



OPEN

Deep learning approach for detecting tomato flowers and buds in greenhouses on 3P2R gantry robot

Rajmeet Singh^{1,2,3}, Asim Khan^{2,3}, Lakmal Seneviratne^{1,2} & Irfan Hussain^{1,2}✉

In recent years, significant advancements have been made in the field of smart greenhouses, particularly in the application of computer vision and robotics for pollinating flowers. Robotic pollination offers several benefits, including reduced labor requirements and preservation of costly pollen through artificial tomato pollination. However, previous studies have primarily focused on the labeling and detection of tomato flowers alone. Therefore, the objective of this study was to develop a comprehensive methodology for simultaneously labeling, training, and detecting tomato flowers specifically tailored for robotic pollination. To achieve this, transfer learning techniques were employed using well-known models, namely YOLOv5 and the recently introduced YOLOv8, for tomato flower detection. The performance of both models was evaluated using the same image dataset, and a comparison was made based on their Average Precision (AP) scores to determine the superior model. The results indicated that YOLOv8 achieved a higher mean AP (mAP) of 92.6% in tomato flower and bud detection, outperforming YOLOv5 with 91.2%. Notably, YOLOv8 also demonstrated an inference speed of 0.7 ms when considering an image size of 1920 × 1080 pixels resized to 640 × 640 pixels during detection. The image dataset was acquired during both morning and evening periods to minimize the impact of lighting conditions on the detection model. These findings highlight the potential of YOLOv8 for real-time detection of tomato flowers and buds, enabling further estimation of flower blooming peaks and facilitating robotic pollination. In the context of robotic pollination, the study also focuses on the deployment of the proposed detection model on the 3P2R gantry robot. The study introduces a kinematic model and a modified circuit for the gantry robot. The position-based visual servoing method is employed to approach the detected flower during the pollination process. The effectiveness of the proposed visual servoing approach is validated in both un-clustered and clustered plant environments in the laboratory setting. Additionally, this study provides valuable theoretical and practical insights for specialists in the field of greenhouse systems, particularly in the design of flower detection algorithms using computer vision and its deployment in robotic systems used in greenhouses.

Keywords Tomato flower, Bud, Deep learning, YOLO, Transfer learning, Pollination robot

In modern agriculture, greenhouse farming assumes a pivotal role, facilitating controlled and efficient crop cultivation. However, the shortage of natural pollinators like bees presents a significant barrier to achieving successful pollination in greenhouse environments. The controlled conditions of greenhouses require effective crop pollination, a task that cannot solely rely on natural pollinators. Despite attempts to boost honeybee populations, the rate of growth has been insufficient to meet rising demand, resulting in pollination deficits and increasing costs for pollination services^{1,2}. The tomato (*Solanum lycopersicum*) emerges as a globally significant and extensively cultivated crop³. Pollination, the transfer of pollen from stamens to pistils, is essential for seed and fruit development. Unlike some plants that depend on cross-pollination, tomatoes can self-pollinate within a single flower once pollen is available. In greenhouse tomato cultivation, the most common pollination methods are: insect pollination and manual flower vibration for artificial pollination. However, ensuring effective insect

¹Department of Mechanical Engineering, Khalifa University, Abu Dhabi, United Arab Emirates. ²Khalifa University Center for Robotics and Autonomous Systems (KUCARS), Khalifa University, Abu Dhabi, United Arab Emirates. ³These authors contributed equally: Rajmeet Singh and Asim Khan. ✉email: irfan.hussain@ku.ac.ae

pollination can be challenging, particularly in greenhouse control (hot and humid) environment when insect (bee) activity decreases, leading to reduced pollination efficiency. Consequently, artificial methods like manual flower vibration are often preferred. This approach involves farm workers identifying mature flowers based on visual cues and using vibrating tools for pollination. However, accurately selecting flowers for artificial pollination requires skilled workers, resulting in increased costs in the cultivation process. Figure 1 illustrates the tomato plant flower structure and most used pollination methods in the agriculture sector for tomato fruit pollination.

The decline in pollinator populations has resulted in an economic impact, leading to a heightened demand for pollination services in the agricultural sector specifically in greenhouses^{4–6}. This increased need underscores the requirement for robust solutions. In recent years, the crucial topic of pollination has attracted attention from a range of researchers, small businesses, and startups, evident in the growing number of patents focusing on artificial pollination devices⁷.

The advancement of cutting-edge technologies like robots, visual servoing, and artificial intelligence is pivotal for attaining automated pollination. Visual servoing, a facet of robot control, has developed from the manipulation control of robotic arm manipulators. Robotic pollination offers benefits such as labor cost reduction and improved pollination efficiency. Therefore, there is a need for an accurate detection model that identifies the flower and bud and the location of flowers to enhance tomato production and quality in greenhouses. Traditional methods of flower detection primarily rely on algorithms i.e. image filtering, image fusion, and edge detection to calculate flower features. Extensive research has been conducted on these approaches. For instance, Albadarnel⁸ implemented flower region detection from images and identified species of flower by fusing color and texture features. Aleya et al.⁹ employed the k-means model to distinguish flowers from background and used histogram distribution to detect broken flowers. Dorj¹⁰ effectively detected citrus flowers by reducing noise and illumination effects using a Gaussian filter. Most of these methods assume a single, simple background, making it easier to detect flowers based on color and shape features. However, in the greenhouse's environment, occlusion issues are prevalent, posing challenges to flower detection.

Deep learning (DL) based object detection technology^{11–14} has shown promising results in flower detection. Singh et al.¹⁵ proposed HAAR cascade classifier to detect the different shape of objects using image processing technique for robot path planning. For instance, Chen¹⁶ utilized a faster region-based convolutional neural network (R-CNN) to detect and count strawberry flowers, as well as their ripeness, achieving satisfactory outcomes. Sun et al.¹⁷ proposed an enhanced CNN method specifically for fruit and flower detection. By employing Resnet-50 with residual blocks rather than conventional feature extraction network, they significantly improved the detection accuracy compared to the Faster R-CNN algorithm. Saad et al.¹⁸ introduced a Faster R-CNN model for identifying pepper fruits and flowers, optimizing parameters of the pepper and flower classification and detection. Huang¹⁹ enhanced the original Yolov3 model by incorporating CSPDarknet53 as the backbone. Their focus was on detecting immature apples. Tian²⁰ presented a flower detection model using Single Shot MultiBox Detector (SSD) algorithm. They further improved the convergence rate and accuracy by employing a Gradient Descent (GD) algorithm with the Adam optimization function. Moreover, the existing object detection methods focused solely on flowers, overlooking the detection of buds. The similarity in appearance and color between tomato buds and flowers adds to the complexity of simultaneously detecting both. Additionally, the detection accuracy and speed of image processing are the crucial factors to ensure the efficient operation of a pollination robot in an intelligent greenhouse. Hence, it is essential to carefully select an appropriate deep learning network that can effectively detect both tomato flowers and buds while maintaining accuracy and fast processing capabilities.

The YOLO (You Only Look Once) algorithm strikes a equilibrium between accuracy and speed by utilizing a single convolutional neural network (CNN) that comprehensively analyzes the whole image during both training and testing. Within the Darknet framework, the well-known YOLO model is YOLOv5, which was released in 2020. As of our study in 2024, the most recent iteration of the YOLO series is YOLOv8, which was released in 2023.²¹ employed the YOLOv5 model, which is a single-stage detector, to detect and analyze phenotyping traits of tomato plants. The main objective of the study was to accurately identify nodes and tomato fruits. Multiple tomato genotypes were included in the experiment. However, it's important to note that the study did not specifically focus on detecting tomato buds, which bear a resemblance to tomato flowers. Xu et al.⁶ introduced a modified version of the YOLOv5s model for detecting the different stages of tomato flowering. Their approach

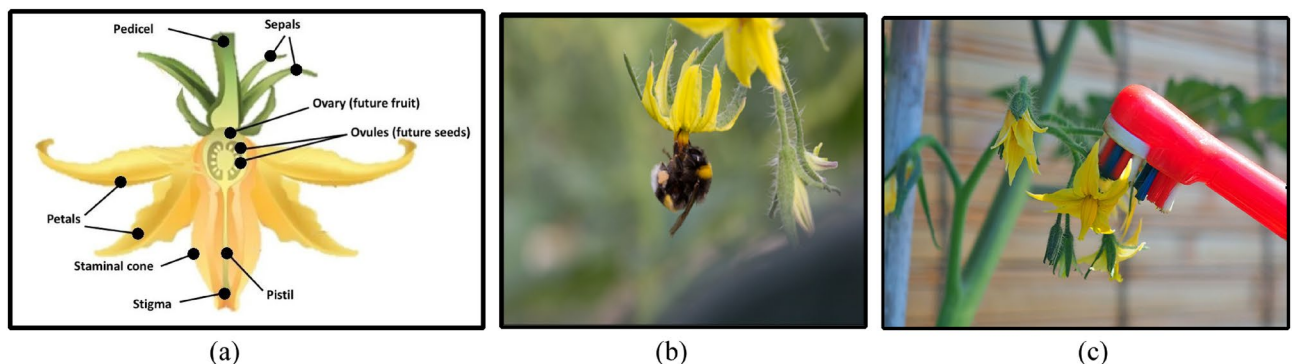


Figure 1. (a) Tomato plant flower structure, (b) insect (bee) pollination, and (c) artificial (vibrator toothbrush) pollination.

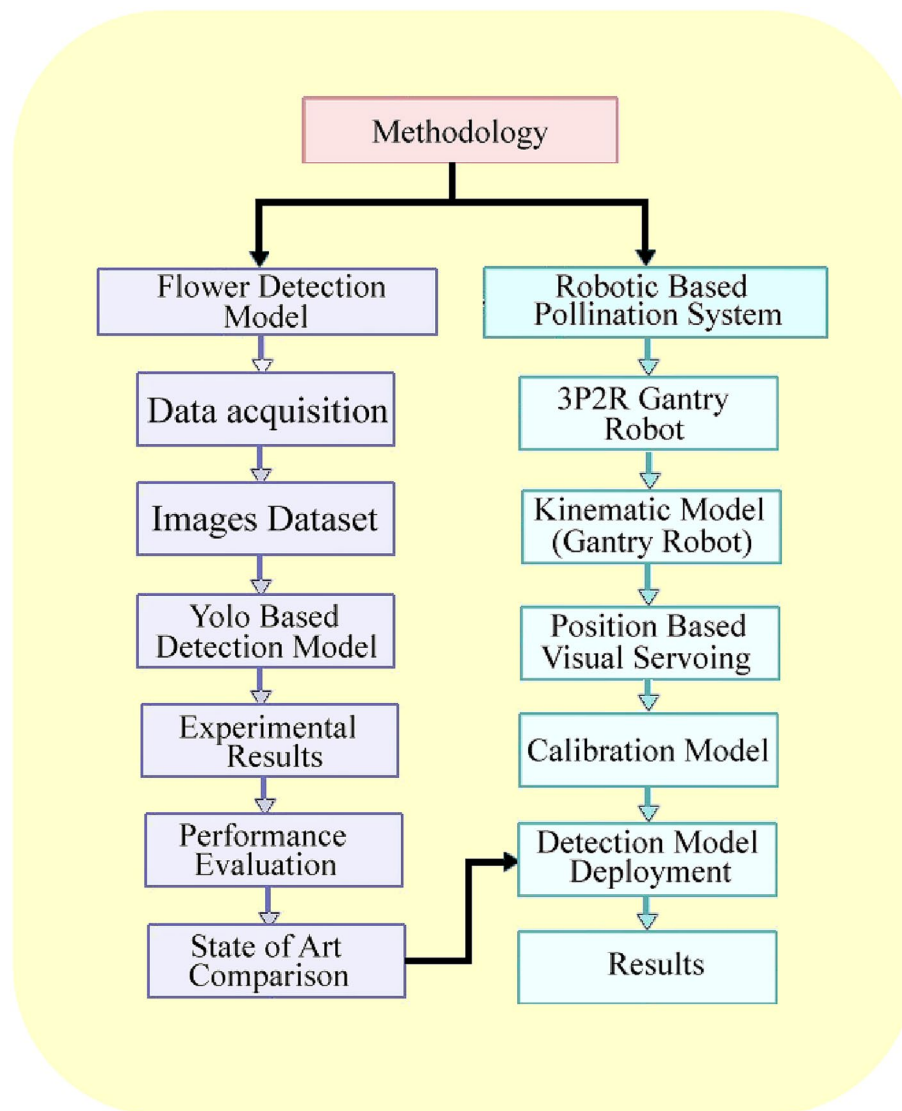


Figure 2. Methodology flow diagram.

Items	Specifications	
Camera	Manufacturer	Intel
	Model	D435i
	FPS	30
	Ideal range	0.3 to 3 m
	Sensor resolution	2 MP
Processor	Manufacturer	Nvidia
	Model	Jetson nano Orin
	CPU	6-core arm cortex
	GPU	1024-core Nvidia
	GPU frequency	625 MHz
Body	Material	ABS 3D printed

Table 1. Data acquisition device specifications.

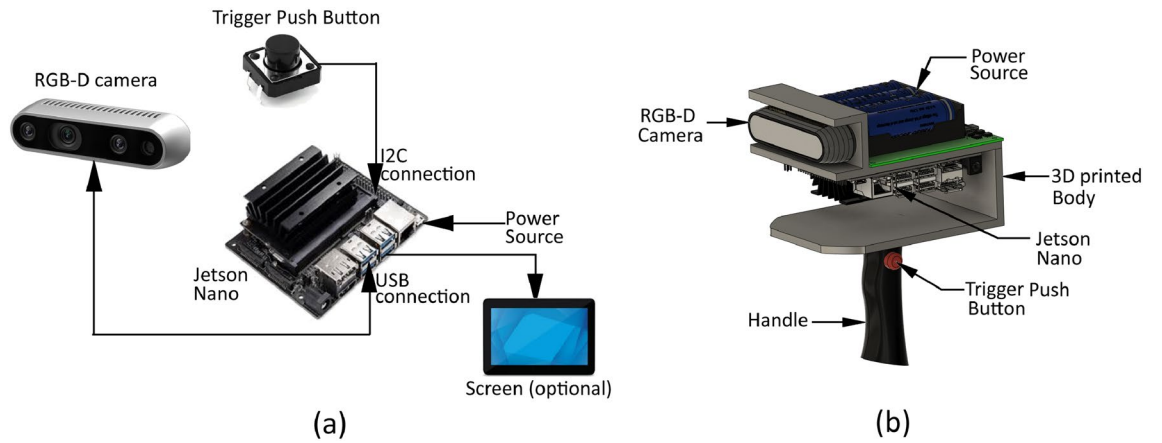


Figure 3. Data acquisition setup (a) schematic diagram, and (b) 3D model of setup.



Figure 4. Examples of tomato flowers images acquired in greenhouse.

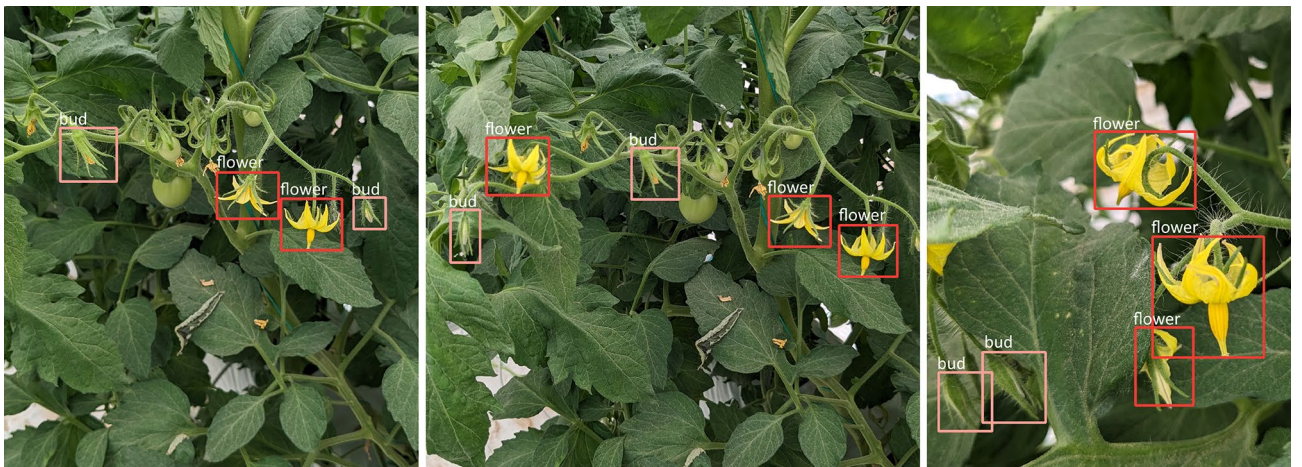


Figure 5. Labeling examples of tomato flower. Flower was labeled as “flower” using red rectangle and Bud was labeled as “bud” using pink rectangle.

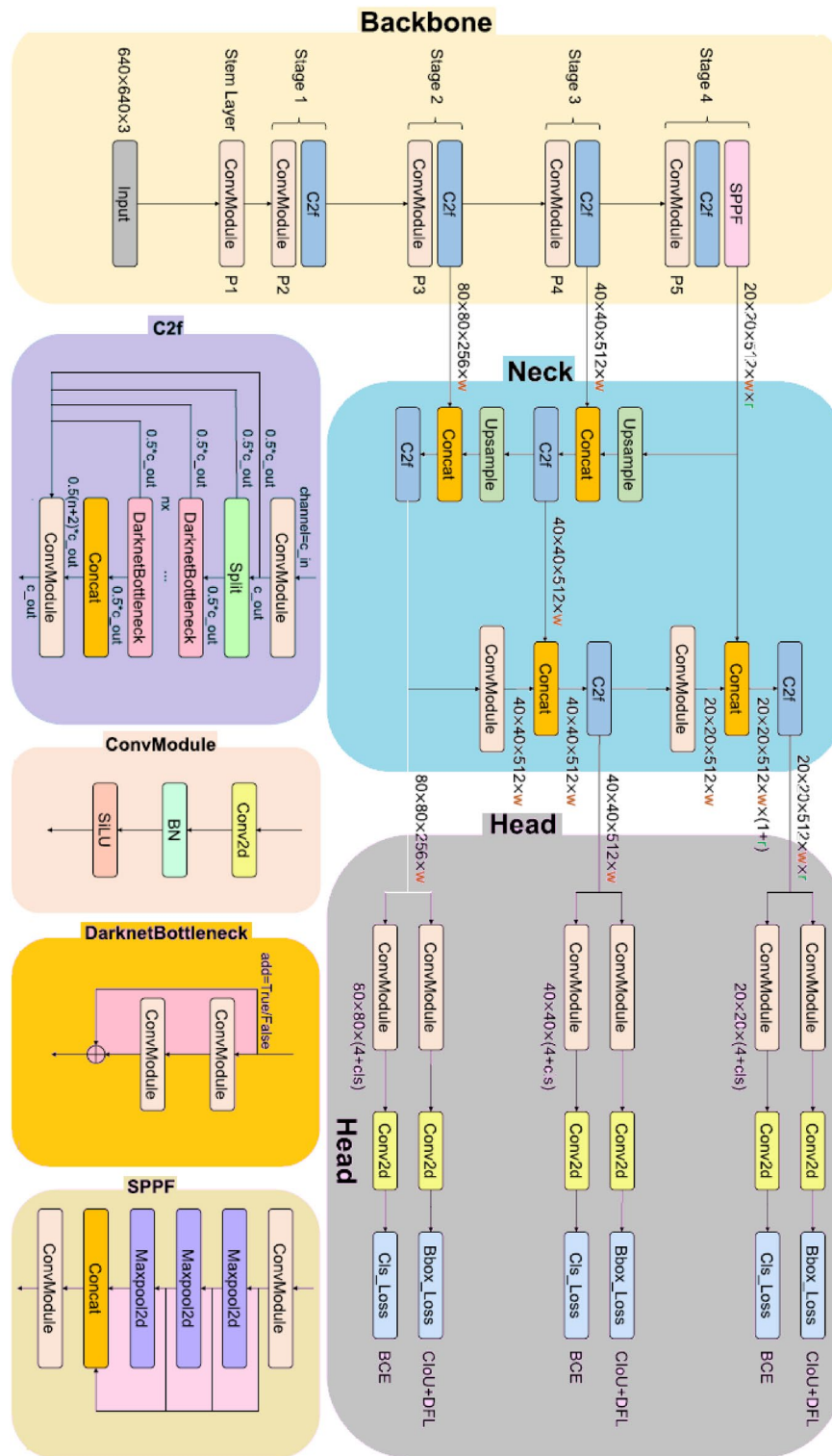


Figure 6. Detailed illustration of YOLOv8 model architecture. The backbone, neck, and head are the three parts of our model, and C2f, ConvModule, DarknetBottleneck, and SPPF are modules.

involved incorporating the Convolutional Block Attention module into the backbone layer of the YOLOv5s model to enhance key performance parameters like mean average precision (mAP) and recall (R). The proposed model achieved an impressive accuracy of 94.2%. However, there is still potential for improving the detection accuracy by leveraging the advancements offered by the latest YOLOv8 model. Zhang et al.²² presented a cascade deep learning network algorithm based on YOLOv5 to address multiple challenges in tomato cultivation.

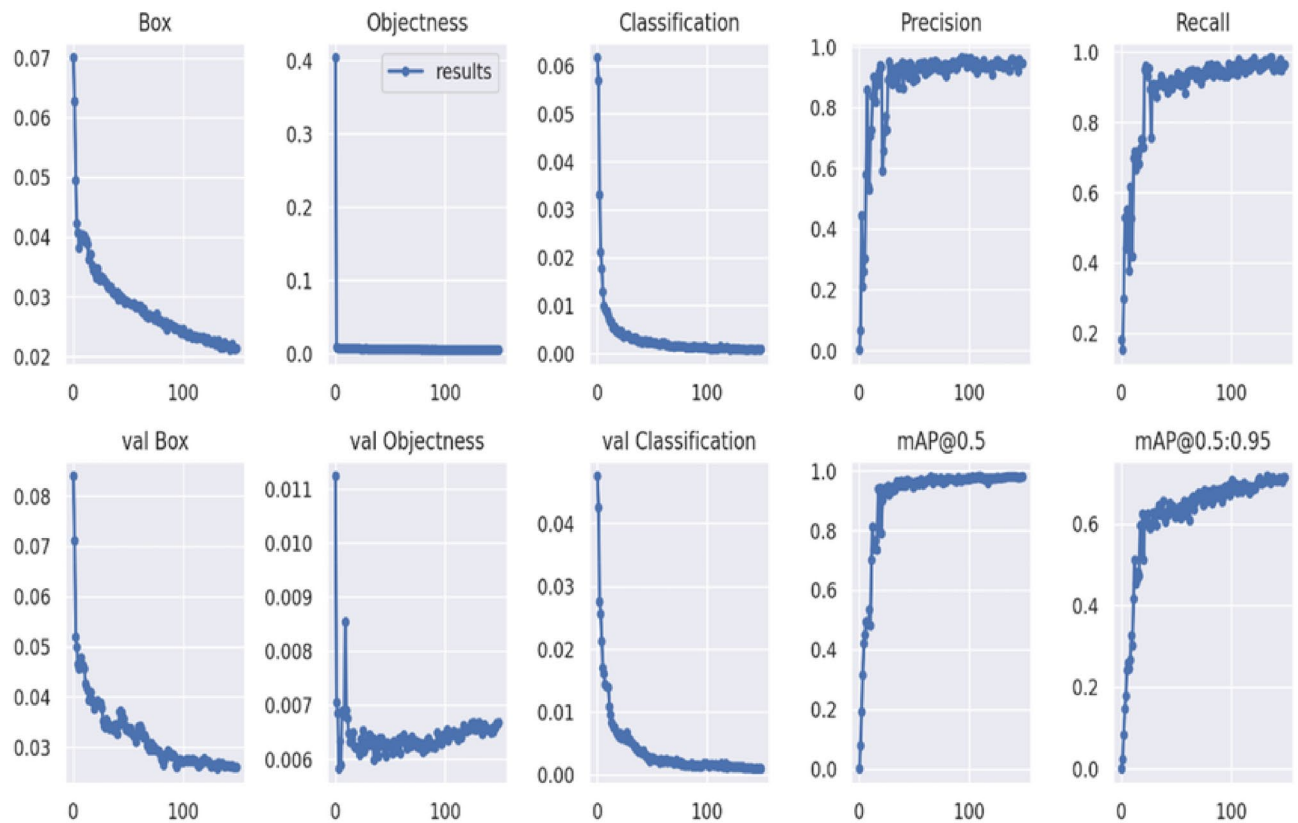


Figure 7. Comprehensive evaluation metrics showcase model performance across key tasks.

Their algorithm aimed to reduce collisions, determine tomato maturity, and estimate 3D poses. However, their study did not encompass the domain of tomato flower detection specifically for robot pollination applications.

Considering the challenges and methodologies, we opted to utilize the YOLOv8m model as the foundation for our detection network. To enhance the model's resilience, we introduced images with varying lighting conditions, specifically captured during morning and evening times, into our dataset. Additionally, we addressed the detection of tomato buds, which bear a striking resemblance to tomato flowers, using the YOLOv8m model, a novel approach not yet documented in existing literature. To facilitate data collection, we developed a custom image capture device that allowed us to gather images of every tomato flower and bud within the greenhouse. Furthermore, we proposed a position based visual servoing model to compute the center of the detected flowers, which would serve as a crucial component in subsequent robot pollination processes.

The study made several contributions, including:

- To overcome the decline in the pollinators (bee) in the greenhouse, the author proposed the robot vision-based pollination approach for tomato flowers.
- To mitigate the impact of lighting on the object detection model, the dataset was augmented with images captured under diverse lighting conditions and at varying times, including both morning and evening. A compact image data capture device was designed and proposed for generating datasets within the greenhouse.
- A comparative analysis was conducted between the latest YOLOv8m model and the YOLOv5m model to assess their state-of-the-art performance and enhance detection accuracy.
- During the training of flower detection model, we considered the detection of tomato buds as well. This decision was driven by the significant similarity between tomato buds and flowers, which could potentially result in improper flower pollination by robots if not properly addressed.
- Authors deployed the proposed detection model on the 3P2R gantry robot and validated it in lab environment for pollination task using artificial tomato plant flowers.

Methodology

Based on the literature gap highlighted earlier, which indicates the decline of natural pollinators like bees in controlled environments such as greenhouses, it is imperative to shift our attention towards alternative solutions, notably robot-based pollination methods. To achieve this goal, the work is divided into two sections: (i) flower detection model and (ii) robotic based pollination system. The overall flow diagram of the methodology is proposed in the Fig. 2. Initially, a data acquisition or capturing device is created and developed, and image datasets are produced for training and validation purposes. The detection model based on the YOLOv8 architecture is introduced and verified using the dataset. A comparison between the proposed detection model and

the state-of-the-art Yolov5m model is conducted. Additionally, this study elaborates on the kinematic model and altered circuit diagram of the 3P2R gantry robot. Subsequently, the proposed detection model is implemented on the gantry robot for pollination purposes.

Detection model

Data acquisition

In Fig. 3, the lab's developed image acquisition setup is depicted. The setup for data acquisition is designed to be compact and highly portable. It includes an RGB-D camera for image capture, a Jetson Nano for efficient processing, a compact power source with a standby battery lasting up to 3 hours, and a trigger button (analogue input) for capturing 1250 (dataset A) images. The device's body is fabricated using 3D printing technology within the lab. The device specifications are mentioned in Table 1. Furthermore, another 300 images (dataset B) were collected from google images. Later, transfer learning method is implemented to test the model performance.

All images captured during the experiment were taken under natural light conditions, encompassing various challenges such as occlusion and overlap. These images were saved in the Portable Network Graphics (PNG) format. Figure 4 provides visual examples of some of the acquired images.

Image datasets

We have divided our dataset into train, validation, and testing sets at 80%, 15%, and 5%, respectively. To facilitate this evaluation, all images in the dataset were manually annotated with rectangular bounding boxes (BB) and labeled as either “flower” or “bud”. The annotations were saved in “txt” format alongside the corresponding image files. A few labeling examples of tomato flowers and buds were shown in Fig. 5.

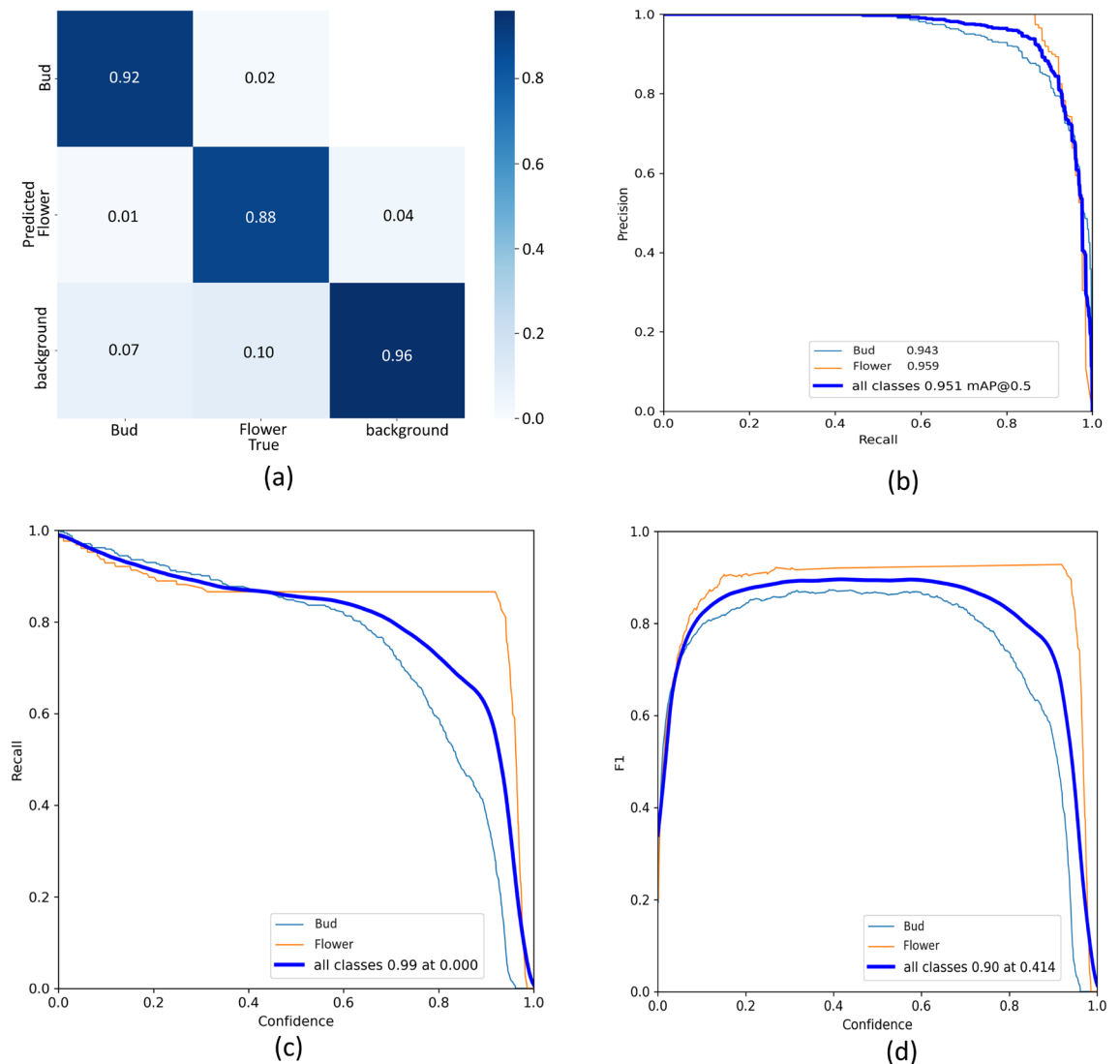


Figure 8. Performance parameter (a) confusion matrix, (b) precision–recall curve, (c) recall–confidence curve, and (d) F1 score curve.

To mitigate overfitting and enhance the generalization capability of the DL model used in this study, data augmentation techniques such as image mirroring and rotation were applied. Image mirroring, including both horizontal and vertical mirroring, was implemented using the “cv2.flip (image, dim)” function from OpenCV in Python. For image rotation, the OpenCV function such as “cv2.getRotationMatrix2D [(cols/2, rows/2), angle, 1]” was implemented. The original image was rotated by specifying the desired angle (90°, 180°, or 270°) through the “angle” parameter of the function.

YOLO network architecture

Numerous versions of YOLO have been created since 2015 in order to facilitate object identification; they include YOLOv1, YOLOv2, YOLOv4, YOLOv5, and YOLOs, among others. Currently, YOLO stands as a prominent object detection model known for its exceptional speed, revolutionizing the field by approaching object detection as a regression problem. This approach allows YOLO to strike an optimal balance between detection accuracy and processing speed, making it a significant advancement in the field of computer vision. Operating as a one-stage network, YOLO partitions an image into regions and predicts boundary boxes, probabilities, and conditional class probabilities. Within the official Darknet framework, the most recent models available are YOLOv8 (released in 2023) and YOLOv7 (released in 2022). For our research, we are implementing YOLOv8 to detect tomato flowers and buds.

Figure 6 displays the YOLOv8 architecture. The company Ultralytics, who also developed the YOLOv5 model, is behind YOLOv8. Compared to YOLOv5, YOLOv8 has several enhancements and modifications to the architecture and development experience. Ultralytics has added many additional upgrades and combinations into the YOLOv8 architecture. First, as YOLOv8 makes use of the CSPDarknet53 feature extractor, its foundation is identical to that of YOLOv5. Some adjustments have been made, such as the replacement of C3 with C2 to enhance detection accuracy by combining contextual information with high-level characteristics. The body converts its initial 6x6 convolution into a 3x3 convolution. The bottleneck's output, which consists of two 3x3 transitions mixed with the remaining connections, is merged in C2f as shown in the Fig. 6, the number of features and the expansion rate are denoted by “f” and “e” respectively. While a block comprising of Conv, BatchNormalisation and a SiLU is presented as CBS. The output from the last bottleneck is utilised in C3.

Second, YOLOv8 uses an anchor-free model with a detachable head to handle object, classification, and regression tasks independently. The total accuracy of the model is improved by this architecture, which lets each branch focus on its specific task. The sigmoid function is used as the activation function for the feature score in the YOLOv8 output layer. This shows how likely it is that there is anything inside the bounding box. The SoftMax function is used to determine the class probability, which describes the chance that an object will belong to a specific class. YOLOv8 provides a semantic segmentation model called the YOLOv8-Seg model for recognition and semantic segmentation of multiple objects. In order to achieve state-of-the-art benchmarks results in these tasks, the model maintains high speed in images and video frames processing. In order to enhance the expected outcomes for tiny objects, the loss functions for binary cross-entropy and bounding box loss are used for classification loss. Fourth, YOLOv8's bottleneck is identical to YOLOv5's, with the exception that the first convolution's kernel size has been altered from 1×1 to 3×3 . As seen in Fig. 6, this modification denotes a change to the ResNet block. A total of five pre-trained model types for object recognition are now available in YOLOv8: YOLOv8n (N), YOLOv8s (S), YOLOv8m (M), YOLOv8l (L), and YOLOv8x (XL). YOLOv8x is the most accurate but slowest of all, whereas YOLOv8s is the small in size and fastest. For our work on implementing the trained model on a robotic platform, we utilized the smaller S-variant of YOLOv5 and YOLOv8. The computational and memory constraints of the robotic system drove the decision to use the smaller S-variants. Deploying the full-sized YOLO models would have resulted in performance issues on the embedded hardware such as latency lag, more inference speed. We achieved efficient inference by selecting the more lightweight S-variants while maintaining the key detection capabilities required for our robotic application.

We utilise the YOLOv8s source code from²³ in our work. Using unique datasets, YOLOv8 enables the optimisation of tomato flower detection models. Tomato dataset serves as the input data for this work. The tomato flowers and buds labels on the dataset images are normalised to a size of 640×640 , while the bounding box is normalised using the format of the yolo in the form of image format. The standard syntax for the bounding box notation in YOLO is (class, x, y, width, height, confidence). The class signifies the title of the object class, and (x, y) indicates the bounding box's centre in relation to the image's dimensions. The box's measurements, in relation to the image's size, are indicated by the width and height. Lastly, the model's confidence in the existence of an object inside the bounding box is indicated by the confidence score.

Model	Training time (h)	Class	Precision	Recall	F1	mAP	Inference time (ms)
YOLOv5s	1.4	All	0.901	0.878	0.881	0.912	0.9
		Bud	0.848	0.845	0.846	0.862	
		Flower	0.952	0.935	0.944	0.962	
YOLOv8s	1.6	All	0.925	0.886	0.894	0.926	0.7
		Bud	0.93	0.894	0.855	0.888	
		Flower	0.958	0.946	0.958	0.977	

Table 2. Metrics results for YOLOv5s and YOLOv8s on validation.

Experimental results

Commonly used basic metrics for object detection are Precision, Recall, and Intersection over Union (IoU). Precision measures the fraction of detected objects that are true positives, while Recall measures the fraction of true positives that are detected. IoU, on the other hand, quantifies the overlap between the predicted bounding box and the ground truth bounding box, with a higher IoU indicating better localization accuracy. Other metrics include the Box loss, Objectness loss, and Classification loss. The Box loss is based on the GIOU (Generalized Intersection over Union) loss function, where a lower value indicates higher accuracy in the bounding box predictions. The Objectness loss represents the target detection loss, with a smaller value corresponding to higher accuracy in object detection. The Classification loss is the mean of the classification loss, and a lower value indicates higher accuracy in the object classification task, as illustrated in Fig. 7.

The network training in this paper utilizes a hardware configuration consisting of an Intel(R) Core (TM) i7-11370@3.30 GHz CPU, 16 GB of RAM, a 1 TB HDD, and an NVIDIA GeForce RTX 3070 GPU. The operating system was Ubuntu, and the PyTorch DL framework was employed. The software tools used include Python 3.9.18, torch-2.1.2+cu118, and OpenCV 3.4.5. Both the YOLOv8m (medium) and YOLOv5m (medium) models accept 640×640 -pixel images as input. The models were trained with an initial learning rate of 0.001, an initial learning rate momentum of 0.90, weight decay of 0.0005, a batch size of 8, and each model was trained for 150 epochs.

Performance evaluation

The trained models were evaluated on the dataset (A) using various evaluation metrics, including precision (P), recall (R), F1 score, average precision (AP), mean average precision (mAP), and detection speed. A commonly used threshold for assessing detection accuracy is an Intersection over Union (IoU) score of over 0.5, which is acceptable. If the IoU score falls below this threshold, the detection is considered unsuccessful or undetected. The IoU score calculates the overlap between the predicted BB and the ground truth BB and is calculated using Eq. (1).

$$IoU = \left| \frac{A \cap B}{A \cup B} \right| \quad (1)$$

where “A” represents the prediction BB, and the “B” represents the true BB. The precision (P) metric measures the relevancy of the detection results, indicating how many of the predicted detections are correct. On the other hand, recall (R) measures the number of truly relevant detections that are successfully retrieved. In Eqs. (2) and (3), P and R are defined, respectively.

$$P = \frac{TP}{TP + FP} \quad (2)$$

$$R = \frac{TP}{TP + FN} \quad (3)$$

The F1 score combines both precision and recall and is calculated using Eq. (4). A flower labeled as class A and correctly detected as class A is considered a true positive (TP). If a flower labeled as class A is mistakenly detected as another class, it is a false negative (FN). Conversely, if a flower is not present but is erroneously detected, it is

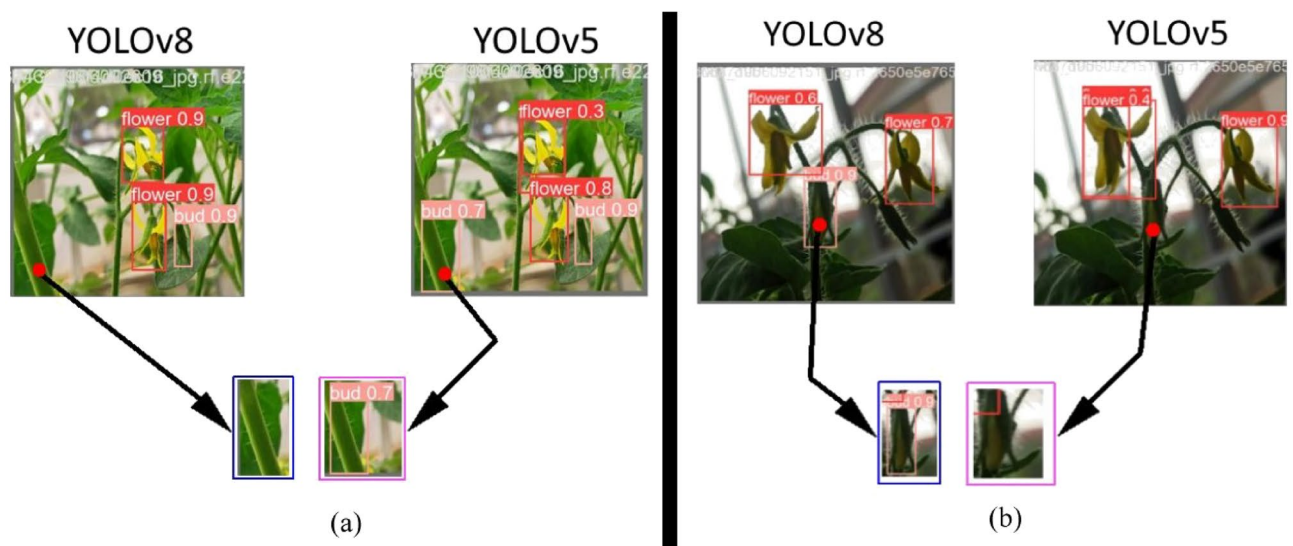


Figure 9. Examples of tomato flower and bud images detected by YOLOv8s and YOLOv5s models. (a) Plant stem is falsely detected as bud by YOLOv5s, and (b) bud is not detected by YOLOv5s and two bounding boxes are drawn for flower detection by YOLOv5s.

a false positive (FP). The F1 score increases when both precision and recall are higher, indicating a better overall performance.

$$F_1 - \text{Score} = \frac{2P * P}{P + R} \quad (4)$$

In this study, the performance of the models in detecting a specific class, such as flowers ($k = 1$), was evaluated using AP_k and mAP metrics. AP_k represents the area below the curve of precision (P_k) plotted against recall (R_k), as defined in Eq. (5). It quantifies the overall detection performance for a particular class. The mAP, defined in Eq. (6), is the average AP across different classes, in this case, the flower class. Higher values of AP and mAP indicate good detection results achieved by the DL model for the object class.

$$AP_k = \int_0^1 P_k(R_k) dR_k \quad (5)$$

$$mAP_k = \sum_{i=1}^k AP_i \quad (6)$$

Confusion matrix

The normalized confusion matrix (Fig. 8a) presents a detailed assessment of the multi-class classification model's performance across the "Bud," "Flower True," and "Background" classes. The diagonal elements of the matrix indicate the model's class-wise accuracy, with the "background" class achieving the highest recognition rate of 96%, followed by the "Bud" class at 92%, and the "Flower True" class at 88%. This suggests the model excels at identifying background instances while experiencing some difficulty consistently distinguishing between the visually similar "Bud" and "Flower True" classes. The off-diagonal elements further reveal the model's misclassification patterns, such as 7% of "Bud" instances being misclassified as "background" and 10% of "Flower True" instances being predicted as "background." These insights can guide targeted refinements to the model architecture and training data to enhance its ability to discriminate between the more challenging "Bud" and "Flower True" classes, ultimately improving the overall classification performance. The comprehensive evaluation provided by the confusion matrix is crucial for developing reliable and robust computer vision models for real-world deployment.

Precision–recall curve

Figure 8b shows a Precision–Recall curve, which is a widely used performance metric for evaluating the performance of classification models. The curve plots the trade-off between precision (the proportion of true positives among all predicted positives) and recall (the proportion of true positives among all actual positives) as the decision threshold is varied. The blue curve represents the Precision–Recall curve for the "Bud" class, the orange curve represents the "Flower" class, and the blue curve labeled "all classes 0.951 mAP@0.5" represents the overall performance across all classes. The "Bud" class has a higher precision–recall performance, with a precision–recall area under the curve (AUC) of 0.943, indicating that the model is generally able to accurately identify "Bud"

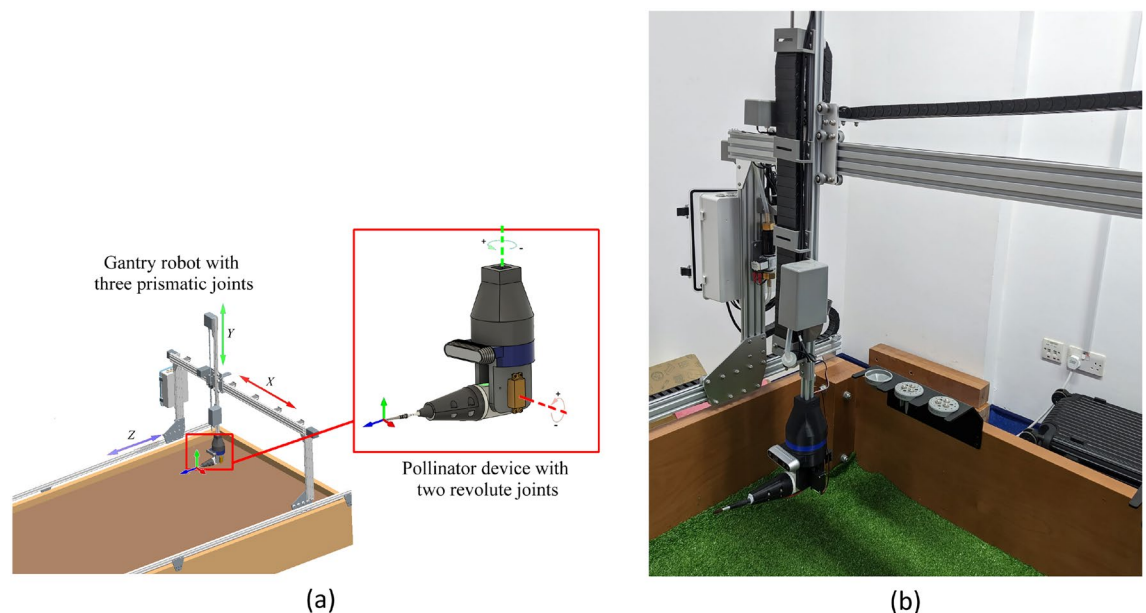


Figure 10. 3P2R Gantry robot with pollination system (a) CAD model, and (b) Prototype model.

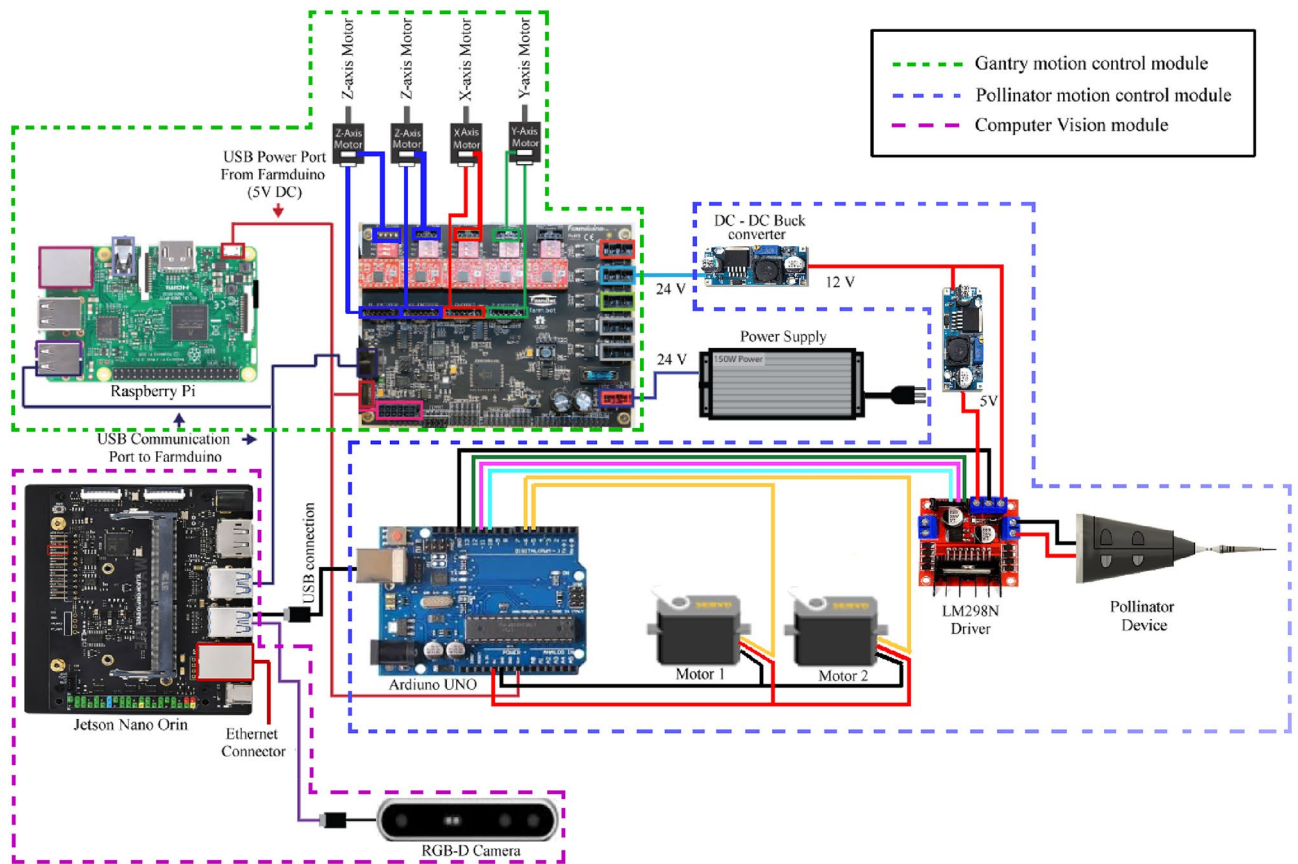


Figure 11. Circuit diagram of 3P2R gantry robot.

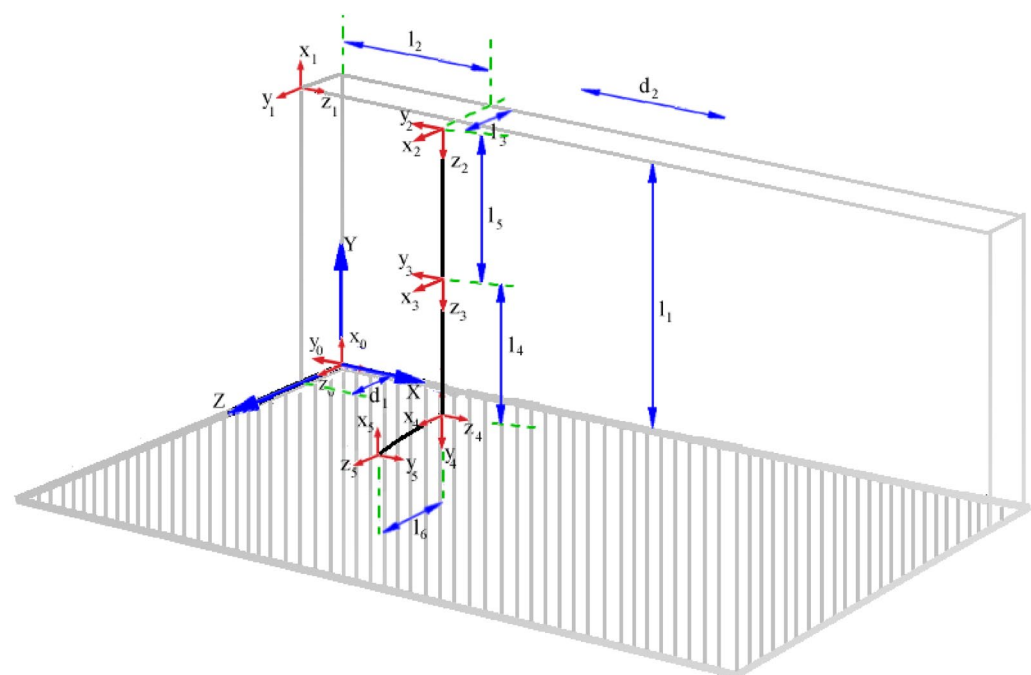


Figure 12. Cartesian system of the gantry robot.

n	θ_n	d_n	a_n	α_n
1	0	l_1	l_1	90
2	90	$l_2 + d_2$	l_3	-90
3	0	l_5	0	0
4	θ_4	l_4	0	-90
5	θ_5	0	l_6	-90

Table 3. 3P2R gantry robot Denavit–Hartenberg (DH) parameters.

instances with high recall. The “Flower” class has a lower precision-recall performance, with a precision-recall AUC of 0.959, suggesting that the model struggles more with identifying “Flower” instances accurately. The overall performance across all classes, as indicated by the blue curve labeled “all classes 0.951 mAP@0.5”, has a precision-recall AUC of 0.951, which is a good overall performance for the model.

The Precision–Recall curve provides valuable insights into the model’s performance, allowing you to understand the trade-offs between precision and recall and select an appropriate decision threshold based on the requirements of the specific application. For example, if high precision is more important, you may choose a higher decision threshold, even if it results in lower recall.

Recall–confidence curve

Figure 8c shows a Recall-Confidence Curve, which is another performance metric that visualizes the relationship between the model’s confidence in its predictions and the recall (the proportion of true positives among all actual positives). The blue curve represents the Recall-Confidence Curve for the “Bud” class, the orange curve represents the “Flower” class, and the blue curve labeled “all classes 0.99 at 0.000” represents the overall performance across all classes. For the “Bud” class, the recall starts very high (close to 1.0) at low confidence thresholds, indicating that the model is able to identify most of the “Bud” instances correctly. As the confidence threshold increases, the recall drops gradually, suggesting that the model becomes more selective in its predictions for the “Bud” class. The “Flower” class exhibits a different pattern, with the recall starting lower and dropping more rapidly as the confidence threshold increases. This implies that the model struggles more with accurately identifying “Flower” instances, especially at higher confidence levels. The “all classes” curve represents the overall recall performance, and it shows a similar trend to the “Bud” class, with a high recall at low confidence thresholds and a gradual decline as the threshold increases. The fact that the “all classes” curve is labeled “0.99 at 0.000” suggests that the model achieves an overall recall of 0.99 when using a confidence threshold of 0.000. The Recall-Confidence Curve provides insights into the model’s ability to balance the trade-off between the number of true positives it can identify (recall) and the confidence it has in its predictions. This information can be useful for selecting an appropriate confidence threshold based on the specific requirements of the application, such as prioritizing high recall or maintaining a certain level of confidence in the predictions.

F1-score

Figure 8d presents an F1-Confidence Curve, a plot showing the relationship between the confidence threshold and the F1-score, a performance metric that combines precision and recall. The F1-score ranges from 0 to 1, with 1 indicating perfect performance. The curve is plotted for two classes, “Bud” and “Flower,” as well as an

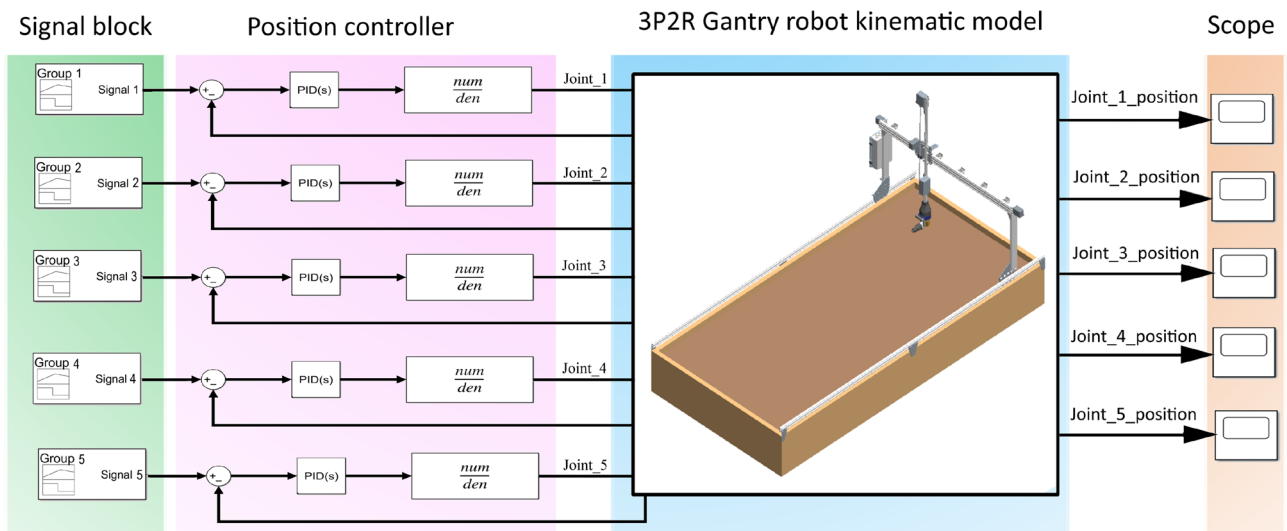


Figure 13. Simulink model of the 3P2R gantry robot.

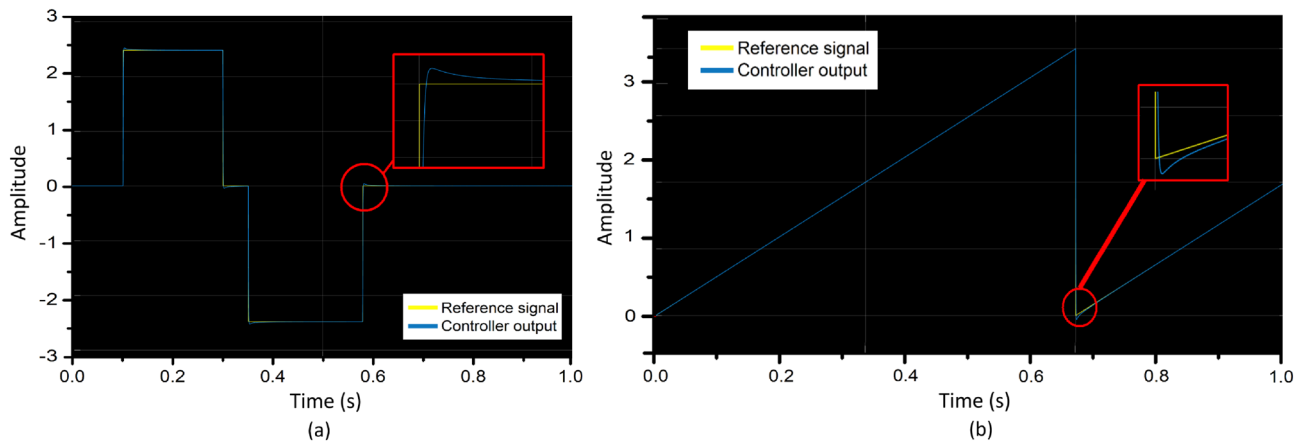


Figure 14. PID controller results (a) step response, and (b) sawtooth response.

PID parameters	Values
Proportional (P)	54185
Integral (I)	1303349
Derivative (D)	2187
Rise time (Rt)	0.55 s
Settling time (St)	0.89 s
Overshoot (Os)	3.12%
Peak time (Pt)	1.54 s

Table 4. PID controller parameters.

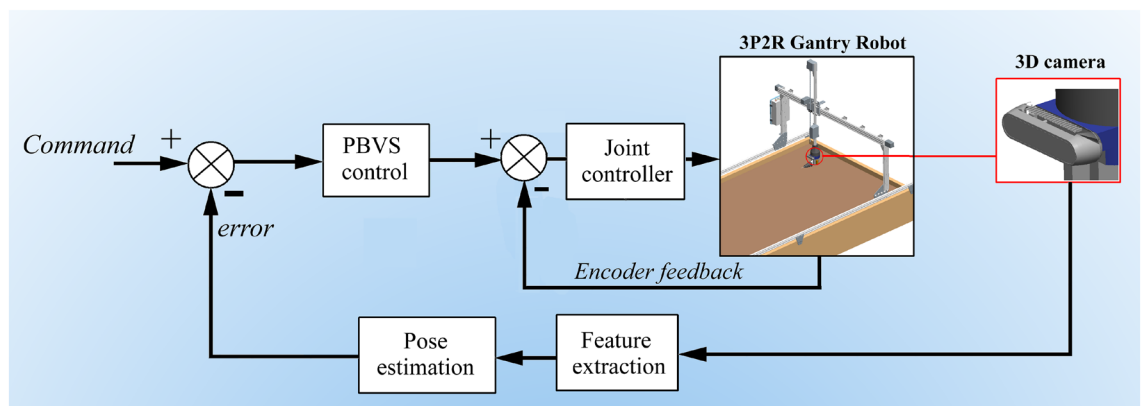


Figure 15. Position based visual servoing model of the 3P2R gantry robot.

“all classes” curve that represents the overall performance across all classes. The “Bud” curve starts with a high F1-score of around 0.92 at a confidence threshold of 0.0, indicating that the model is highly confident in its predictions for the “Bud” class. As the confidence threshold increases, the F1-score for “Bud” drops gradually, reaching around 0.80 at a threshold of 0.4. The “Flower” curve starts with a lower F1-score of around 0.70 at a confidence threshold of 0.0, suggesting that the model is less confident in its predictions for the “Flower” class. As the threshold increases, the F1-score drops more rapidly, reaching around 0.40 at a threshold of 0.4. The “all classes” curve provides an overall assessment of the model’s performance, with an F1-score of around 0.90 at a confidence threshold of 0.0, and a gradual decline to around 0.414 at a threshold of 0.4. The F1-Confidence Curve can be used to select an appropriate confidence threshold for the model, balancing the trade-off between precision and recall. For example, a threshold of 0.4 might be a reasonable choice, as it maintains a relatively high F1-score for the “Bud” class while still providing good overall performance across all classes.

Comparison between YOLOv5 and YOLOv8

The Table 2 compares the performance of two object detection models, YOLOv5s and YOLOv8s, on a validation dataset. The models were evaluated on three different classes: “All” (which represents the overall performance

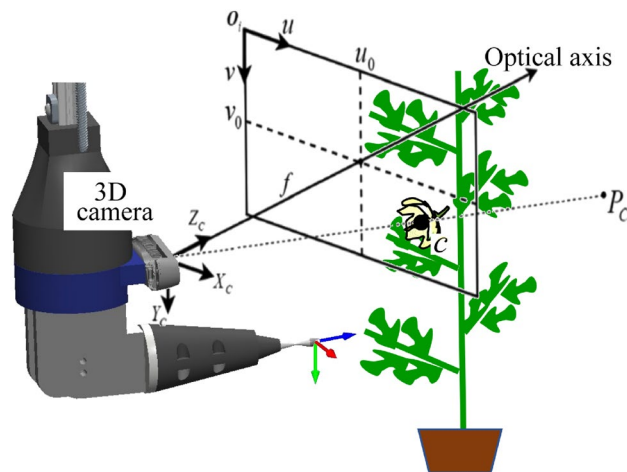


Figure 16. Camera model.

across all classes), “Bud”, and “Flower”. Starting with the overall performance, the YOLOv8s model demonstrates slightly better results compared to YOLOv5s. YOLOv8s has a precision of 0.925, meaning 92.5% of its detections are correct. Its recall is 0.886, indicating it is able to detect 88.6% of the ground truth objects. This leads to an F1-score of 0.894, which is a balanced metric combining precision and recall. The mean average precision (mAP) for YOLOv8s is 0.926, suggesting it is able to accurately detect and localize objects 92.6% of the time.

In contrast, the YOLOv5s model has a precision of 0.901, recall of 0.878, F1-score of 0.881, and mAP of 0.912. While these metrics are still strong, they are slightly lower than the performance of YOLOv8s on the overall “All” class. Looking at the individual classes, both models perform very well on the “Flower” class. YOLOv8s achieves a precision of 0.958, recall of 0.946, F1-score of 0.958, and mAP of 0.977 for this class. YOLOv5s is also highly accurate on the Flower class, with a precision of 0.952, recall of 0.935, F1-score of 0.944, and mAP of 0.962. The “Bud” class appears to be more challenging for both models. YOLOv8 has a higher precision of 0.93 and recall of 0.894 compared to YOLOv5’s 0.848 precision and 0.845 recall. However, YOLOv5s has a slightly higher F1-score of 0.846 versus 0.855 for YOLOv8s on this class. The mAP is also higher for YOLOv5 at 0.862 compared to 0.888 for YOLOv8. In terms of inference time, YOLOv5s has inference time of 0.9 ms per image, compared to 0.7 ms for YOLOv8s. This suggests YOLOv5 may be more suitable for real-time applications where faster inference is critical. Overall, the results indicate that both YOLOv5s and YOLOv8s are highly capable object detection models, with YOLOv8s offering slightly better overall performance. The choice between the two models may depend on the specific requirements of the application, such as the need for speed versus the need for the highest possible accuracy.

Figure 9 illustrates the detection results obtained by both YOLOv5s and YOLOv8s models. The evaluation of model performance involved using morning and evening images. In Fig. 9a, it can be observed that YOLOv5s misclassifies the stem as a bud, whereas the detection confidence level of YOLOv8 is notably higher. Additionally, in the night image Fig. 9b, YOLOv8s successfully detects the bud, whereas YOLOv5s fails to do so. Overall, YOLOv8s demonstrates superior detection performance for flowers and buds, which proves beneficial for achieving better pollination outcomes.

Robotic pollination system overview

While the primary focus of our study was on achieving significant detection results using the YOLOv8 model, we are currently engaged in further research regarding the visual servoing aspect of a robotic pollination system²⁴. This involves the utilization of ArUco markers and their deployment on a 3P2R gantry robot, which, to the best of the author’s knowledge, has not been extensively documented in existing literature. To validate the visual servoing approach, we conducted experiments on both un-clustered and clustered plants before applying it to a dataset of tomato plants. To determine the target’s depth from the robot pollinator device, we employed a 3D-RGB camera for depth prediction. Additionally, the design of our pollinator is innovative and can be adapted for other contact pollination farming practices such as strawberry and blueberry crops.

This section begins by introducing the mathematical and Simulink modeling of the 3P2R gantry robot. Additionally, it highlights the modifications made to the onboard circuit diagram to enable real-time detection and deployment on the hardware. The validation of the proposed detection and visual servoing algorithm is then conducted within a controlled laboratory environment. The experimentation took place at the AgriRobotics Lab in Khalifa University. The experimental conditions included artificial room lighting with white light, the utilization of paper-printed ArUco markers for camera calibration, and 3D printed tomato flowers for the pollination task. Furthermore, a dataset of greenhouse tomato flower images was employed to validate the detection and visual servoing algorithms.

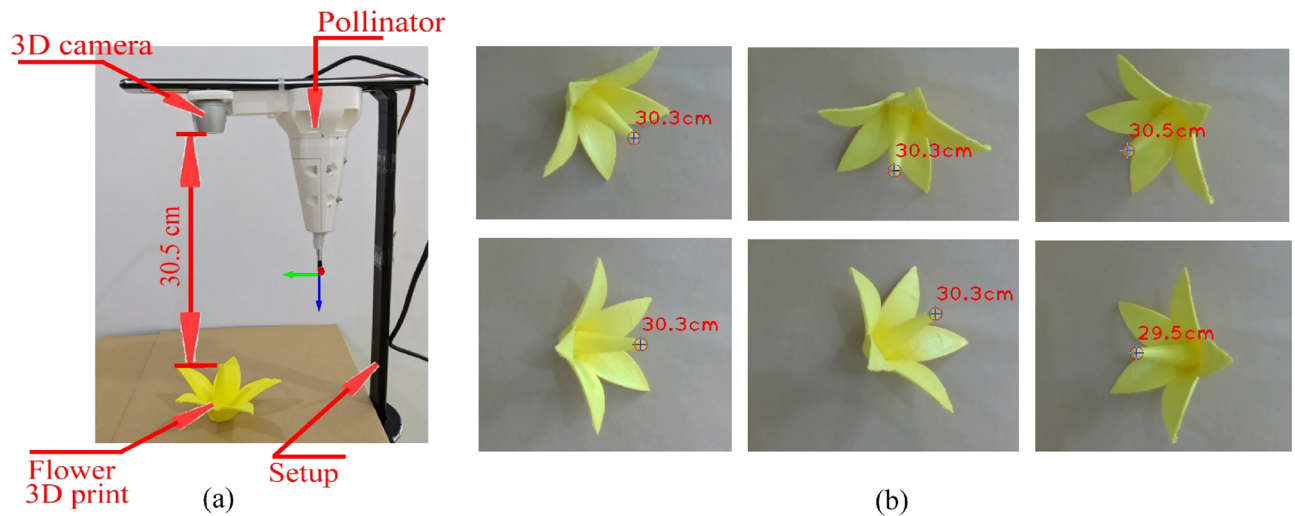


Figure 17. Depth estimation results (a) lab experiment setup, and (b) depth estimation results for different flower orientations.

3P2R gantry robot

The research utilizes a 3P2R (3 prismatic and 2 revolute) joints based gantry robot for automated pollination tasks. It is the prototype model of the large gantry systems used in the greenhouses²⁵. Figure 10 illustrates the overall system design of the 3P2R gantry robot. This robot features three prismatic joints enabling movement of the end effector along the X, Y, and Z axes. Additionally, the proposed pollinator incorporates two revolute joints for precise positioning and approach towards the target flower. The three servo motors with encoders are used to control the linear movements along the X, Y, and Z directions. Its electronics stack consists of a Raspberry Pi 3, Jetson nano Orin, and Arduino uno with a RAMPS 1.4 shield to control the various movements of the end effector and real time tomato flower detection for pollination application.

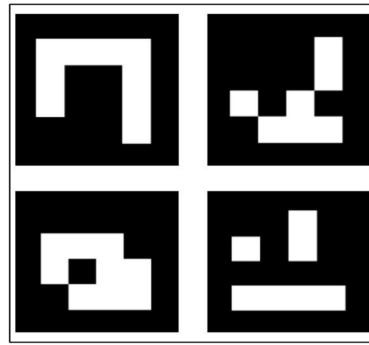
Figure 10a showcases the computer-aided design (CAD) model of the gantry system, featuring the proposed pollinator as the end effector. The actual prototype of the system is depicted in Fig. 10b. The pollinator itself is fabricated using ABS material via 3D printing and is outfitted with a 3D camera, specifically the Intel RealSense D435. Two servo motors are employed to enable yaw and pitch movements of the pollinator, facilitating precise adjustments. In the context of pollination, the rolling movement is deemed unnecessary and therefore not considered in the application. The Python-written program is responsible for controlling the movement of the 3P2R gantry robot. The deployed YOLOv8 detection model is utilized on the Jetson Nano onboard system in the ONNX format.

Figure 11 depicts the circuit diagram of the 3P2R gantry robot. The circuit design is based on the FarmBot circuit board²⁵, with modifications to accommodate the Jetson Nano Orin, arduino uno, and RGB-D camera. These components are connected to the FarmBot board, along with the additional circuitry required to control the pollinator's motion. The Jetson Nano processes captured images and classifies them using a trained model to identify flowers and buds. Upon detection, the visual servoing control system guides the linear movements of the end effector, precisely positioning it near the target. The pollination module then activates, performing the pollination process for 30-60 seconds during scanning process. The robot subsequently resumes its search for the next flower. The green colour dotted lines represent the gantry motion control mode, blue line represents the pollinator motion control module and pink line depicts the computer vision module.

Kinematic modelling of the 3P2R gantry robot

Robot kinematics mainly consists of two types: forward kinematics and inverse kinematics. Two types of coordinate system are useful for describing the robotic system's configuration. If we focus on the task and the end effector, we prefer to use Cartesian coordinates or end effector coordinates. The set of all such coordinates is generally called the Cartesian or end effector space. The other set of coordinates is the so-called joint coordinates, which are useful for describing the configuration of the mechanical linkage. The set of all such coordinates is generally called the joint space. In robotics, it is often necessary to be able to "map" joint coordinates to end effector coordinates. This map or the procedure used to obtain end effector coordinates from joint coordinates is called direct kinematics. Figure 12 shows the kinematic diagram of the 3P2R gantry robot. The initial frame of the robot is (x_0, y_0, z_0) and final frame for end effector is (x_5, y_5, z_5) .

The D-H parameters for the gantry robot is shown in the Table 3. The Eq. (7) is the 4 by 4 Homogeneous transformation matrix for gantry robot end effector with respect to base coordinate. The overall transformation matrix is calculated by using Eq. (8). By using the Eq. (8) the relation between the pollinator end effector with respect to base frame is derived.



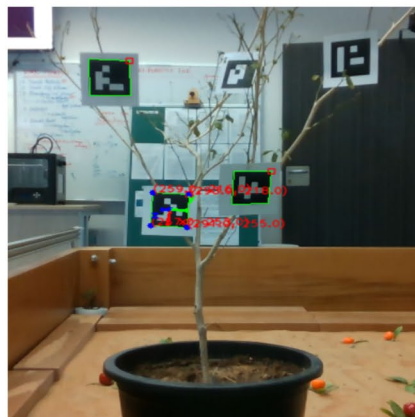
(a)



(b)



(c)



(d)



(e)

Figure 18. Camera Robot calibration (a) ArUco markers, (b) un-cluster plant, (c) cluster plant, (d,e) calibration results.

$$H_{n-1}^n = \begin{bmatrix} C(\theta_n) & -S(\theta_n)C(\alpha_n) & S(\theta_n)S(\alpha_n) & a_nC(\theta_n) \\ S(\theta_n) & C(\theta_n)C(\alpha_n) & -C(\theta_n)S(\alpha_n) & a_nS(\theta_n) \\ 0 & S(\theta_n) & C(\theta_n) & d_n \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad (7)$$

$$H_0^5 = H_0^1.H_1^2.H_2^3.H_3^4.H_4^5 \quad (8)$$

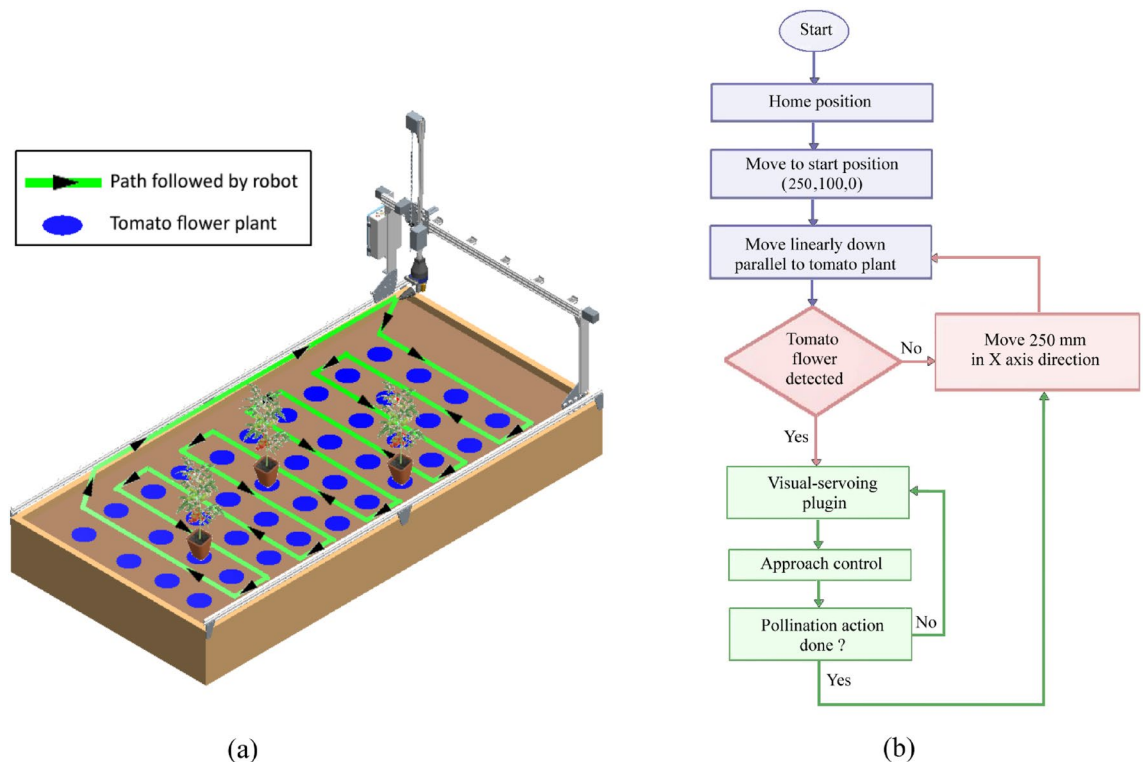


Figure 19. (a) Robot path coverage visualisation, and (b) flow diagram of searching algorithm during pollination.

Simulink model

In this section, we present the Simulink model of the 3P2R gantry robot, which was developed using the Physical Modeling Toolbox²⁶. Figure 13 illustrates the Simulink model of the gantry robot. To create the model, the CAD model of the 3P2R gantry robot was exported into an .xml format using the SimMechanics toolbox within the MATLAB environment. The .xml file was then converted into the Simulink (.slx) format using the “*simimport*” command. Custom signal builder blocks were utilized as reference input signals for the robot joints’ position. A PID controller was employed to regulate the desired position of the robot joints. The current joint position served as a feedback signal to map against the desired position. The objective of the PID controller was to minimize the error between the desired and current joint positions.

Figure 14 shows the results of the PID controller output for joint 1 with different input signals. Figure 14a depicts the step response and Fig. 14b shows the sawtooth signal response. The controller performed well and PID controller parameters are shown in Table 4.

Position based visual servoing

Position-based visual servoing is a technique used in robotics to control the motion of a robot based on visual information. It involves using visual feedback from a camera or sensor to compute the desired position or pose of the robot’s end effector or specific parts of the robot. The goal is to bring the robot to a desired position or track a desired trajectory using visual cues such as AurCo marker²⁷. Zhu et al. proposed the adaptive sliding mode control based position based visual servoing method to control the parallel robot²⁸. The position based control method was proposed for citrus fruit harvesting²⁹. Input of the visual servo control are the time-varying position errors between the target fruit and end-effector.

In our proposed approach for the pollination process, we introduced a hand-to-eye-based position visual servoing method to detect tomato flowers. To accomplish this, we mounted a 3D camera on the pollinator device, enabling the detection of the target flowers. Figure 15 provides an overview of the overall position-based visual servoing model employed with the 3P2R gantry robot. The control algorithm in this approach computes the error by comparing the desired and current poses, utilizing visual features extracted from the camera images. These errors are then utilized to generate control signals that adjust the robot’s joints, allowing it to move closer to the desired position. Additionally, the current joint positions obtained through encoders serve as feedback, facilitating more precise movements to reach the target position. To the best of the author’s knowledge, the proposed position-based visual servoing method includes a novel double feedback mechanism for effectively reaching the desired target, specifically the detected flower is not reported in literature. This unique approach combines two feedback sources: the first feedback mechanism utilizes visual information acquired from the camera, while the second feedback mechanism incorporates the use of encoders. By integrating these two feedback sources, the system accurately positioning the robot to reach the desired target position.

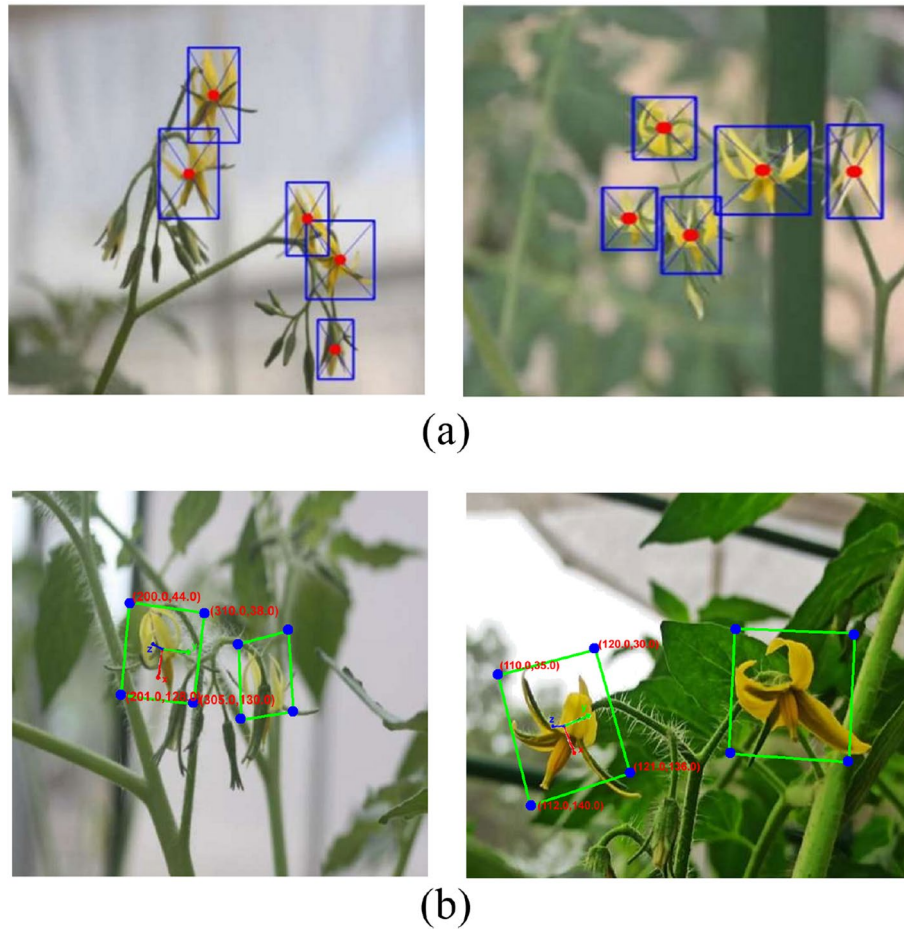


Figure 20. (a) Centre position detection, and (b) visual servoing pose detection for detected flowers.

Camera model

The projection of the camera onto the image plane is depicted in Fig. 16, where the pixel point $p(u,v)$ is represented in the 2D coordinate system. This projection illustrates the mapping of the three-dimensional (3D) world onto the two-dimensional (2D) image plane captured by the camera. The 3D scene point and camera coordinate system are represented as P_c and (x_c, y_c, z_c) . The c is the center point of the detected flower on image plane O_i and f is focal length of the camera. The relationship between the 3D coordinates and the corresponding image coordinates is established using the pinhole model. This model describes the geometric principles governing the projection of three-dimensional points onto a two-dimensional image³⁰.

$$\begin{bmatrix} u \\ v \end{bmatrix} = \begin{bmatrix} f \frac{x_c}{z_c} \\ f \frac{y_c}{z_c} \end{bmatrix} \tag{9}$$

However, it should be noted that in practical scenarios, the principal point of the camera may not align perfectly with the center of the image. Therefore, it becomes necessary to consider the offset (u_o, v_o) in the image plane. Consequently, Eq. (9) is modified to incorporate this offset and account for any deviation from the ideal alignment.

$$z_c \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f_x & 0 & u_o \\ 0 & f_y & v_o \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x_c \\ y_c \\ z_c \end{bmatrix} \tag{10}$$

Depth estimation

Obtaining accurate depth information is crucial for successfully performing the pollination task on the detected tomato flowers. To achieve precise depth measurements, we employed an Intel RealSense RGB-D camera. This 3D camera was mounted on the end effector of the 3R2P gantry robot. By positioning the RGB-D camera on the gripper, it was able to determine the depth information of the target flower based on its location within the camera’s field of view. This depth information played a critical role in enabling the robot to effectively plan and

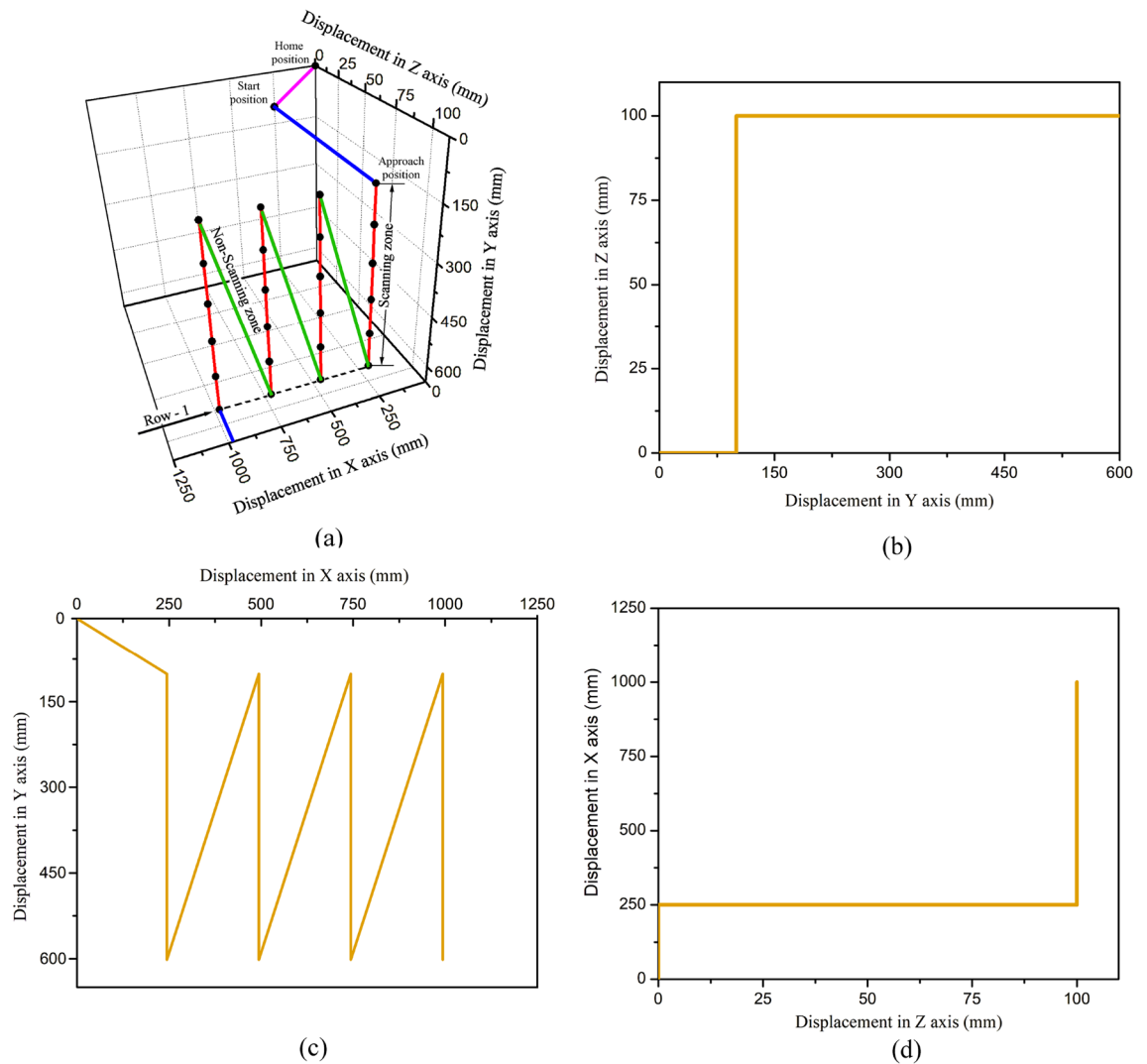


Figure 21. Trajectory result of the 3R2P gantry robot for first row (a) 3D trajectory of the pollinator end effector, (b) Y-Z plane, (c) X-Y plane, and (d) Z-X plane.

execute the approaching action required for the pollination process. In our study, we employed a laboratory setup to estimate depth while accounting for the variations in flower orientations, as depicted in Figure 17. In this experiment author's used the 3D printed tomato flower and placed in different orientation for better depth estimation calibration results.

Hand to eye robot calibration model

The calibration process in this study involved the use of a printed ArUco marker board (Fig. 18a) to calibrate both the camera and the robot. The authors considered two different environments for the calibration process, as depicted in Fig. 18b, c. The first environment represented an un-clustered plant, while the second environment depicted a cluster plant setting. To perform the calibration, a Python-based camera calibration package called “visphand2eyecalibration” was utilized. This package consisted of two essential transformations: (i) the camera-to-object transformation and (ii) the world-to-hand transformation. The calibration process involved running the “arucorealsense.launch” file and placing the ArUco plane within the camera's field of view. The package utilized a calibrator node to perform the hand-to-eye calibration. As a result, two transformation frames, namely “/cameralink” and “/armarker,” were published within the system. These frames represented the spatial relationships between the marker and the camera attached to the robot. The purpose of this calibration was to ensure accurate perception and coordination between the robot's hand and the camera during pollination tasks. The ArUco marker detection for un-cluster and cluster environment using visual servoing method are depicted in Fig. 18d, e.

Tomato flower searching algorithm

The robot is always positioned at the top, which is known as the home position. The pollinator mounted camera moves to start position and follows the path as depicted in Fig. 19a. Every time the robot camera scan the tomato

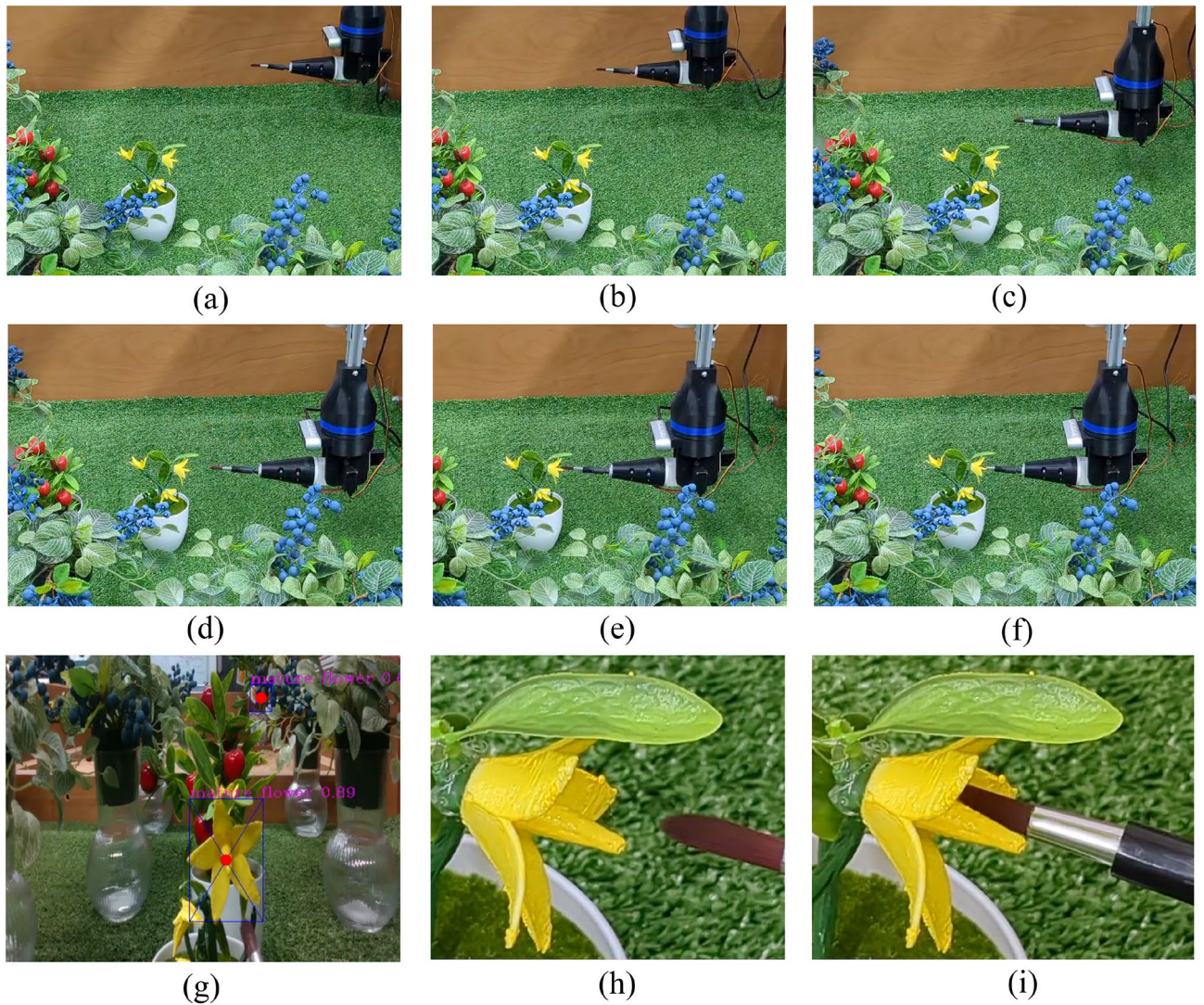


Figure 22. (a–f) Robot path during the pollination task, (g) tomato flower detection, (h,i) zoom view of the pollination process.

plant linearly and then move 250 mm in X axis for next plant in the row. The scanning path is pre-defined for this work. After covering all the plants in the workspace the robot move back to home position. Fig. 19b depicts the flow diagram of the searching algorithm for 3R2P gantry robot.

Visualization and results

This section showcases the visualization of tomato flower detection and pollination process. The pollinator with depth camera (end effector) classify the tomato flower and buds and detects the tomato flower for pollination process. Figure 20 depicts the tomato flower detection and calculate the centre pose of the detected flowers. The center pose coordinates are used to calculate the desired motion of the robot in x, y, and z axis using visual servoing approach.

Figure 21 provides a comprehensive visualization of the pollination process for the tomato flower in the first row. The robot's end effector, known as the pollinator, initiates the process by moving from a pre-defined home position to the start position at coordinates (250, 100, 0). Subsequently, the robot follows the approach position at coordinates (250, 100, 100). Meanwhile, the robot's camera begins descending in the Y direction to a depth of 600 mm while scanning the tomato plant. Once it reaches the position (250, 100, 600), the robot moves diagonally to the next pre-defined coordinate to scan the next plant. Figure 21a illustrates the 3D trajectory of the robot's end effector as it scans the plants in the first row. On the other hand, Fig. 21b–d depict the 2D trajectories of the end effector, providing a visual representation of its movement during the pollination process. Figure 22 illustrates the lab experimentation pictures Fig. 22a–f depicts the path followed by the 3P2R gantry robot end effector (pollinator). Figure 22g shows the detected tomato flower, h, i depicts the zoom view of the pollination process.

Currently, the assessment of the overall pollination success rate is visually determined, as done by the greenhouse operators. This success rate is contingent on the total productivity of tomato fruit crops per unit area. To ascertain this success rate, multiple trials need to be conducted in the greenhouse, which is part of our

forthcoming research plan. Additionally, the pollination success rate of the flower will be evaluated by examining the quantity of pollen collected in the fruit (female ovary) section through microscopic analysis, a step we intend to address in future endeavors. This study primarily emphasizes deploying the refined detection model on the robotic based pollination system and aims to discover an autonomous alternative to natural pollinators (bees) in greenhouse settings.

Conclusion

This study addressed the need for precise and efficient detection of tomato flowers in the context of robotic pollination within smart greenhouses. By employing transfer learning techniques, specifically utilizing YOLOv5 and YOLOv8 models, the researchers developed a methodology for labeling, training, and detecting tomato flowers. The results demonstrated that YOLOv8 outperformed YOLOv5 in terms of average precision (mAP) with a score of 92.6% for tomato flower and bud detection. The model also exhibited a fast inference speed of 0.7 ms, making it suitable for real-time applications. The datasets were acquired under different lighting conditions to ensure robustness. The findings showcased the potential of YOLOv8 in estimating flower blooming peaks and facilitating robotic pollination tasks. In addition to the detection aspect, the study focused on the deployment of the detection model on the 3P2R gantry robot. The researchers proposed a kinematic model and a modified circuit for the robot, and the position-based visual servoing method was utilized for approaching the detected flowers during the pollination process. The effectiveness of the visual servoing approach was validated in both unclustered and clustered plant environments in the laboratory. Overall, this research contributes valuable insights to the field of greenhouse systems. The developed methodology and the superior performance of YOLOv8 offer promising prospects for real-time detection, estimation of blooming peaks, and enhancing the effectiveness of robotic pollination tasks in smart greenhouses. In terms of future scope, the authors plan to address challenges related to flower overlapping and occlusions in order to develop more robust solutions (Supplementary Video).

Data availability

The data that support the findings of this study are available from ASPIRE, Abu Dhabi, but restrictions apply to the availability of these data, which were used under license for the current study and so are not publicly available. Data are, however, available from the authors upon reasonable request and with permission of ASPIRE, Abu Dhabi.

Received: 1 July 2024; Accepted: 23 August 2024

Published online: 04 September 2024

References

1. Aizen, M. A., Garibaldi, L. A., Cunningham, S. A. & Klein, A. M. Global agricultural productivity is threatened by increasing pollinator dependence without a parallel increase in crop diversification. *Glob. Change Biol.* **10**, 3516–3527 (2019).
2. Aizen, M. A., Garibaldi, L. A., Cunningham, S. A. & Klein, A. M. Long-term global trends in crop yield and production reveal no current pollination shortage but increasing pollinator dependency. *Curr. Biol.* **18**, 1572–1575 (2008).
3. Cui, X., Guan, Z., Morgan, K. L., Huang, K.-M. & Hammami, A. M. Multitiered fresh produce supply chain: The case of tomatoes. *Horticulturae* **8**, 1204 (2022).
4. Colucci, G., Tagliavini, L., Botta, A., Baglieri, L. & Quaglia, G. Decoupled motion planning of a mobile manipulator for precision agriculture. *Robotica* **41**, 1872–1887 (2023).
5. Murphy, J. T., Breeze, T. D., Willcox, B., Kavanagh, S. & Stout, J. C. Globalisation and pollinators: Pollinator declines are an economic threat to global food systems. *People Nat.* **4**, 773–785 (2022).
6. Xu, T. *et al.* A neural network structure with attention mechanism and additional feature fusion layer for tomato flowering phase detection in pollination robots. *Machines* **10**, 1076 (2022).
7. Broussard, M. A., Coates, M. & Martinsen, P. Artificial pollination technologies: A review. *Agronomy* **13**, 1351 (2023).
8. Albadarneh, A. A. *Automated flower species detection and recognition from digital images*. Ph.D. thesis, Princess Sumaya University for Technology (Jordan) (2016).
9. Aleya, K. F. Automated damaged flower detection using image processing. *J. Glob. Res. Comput. Sci.* **4**, 21–24 (2013).
10. Dorj, U.-O. & Lee, M. A new method for tangerine tree flower recognition. In *International Conference on Bio-Science and Bio-Technology*. 49–56 (Springer, 2012).
11. Zheng, Y.-Y. *et al.* Probability fusion decision framework of multiple deep neural networks for fine-grained visual classification. *IEEE Access* **7**, 122740–122757 (2019).
12. Khan, A., Nawaz, U., Ulhaq, A. & Robinson, R. W. Real-time plant health assessment via implementing cloud-based scalable transfer learning on AWS deepfens. *Plos one* **15**, e0243243 (2020).
13. Khan, A., Nawaz, U., Kshetrimayum, L., Seneviratne, L. & Hussain, I. Early and accurate detection of tomato leaf diseases using tomformer. In *2023 21st International Conference on Advanced Robotics (ICAR)*. 645–651 (IEEE, 2023).
14. Khan, A. *et al.* Tomato maturity recognition with convolutional transformers. *Sci. Rep.* **13**, 22885 (2023).
15. Singh, R., Bera, T. K. & Chatti, N. A real-time obstacle avoidance and path tracking strategy for a mobile robot using machine-learning and vision-based approach. *Simulation* **98**, 789–805 (2022).
16. Chen, Y. *et al.* Strawberry yield prediction based on a deep neural network using high-resolution aerial orthoimages. *Remote Sens.* **11**, 1584 (2019).
17. Sun, J. *et al.* Detection of key organs in tomato based on deep migration learning in a complex background. *Agriculture* **8**, 196 (2018).
18. Saad, W., Karim, S., Razak, M., Radzi, S. & Yussof, Z. Classification and detection of chili and its flower using deep learning approach. *J. Phys. Conf. Ser.* **1502**, 012055 (IOP Publishing, 2020).
19. Huang, Z., Zhang, P., Liu, R. & Li, D. Immature apple detection method based on improved yolov3. *ASP Trans. Internet Things* **1**, 9–13 (2021).
20. Tian, M., Chen, H. & Wang, Q. Detection and recognition of flower image based on SSD network in video stream. *J. Phys. Conf. Ser.* **1237**, 032045 (IOP Publishing, 2019).
21. Cardellicchio, A. *et al.* Detection of tomato plant phenotyping traits using yolov5-based single stage detectors. *Comput. Electron. Agric.* **207**, 107757 (2023).

22. Zhang, J. *et al.* Greenhouse tomato detection and pose classification algorithm based on improved yolov5. *Comput. Electron. Agric.* **216**, 108519 (2024).
23. Joher, G., Chaurasia, A. & Qiu, J. *Ultralytics yolov8* (2023).
24. Hiraguri, T. *et al.* Shape classification technology of pollinated tomato flowers for robotic implementation. *Sci. Rep.* **13**, 2159 (2023).
25. Aronson, R. L. Humanity's open-source automated precision farming machine. *Creative Commons Attrib.* **4** (2013).
26. Singh, R. & Bera, T. Walking model of Jansen mechanism-based quadruped robot and application to obstacle avoidance. *Arab. J. Sci. Eng.* **45**, 653–664 (2020).
27. Thuilot, B., Martinet, P., Cordesses, L. & Gallice, J. Position based visual servoing: Keeping the object in the field of vision. In *Proceedings 2002 IEEE International Conference on Robotics and Automation (Cat. No. 02CH37292)*. Vol. 2. 1624–1629 (IEEE, 2002).
28. Zhu, N., Xie, W.-F. & Shen, H. Position-based visual servoing of a 6-RSS parallel robot using adaptive sliding mode control. *ISA Trans.* **144**, 398–408 (2024).
29. Mehta, S. & Burks, T. Vision-based control of robotic manipulator for citrus harvesting. *Comput. Electron. Agric.* **102**, 146–158 (2014).
30. Ayaz, S. M., Kim, M. Y. & Park, J. Survey on zoom-lens calibration methods and techniques. *Mach. Vis. Appl.* **28**, 803–818 (2017).

Acknowledgements

We acknowledge Khalifa University Center for Robotics and Autonomous Systems (KUCARS) for their facilities and labs.

Author contributions

Rajmeet Singh: Pollination conceptualization, Methodology, robot experimentation, drafting robotics section, Visualization. Asim Khan: Detection model conceptualization, Methodology, computer vision experimentation, drafting computer vision section, Visualization. Lakmal Seneviratne: Supervision, Project administration. Irfan Hussain: Supervision, Project administration, Funding acquisition.

Competing interests

The authors declare no competing interests.

Additional information

Supplementary Information The online version contains supplementary material available at <https://doi.org/10.1038/s41598-024-71013-1>.

Correspondence and requests for materials should be addressed to I.H.

Reprints and permissions information is available at www.nature.com/reprints.

Publisher's note Springer Nature remains neutral with regard to jurisdictional claims in published maps and institutional affiliations.

Open Access This article is licensed under a Creative Commons Attribution-NonCommercial-NoDerivatives 4.0 International License, which permits any non-commercial use, sharing, distribution and reproduction in any medium or format, as long as you give appropriate credit to the original author(s) and the source, provide a link to the Creative Commons licence, and indicate if you modified the licensed material. You do not have permission under this licence to share adapted material derived from this article or parts of it. The images or other third party material in this article are included in the article's Creative Commons licence, unless indicated otherwise in a credit line to the material. If material is not included in the article's Creative Commons licence and your intended use is not permitted by statutory regulation or exceeds the permitted use, you will need to obtain permission directly from the copyright holder. To view a copy of this licence, visit <http://creativecommons.org/licenses/by-nc-nd/4.0/>.

© The Author(s) 2024