574]	1.240×10^{-2}
5	[0.5039, 5, 0.7947, 0.9031, 0.5425]	0.0
6	[0.6414, 2, 0.5392, 0.3604, 0.1574]	1.240×10^{-2}
7	[0.5039, 5, 0.9005, 0.1662, 0.0266]	0.0795
8	[0.6414, 2, 0.5392, 0.3604, 0.1574]	3.8489×10^{-4}
9	[0.5039, 5, 0.5392, 0.3604, 0.1574]	2.6204×10^{-4}
10	[0.6414, 2, 0.5392, 0.3604, 0.1574]	1.240×10^{-2}



(a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

Fig. 7. Examples of synthetic images generated using the optimal parameters obtained in the optimization stage through the genetic algorithm. (a) Sample 1 (b) Sample 2 (c) Sample 3 (d) Sample 4

C. Final Stage

With the θ values obtained by the genetic algorithm, they were used in the T function (Equation (8)) to generate a synthetic dataset, with the objective of further training the object detection model. In this case, YOLOv5m, a more robust version of this architecture, used in the optimization stage, was employed. After training, the model performance was evaluated using the D_r dataset for validation, obtaining the performance metrics presented in Table III.

TABLE II
PERFORMANCE METRICS OBTAINED FROM THE YOLOV5M MODEL USING D_c (CONTROLLED IMAGE DATASET) AS INPUT

Class	Instances	P	R	mAP@50
All	238	0.0105	0.0222	0.00381
Red	29	0	0	0
Red-orange	23	0	0	0
Orange	12	0	0	0
Striped	14	0.0138	0.0714	0.00565
Salmon	21	0.0133	0.0476	0.000957
Green	139	0.0358	0.0144	0.0163

TABLE III
PERFORMANCE METRICS OBTAINED FROM THE YOLOV5M MODEL USING D_g (GENERATED DATASET) AS INPUT

Class	Instances	P	R	mAP@50
All	238	0.186	0.145	0.131
Red	29	0.0902	0.0345	0.033
Red-orange	23	0.418	0.261	0.364
Orange	12	0	0	0.0165
Striped	14	0.0649	0.214	0.0999
Salmon	21	0.0439	0.0952	0.0263
Green	139	0.498	0.266	0.249

In order to compare and analyze the improvement obtained with the proposed methodology, we trained the same detection model using D_c as the input training set. It is important to emphasize that this training incorporated YOLOv5's inherent data augmentation pipeline—including flips, rotations, scaling, and mosaic creation—applied consistently to the input images throughout the entire process. The resulting performance metrics, which highlight the limitations of traditional augmentation when facing a significant domain gap, are presented in Table II.

When comparing the Tables II and III, it is observed that the use of the set D_g, obtained by using the values of θ found as input parameter for the function T, improves the performance by approximately 3438.32%. This improvement is evidenced by analyzing the mAP@50 metric obtained for all classes in each experiment. This demonstrates that our methodology, based on adapting a dataset in a controlled context to a more realistic environment, such as a greenhouse, is effective. Furthermore, when analyzing the metrics obtained in the experiment using D_g, it is observed that the improvement is more noticeable in the green and red-orange classes (see Fig. 8). This is because these colors show less variation in intensity under different illumination conditions compared to the other classes. However, the high variability in illumination remains one of the main limitations of the model.

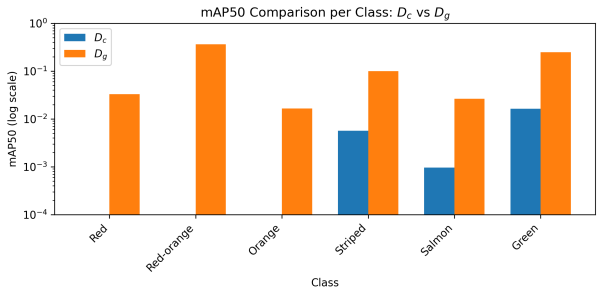


Fig. 8. Comparative graph of mAP@50 values by class with respect to the database with controlled images (D_c) and the generated database (D_g)

TABLE IV
PERFORMANCE METRICS OBTAINED USING DIFFERENT INPUT DATASETS

Input Dataset	P	R	mAP@50
D_c	0.0105	0.0022	0.00381
D_g	0.186	0.145	0.131
$D_{r(\text{green only})}$	0.941	0.0647	0.15
$D_{r(\textbf{green only})} + D_g$	**0.271**	**0.205**	**0.217**

(a) Frame 1 (b) Frame 2 (c) Frame 3

Fig. 9. Examples of inference results obtained with the best performing model, trained with $D_{r(\text{green only})} + D_g$. (a) Frame 1 (b) Frame 2 (c) Frame 3

To better evaluate the improvements obtained by using this set, additional experiments were carried out, the results of which are presented in Table IV. This table shows the average metrics obtained for all classes.

The experiments consisted of training the YOLO model with the same hyperparameters used in the final stage, but varying the input datasets. In addition, a new dataset called $D_{r(greenonly)}$ was added, composed of images of tomatoes inside a greenhouse, but exclusively at the "green" ripening stage.

The results show that the best performance was obtained when the training set was composed of $D_{r(greenonly)} + D_g$, demonstrating that the synthetic set contributes to the generalization of the other color classes that were not present in $D_{r(greenonly)}$ under real conditions.

Some examples of the inferences made by the final model can be seen in Fig. 9. For the sake of comparison in the inference, Fig. 10 shows the results obtained when training the model using only D_C as input.

D. Distributional Alignment Validation

To empirically validate the proposed domain proximity premise, we quantify the distributional distance between the controlled (D_c), synthetic (D_g), and real (D_r) datasets in the embedding space induced by ResNet101.

Feature embeddings were extracted for all images and compared using multiple statistical distances, including Jensen–

(a) Frame 1 (b) Frame 2 (c) Frame 3

Fig. 10. Examples of inference results obtained with the model trained using only D_c as input dataset. (a) Frame 1 (b) Frame 2 (c) Frame 3

TABLE V
DISTANCE METRICS EVALUATION ($N = 132$, RUNS$= 30$, $K = 200$)

Metric	Value
JS (D_g, D_r)	0.822108 ± 0.007797
JS (D_c, D_r)	0.832555 ± 0.000000
KL_{sym} (D_g, D_r)	22.844757 ± 0.558560
KL_{sym} (D_c, D_r)	23.700478 ± 0.048516
Frechet (D_g, D_r)	0.879 ± 0.033
Frechet (D_c, D_r)	1.349 ± 0.012
SWD (D_g, D_r)	0.011373 ± 0.000745
SWD (D_c, D_r)	0.016451 ± 0.001304

Shannon (JS) divergence, symmetric Kullback–Leibler divergence (KL_{sym}), Fréchet distance, and Sliced Wasserstein Distance (SWD). For robustness, the comparison was performed over 30 balanced sampling runs, where the number of images per domain was matched in each iteration.

Table V reports the mean and standard deviation of each metric. Across all evaluated measures, the synthetic dataset consistently exhibits lower distributional distance to the real dataset compared to the controlled dataset. In particular, the Fréchet distance decreases from 1.349 ± 0.012 (D_c vs D_r) to 0.879 ± 0.033 (D_g vs D_r), representing an approximate 35% reduction in global geometric discrepancy within the embedding space. Similarly, the Sliced Wasserstein Distance decreases from 0.01645 ± 0.00130 to 0.01137 ± 0.00075, indicating improved alignment under a transport-based metric. Probabilistic discrepancies show a consistent tendency, with both JS divergence and KL_{sym} exhibiting lower values for D_g–D_r comparison relative to D_c–D_r. In consequence, we can affirm that the experimentation provides positive evidence of the validity of our premise.

VI. CONCLUSIONS AND FUTURE WORK

This work has demonstrated that explainable synthetic data generation is an effective strategy for detecting ripening stages of tomatoes in greenhouses. Variability in lighting conditions makes it difficult to capture real images that represent all ripening stages, which makes the use of synthetic data a valuable alternative for training detection models. Thanks to this methodology, it was possible to develop a system capable of identifying different maturation stages using images beyond the specific context of the problem.

The optimization algorithm employed made it possible to efficiently adjust the parameters of the transformation function, thus improving the quality of the synthetic images and, consequently, significantly increasing the generalization capacity of the model. In addition, the flexibility of this methodology allows its application to other crops, opening the possibility of using it in the detection and monitoring of various agricultural products. The experiments carried out demonstrated that the YOLO model, trained with synthetic images generated using our methodology, was able to detect tomatoes at different stages of ripeness. In contrast, the model trained exclusively with the original set of images captured in controlled environments failed to make effective detections. Although our approach still has limitations, the results show a significant improvement in the model's generalization ability, allowing it to identify tomatoes in realistic scenarios.

For future work, we propose to explore different color spaces to improve the representation of illumination variations in tomatoes and increase the robustness of the model. Furthermore, we intend to evaluate the influence of genetic algorithm hyperparameters, specifically population size and number of generations, which were kept constant in this study. Exploring these parameters will allow us to determine their impact on the quality of the generated dataset and the optimization of the domain gap. Additionally, we will seek to optimize the implementation of the genetic algorithm and improve image processing in the transformation, with the goal of reducing run time and extending the system's ability to adapt to variations in the real environment.

REFERENCES

[1] H. Nie, X. Yang, S. Zheng, and L. Hou, "Gene-based developments in improving quality of tomato: Focus on firmness, shelf life, and pre-and post-harvest stress adaptations," *Horticulturae*, vol. 10, no. 6, p. 641, 2024, DOI: https://doi.org/10.3390/horticulturae10060641.

[2] S. Akter, M. Khatun, M. Rahman, and M. F. Mondal, "Maturity stage and post-harvest treatments on quality and shelf life of tomatoes," *Applied Agriculture Sciences*, vol. 4, no. 1, pp. 46–56, 2024, DOI: https://doi.org/10.25163/agriculture.2110046.

[3] Y. Edan, G. Adamides, and R. Oberti, "Agriculture automation," *Springer handbook of automation*, pp. 1055–1078, 2023, DOI: https://doi.org/10.1007/978-3-030-96729-1_49.

[4] K. H. Coble, A. K. Mishra, S. Ferrell, and T. Griffin, "Big data in agriculture: A challenge for the future," *Applied Economic Perspectives and Policy*, vol. 40, no. 1, pp. 79–96, 2018, DOI: https://doi.org/10.1093/aepp/ppx056.

[5] N. Giakoumoglou, E. M. Pechlivani, and D. Tzovaras, "Generate-paste-blend-detect: Synthetic dataset for object detection in the agriculture domain," *Smart Agricultural Technology*, vol. 5, p. 100258, 2023, DOI: https://doi.org/10.1016/j.atech.2023.100258.

[6] M. Afonso and V. Giufrida, "Synthetic data for computer vision in agriculture," *Frontiers in Plant Science*, vol. 14, p. 1277073, 2023, DOI: https://doi.org/10.3389/fpls.2023.1277073.

[7] Laboro.ai, "Laboro tomato dataset for object detection," https://github.com/laboroai/LaboroTomato, 2020.

[8] D. Wu, S. Jiang, E. Zhao, Y. Liu, H. Zhu, W. Wang, and R. Wang, "Detection of camellia oleifera fruit in complex scenes by using yolov7 and data augmentation," *Applied sciences*, vol. 12, no. 22, p. 11318, 2022, DOI: https://doi.org/10.3390/app122211318.

[9] P. Enkvetchakul and O. Surinta, "Effective data augmentation and training techniques for improving deep learning in plant leaf disease recognition," *Applied Science and Engineering Progress*, vol. 15, no. 3, pp. 3810–3810, 2022, DOI: https://doi.org/10.14416/j.asep.2021.01.003.

[10] R. Nithya, B. Santhi, R. Manikandan, M. Rahimi, and A. H. Gandomi, "Computer vision system for mango fruit defect detection using deep convolutional neural network," *foods*, vol. 11, no. 21, p. 3483, 2022, DOI: https://doi.org/10.3390/foods11213483.
[11] J. J. Bird, C. M. Barnes, L. J. Manso, A. Ekárt, and D. R. Faria, "Fruit quality and defect image classification with conditional gan data augmentation," *Scientia Horticulturae*, vol. 293, p. 110684, 2022, DOI: https://doi.org/10.1016/j.scienta.2021.110684.
[12] L. Divyanth, D. Guru, P. Soni, R. Machavaram, M. Nadimi, and J. Paliwal, "Image-to-image translation-based data augmentation for improving crop/weed classification models for precision agriculture applications," *Algorithms*, vol. 15, no. 11, p. 401, 2022, DOI: https://doi.org/10.3390/a15110401.
[13] B. Min, T. Kim, D. Shin, and D. Shin, "Data augmentation method for plant leaf disease recognition," *Applied Sciences*, vol. 13, no. 3, p. 1465, 2023, DOI: https://doi.org/10.3390/app13031465.
[14] H. Tan, Y. Hu, B. Ma, G. Yu, and Y. Li, "An improved dcgan model: Data augmentation of hyperspectral image for identification pesticide residues of hami melon," *Food Control*, vol. 157, p. 110168, 2024, DOI: https://doi.org/10.1016/j.foodcont.2023.110168.
[15] M. Momeny, A. Jahanbakhshi, A. A. Neshat, R. Hadipour-Rokni, Y.-D. Zhang, and Y. Ampatzidis, "Detection of citrus black spot disease and ripeness level in orange fruit using learning-to-augment incorporated deep networks," *Ecological Informatics*, vol. 71, p. 101829, 2022, DOI: https://doi.org/10.1016/j.ecoinf.2022.101829.
[16] G. Dai, J. Fan, Z. Tian, and C. Wang, "Pplc-net: Neural network-based plant disease identification model supported by weather data augmentation and multi-level attention mechanism," *Journal of King Saud University-Computer and Information Sciences*, vol. 35, no. 5, p. 101555, 2023, DOI: https://doi.org/10.1016/j.jksuci.2023.101555.
[17] A. Rahman, Y. Lu, and H. Wang, "Performance evaluation of deep learning object detectors for weed detection for cotton," *Smart Agricultural Technology*, vol. 3, p. 100126, 2023, DOI: https://doi.org/10.1016/j.atech.2022.100126.
[18] H. Li, W. Guo, G. Lu, and Y. Shi, "Augmentation method for high intra-class variation data in apple detection," *Sensors*, vol. 22, no. 17, p. 6325, 2022, DOI: https://doi.org/10.3390/s22176325.
[19] X. Fu, S. Zhao, C. Wang, X. Tang, D. Tao, G. Li, L. Jiao, and D. Dong, "Green fruit detection with a small dataset under a similar color background based on the improved yolov5-at," *Foods*, vol. 13, no. 7, p. 1060, 2024, DOI: https://doi.org/10.3390/foods13071060.
[20] J. Gao, J. Zhang, F. Zhang, and J. Gao, "Lacta: A lightweight and accurate algorithm for cherry tomato detection in unstructured environments," *Expert systems with applications*, vol. 238, p. 122073, 2024, DOI: https://doi.org/10.1016/j.eswa.2023.122073.
[21] A. Fadihel *et al.*, "Tom2024: Datasets of tomato, onion, and maize images for developing pests and diseases ai-based classification models," *Data in Brief*, vol. 53, p. 110192, 2024, enfoque en detección de plagas y enfermedades, DOI: https://doi.org/10.1016/j.dib.2024.110192.
[22] A. Preatoni, "Tomato detection dataset," https://www.kaggle.com/datasets/andrewmvd/tomato-detection, 2020.
[23] up2metric, "tomatod dataset for tomato fruit localization and ripening classification," https://github.com/up2metric/tomatOD, 2023.
[24] A. Hernández, J. I. Vasquez-Gomez, J. C. Herrera-Lozada, and J. H. Abril-García, "Detection of tomato ripening stages using yolov3-tiny," *Research in Computing Science*, vol. 151, no. 10, pp. 217–231, 2022, DOI: https://doi.org/10.48550/arXiv.2302.00164, ISSN: 1870-4069.
[25] T. et al., "Labelimg." [Online]. Available: https://github.com/heartexlabs/labelImg
[26] I. Goodfellow, Y. Bengio, and A. Courville, *Deep Learning*. MIT Press, 2016, http://www.deeplearningbook.org.
[27] S. Ben-David, J. Blitzer, K. Crammer, A. Kulesza, F. Pereira, and J. W. Vaughan, "A theory of learning from different domains," *Machine learning*, vol. 79, no. 1, pp. 151–175, 2010, DOI: https://doi.org/10.1007/s10994-009-5152-4.
[28] A. E. Eiben and J. E. Smith, "Popular evolutionary algorithm variants," in *Introduction to Evolutionary Computing*. Springer, 2015, pp. 99–116, DOI: https://doi.org/10.1007/978-3-662-44874-8_6.

Gerardo Antonio Alvarez Hernandez received his M.Sc. degree from the Center for Innovation and Technological Development in Computing at the National Polytechnic Institute (IPN) in 2023. He earned his B.S degree in Communications and Electronics Engineering from the Higher School of Mechanical and Electrical Engineering (IPN) in 2020. Since 2024, he has been pursuing a PhD in Robotic and Mechatronic Systems Engineering at the IPN's Center for Innovation and Technological Development in Computing. His research interests include computer vision, deep learning, machine learning, and robotics applied to agriculture.

Juan Irving Vasquez received his M.Sc. and Ph.D. degrees from the National Institute for Astrophysics, Optics, and Electronics (INAOE), Mexico, in 2009 and 2014, respectively. He earned his B.S. degree in Computer Sciences from the Tehuacan Institute of Technology, Mexico, in 2006. From 2016 to 2021, he served as a researcher at the National Council of Science and Technology (CONACYT) in Mexico. Since 2021, he has been a full-time professor at the National Polytechnic Institute (IPN). His research interests include robotics, motion planning, view planning, and their applications to object reconstruction, inspection, and surveillance.

Abril Valeria Uriarte Arcia received her M.Sc. and Ph.D. degrees from the National Polytechnic Institute (IPN), Mexico, in 2012 and 2016, respectively. She earned her B.S. degree in Computer Sciences from the National University of Engineering, Nicaragua, in 2008. Since 2016, she has been a full-time professor at IPN. Her research interests include machine learning, time series, data streams, and their applications to different fields such as medical, environmental, and agriculture.

Luis Alberto Tovar-Ortiz is pursuing his Ph.D. program in Robotic and Mechatronic Systems Engineering at the Center for Innovation and Technological Development in Computing of the National Polytechnic Institute (IPN). His academic interest is in image processing for industrial inspection and maintenance information systems, particularly overhead cranes, works with embedded systems and mechatronics design.