Complex & Intelligent Systems (2022) 8:2955–2969
https://doi.org/10.1007/s40747-021-00522-7

ORIGINAL ARTICLE

A peduncle detection method of tomato for autonomous harvesting

Jiacheng Rong[1] · Guanglin Dai[1] · Pengbo Wang[1]

Received: 29 March 2021 / Accepted: 31 August 2021 / Published online: 28 September 2021

Abstract
For automating the harvesting of bunches of tomatoes in a greenhouse, the end-effector needs to reach the exact cutting point and adaptively adjust the pose of peduncles. In this paper, a method is proposed for peduncle cutting point localization and pose estimation. Images captured in real time at a fixed long-distance are detected using the YOLOv4-Tiny detector with a precision of 92.7% and a detection speed of 0.0091 s per frame, then the YOLACT + + Network with mAP of 73.1 and a time speed of 0.109 s per frame is used to segment the close-up distance. The segmented peduncle mask is fitted to the curve using least squares and three key points on the curve are found. Finally, a geometric model is established to estimate the pose of the peduncle with an average error of 4.98° in yaw angle and 4.75° in pitch angle over the 30 sets of tests.

Keywords Harvesting robot · Peduncle detection · Deep learning · Pose estimation

Introduction

Harvesting cherry tomatoes relies mainly on manual picking, which is time consuming and labor intensive [1]. Moreover, the aging population and increasing labor costs have led to a shortage of labor. Therefore, there is an urgent need to integrate robot technology with artificial intelligence in agriculture, using automated picking instead of manual picking [2]. At present, the research of picking robots is focused on the following directions: fruit localization, ripeness determination, obstacle localization, task planning and motion planning [3]. In this paper, we focus on fruit detection and localization, which is a hot topic of current research and a prerequisite for automated robotic picking.

The use of deep learning to detect fruit has become very popular and has been mentioned in some prior work [4]. Koirala et al. improved the YOLOv3 detector by reducing the model to 33 layers for better connecting the deeper semantic information with the shallower fine-grained information to detect mangoes with an F1-score of 0.89 and a detection speed of 70 ms per image(on an NVIDIA GTX 1070 Ti GPU) [5]. Li et al. proposed an improved YOLOv3 network to detect apples by replacing the backbone from DarkNet53 method to DenseNet method and cost 0.304 s on a 3000 × 3000 pixels image [6]. Liu et al. used a circular bounding box instead of a rectangular bounding box for individual tomato localization based on YOLOv3, achieving the correct identification rate of 94.58% with an average detection time of 0.054 s per image under slight occlusion conditions [7]. Birrell et al. automated the harvesting of mature lettuce in the natural environment [8]. The images are captured in real time by a stationary camera and passed into the YOLOv3 network for coarse positioning, and the detected images are cropped and passed into the darknet network for fine classification, enabling the differentiation between immature lettuce, mature lettuce, and lettuce affected by pests and diseases. Isaac et al. improved Mask RCNN for strawberry fruit segmentation and also performed well on the detection of partially obscured strawberries [9]. Song et al. used DeepLabV3 + to segment calyces, branches and wires on kiwifruit canopy images, which achieved mIoU of 0.694 with a detection time of 210.0 ms [10]. As deep learning models push the envelope, detectors are able to detect fruit faster and better, for example using YOLOv4 to detect citrus with 3.15% better accuracy compared to YOLOv3 [11].

Although the current research has been able to solve the problem of fruit identification and positioning, such as harvesting the bunch of tomatoes, the robotic end-effector must be able to accurately grip the peduncle attached to the

✉ Pengbo Wang
pbwang@suda.edu.cn

[1] School of Mechanical and Electric Engineering, Jiangsu Provincial Key Laboratory of Advanced Robotics, Soochow University, Suzhou 215123, China

fruit. However, compared to the detection and localization of fruit, the peduncles are smaller and more complex, and varied in shape, and the detector is susceptible to interference from complex backgrounds such as the main stem. Many attempts have been made by scholars to address these issues. Sa et al. [12] proposed a three-dimensional vision detection method for fruit stem recognition of automatic picking sweet pepper in the greenhouse. Based on HSV color space and geometric features (different curvature of stem and fruit surface), the detection features are extracted and classified by the support vector machine. Luo et al. studied the detection of cutting points on peduncles of overlapping grape clusters in the unstructured vineyard [13]. After segmenting individual clusters using k-means clustering and effective color component, a geometric constraint method is used to determine the cutting point in the region of interest of each grape cluster's peduncle. Yoshida et al. used the support vector machine to classify the point cloud data, clustering to obtain fruit peduncle pixels, and then looking for cutting points [14, 15]. For solving the problem of the gripping pose of the end-effector of the pepper picking robot, Barth et al. first semantically segmented the pixels of the fruit and the main stem and then determined the spatial relationship between the fruit and the main stem to estimate the pose of the sweet pepper [16]. To solve the pose estimation problem for automated strawberry picking, Yu et al. [17] used a labeled dataset of bounding boxes with angles based on YOLO object detection and let the network learn the rotation angle of the bounding boxes to estimate the growing pose of strawberries. Xiong et al. [18] proposed a nighttime litchi fruit and fruit stalk detection method. The method is based on YOLOv3 to detect litchi fruits in the natural environment at night and determine the region of interest of fruit stalks based on the predicted litchi fruit bounding box. Finally, the litchi fruit stalks are segmented based on U-Net networks.

Existing methods allow for the identification and positioning of fruit peduncles to some extent, grapes and lychees are usually vertical downwards, the methods proposed by Luo et al. and Xiong et al. work well for vertical downwards growing stalks, but peduncles of tomato can deflect to any angle [13, 18]. The method proposed by Barth et al. estimates the angle of a pepper relative to the main stem, but only estimates a pose that is only applicable to the unique end-effector designed by themselves [16]. Therefore, to realize the fully autonomous picking of the robot, the current vision system needs to be continuously improved. The purpose of this work is to study the cutting point positioning and pose estimation of the peduncle of tomatoes, provide coordinates for the robot end-effector grasping and adjust the grasping posture of the hand, so as to improve the picking success rate.

In this paper, we present a two-stage peduncle localization method for cutting point localization of peduncles for bunches of cherry tomatoes in an unstructured environment. First, the tomatoes are coarsely located using a YOLOv4-Tiny detector. Second, YOLACT + + is used to segment the fruit pixels and peduncle pixels. Finally, the peduncle mask is used to fit the curve and three key points are found to construct a geometric model to estimate the peduncle pose. The outline of this paper is as follows: in Sect. 2, the materials and methods used in the study including image acquisition and detection methods are described. In Sect. 3, the experimental results are presented and discusses. Lastly, in Sect. 4, the research work is summarized and future work is prospected.

Materials and methods

This study focused on estimating the cutting point and cutting pose of the bunch of cherry tomatoes. Since ripe tomatoes are not distributed at a similar height, the camera needs to collect images at a relatively long distance to obtain a field of view that is sufficient to cover most ripe tomatoes. However, the image obtained at a long distance has too few peduncle pixels, which makes it difficult to use the algorithm to segment peduncle of tomato bunches. If the camera is close to the peduncle to collect images, the larger pixel area of the peduncle is easier to be segmented, and a better point cloud data for the peduncle can be obtained by RGB-D camera [19]. Besides, close-up images can also avoid background interference to a certain extent.

Combining the advantages of a large field of view for long-shot detection and a large pixel area of peduncle for close-shot detection, we developed a two-stage method for segment peduncle pixels corresponding to a bunch of cherry tomatoes. The flowchart of the two-stage peduncle segmentation method is illustrated in Fig. 1. In step 1 and step 2, images captured at different distances need to be collected, images input to YOLOv4-Tiny need to be captured at a fixed distance of 0.5 to 0.7 m from the cherry tomatoes (see Fig. 2a), while images input to YOLACT + + need to be captured by moving the camera installed on the end of the robotic arm to 0.3 m in front of the tomato bunch (see Fig. 2b). Therefore, the camera is mounted at the end of the robotic arm using an Eye-in-hand method, which allows the robotic arm to reach the position where the camera able to collect close-up images. Step 1 is the coarse positioning of the cherry tomato bunch which is the center of a tomato bunch. An image is collected at 0.5–0.7 m that has a field of view covering the entire pickable area of the ripe cherry tomatoes, as a result of which the pixel area of the individual peduncle will be very small and difficult to detect. The YOLOv4-Tiny object detection algorithm is, therefore, first adopted to detect the pickable tomato bunch in the image and obtain a bounding box containing the tomato

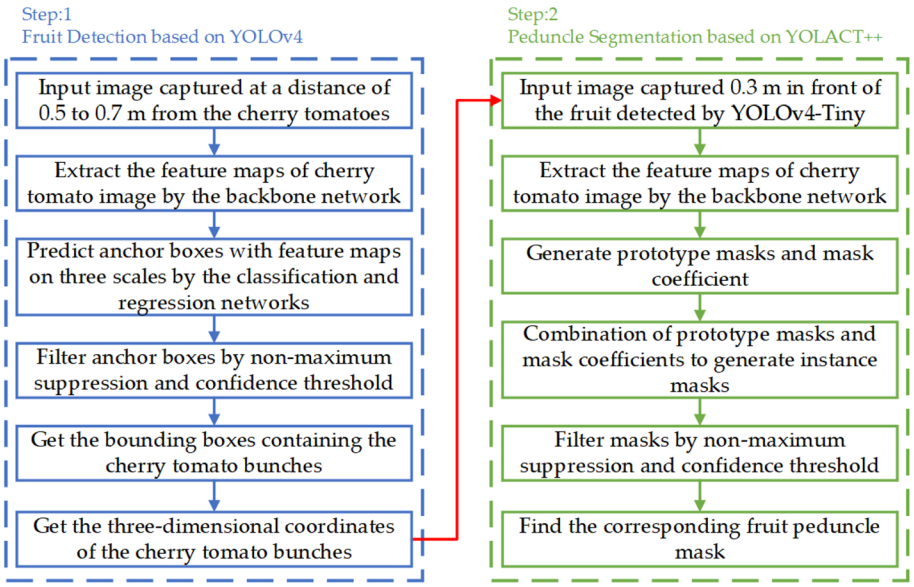
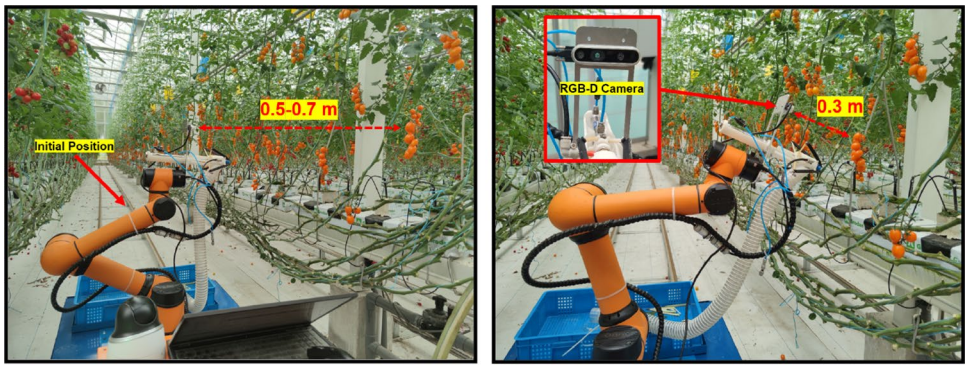


Fig. 1 Flowchart of the two-stage peduncle segmentation method



(a) The position of the robotic arm in step 1 **(b)** The position of the robotic arm in step 2

Fig. 2 The position change of the robotic arm in different steps: **a** the position of the robotic arm when an image is captured at a long distance; **b** the position of the robotic arm when a close-up image captured 0.3 m in front of the tomato bunch

bunch. After tomato bunches are detected, the state of the mobile robot platform used to install the robotic arm changes from moving to stopped. Then, the robotic arm approaches coarse positions (the coarse positions of tomato bunches are obtained in step 1, and robotic arm is moved to make the RGB-D camera reach the position 0.3 m in front of tomato bunch with a pose of 20 degrees counterclockwise around the X-axis of the camera coordinate system, as shown in Fig. 3) and perform operations in step 2. Step 2 is to segment the peduncle pixel that corresponds to the pickable tomato bunch. Compared to the image captured at a longer distance, the pixel area of the peduncle is larger in the image captured 0.3 m in front of the fruit, with a relative reduction in the pixel size of the complex background. In step 2, the YOLACT++ instance segmentation algorithm is adopted to produce the masks for tomato bunches and peduncles. However, the output of the instance segmentation algorithm lacks a correlation between masks for the tomato bunch and the peduncle. To better construct the connection between masks for the tomato bunch and the peduncle, an algorithm

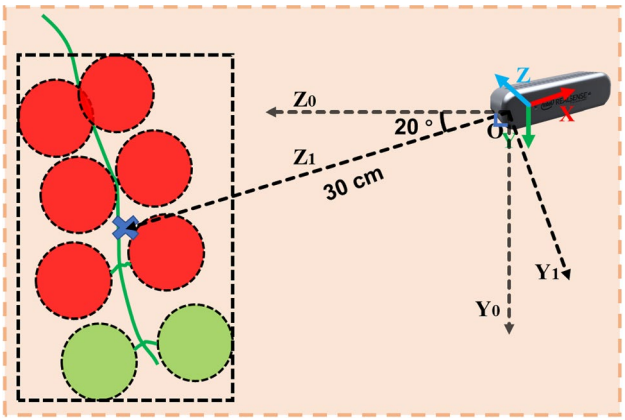


Fig. 3 Schematic diagram of close-up image collection location

to suppress non-corresponding peduncle masks is developed inspired by the non-maximum suppression algorithm. After obtaining the peduncle masks, we use the mask to find three key points on the peduncle masks (the upper point, the lower

point, and the cutting point) and map the two-dimensional coordinates on the point cloud image to obtain the corresponding three-dimensional coordinates to construct the geometry model to predict the pose of the peduncle.

Image acquisition

The cherry tomato datasets used in this paper were collected in a period from September 2020 to January 2021 in Nijiawan Water Field Greenhouse, Xiangcheng District, Suzhou, China. The images were captured using multiple RGB-D cameras (Intel RealSense D435i and D415) with 1280×720-pixel resolution. To allow the visual system of the harvesting robot to have a wide range of adaptability in unstructured environments, images of tomatoes with different growth cycles under different lighting conditions were collected.

Different datasets were required to train the models for YOLOv4-Tiny and YOLACT++. To meet this requirement, images captured at the long-distance and short-distance were collected separately. A total of 1528 images were collected, of which 828 images were acquired with the camera at a distance of 0.5–0.7 m from the cherry tomatoes and another 700 at 0.25–0.35 m. Figure 4 shows some samples in different poses and captured at a different distance from the dataset. Before the tomato matures, to facilitate light transmission and reduce diseases and pests, the staff will remove the leaves from the area to be mature. Therefore, all images we collect in this paper are tomatoes without leaves.

To make the image algorithm based on the deep neural network learn the characteristics of the tomatoes, it is necessary to label the acquired images and mark the tomatoes and peduncles to be detected. In this paper, the rectangular box in the 'labelImg' calibration tool(see Fig. 5a) is used to label

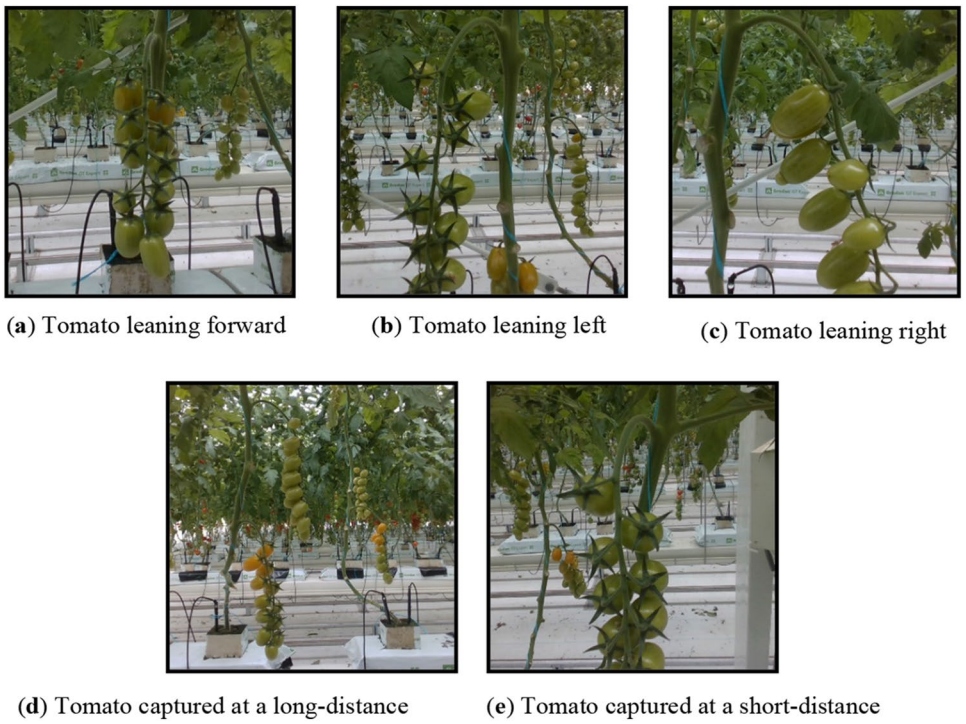


Fig. 4 Tomato samples in different poses and captured at a different distance: **a** tomato leaning forward; **b** tomato leaning left; **c** tomato leaning right; **d** tomato captured at a long distance; **e** tomato captured at a short distance

Fig. 5 Tomato labeling methods: **a** the rectangular boxes are used to label tomatoes captured at a long distance; **b** two polygon boxes are used to label a bunch of tomatoes and the peduncle

images captured at a long distance, while the polygon box in the 'labelme' calibration tool (see Fig. 5b) is used to mark images along the edge of the object captured at a short distance. As shown in Fig. 5b, there are two polygon boxes in total: the first one labels cherry tomatoes with the peduncle and the second one marks only the peduncle.

Tomato bunch detection based on YOLOv4-Tiny

YOLOv4 is the latest single-stage object detection algorithm, which is upgraded from YOLOv3 and can achieve better performance without reducing the speed [20–23]. YOLOv4 chooses CSPDarknet53 as the backbone, which consists of a multiple residual module and downsampling, and uses the Cross Stage Partial (CSP) Module to divide the feature mapping of the 'Resblock_body' into two parts and merge them in a cross-stage hierarchy (see Fig. 6). The Spatial Pyramid Pooling (SPP) module is added to CSPDarknet53 to enhance the receptive field and isolate important contextual features. After a comparative experiment on our tomato dataset, YOLOv4-Tiny detector is adopted in this work, which is a simplified version of YOLOv4 and has a higher detection speed without reducing detection accuracy. It uses the backbone with fewer layers, cancels the SPP module, and reduces the number of detector heads from three to two. As the robot moves, the vision system needs to be able to detect the presence of cherry tomatoes in the field of view in real time. Therefore, the YOLOv4-Tiny detector is adopted to coarsely locate the cherry tomato fruit in a complex unstructured environment and to obtain information on the position of the fruit in the image.

Peduncle segmentation based on YOLACT++

After the cherry tomato fruit has been detected at a long distance, the camera mounted at the end of the robotic arm moves in front of the fruit. The fruit detected in the YOLOv4 detector and the corresponding peduncle appears in the center of the camera's field of view, with an increased proportion of the peduncle pixel area in the overall image and relatively less interference from the complex background. This significantly reduces the difficulty of segmenting the peduncle pixels and the point cloud information provided by the RealSense D415 is more accurate. The ultimate goal of step 2 is to find the peduncle that corresponds to the cherry tomato fruit. The instance segmentation network simultaneously separates the pixels of the whole bunch of cherry tomato and the pixels of the individual peduncle and finally uses a relative position of the segmented pixels to suppress the peduncle pixels that are not in the whole bunch of cherry tomato. The YOLACT++ instance segmentation neural network is used as a detector to find a whole bunch of tomatoes and the corresponding peduncle in the close-up image.

YOLACT++ uses ResNet-101 as the feature backbone, enabling deeper abstract semantic information features to be extracted [24, 25]. Different from the single-stage object detector, YOLACT++ divides the instance segmentation task into two parallel processes, as shown

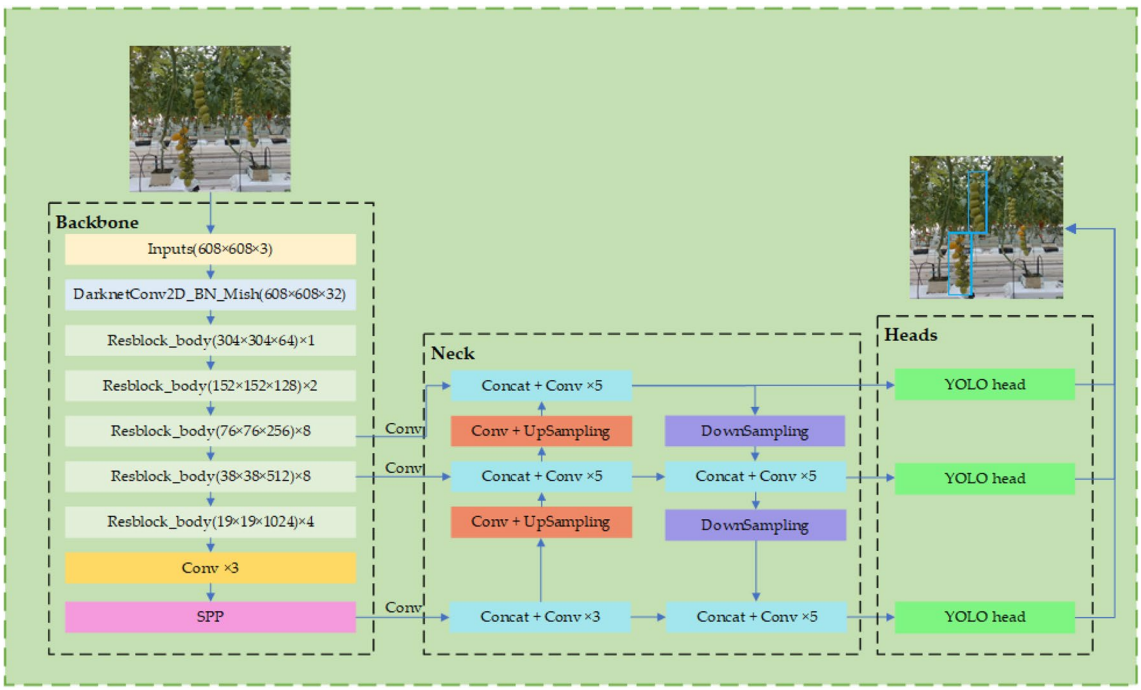


Fig. 6 The structure of YOLOv4

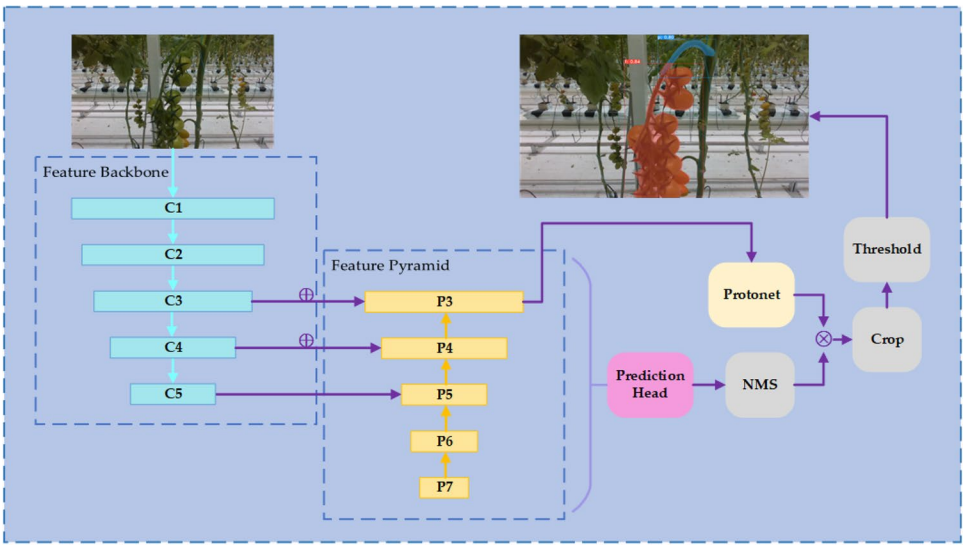


Fig. 7 The structure of YOLACT++

in Fig. 7. The first branch uses Fully Convolutional Networks (FCN) to generate a series of prototype masks that are consistent with the original image size. At the same time, the second branch is to add another output based on the object detector. In addition to predicting the category confidence scores and bounding box coordinates, a series of mask coefficients need to be predicted for each instance. Finally, to produce the instance masks, the branch that produces the prototype masks and the branch that produces the mask coefficients are combined in linear combinations, and the results of the combinations are nonlinearized using the Sigmoid function to obtain the final masks.

In this paper, YOLACT++ is selected to split the pixels of the two types of objects in the close-up image, one is the whole bunch of cherry tomatoes containing peduncle and fruit, and the other is only the peduncle. When processing image data, we have labeled the collected data set accordingly. The network will learn the feature information of these two types of objects in the training phase, and finally, be able to make robust predictions in the test set.

Background interference filter

In some cases, even if the camera is close enough to the whole bunch of cherry tomatoes to be picked for image acquisition, multiple bunches of tomatoes may appear in the close-up image or other peduncles are misidentified (see Fig. 8), because the other bunches of tomatoes or peduncles are too close to the tomatoes to be picked. At this time, YOLACT++ detects multiple bunches of tomatoes in the field of view simultaneously. But in subsequent processing, only one bunch of tomatoes is needed to find its corresponding peduncle, so the masks for the remaining whole bunches of tomatoes and the peduncles need to be suppressed.

Based on this requirement, the output of YOLACT++ is post-processed, as shown in Fig. 9. First, the output obtained is the category, confidence score, bounding box, and binary mask of each detected instance. If bunches of tomatoes and peduncle instances are detected in the image, all binary masks are stored separately according to categories ('peduncle' binary masks and 'fruit' binary masks). Next, we find the mask with the highest confidence score in the 'fruit' binary masks, which is usually the mask for the whole bunch of cherry

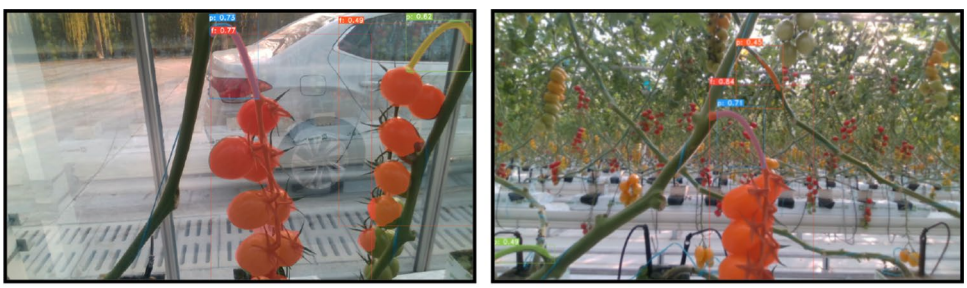


Fig. 8 Multiple bunches of cherry tomatoes in the field of view: **a** multiple bunches of tomatoes appear in an image; **b** multiple peduncles are detected

(**a**) Multiple bunches appear in the image (**b**) Multiple peduncles are detected

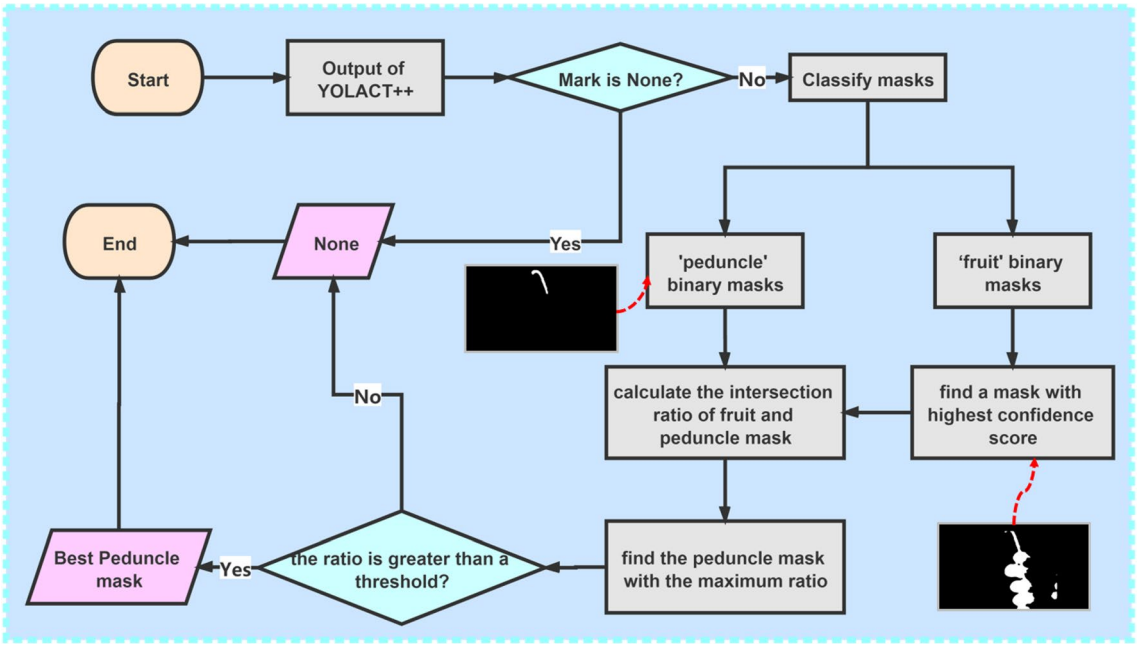


Fig. 9 The process of obtaining the mask for the most suitable peduncle

tomatoes that can be picked. After obtaining the 'fruit' binary mask with the highest confidence score, multiply it with the 'peduncle' binary masks (each binary mask is a matrix of the same size) to have the new masks and calculate the intersection ratio of fruit and peduncle masks (the ratio of the number of elements with a value of 1 in the new mask to the number of elements with a value of 1 in the peduncle mask). Finally, find the peduncle mask with the maximum ratio and judge whether the ratio is greater than a threshold. This method is inspired by the Non-Maximum Suppression (NMS) algorithm and can find the peduncle corresponding to the pickable bunch of cherry tomatoes [26].

Cutting point and pose estimation

After obtaining the peduncle mask corresponding to the pickable bunch of tomatoes by the two-stage peduncle segmentation method above, the appropriate cutting point and pose of the peduncle are estimated using a geometric model so that the end effector can reach the correct cutting point position in a suitable pose for a harvesting operation.

The method of estimating the cutting point of the peduncle is illustrated in Fig. 10. For the segmented peduncle mask, the pixel value of the mask position representing the peduncle is set to 1 (white) and the rest of the region is set to 0 (black). Then, all pixel points set to 1 are found and the curve is fitted using the least-squares polynomial by calling the function 'polyfit' in NumPy. Afterward, the minimum and maximum coordinates of

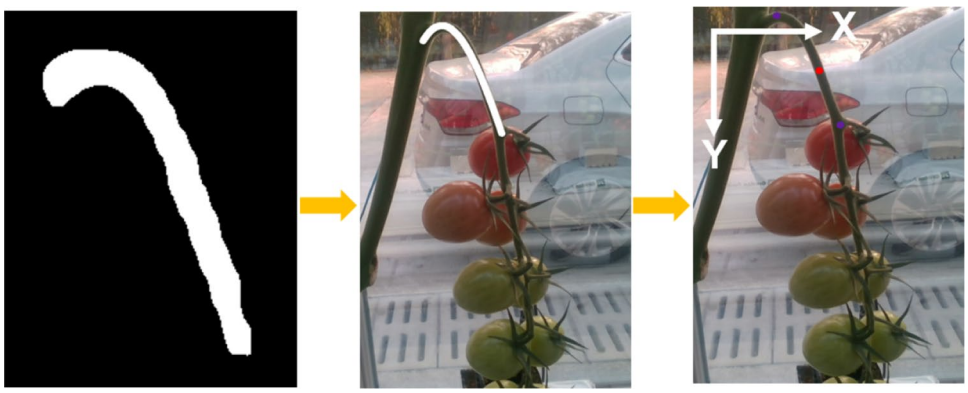


Fig. 10 Estimation of peduncle cutting point

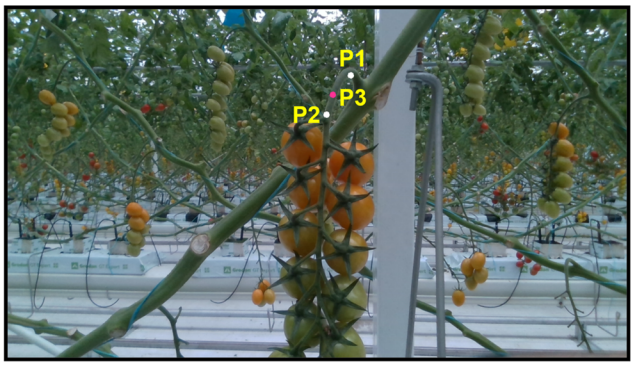
the contour in the Y-direction are obtained by searching the upper and lower extreme value points of the peduncle mask whose coordinates corresponding to the X-direction are found, and the two extreme value points are regarded as the upper and lower points of the peduncle. Ultimately, the coordinate of the cutting point of the peduncle in the Y-direction is obtained by averaging the highest and lowest points of the peduncle, while the coordinates must be satisfied on the fitted curve, and calculate the coordinate in the X-direction inverting the polynomial using formula (1):

$$y_{\text{cutting point}} = \alpha * x^{n}_{\text{cutting point}} + \beta * x^{n-1}_{\text{cutting point}} + \cdots + \gamma, \tag{1}$$

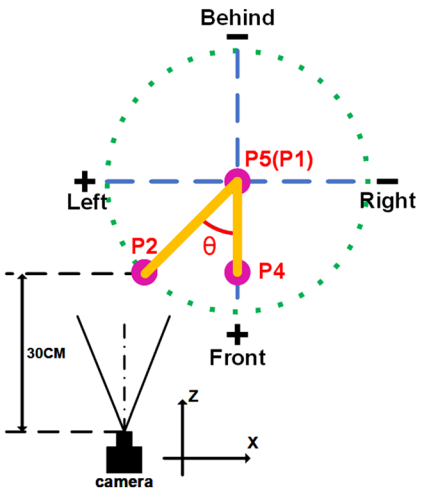
where α, β and γ are coefficient obtained by fitting the peduncle mask, the polynomial power(n) is set to 2, $y_{\text{cutting point}}$ is the average of the highest and lowest points, and $x_{\text{cutting point}}$ is the coordinate of the cutting point to be solved in the X-direction.

The robot can easily harvest bunches of tomatoes with the peduncles vertically in front of them. However, the peduncles of tomatoes in production greenhouses are often irregular and the success rate of picking bunches of tomatoes in a fixed pose is low, so a vision system is needed to estimate the pose of the peduncles and drive the robot to be able to adapt to the pose of the peduncles for picking. A mathematical geometric model is, therefore, established based on three points to find the pose of the peduncle after obtaining the upper point ($P_1 = (x_1, y_1, z_1)$), the lower point ($P_2 = (x_2, y_2, z_2)$), and the cutting point ($P_3 = (x_3, y_3, z_3)$) of the peduncle (see Fig. 11a), of which 3D coordinates are obtained from the point cloud map.

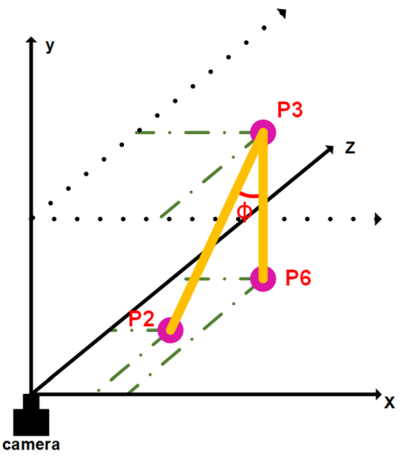
In the greenhouse, except in cases where the peduncle is shaded by the main stem, the peduncles of tomatoes vertically downward, tilt to the left or the right. Therefore, it is necessary to estimate the angles of yaw and



(**a**) Results of three points predicted



(**b**) Top view of the model used to estimate peduncle pose



(**c**) Three-dimensional view of the model used to estimate peduncle pose

Fig. 11 Estimation of the peduncle pose: **a** the results of three points predicted; **b** the top view of the model used to estimate the yaw angle θ of the peduncle; **c** the three-dimensional view of the model used to estimate the pitch angle φ of the peduncle

pitch rotations of peduncles relative to the ideal vertical downward situation. As shown in Fig. 11b, the upper point P_1 and the lower point P_2 are used to calculate the angle of yaw rotation, and P_1 and P_2 are projected to form points $P_4(P_4 = (x_1, y_2, z_2))$ and $P_5(P_5 = (x_1, y_2, z_1))$ with angle θ being the angle of yaw rotation for the peduncle. Then, the projection of the lower point P_2 and the cutting point P_3 is used to get point $P_6(P_6 = (x_3, y_2, z_3))$, and the angle ψ is the angle of pitch rotation for the peduncle (see Fig. 11c). P_2, P_4, P_5 and P_6 are all in the same plane parallel to xoz plane. The formulas (2) and (3) to solve the angles of yaw and pitch rotations for peduncles is as follows:

$$\theta = \cos^{-1}\frac{P_4P_5}{P_2P_5} = \cos^{-1}\frac{\sqrt{(z_1 - z_2)^2}}{\sqrt{(x_1 - x_2)^2 + (z_1 - z_2)^2}} \tag{2}$$

$$\psi = \cos^{-1}\frac{P_3P_6}{P_2P_3} = \cos^{-1}\frac{\sqrt{(y_1 - y_3)^2}}{\sqrt{(x_1 - x_3)^2 + (y_1 - y_3)^2 + (z_1 - z_3)^2}} \tag{3}$$

Experiments and discussion

All field experiments in this paper are conducted in Caoyang agricultural technology demonstration greenhouse, Xiangcheng District, Suzhou, China. Before our experiments, the staff trimmed the leaves of tomatoes in the area to be matured, which is part of gardening, in order to ventilate light and reduce pests and diseases. Therefore, we do not consider the situation where the leaves occlude tomatoes and peduncles. Besides, we choose tomato varieties with long peduncles as our research object, and peduncles of tomato bunches are usually not blocked. Therefore, we acquiesce that peduncles of tomatoes appear in the images collected by the camera, and the detection of occluded peduncles is not in the study of this paper. In this field experiments, we focus on the detection effect of tomato bunches in the images captured at a long distance, the segmentation effect of peduncles in the close-up images and the accuracy of the peduncle pose estimation.

In this demonstration, an industrial computer with an AMD Ryzen5-2600X CPU, an NVIDIA GeForce RTX 2060 SUPER GPU with 8 GB memory, and Windows 10 system is used for training the object detection algorithm and instance segmentation algorithm. A laptop with an Intel Core i7-8750H, an NVIDIA GeForce GTX 1080 is used for testing in the greenhouse.

Table 1 Number of tomato bunch datasets captured at the long-distance

Training sets	Validation sets	Test sets	Total
496	166	166	828

Table 2 Testing results of the YOLOv4 model

Methods	Speed (ms)	Recall	Precision	F1
YOLOv3	28.7	95.6	**95.9**	**95.7**
YOLOv4	34.4	95.4	92.2	93.8
YOLOv4-Tiny	**9.1**	**96.2**	92.7	94.4

Bold values indicate better results than other methods under the current index

Results of tomato bunch detection

As the representative work of single-stage object detectors, YOLO series achieves a balance between accuracy and speed. With the appearance of YOLOv3 and YOLOv4 detectors, the accuracy of real-time object detection algorithms has been continuously updated, achieving the best balance between speed and accuracy. Whereas the real-time requirement is important in the practical application of picking robots, so we compare several single-stage detectors to find a real-time detector with the best accuracy for detecting tomato bunches.

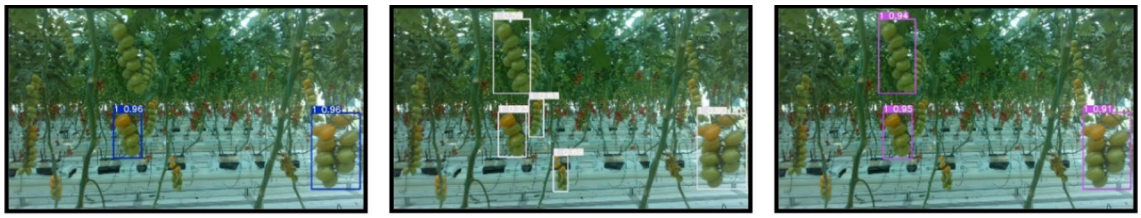
In this paper, experiments are carried out to compare the performance of the tomato bunch detection based on the YOLOv3, YOLOv4, and YOLOv4-Tiny. A total of 828 images are divided into training sets, validation sets, and test sets, as shown in Table 1. These models are trained for 600 epochs with a batch size of 8, an initial learning rate of 0.00261, a momentum of 0.949, and a decay of 0.0005.

The prediction results can be divided into the following four types: TP, actual true samples are predicted to be positive; FP, actual false samples are predicted to be positive; TN, actual false samples are predicted to be negative and FN, actual true samples are predicted to be negative. Speed, recall, precision, and F_1 score are used to evaluate the model performance. The indexes for evaluation of the trained model are defined as follows:

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}} \tag{4}$$

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}} \tag{5}$$

$$F_1 = \frac{2 \times \text{Recall} \times \text{Precision}}{\text{Recall} + \text{Precision}}. \tag{6}$$



(a) YOLOv3 (b) YOLOv4 (c) YOLOv4-Tiny

Fig. 12 Detection results of different methods: **a** YOLOv3; **b** YOLOv4; **c** YOLOv4-Tiny

Table 2 shows the comparison results of tomato bunch detection by three methods. Compared to YOLOv3 and YOLOv4, YOLOv4-Tiny is the fastest, taking only 9.1 ms to detect an image resized to 512. YOLOv3 has the best accuracy of 95.9%, which is 3.7 and 3.2% higher than YOLOv4 and YOLOv4-Tiny, respectively. In terms of recall, YOLOv4-Tiny achieves 96.2% that is higher than the other two methods. Considering the recall and precision together (F1 score), YOLOv3 performs best. However, YOLOv4-Tiny is 19.6 ms faster than YOLOv3 at the expense of a 1.3% F1 score, which means that YOLOv4-Tiny has lower hardware requirements and better real-time performance for a similar accuracy.

Discussion of tomato bunch detection

Figure 12 shows the prediction results of different methods tested on one of the samples. All the tomato bunches in this image have been detected by YOLOv4-Tiny, while YOLOv3 misses one and YOLOv4 detects two tomato bunches in the background as well. Such a result, also seen in the other test samples, is the reason for the lower accuracy of YOLOv4 compared to the other two methods. The reason for this phenomenon can be that YOLOv4 has better performance for detecting multi-scale targets, with a deeper network and a better multi-scale feature map fusion method enabling tomato bunches in the background to be detected as well. The tomato bunch in the background and the tomato bunch in the foreground have some similar characteristics, but their scales are different, fusing the output of different layers will destroy this characteristic of scale changes. The test results of YOLOv4-Tiny suggest that for simpler classification, such as detecting a single target of tomato bunches, a lightweight detector with the shallower depth may be better, especially the speed advantage will be more obvious.

By comparison, we choose YOLOv4-Tiny as the object detector to detect and coarsely locate tomato bunches while the robot is moving.

Table 3 Number of tomato datasets captured at the close-up distance

Training sets	Validation sets	Test sets	Total
410	170	120	700

Table 4 Testing results of the YOLACT++model

Backbone	FPS	mAP	mAP50	mAP75
R-50-FPN	**17.7**	61.9	94.2	71.0
R-101-FPN	9.2	**73.1**	**96.9**	**87.3**

Bold values indicate better results than other methods under the current index

Results of peduncle cutting point positioning

After the tomato bunch is detected, the robot arm approaches a specific whole bunch of tomatoes for a close-up shot (at this position the camera's field of view can cover the whole bunch of tomatoes and make it appear in the middle of the image), then we applied an instance segmentation model to find the pixel mask of the tomato bunch and corresponding peduncle. Therefore, we need to verify the effect of instance segmentation and use the YOLACT++model with different backbones to test the effect of instance segmentation. The instance segmentation model YOLACT++receives 410 images as the training sets and 170 images as the validation sets (see Table 3), adjusting the resolution from 1280×720 pixels to 700×700 as the input to the network. In this paper, we don't consider the situation that the peduncle is blocked. If the camera cannot see the peduncle corresponding to the tomato bunch, the robot will not pick it. Therefore, only close-up images that peduncles have not been blocked are in the dataset. Due to GPU memory constraints, the batch size is had to be set to 3, which affects the performance of the network to some extent. The momentum and weight decay are, respectively, set to 0.9 and 0.0005. The model is trained for 600 epochs with an initial learning rate of 10^{-3}, which is then divided by 10 after 100 epochs.

The trained model parameters are loaded into YOLACT++to test 120 images in the test set. We introduce

FPS (Frames Per Second) to evaluate the running speed of the model, and AP (Average Precision) is used to evaluate the quality of the mask predicted by the model. As shown in Table 4, the YOLACT++network with R-101-FPN backbone improves mAP by 11.2 compared to the backbone being R-50-FPN. However, the model using the R-101-FPN backbone is 9.2 FPS, while the model using R-50-FPN is able to achieve 17.7 FPS. The experimental results show that using a backbone with more layers improves the accuracy and precision of the model, but deeper backbones consume more time. In this paper, the close-up peduncle segmentation task requires better accuracy, and a deeper backbone network can bring significant accuracy improvements, hence the backbone of YOLACT++chooses R-101-FPN.

Then, we use the peduncle mask predicted YOLACT++to fit a curve, and find the key points on the curve. The prediction result of the field experiment is shown in Fig. 13. In this way, three key points of the fruit corresponding to the fruit stem can be found. In 120 tests conducted, the algorithm can accurately find the key points of the corresponding peduncle in 112 images and misses 8 peduncles.

Discussion of peduncle cutting point positioning

In the first detection, we have balanced the accuracy and speed of the object detector, and have higher requirements for real-time performance. When considering the peduncle segmentation algorithm, we pay more attention to the accuracy of detection. The reason is that the robot is moving at the first detection. Only when the tomato bunch is detected, the robot stops, then the camera mounted on the robot arm approaches the tomato bunch for a second detection. In the second detection, the robot is static, so the accuracy of peduncle segmenting is given priority.

Table 5 Analysis of the failure detection

Reasons for failure of detection	Number in test sets
Disturbed by neighboring tomatoes	1
Incomplete or undetectable masks	6
Light interference	1

Fig. 13 Field experimental results of positioning the key points of peduncles

According to the experimental results, we analyzed the failure reasons for peduncle cutting point positioning failure. Table 5 shows the reasons why the algorithm does not find the key points correctly. In one of the images, two fruits and a peduncle are detected, and the masks of the fruit with the highest confidence level and the peduncle do not intersect, so this detection fails. The absence of any peduncles in the predictions of the YOLACT + + model is the cause of detection failure in the six test images. Finally, the presence of strong sunlight in the background also makes a detection failure. By analyzing and summarizing the reasons for the failures, we find that when the peduncle is vertically downward and overlaps the main stem, the predicted mask will be incomplete or undetectable. The possible reason is that there are too few data samples of peduncles that overlap the main stem vertically so that the network does not distinguish the characteristics of this different object well. In the future, we will add these samples that may cause failure in the training set and validation set to improve the detection effect.

Results of pose estimation

In Sect. 3.4, we try to find three key points of fruit peduncles, based on which a geometric model is established to estimate the angles of yaw and pitch rotations of peduncles relative to the ideal vertical downward situation, allowing the picking robot to adaptively adjust the end-effector to grip the peduncle in a more reasonable pose. For the performance evaluation, we test 30 different bunches of tomatoes in a greenhouse (all 30 test data can obtain peduncle masks) and then measure two angles of the peduncle using an angle ruler. Finally, the error between the predicted angle and the measured angle is analyzed as the basis for evaluating the performance of the pose estimation algorithm. Table 6 shows the pose prediction and measurement results of the

Table 6 Testing results on the estimation of the peduncle pose

No.	L (+) /R (−)	F (+)/B (−)	Estimated yaw angle θ(°)	Estimated pitch angle ϕ(°)	Measured yaw angle θ (°)	Measured pitch angle ϕ(°)
1	−	+	26.0	29.9	22.4	32.0
2	−	−	53.9	32.3	60.3	32.0
3	−	−	37.0	21.6	33.6	23.5
4	−	−	72.0	26.9	69.9	35.4
5	−	+	47.3	25.1	43.7	31.6
6	−	+	85.4	36.2	81.2	33.6
7	−	−	58.8	29.3	56.2	31.9
8	−	−	33.4	18.8	33.1	22.7
9	−	−	52.7	42.6	57.2	55.2
10	−	−	80.9	24.8	82.2	29.6
11	−	−	43.6	23.9	53.3	30.6
12	−	−	84.2	32.1	83.7	31.5
13	−	+	42.4	16.8	44.5	17.5
14	−	−	68.9	23.9	77.1	16.4
15	−	−	31.7	27.5	35.4	37.8
16	−	−	22.1	17.1	32.7	26.5
17	−	+	90.0	30.3	85.1	29.2
18	−	−	47.7	21.1	58.9	20.9
19	−	+	82.9	43.6	85.8	38.8
20	−	+	79.2	35.7	87.3	35.9
21	−	+	34.5	15.7	28.3	26.8
22	−	+	59.2	27.8	57.4	30.5
23	+	+	38.3	15.9	28.7	23.7
24	−	+	86.6	22.8	80.5	31.5
25	−	+	68.9	33.4	75.1	28.5
26	−	+	44.8	15.9	43.0	16.0
27	−	+	73.4	29.9	74.3	27.5
28	+	+	73.3	26.9	63.0	31.4
29	+	+	77.3	29.9	66.7	37.5
30	−	+	77.6	32.8	75.7	38.3

corresponding peduncle of 30 different groups of bunches of tomatoes. L(+) means that the peduncle is inclined to the left and R(-) means that the peduncle is inclined to the right. As shown in Fig. 11, if the coordinate of Point P_1 in the X-direction is greater than Point P_2, it is considered to be inclined to the left. F(+) means that it is inclined forward and B(−) means that it is inclined backward. If the coordinate of Point P_1 in the Z-direction is greater than Point P_2 (see Fig. 11), it is considered to be inclined to the forward. The pose predictions of the 30 groups of peduncles can well predict whether the fruit peduncles are forward or backward, left or right. Yaw angle and Pitch angle are θ and φ which are defined in Fig. 11b and c respectively. Estimated angles are calculated by the pose estimation algorithm and measured angle is measured by an angle ruler. Since the measured angle is manually measured in the greenhouse using an angle ruler, there is a measurement error of ±5°.

During the experiment, we excluded the effects caused by missing data in the point cloud images, and the angle estimation results were all based on the successful predictions of YOLACT++. The results of the 30 sets of peduncle pose predictions are analyzed to obtain prediction errors for yaw and pitch angles as shown in Fig. 14. The mean error of the predicted yaw angle θ is 4.98° with a maximum error of 11.2°, while the mean error of the pitch angle φ is 4.75° with a maximum error of 12.6°. The errors above are all based on the fact that the segmentation algorithm can correctly segment the pixel area of the peduncles in 93.3% of cases, thus being able to find the correct three key points on the fitted curve.

Discussion of pose estimation

The peduncle pose estimation algorithm constructs a polynomial curve by fitting the peduncle mask predicted by YOLACT++ to obtain three key points and finds the corresponding three-dimensional coordinates and then establishes a mathematical model about these three points to obtain the relative posture of the peduncle(the pose is relative to the ideal vertical downward peduncle). The error of the pose estimation depends on the accuracy of the prediction of the peduncle mask and the accuracy of the point cloud obtained by the RGB-D camera. We have done a separate experimental analysis on the results of peduncle masks predicted by YOLACT++. Therefore, most of the measurement errors above in pose estimation are caused by inaccurate point cloud information. Also, to reduce the impact caused by missing holes, the adjacent point cloud data is used to fill the missing data, which further increasing the error in the point cloud data.

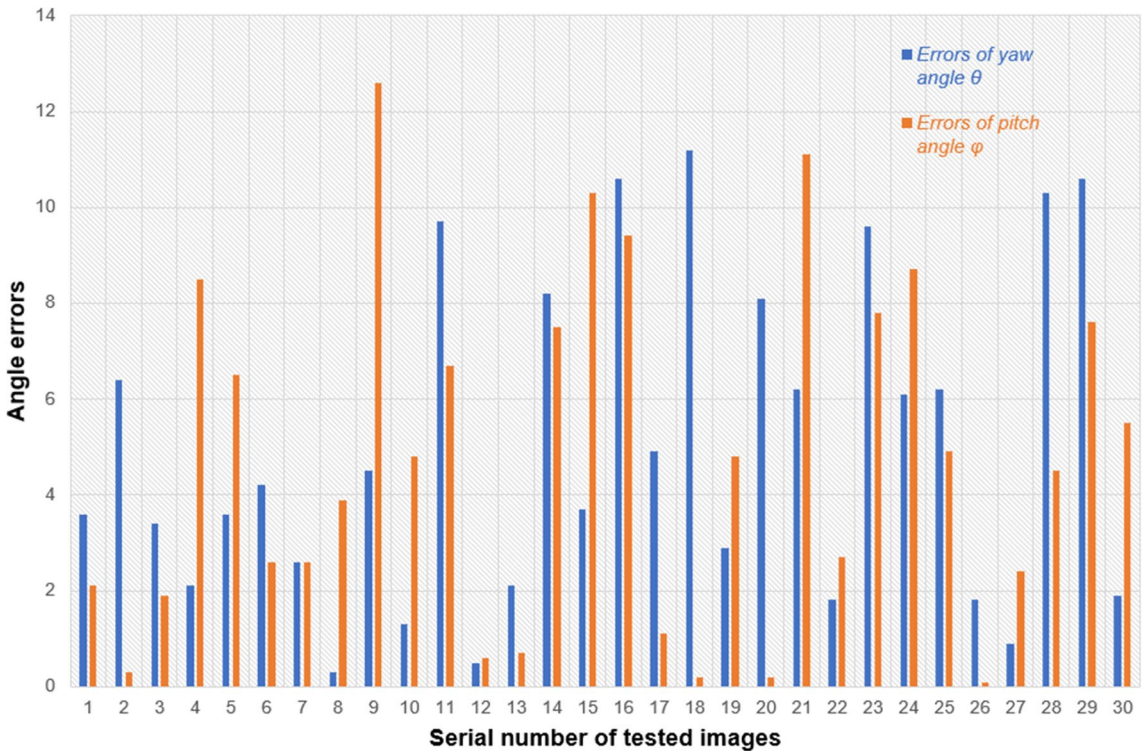


Fig. 14 Error analysis of the estimated peduncle pose angle for the 30 tested images

Conclusions

In this paper, we develop a two-stage peduncle segmentation method and establish a geometric model of the peduncle mask corresponding to the bunch of tomatoes to find the cutting point and pose. The performance of the proposed method is tested in Sect. 3 and the results demonstrate the effectiveness of using this method to find the cutting point and the cutting pose of the peduncle in the greenhouse.

YOLOv4-Tiny is chosen as the object detector to find bunches of tomatoes in real-time as the robot moves, with a detection time of 0.0091 s per frame and an accuracy of 92.7%. After YOLOv4 detects tomatoes, the camera moves sequentially to the front of the tomato to take a close-up image. YOLACT++is then selected as the network for segmenting the peduncle mask, and in test sets, the mAP is 73.1 with a time speed of 0.109 s per frame. Finally, a geometric model is developed to estimate the cutting point and pose of the peduncle based on the peduncle mask segmented by YOLACT++. To find the peduncle mask corresponding to the bunch of tomatoes, the ratio of the intersection of the bunch of tomatoes mask and the peduncle mask is calculated, and only the peduncle mask with intersection ratios greater than a set threshold is output. Finally, a mathematical geometric model is developed using the three key points to predict the pose of the peduncle, with an average error of 4.98° in yaw angle and 4.75° in pitch angle over the 30 sets of tests.

When predicting peduncle masks, the segmentation success rate is low for peduncles that grow vertically downward. The reason is that this kind of peduncle data accounts for a relatively small proportion in the training sets. Therefore, we will expand our data set for better performance in the future. Besides, the quality of the point cloud data output from RGB-D cameras affects the accuracy and stability of pose estimation. Compare different RGB-D cameras and select the one more suitable for use in the greenhouse.

Future work is to design an end-effector and build a harvesting robot system. Using the peduncle cutting point positioning and pose estimation methods proposed in this paper, the gripper mounted at the end of the robotic arm will be able to pick bunches of tomatoes adaptively and the performance of the robotic system will be evaluated.

Author contributions JR: methodology, software, validation, writing. GD: methodology, software, validation. PW: conceptualization, funding acquisition, methodology, validation, writing.

Funding This research was funded by National Key Research and Development Program of China (2017YFD0701502).

Declarations

Conflict of interest The authors declare no conflict of interest.

Availability of data and materials Not applicable.

Code availability Not applicable.

Humans and/or animal rights Not applicable.

Ethics approval Not applicable.

Consent to participate Not applicable.

Consent for publication Not applicable.

References

1. Fu L, Wang B, Cui Y, Su S, Gejima Y, Kobayashi T (2015) Kiwifruit recognition at nighttime using artificial lighting based on machine vision. Int J Agric Biol Eng 8(4):52–59
2. Zhao Y, Gong L, Huang Y, Liu C (2016) A review of key techniques of vision-based control for harvesting robot. Comput Electron Agric 127:311–323
3. Bac CW, Henten EJ, Hemming J, Edan Y (2014) Harvesting robots for high-value crops: state-of-the-art review and challenges ahead. J Field Robot 31(6):888–911
4. Kamilaris A, Prenafeta-Boldú FX (2018) Deep learning in agriculture: a survey. Comput Electron Agric 147:70–90
5. Koirala A, Walsh KB, Wang Z, McCarthy C (2019) Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of "MangoYOLO." Precis Agric 20(6):1107–1135
6. Tian Y, Yang G, Wang Z, Wang H, Li E, Liang Z (2019) Apple detection during different growth stages in orchards using the improved YOLO-V3 model. Comput Electron Agr 157:417–426
7. Liu G, Nouaze JC, Mbouembe PLT, Kim JH (2020) YOLO-Tomato: a robust algorithm for tomato detection based on YOLOv3. Sensors (Basel) 20(7):2145
8. Birrell S, Hughes J, Cai J, Iida F (2020) A field-tested robotic harvesting system for iceberg lettuce. J Field Robot 37(2):225–245
9. Perez-Borrero I, Marin-Santos D, Gegundez-Arias ME, Cortes-Ancos E (2020) A fast and accurate deep learning method for strawberry instance segmentation. Comput Electron Agric 178:105736
10. Song Z, Zhou Z, Wang W, Gao F, Fu L, Li R, Cui Y (2021) Canopy segmentation and wire reconstruction for kiwifruit robotic harvesting. Comput Electron Agric 181:105933

11. Chen W, Lu S, Liu B, Li G, Qian T (2020) Detecting citrus in orchard environment by using improved YOLOv4. Sci Program Neth 2020:13
12. Sa I, Lehnert C, English A, McCool C, Dayoub F, Upcroft B, Perez T (2017) Peduncle detection of sweet pepper for autonomous crop harvesting-combined color and 3-D information. IEEE Robot Autom Let 2(2):765–772
13. Luo L, Tang Y, Lu Q, Chen X, Zhang P, Zou X (2018) A vision methodology for harvesting robot to detect cutting points on peduncles of double overlapping grape clusters in a vineyard. Comput Ind 99:130–139
14. Yoshida T, Fukao T, Hasegawa T (2018) Fast detection of tomato peduncle using point cloud with a harvesting robot. J Robot Mechatron 30(2):180–186
15. Yoshida T, Fukao T, Hasegawa T (2020) Cutting point detection using a robot with point clouds for tomato harvesting. J Robot Mechatron 32(2):437–444
16. Barth R, Hemming J, Van Henten EJ (2019) Angle estimation between plant parts for grasp optimisation in harvest robots. Biosyst Eng 183:26–46
17. Yu Y, Zhang K, Liu H, Yang L, Zhang D (2020) Real-time visual localization of the picking points for a ridge-planting strawberry harvesting robot. IEEE Access 8:116556–116568
18. Liang CX, Xiong JT, Zheng ZH, Zhong Z, Li ZH, Chen SM, Yang ZG (2020) A visual detection method for nighttime litchi fruits and fruiting stems. Comput Electron Agr 169:105192
19. Liu JZ, Yuan Y, Zhou Y, Zhu XX, Syed TN (2018) Experiments and analysis of close-shot identification of on-branch citrus fruit with RealSense. Sensors (Basel) 18(5):23
20. Redmon J, Divvala S, Girshick R, Farhadi (2016) A You only look once: unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 779–788
21. Redmon J, Farhadi A (2017) YOLO9000: better, faster, stronger. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp 7263–7271
22. Redmon J, Farhadi A (2018) YOLOv3: an incremental improvement. 1804.02767
23. Bochkovskiy A, Wang C, Liao H (2020) Yolov4: Optimal speed and accuracy of object detection.
24. Bolya D, Zhou C, Xiao F, Lee Y (2019) Yolact: real-time instance segmentation. In: Proceedings of the IEEE/CVF international conference on computer vision, pp 9157–9166
25. Bolya D, Zhou C, Xiao F, et al (2019) YOLACT++: better real-time instance segmentation [J]
26. Neubeck A, Van Gool L (2006) Efficient non-maximum suppression. In: Eighteenth international conference on pattern recognition (ICPR'06), pp 850–855