ESTIMATION OF 3D POSITION AND SIZE OF FRUITS USING AN INTEL REALSENSE D435 CAMERA

Emanuel F. Chimuando[1], João A. Fabro[2], Sérgio F. Pichorim[3]

1-Master's Student in Electrical Engineering and Industrial Informatics – CPGEI – UTFPR – Curitiba, Brazil

2-Department of Informatics – UTFPR – Curitiba, Brazil

3-Department of Electronics DAELN – CPGEI – UTFPR – Curitiba, Brazil

A paper from the Proceedings of the 17th International Conference on Precision Agriculture and 11th Brazilian Congress on Precision and Digital Agriculture 13-15 July 2026 Porto Alegre, Brazil

Abstract.

This work addresses the development of a computer vision algorithm for an agricultural robot whose task is to harvest tomatoes in a plantation. The ability to detect fruits, as well as to estimate their 3D position and size in the world coordinate system, is fundamental to accomplishing this task. This topic has been widely discussed in the literature, particularly due to the challenge of obtaining accurate estimates of the spatial coordinates of objects. This process is strongly affected by several factors, such as irregular fruit shapes, curved and reflective surfaces, and partial occlusions, all of which significantly impact the accuracy of depth estimation.

In this research, we present a proposed architecture that employs the k-means algorithm to segment a specific sampling region in the depth map. A depth-gradient threshold filter is then applied to identify and retain planar regions within the selected sampling area. A Convolutional Neural Network (CNN) is used for tomato fruit detection, while an Intel RealSense D435 camera is employed to acquire color and depth frames.

Textured calibration targets, together with the Intel RealSense Viewer tool, were used to perform on-chip, tare, and focal length calibrations to ensure maximum accuracy of the camera's depth data. These procedures resulted in a Health Check value of 0.05 and a focal length imbalance of 0.006%.

The experiment was conducted under the assumption that the camera coordinate system and the world coordinate system coincide. To achieve this, the camera coordinate system was aligned with the world coordinate system. As a reference for the world coordinate system, two orthogonal laser beams (X-axis and Y-axis) projected onto a wall were used, generated by a laser level device. The device features a self-leveling mechanism, ensuring precise alignment with a vertical and horizontal accuracy of ±1.5 mm over 5 m. To determine measurement errors, the algorithm's results were compared with ground-truth position and size values measured manually using a laser distance meter and a digital caliper, with accuracies of ±2 mm and ±0.02 mm, respectively.

The proposed algorithm proved capable of identifying tomatoes at different stages of ripeness, while accurately estimating their positions and dimensions. In controlled-environment experiments, the method achieved Root Mean Square Errors (RMSE) of 4.1 mm, 4.0 mm, and

2.6 mm for the X, Y, and Z axes, respectively, and 3.0 mm and 4.3 mm for height and width, respectively. In field experiments, the RMSE values were 13.3 mm, 11.5 mm, and 21.5 mm for the X, Y, and Z axes, respectively, and 9.7 mm and 10.6 mm for height and width, respectively.

Regarding computational performance, the algorithm exhibited an average latency of 1.01 seconds per frame. It was observed that the CNN inference stage constitutes the main bottleneck of the system, accounting for approximately 97% of the total latency. This behavior is primarily due to the absence of GPU acceleration, as all processing was performed exclusively on the CPU, significantly limiting the overall system performance.

Therefore, the proposed algorithm does not meet strict real-time requirements. However, it may be suitable for applications with low dynamics, where latencies on the order of one second are considered acceptable, such as the application proposed in this work, in which the target objects exhibit slow or negligible movement.

Keywords: *YOLOv8, OpenCV, k-means, Stereo Vision, Depth Estimation, Intel RealSense D435, Bounding Box, 3D Position, Tomatoes, Calibration.*

Introduction

Autonomous agricultural robots are already being deployed in various farming operations, replacing traditional machinery and performing tasks such as planting, harvesting, and crop monitoring with greater speed and precision (Vijay & Ponnusamy, 2023). Automatic fruit harvesting requires computer vision algorithms capable of identifying fruits at the optimal stage of ripeness. The research and development challenges associated with these algorithms are considerable due to the variability of agricultural environments, the need for high accuracy, and the requirement that the system not only recognize the fruit but also determine its exact location. Convolutional Neural Networks (CNNs), particularly the YOLO (You Only Look Once) family of models, have revolutionized real-time object detection by eliminating the need for manual feature engineering while providing robust visual recognition capabilities. In parallel, the acquisition of 3D data has become essential for modern applications such as robotics and fruit harvesting. Although several distance measurement technologies exist, including LiDAR and radar, RGB-D cameras such as the Intel RealSense and Kinect have emerged as the preferred choice. Their low cost, compact size, and high accuracy at short and medium ranges have established them as an ideal solution for spatial coordinate mapping and scene reconstruction in intelligent systems. The integration of deep learning techniques with depth sensors has attracted significant scientific interest due to its effectiveness in detecting and measuring real-world objects. This versatile approach has driven advances in several fields and serves as the foundation for the development of increasingly robust solutions.

Suwandi et al. (2022) developed a computer vision system for a robotic soccer player using a Deep Neural Network (DNN) for real-time ball detection and an Intel RealSense D455 camera to acquire color and depth frames. Their algorithm converts the detected image into grayscale and divides it into a 3 × 3 matrix to calculate the standard deviation of each region. The center of the ball is then determined, and the average depth of the corresponding region is estimated, allowing the calculation of its spatial coordinates. Using this approach, the authors reported mean errors of 69.53 mm, 46.11 mm, and 65.77 mm for the X, Y, and Z coordinates, respectively.

Mengoli et al. (2022) proposed an architecture based on the YOLOv5 CNN for apple fruit detection, using an Intel RealSense D435i camera to acquire color and depth frames. The core of the approach consists of converting the detected fruit image to grayscale and subsequently applying the Hough Transform (Jamundá, 2000) to detect circular shapes, thereby obtaining the fruit dimensions in pixels. To estimate depth and, consequently, the fruit size in millimeters, a rectangular Region of Interest (ROI) is selected in the depth frame, specifically in the central

region of the detected fruit, and the average depth within this ROI is calculated. This approach achieved a mean measurement error ranging from 7 to 9 mm.

Andriyanov (2022) conducted a comparative study using Intel RealSense D415 and Intel RealSense D455 cameras for object coordinate estimation. Object detection was performed using the YOLOv5 CNN. The proposed algorithm uses the center point of the YOLOv5 bounding box (bbox) to estimate depth and subsequently determine the remaining spatial coordinates. The reported results showed errors ranging from 5 to 7 mm along the X and Y axes and approximately 15 mm along the Z axis when using the D415 camera. For the D455 camera, the errors ranged from 2 to 3 mm along the X and Y axes and approximately 8 mm along the Z axis.

Cuong Vo-Le et al. (2020) used an Intel RealSense D415 camera to estimate object dimensions. Their approach employs depth thresholding and the OpenCV FindContours method to detect objects in the depth frame and map them onto the color frame. Subsequently, the Good Features to Track (GFTT) algorithm is applied to identify key points within the object region, such as the corners of a rectangle. The Convex Hull algorithm is then used to determine the smallest polygon enclosing all detected points. Finally, an interpolation process is applied to compensate for depth inaccuracies along object boundaries before the object dimensions are calculated using the Euclidean distance between the 3D coordinates of the key points. The proposed method achieved a percentage error of 0.99%.

Materials and Methods

This section describes the materials and methods used in the development of this research. Figure 1 presents a flowchart of the proposed system, which employs a previously calibrated Intel RealSense D435 camera and utilizes the k-means algorithm to segment regions with different depth values. A specific sampling region corresponding to the fruit foreground is then selected from the depth map. Within this sampling region, a filter based on a depth-gradient threshold is applied to identify and retain planar regions. Subsequently, projection and perspective equations are used to estimate the 3D coordinates and dimensions of each detected fruit.

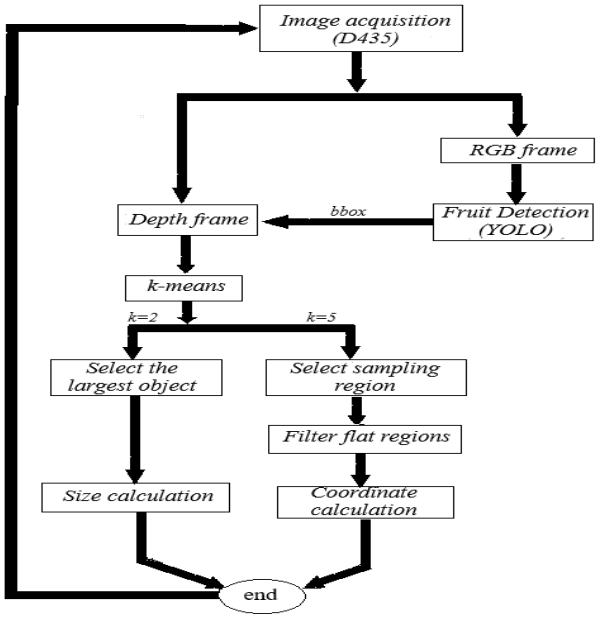

Fig. 1. Flowchart of the proposed system

The experiments were conducted on a computing platform equipped with an 8th-generation Intel Core i7-8650U processor. The processor operates at a base frequency of 1.90 GHz and can achieve higher clock speeds depending on workload demands. The system was equipped with 24 GB of RAM, providing sufficient computational resources to execute multiple applications simultaneously.

Intel RealSense D435 Camera

The Intel RealSense D435, shown in figure 2, is a depth camera belonging to the Intel RealSense D400 series, whose depth estimation is based on stereo vision. It is equipped with two infrared (IR) cameras for depth acquisition, an infrared projector, and an RGB camera for color image acquisition (Yoshida, 2022). Consequently, the device is capable of providing synchronized RGB-D data.

Fig. 2. Intel RealSense D435 camera (adapted from Grunnet-Jepsen et al., 2022)

All camera sensor modules are manufactured to achieve high robustness and are enclosed in laser-sealed steel housings to preserve calibration and maintain performance throughout their operational lifetime (Grunnet-Jepsen et al., 2022). Nevertheless, during use or storage, certain conditions may lead to calibration degradation, such as exposure to extreme temperature cycles, excessive shocks, or vibrations (Grunnet-Jepsen et al., 2022).

Intel RealSense D435 Camera Calibration

The purpose of camera calibration is to correct intrinsic and extrinsic errors. Intrinsic errors arise primarily from microscopic displacements in lens positions. Extrinsic errors are associated with small bends and twists in the supporting structure on which the two stereo sensors are mounted (Grunnet-Jepsen et al., 2022).

To restore camera calibration, Intel provides a set of tools, including the *Dynamic Calibration Tool*, *Intel RealSense Viewer*, and *Depth Quality Tool*, which enable recalibration in a fast, simple, and efficient manner. To optimize the camera's depth performance in terms of depth-noise reduction, the on-chip self-calibration procedure was performed. For this process, the camera was pointed toward a textured target (Figure 3a) mounted on a flat surface, as recommended by Grunnet-Jepsen et al. (2022), and calibration was carried out using the *Intel RealSense Viewer* tool. Subsequently, to improve the accuracy of the camera's depth measurements, the tare calibration procedure was executed. Tare calibration requires the use of a target with objects of known dimensions (Figure 3b) to determine the distance from the left camera origin to the target and then perform the calibration process. In addition, focal length calibration was performed to detect and correct focal length imbalance. This technique also uses a textured target as a reference (Figure 3b). Both tare calibration and focal length calibration were carried out using the *Intel RealSense Viewer*.

Once calibration is completed, the software returns a calibration result. For on-chip calibration, the output is an index known as the *Health Check*, which indicates the degree of calibration deviation from the optimal condition. Values close to zero are the most desirable; when the value is equal to or less than 0.25, the camera is considered to be nearly optimally calibrated. Conversely, values above this threshold indicate that the camera's performance may be improved through recalibration.

For focal length calibration, the reported metric is the *focal length imbalance*. This measurement is typically accurate within ±0.03%. In most cases, a focal length imbalance of up to ±0.2% has a negligible impact on performance and can be tolerated without correction (Grunnet-Jepsen et al., 2022).

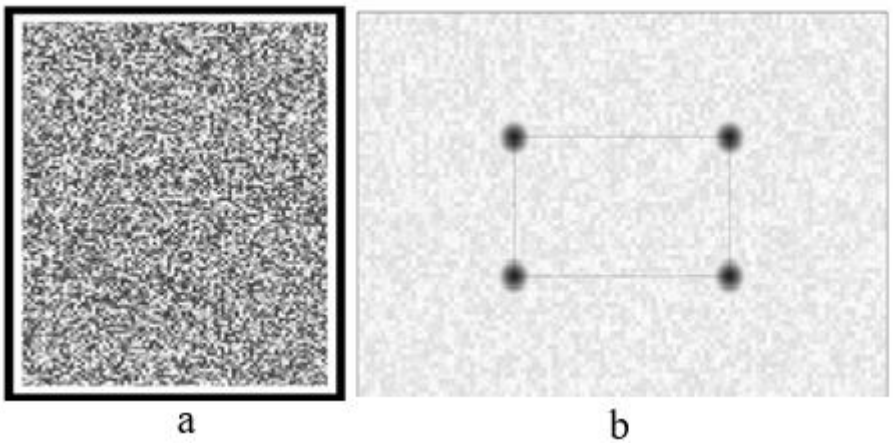

Fig. 3. A4 and A3 textured calibration targets used for on-chip self-calibration (a) and for tare and focal length calibration (b). Adapted from Grunnet-Jepsen et al. (2022)

Following the execution of the calibration procedures, an on-chip self-calibration *Health Check* value of 0.05 was obtained, while the focal length calibration resulted in a *focal length imbalance* of 0.006%.

Camera Data Preprocessing

Since the RGB and depth sensors of the Intel RealSense D435 are physically separated, each sensor has its own coordinate system. Therefore, it is necessary to align the depth image with the coordinate system of the color image. As a result, pixel-coordinate information can be mapped between the color and depth frames as if both belonged to a single coordinate system.
The *Librealsense* and *OpenCV* libraries were used to access camera data and perform frame alignment. Both color and depth frames were acquired at a resolution of 640 × 480 pixels and a frame rate of 30 frames per second (FPS).

Fruit Detection

To identify fruits within the scene from the color frames provided by the camera, a YOLOv8 Convolutional Neural Network (CNN) was trained. The YOLOv8 architecture with pretrained weights was selected due to its favorable balance between detection performance, accuracy, computational efficiency, and adaptability to the dataset. It also outperformed the YOLOv11

architecture, which was trained using the same dataset. The selected model was the YOLOv8s (small) variant, comprising approximately 11.2 million parameters, offering a good compromise between accuracy and computational cost. Initially, the pretrained yolov8s.pt model was trained for 30 epochs using an image size of 960 × 960 pixels (*imgsz = 960 × 960*) and a batch size of 5 (*batch = 5*). After validation, the model achieved mean Average Precision (mAP) values of 84.8% and 72.4% for mAP50 and mAP50–95, respectively.

To further improve model accuracy, a fine-tuning stage was performed using 50 epochs, an image size of 1280 × 1280 pixels, and a batch size of 16. This resulted in improved performance, achieving mAP50 and mAP50–95 values of 88.0% and 79.3%, respectively. Figure 4 presents examples of tomato fruit detection obtained with the trained model under two different scenarios: a controlled environment without occlusions (Figure 4a) and a field condition in which occlusions are present (Figure 4b). For each detected fruit, four pieces of information are displayed: its identification number (ID), size category, ripeness stage, and detection confidence score. For example, the label "*id:1 b_half_ripened 0.81*", shown in Figure 4a, indicates that the fruit has an ID of 1, belongs to the large size category (big), is classified as half-ripened, and was detected with a confidence score of 81%.

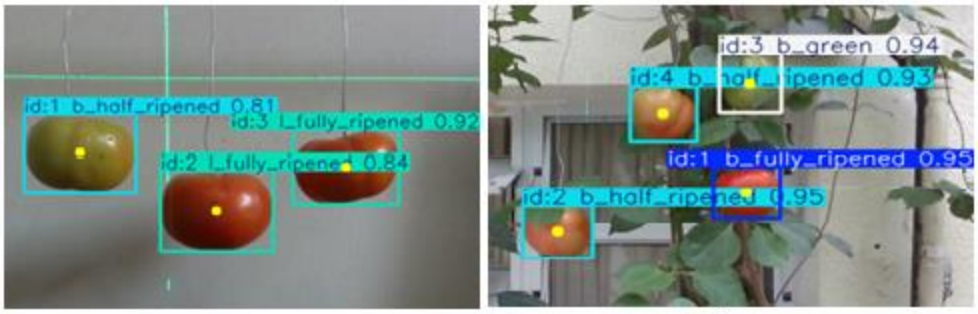

Fig. 4. Detection of tomato fruits in a controlled environment (a) and in a field environment (b)

The dataset used to train and validate the model is called *Laboro Tomato*, obtained from Kaggle. It consists of 804 images, of which 643 were used for training and 161 for validation. Each tomato is classified into two size categories (*regular-sized tomato* and *cherry tomato*) and three ripeness stages: *fully ripe*, *half-ripened*, and *green*. Additional information about the dataset can be found at the following link: https://www.kaggle.com/datasets/nexuswho/laboro-tomato.

Depth Estimation

In some of the methods described in the previous section, object depth is estimated using the center point of the bounding box, as in the approach presented by Andriyanov (2022). However, this strategy may introduce depth estimation errors. Under ideal conditions, the center of the detected object should coincide with the center of the bounding box. In practice, however, the bounding box often does not perfectly fit the detected object, resulting in an inaccurate estimate of the object's center position (Suwandi et al., 2022). The proposed depth estimation method uses the k-means algorithm to segment the foreground depth region of the detected object and subsequently calculate the average depth of that region. This approach improves depth estimation accuracy because it relies on a specific sampling region rather than a single point defined by the center of the bounding box. Consequently, the method reduces depth errors associated with bounding-box inaccuracies and surface irregularities of the detected objects.

After object detection by the neural network, the coordinates of the detected object's bounding box are mapped onto the depth frame, thereby defining a Region of Interest (ROI) within the depth image. The k-means algorithm is then applied to this ROI.

Among the techniques used to organize and interpret data acquired from RGB-D cameras, the k-means algorithm stands out due to its simplicity and efficiency in segmenting points or regions with similar characteristics. The k-means is a partition-based clustering method whose fundamental principle is to divide data into clusters, where each point belongs to the cluster whose centroid is nearest (Sidorova, 2022). The algorithm iteratively minimizes a cost function and converges-to a local minimum. Euclidean distance is used as the dissimilarity metric to determine the distance between each pixel and the centroid of each cluster (Baid et al., 2017).

In the proposed approach, the *k-means* algorithm is used to identify and separate regions with different depth values. For an object such as a tomato, depth may vary across different areas of the fruit surface due to its curved geometry. The foreground corresponds to the region with the shortest distance between the fruit surface and the camera. Figure 5 illustrates the detected fruit, showing the corresponding RGB image crop (Figure 5a) and the result of applying *k-means* clustering with $k = 5$ to the depth map (Figure 5b), where the gray-colored region corresponds to the foreground layer.

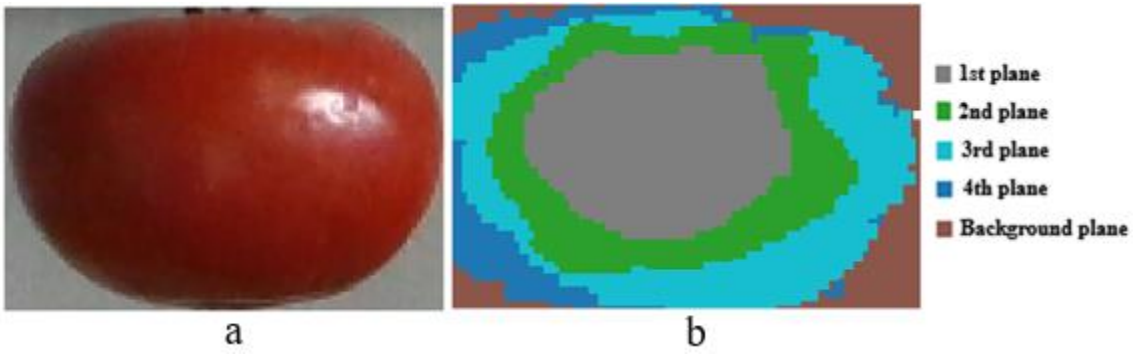

Fig. 5. In the first image (a), the Region of Interest (ROI) extracted from the color frame of the detected fruit is shown. In the second image (b), the same ROI is mapped onto the depth frame and segmented according to depth values after applying the k-means algorithm with *k* = *5* (five segmented clusters)

After applying the k-means algorithm, the average depth of each cluster is computed using the depth values of the pixels obtained from the depth map generated by the camera. Subsequently, a mask is applied (Figure 6) to retain only the region corresponding to the cluster with the smallest average depth value.

The equation used to calculate the average depth of each cluster is given by:

$$d_m = \frac{1}{N} \sum_{i=1}^{N} d(x_i\ ,\ y_i) \tag{1}$$

where *N* is the number of pixels within the cluster, x_i and y_i are the coordinates of each pixel in the depth frame, and $d(x_i\ ,\ y_i)$ represents the depth value of each pixel belonging to the selected cluster, obtained directly from the depth frame.

To determine the actual distance between the camera's front glass surface and the fruit surface, it is necessary to apply a correction to the depth values before computing d_m. For the Intel RealSense D435 camera, this correction is performed by adding 4.2 mm to the measured depth values. This adjustment is required because the depth values provided by the camera are not

referenced to the front glass surface of the camera, which is typically considered the physical origin for distance measurements. Instead, the camera's internal coordinate system is defined relative to the optical center of the depth sensor, resulting in a small offset that must be compensated for when calculating real-world distances.

$$D(x_i\ ,\ y_i)\ = d(x_i\ ,\ y_i)\ + 4.2 \tag{2}$$

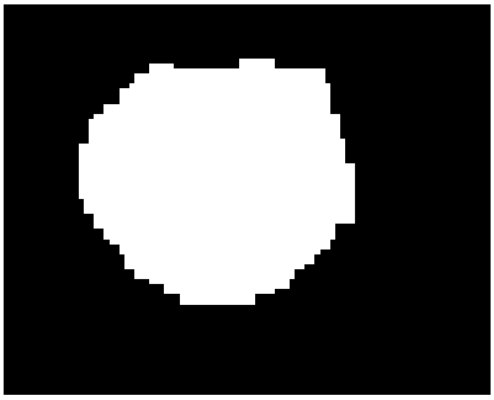

Figure 6. Mask applied to filter the cluster with the lowest mean depth value in the depth map shown in figure 5B

A gradient-magnitude threshold-based filter is then applied to the selected cluster in order to retain only the regions considered planar, since the curved geometry of tomato fruits introduces depth variations across their surfaces. First, the gradients along the *x* and *y* directions of the ROI (Region of Interest) are computed. Subsequently, the gradient magnitude is calculated. Mathematically, the gradient magnitude can be expressed as:

$$\boldsymbol{\nabla f_x} = \frac{\partial D(x_i\ ,y_i)}{\partial x} \tag{3}$$

$$\boldsymbol{\nabla f_y} = \frac{\partial D(x_i\ ,y_i)}{\partial y} \tag{4}$$

$$G_{(x_i,y_i)} = \sqrt{\boldsymbol{\nabla f_x}^2 + \boldsymbol{\nabla f_y}^2} \tag{5}$$

where ∇f_x and ∇f_y are the depth gradients along the *x* and *y* directions, respectively, and $G_{(x_i,y_i)}$ represents the gradient magnitude at pixel (x_i, y_i).

Subsequently, a threshold is defined as a function of the mean (μ) and standard deviation (σ) of the gradient magnitude values. This threshold is computed according to the following equation:

$$T = \mu_{(G)} + A.\sigma_{(G)} \tag{6}$$

where *A* is a filter adjustment coefficient. The larger the value of *A*, the more permissive the filter becomes, allowing regions with greater surface variation to be retained. Conversely, smaller values of *A* make the filter more selective, preserving only regions that are closer to being planar.

The filtering process consists of retaining only those pixels whose gradient magnitude (*G*) is lower than the defined threshold (*T*). Figure 7 presents the filter mask (Figure 7a) and the result obtained using *A* = 0.01 (Figure 7b), demonstrating that only regions classified as planar are retained.

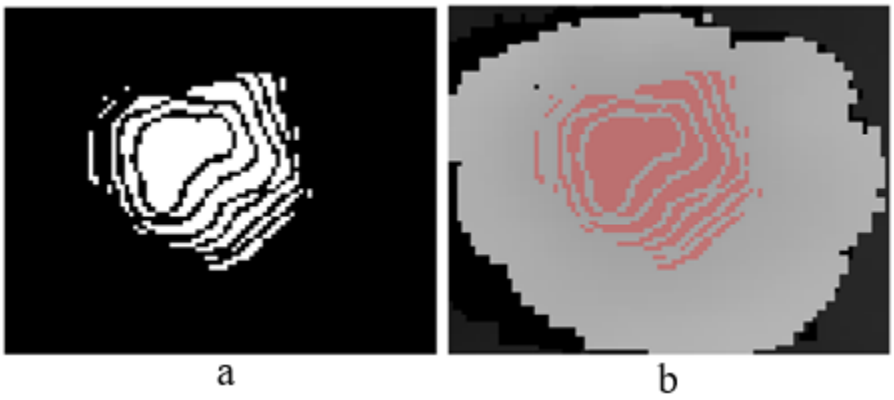

Fig. 7. Filter mask (a) and the result of its application to the selected cluster (b)

Using the depth elements obtained after the filtering process, the average depth is calculated once again using Equation (1). This refined depth value is then used to estimate the position and dimensions of the tomato fruit.

Fruit Position and Size Estimation

As previously mentioned, Intel RealSense cameras employ stereo vision to estimate scene depth. The underlying principle of stereo vision is the use of two images of the same scene captured from different viewpoints to infer depth information. To achieve this, the technique relies on two identical sensors that are physically separated and simultaneously observe the scene.

This configuration enables the mapping of image coordinates 2D to scene coordinates 3D, and vice versa. Consequently, the 3D position of the fruit can be estimated using the following equations:

$$X = \frac{Z}{f_x}(u_l - px) \tag{7}$$

$$Y = \frac{Z}{f_y}(v_l - py) \tag{8}$$

$$z = \frac{b}{(u_l - u_r)} f_x \tag{9}$$

where $(u_r,\ u_r)$ and $(u_l,\ u_l)$ are the pixel coordinates of a point in the 2D image plane of the right and left infrared cameras, respectively; (b) is the baseline distance between the two infrared cameras; f_x and f_y are the focal lengths; and (p_x, p_y) is the coordinates of the principal point (in pixels), defined as the point where the optical axis intersects the image plane.

The focal lengths and principal point coordinates are intrinsic camera parameters and can be

readily obtained using the OpenCV library. Equation (9) is internally employed by the Intel RealSense D435 camera to compute the depth value of each pixel in the depth map. Therefore, for the estimation of the X and Y coordinates, the variable Z is replaced by the average depth value d_m, obtained after applying the filtering procedure described in the previous section.

The Y value must be adjusted to account for the height at which the camera is positioned. Assuming the camera is mounted 800 millimeters above the ground, the corrected Y coordinate can be expressed as:

$$\mathrm{Y} = 800 - \frac{Z}{f_y}(v_l - py) \tag{10}$$

To estimate the fruit dimensions, after the fruit is detected by the neural network and its bounding box is mapped on to the depth frame, the k-means algorithm with (k = 2) is applied to the corresponding ROI. The purpose of this step is to recover the fruit shape from the depth values measured across its surface.

Subsequently, the OpenCV *findContours* method is employed. This method is based on the Suzuki–Abe algorithm for tracing object boundaries (contours) in a binary image (Baid et al., 2017). The use of this technique enables the extraction of all object contours present within the ROI. The largest contour is then selected, providing the geometric representation of the fruit in the image plane, as illustrated in figure 8.

To calculate the fruit width and height, the selected contour is enclosed by a bounding box, and the corresponding bounding-box dimensions (h, w), expressed in pixels, are extracted. This procedure assumes that the fruit's vertical axis is always parallel to the vertical axis of the camera, meaning that the fruit is not tilted with respect to the image plane. Under this assumption, the fruit width and height can be directly estimated from the dimensions of the bounding box.

For tilted fruits, it would be necessary to employ a YOLO-OBB (*You Only Look Once – Oriented Bounding Box*) model (Cong et al., 2025), capable of performing oriented object detection and estimating the object's inclination angle. This information would then allow the correct computation of the fruit dimensions using the proposed method.

Fig. 8. Approximate geometric shape of the detected fruit obtained using the OpenCV *findContours* method applied to the depth map

To obtain the fruit width and height in millimeters, we use the equation that relates the size of an object in the image plane (in pixels) to its actual size in the scene (in millimeters). According to Grunnet-Jepsen et al. (2022), this relationship is given by:

$$w = \frac{z.w_i}{f_x} \quad (11)$$

where *w* is the actual width of the object in the scene, w_i is the corresponding object width in the image plane, z is the object depth, and f_x is the focal length along the x axis. The same equation was used to calculate the actual height (h) of the object by replacing the corresponding parameters in Equation (11), namely h_i and f_y.

Results

An experiment was conducted to validate and evaluate the position and size estimation method proposed in this work. To perform this evaluation, several conditions were established to enable the proper assessment of the method. In particular, it was assumed that the camera coordinate system and the world coordinate system coincide. To satisfy this assumption, the origin of the camera coordinate system was aligned with the origin of the world coordinate system.

As a reference for the world coordinate system, two mutually orthogonal laser beams representing the *X* and *Y* axes were projected onto a wall using a laser level device (Figure 9). The device is equipped with a self-leveling mechanism that ensures accurate alignment, providing a vertical and horizontal leveling accuracy of ±1.5 mm over a distance of 5 m.

Fig. 9. Materials used in the experimental setup

The experiment was conducted using seven tomato fruits under two different scenarios: a controlled environment, characterized by controlled illumination and the absence of leaf or branch occlusions, and a field environment, where such elements were present. To simulate real-world field conditions, a tomato plant was used in the field scenario. In both environments, the proposed algorithm was able to detect tomato fruits reliably at distances shorter than 1 m. Furthermore, the method successfully estimated both the position and size of the fruits in the two scenarios.

To determine measurement errors, the estimated results were compared with ground-truth position and size values obtained through manual measurements using a conventional tape measure, a laser distance meter, and a digital caliper, with accuracies of ±2 mm and ±0.02 mm, respectively. Measurements were performed carefully, taking as reference the distance between

the front glass plane of the left infrared camera and the central surface of the fruit when evaluating depth. For the spatial coordinates X and Y axes, measurements were obtained relative to the fixed reference coordinate system established in the experimental environment.

Tables 1 and 2 present the ground-truth and estimated values of the 3D positions and fruit dimensions, respectively, together with the corresponding errors expressed in millimeters.

Table 1. Ground-truth and estimated coordinates (mm).

Controlled Environment										Field Environment								
Nº	Estimated Values			Ground-Truth Values			Errors			Estimated Values			Ground-Truth Values			Errors		
	X	Y	Z	X	Y	Z	E_x	E_y	E_z	X	Y	Z	X	Y	Z	E_x	E_y	E_z
1	-77.7	844	364	-78	850	365	0.3	5.6	0.3	-45.4	851	453	55	860	450	9.5	8.8	3.8
2	80.0	805	378	75	805	378	5.0	0.2	0.5	91.4	916	398	83	910	406	8.4	6.7	7.9
3	151.8	830	389	14	830	394	2.8	0.3	4.6	145	835	392	140	831	401	5.4	4.5	8.2
4	-79.0	824	324	-76	828	327	3.0	4.0	3.0	73.7	780	280	104	761	335	30.3	19.0	55
5	-4.80	791	320	0	798	322	4.8	6.4	1.9	57.0	921	715	49	939	721	8.0	18.0	6.0
6	67.9	820	344	72	823	341	4.1	2.8	3.0	142,2	955	716	141	961	721	1.6	5.6	5.0
7	123.7	781	340	13	786	343	6.3	4.1	2.5	-49.9	783	737	-58.0	792	742	8.1	8.9	4.7
RMSE							4.1	4.0	2.6							13.3	11.5	21.5

Table 2. Ground-truth and estimated fruit dimensions (mm).

Controlled Environment							Field Environment					
Nº	Estimated Values		Ground-Truth Values		Errors		Estimated Values		Ground-Truth Values		Errors	
	h	*w*	*h*	*w*	E_h	E_w	*h*	*w*	*h*	*w*	E_h	E_w
1	46.3	68.6	50.0	69.9	3.7	1.3	46.4	65.7	47.1	66.4	0.7	0.7
2	53.6	73.7	55.9	74.7	2.3	1.0	60.9	59.4	54.7	77.1	6.2	17.7
3	47.5	69.5	48.8	69.5	1.3	0.0	45.3	52.2	51.1	69.1	5.8	16.9
4	51.3	68.0	51.9	67,9	0.6	0.1	47.2	79.3	51.4	69.2	4.2	10.1
5	58.2	65.8	62.5	72.4	4.3	6.6	50.6	70.9	52.2	65.6	1.6	5.3
6	66.4	57.0	70.5	57.7	4.1	0.7	55.6	52.7	68.2	60.0	12.6	7.3
7	52.4	53.1	55.0	62.3	2.6	9.2	41.9	69.3	62.2	73.5	20.3	4.2
RMSE					3.0	4.3					9.7	10.6

Tables 1 and 2 demonstrate that the proposed algorithm achieved high performance in estimating the 3D position and dimensions of tomato fruits in the controlled-environment scenario, with Root Mean Square Error (RMSE) values below 5 mm. However, in the field environment, a significant increase in error was observed due to the presence of leaf and branch occlusions, as illustrated

in figure 10a for Tomato No. 4, resulting in RMSE values exceeding 10 mm.

Figure 10b presents an example of the errors produced by the proposed method under severe occlusion conditions. In this case, the occlusions led to incorrect clustering of the depth data, which consequently affected the accuracy of both the position and size estimations of the fruit.

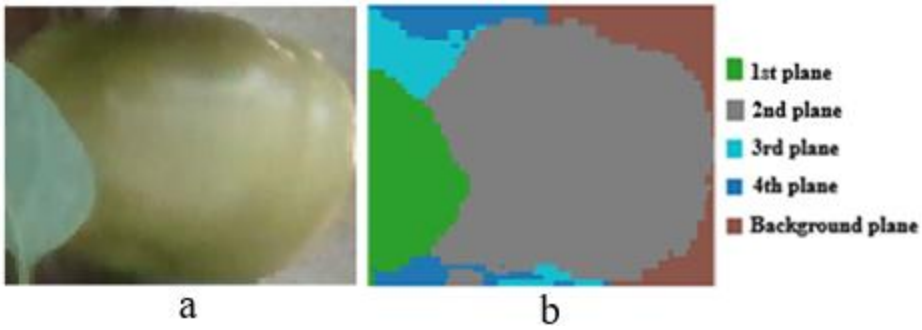

Fig. 10. Occlusion of Tomato No. 4 by a leaf (a); selection of the leaf as the foreground cluster of Tomato No. 4 (b)

Despite the errors observed in the field scenario, the obtained results can still be considered satisfactory, since the proposed method outperformed some of the approaches reported in the literature. Furthermore, because none of the previously discussed studies evaluated their methods under significant occlusion conditions caused by unwanted objects, it is more appropriate to compare their results with those obtained in our controlled-environment experiments.

For instance, Suwandi et al. (2022) reported errors of 69.53 mm, 46.11 mm, and 65.77 mm along the X, Y, and Z axes, respectively. Although their application involved a moving object, the use of the color image to determine the central region of the ball makes the method more susceptible to illumination-related errors, potentially affecting depth estimation accuracy.

In the work of Mengoli et al. (2022), which focuses on fruit size estimation, relatively good results were achieved, with errors ranging from 7 to 9 mm. However, the proposed approach relies on the Hough Transform to detect the circle enclosing the fruit. This technique is prone to oversizing errors, which may reduce the accuracy of fruit dimension measurements.

The results reported by Andriyanov (2022) are very close to those obtained by the proposed method. In some cases, the reported errors along the X and Y axes are lower by less than 3 mm. However, our method achieved substantially better performance along the Z axis, with an improvement of approximately 13 mm. While Andriyanov's approach estimates depth using a single image point (x, y), the proposed method selects a region of interest and computes its average depth. As a result, the proposed approach is less sensitive to depth estimation errors associated with the selection of individual pixels.

On the other hand, the computational performance of the proposed algorithm was evaluated by measuring the processing times of the main stages of the pipeline, namely image acquisition, object-detection inference, and 3D coordinate and size estimation. The experimentally obtained average latencies are presented in Table 3.

Table 3. Average processing latency for each stage of the proposed algorithm.

Processing Stage	Average Time (s)
Image acquisition	0.0075
Inference (YOLOv8s)	0.9855
Coordinate and Size Estimation	0.0205
Total Time	1.0136

The results show that the total processing time per frame is approximately 1.01 seconds (less than 1 FPS). As shown in Table 3, the inference stage is the primary bottleneck, accounting for approximately 97% of the total processing time, whereas image acquisition and the computation of 3D coordinates and fruit dimensions contribute only minimally to the overall latency. These findings indicate that system performance is mainly constrained by the computational cost of the detection model, particularly due to the absence of GPU acceleration in the hardware platform used. Nevertheless, the remaining processing stages proved to be computationally efficient.

It can therefore be concluded that the proposed system does not satisfy strict real-time requirements. However, it may still be suitable for low-dynamic applications, such as precision agriculture, where a latency of approximately one second can be considered acceptable.

Conclusion

In this study, a computer vision algorithm was developed to estimate the 3D position and dimensions of tomato fruits using an Intel RealSense D435 camera. To improve depth-estimation accuracy on the curved and reflective surfaces of tomato fruits, the k-means algorithm was employed to segment the region corresponding to the shortest depth between the camera and the fruit surface, while a gradient-magnitude-based filter was applied to identify planar regions within the selected area.

The proposed algorithm successfully detected tomato fruits and estimated their 3D positions and dimensions under the different experimental scenarios considered in this work. In the controlled-environment scenario, the method achieved Root Mean Square Errors (RMSEs) of 4.0 mm, 4.1 mm, and 2.6 mm along the X, Y, and Z axes, respectively. For fruit size estimation, RMSE values of 3.0 mm and 4.3 mm were obtained for fruit height and width, respectively. In the field scenario, the corresponding RMSE values for position estimation were 13.3 mm, 11.5 mm, and 21.5 mm along the X, Y, and Z axes, while fruit height and width were estimated with RMSEs of 9.7 mm and 10.6 mm, respectively.

Future work will focus on improving performance under field conditions by adopting a cultivation system based on V-shaped trellises, similar to the approach proposed by Yoshida et al. (2022). This cultivation arrangement can provide a more favorable viewing angle for the vision system, reducing significant occlusions when the camera is positioned below the fruit level and oriented upward.

The computational performance analysis revealed that the main limitation of the proposed system is the inference time of the YOLOv8 model, which is highly dependent on the available hardware resources, particularly the lack of GPU acceleration. Potential strategies for improving

performance include the use of GPU-based processing, the adoption of lighter detection models, and image resolution reduction.

Acknowledgments

The authors would like to thank the Brazilian Coordination for the Improvement of Higher Education Personnel (CAPES) for the financial support provided through a graduate scholarship, the Laboratory of Statistical Signal Processing & Inverse Problems (LASSIP) at UTFPR for providing the camera used in the experiments, and the Federal University of Technology – Paraná (UTFPR) for its institutional support.

References

Andriyanov, N. (2022). Estimating object coordinates using convolutional neurais networks and Intel RealSense D415/D455 depth maps. In *Proceedings of the 2022 VIII International Conference on Information Technology and Nanotechnology (ITNT)* (pp. 1–4). IEEE.

Baid, U., Talbar, S., & Talbar, S. N. (2017). Comparative study of K-means, Gaussian mixture model, and fuzzy C-means algorithms for brain tumor segmentation. In B. Iyer, S. Nalbalwar, & R. Pawade (Eds.), *International Conference on Advances in Signal Processing (ICASSP/ICMMD) Proceedings* (Vol. 137, pp. 592–597). Atlantis Press.

Cong, V. D., & Phuong, L. H. (2025). Improving robotic grasping accuracy through oriented bounding box detection with YOLOv11-OBB. *Heliyon, 11*, e43512. https://doi.org/10.1016/j.heliyon.2025.e43512

Grunnet-Jepsen, A., Sweetser, J., Khuong, T., Dorodnicov, S., Tong, D., Mulla, O., Eliyahu, H., & Raikhel, E. (2022). *Intel RealSense self-calibration for D400 series depth cameras* (Rev. 2.7). Intel Corporation.

Jamundá, T. (2000). *Shape recognition: The Hough transform*. Computer Vision Seminar, Graduate Program in Computer Science, Federal University of Santa Catarina (UFSC).

Mengoli, D., & Bortolotti, G. (2022). On-line real-time fruit size estimation using a depth-camera sensor. In *2022 IEEE Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)* (pp. 86–90). IEEE. https://doi.org/10.1109/MetroAgriFor55389.2022.9964960

Sidorova, D. N. (2022). Description of software for comparative analysis of clustering methods. In *IEEE International Multi-Conference on Engineering, Computer and Information Sciences (SIBIRCON)*. IEEE. https://doi.org/10.1109/SIBIRCON56155.2022.10017038

Suwandi, D. P., Kurnianto, I., & Bachtiar, M. M. (2022). Estimation of ball position using depth camera for middle-size goalkeeper robot. In *2022 International Electronics Symposium (IES)*. IEEE. https://doi.org/10.1109/IES55876.2022.9888575

Suzuki, S., & Abe, K. (1985). Topological structural analysis of digitized binary images by border following. *Computer Vision, Graphics, and Image Processing, 30*(1), 32–46.

Vo-Le, C., Muoi, P. V., Son, N. H., San, N. V., Duong, V. K., & Huyen, N. T. (2021). Automatic method for measuring object size using 3D camera. In *Proceedings of the 2020 IEEE 8th International Conference on Communications and Electronics (ICCE)* (pp. 365–369). IEEE. https://doi.org/10.1109/ICCE48956.2021.9352115

Vijay, S., & Ponnusamy, V. (2023). A review on application of robots in agriculture using deep learning. *AIP Conference Proceedings, 2946*, 050005. https://doi.org/10.1063/5.0177964

Yoshida, T., Kawahara, T., & Fukao, T. (2022). Fruit recognition method for a harvesting robot with RGB-D cameras. *ROBOMECH Journal, 9*, Article 15. https://doi.org/10.1186/s40648-022-00230-y