Article

Deep-learning-based in-field citrus fruit detection and tracking

Wenli Zhang1,*, Jiaqi Wang1, Yuxin Liu1, Kaizhen Chen1, Huibin Liu2, Yulin Duan2, Wenbin Wu2, Yun Shi2 and Wei Guo3,*

1 Information department, Beijing University of Technology, Beijing, 100022, China
2 Institute of Agricultural Resources and Regional Planning, Chinese Academy of Agricultural Sciences, Beijing 100081, China
3 International Field Phenomics Research Laboratory, Institute for Sustainable Agro-ecosystem Services, The University of Tokyo, Tokyo 188-0002, Japan
*Corresponding authors. E-mail: zhangwenli@bjut.edu.cn, guowei@g.ecc.u-tokyo.ac.jp

Abstract

Fruit yield estimation is crucial for establishing fruit harvest and marketing strategies. Recently, computer vision and deep learning techniques have been used to estimate citrus fruit yield and have exhibited notable fruit detection ability. However, computer-vision-based citrus fruit counting has two key limitations: inconsistent fruit detection accuracy and double-counting of the same fruit. Using oranges as the experimental material, this paper proposes a deep-learning-based orange counting algorithm using video sequences to help overcome these problems. The algorithm consists of two sub-algorithms, OrangeYolo for fruit detection and OrangeSort for fruit tracking. The OrangeYolo backbone network is partially based on the YOLOv3 algorithm, which has been improved upon to detect small objects (fruits) at multiple scales. The network structure was adjusted to detect small-scale targets while enabling multiscale target detection. A channel attention and spatial attention multiscale fusion module was introduced to fuse the semantic features of the deep network with the shallow textural detail features. OrangeYolo can achieve mean Average Precision (mAP) values of 0.957 in the citrus dataset, higher than the 0.905, 0.911, and 0.917 achieved with the YOLOv3, YOLOv4, and YOLOv5 algorithms. OrangeSort was designed to alleviate the double-counting problem associated with occluded fruits. A specific tracking region counting strategy and tracking algorithm based on motion displacement estimation were established. Six video sequences taken from two fields containing 22 trees were used as the validation dataset. The proposed method showed better performance (Mean Absolute Error (MAE) = 0.081, Standard Deviation (SD) = 0.08) than video-based manual counting and produced more accurate results than the existing standards Sort and DeepSort (MAE = 0.45 and 1.212; SD = 0.4741 and 1.3975).

Introduction

With the introduction of computer vision in the smart agriculture domain, fruit detection and tracking technologies, which can help to obtain fruit yield statistics and assist with automatic fruit picking and automated orchard management, have become important areas of research. Anderson et al. [1] pointed out that yield prediction can help growers with farming decisions for the season, especially in regard to labor allocation for harvesting, fruit transportation, and storage methods. An experienced agronomist can use these resources and combine this information with knowledge of fruit tree physiology to provide advice on the current season and make recommendations on management practices for optimizing orchard development. Anderson et al. [1] also pointed out that fruit yield prediction can now be broadly classified into five methods. Fruit yield estimation is a labor-intensive, monotonous, and tedious task. However, the use of computer vision techniques can quickly detect fruits and thus predict the overall yield of an orchard, and fruit yield prediction methods based on computer vision with deep learning have shown considerable potential in recent years [2]. Nonetheless, computer-vision-based fruit counting has two limitations: inconsistent fruit detection accuracy and double-counting of the same fruit.

To solve the problem of fruit detection accuracy, many researchers have proposed deep-learning-based detection algorithms. Fruit detection algorithms can be broadly classified into detection based [3–10] and segmentation based [11–14] algorithms. However, fruits appear at multiple scales, and it is difficult to extract the effective features from both small- and large-scale fruits in order to detect fruits accurately. Although existing techniques can effectively enhance the accuracy of fruit detection, the presence of small-scale targets has not been considered. In particular, Mu et al. [5] developed a tomato detection model based on a faster region-based convolutional neural network (R-CNN) with ResNet-101 and deep learning approaches to automatically detect intact green tomatoes, regardless of fruit growth.
stage or the presence of occlusions. Detection on the test dataset corresponded to an average precision of 87.83% (intersection over union ≥0.5). The authors reported that large tomatoes tended to have high scores, whereas some small tomatoes had scores of less than 0.2. Jordi [13] used a mask R-CNN instance segmentation neural network to detect apples, and the results showed that some small apples were difficult to detect. Existing techniques typically ignore the problem of small fruit size and severely overlapping fruits. Therefore, although existing algorithms can achieve high-precision fruit detection in scenarios with simple backgrounds and few fruit targets, it is difficult to achieve satisfactory results in actual orchards because the fruit sizes tend to be variable. In addition, some researchers have found that matching the perceptual field to the target scale can improve the detection accuracy for small targets [25, 26]. The Receptive Field is the area size of the pixel points on the output feature map from each layer of the convolutional neural network mapped on the input image. The attention mechanism can enhance the effective information, such as the weight of features, and reduce the influence of invalid information, thereby enhancing the performance of computer vision tasks [35, 36]. This approach can be adopted for fruit detection applications to improve detection accuracy.

To avoid double-counting of the same fruit, some researchers have proposed solutions based on static images [3, 4, 11, 12, 15, 16] and video sequences [17–23]. The video-sequence-based counting method, in which fruit images are collected from multiple viewpoints, thereby enabling the observation of more fruit, shows promise for enabling fruit counting algorithms to be used in actual orchard fields. Multi-objective tracking algorithms are the key to solving the counting problem. Among them, Sort [28] and DeepSort [29] combine the Kalman filter and Hungarian algorithm to accomplish the multi-target tracking task, and they have a wide range of applications in the field of foot traffic counting. Although these methods have been validated in the MOT dataset [38] and have yielded satisfactory tracking results, they do not work well for fruit counting. The main reason is the dense fruit growth and heavy fruit overlap in real orchards. Notably, Wang et al. [17] combined MangoYolo and the Hungarian algorithm to track and count mangoes in a video of mango trees. They demonstrated experimentally that the counting method based on video sequences was considerably more accurate than counting based on static images [3]. In addition, some researchers have used 3D techniques for counting. For example, Gan et al. [23] used the Global Navigation Satellite System, an inertial measurement unit (IMU), and LIDAR to achieve simultaneous localization and used an extended Kalman filter to improve the reliability and accuracy of the localization. However, the 3D counting method requires expensive equipment, and the system is highly complex, making it difficult to obtain useful 3D reconstruction results.

In summary, the counting method based on video sequences is an effective and economical solution that can be successfully implemented. However, in actual orchards, oranges grow densely, and fruits are thus easily occluded by one another or by the foliage. The occluded state of the oranges may vary when fruit images are acquired from different viewpoints. In this case, the abovementioned video-based counting methods may exhibit a tracking target loss owing to the complex occlusion present in global video sequences.

In this paper, using oranges as the target fruit, a deep-learning-based in-field orange fruit counting method is proposed, which includes the fruit detection algorithm OrangeYolo and the fruit tracking algorithm OrangeSort. OrangeYolo was modified from the object detection algorithm Yolov3 [24]. The network architecture was adapted to enable small-scale target detection by matching the feature map receptive field [25] with the target scale. A dual-attention multiscale fusion module was also introduced based on channel attention and spatial attention. OrangeSort is modified from the object tracking algorithm Sort [28]. A motion displacement estimation tracking algorithm was established, and a specific tracking region counting strategy was formulated to avoid the double-counting problem caused by the complex occlusion of fruits in global video sequences.

Method
Overall structure of the proposed algorithm
This study uses the same dataset acquisition method and criteria that were adopted in our previous work [38]. A field rover equipped with the DJI Osmo Action camera (DJI Technology Co., Ltd., Shenzhen, China) was used to acquire the video by moving at a uniform speed (2 m/s) along the tree rows.

DJI Osmo Action camera: The camera shooting direction was perpendicular to the tree rows. The camera has a frame rate of 60 fps, a resolution of 1920 × 1080, a 1/2.3 CMOS sensor, and a field of vision (FOV) of 145° F2.8.

Our self-developed field rover: For better operation in the orchard, we have customized a field rover. It has a length, width, and height of 120 × 90 × 70 cm, is motor driven through an ROS control interface, and has an automatic cruise function, which allows it to travel through the orchard according to a set track and speed. The functions are close to those of existing orchard work assistance robots already in agricultural use.

Figure 1 shows a schematic of the data acquisition method and field rover, as well as the overall structure of the fruit detection and tracking counting algorithm, which consists of the OrangeYolo and OrangeSort modules. The input is the video sequence from the orchard field. First, OrangeYolo detects the fruit frame-by-frame. The information for each detected fruit is fed to OrangeSort, which assigns a unique tracking ID to each
individual fruit and correlates the same fruit data in different video consequences. Finally, the number of tracking IDs is counted.

**OrangeYolo**

The network structure of the OrangeYolo detection algorithm is shown in Figure 1(a). First, DarkNet53 [24] was used as the algorithm backbone network to extract the target features. DarkNet53 has been used by many researchers as a backbone network owing to its simplicity, efficiency [30, 31], and effectiveness in enhancing detection performance without modifying network structure and incurring additional computational costs. Second, in order to be able to detect targets at various scales in the images, this study analyzes the target scale characteristics of the field fruit dataset. Based on the receptive field scale matching principle, three detection branches were designed in the 18th, 26th, and 43rd convolutional layers of the backbone network to detect targets of various scales.
Receptive field scale analysis

Luo et al. [25] examined the receptive field problem in deep convolutional networks and introduced the concept of the effective receptive field. They noted that the best detection results are obtained only when the effective receptive field scale matches the target scale. On the left side of Figure 7 (Appendix Figure), the blue and red boxes indicate the receptive field of the feature map and the effective receptive field area, respectively. According to the abovementioned principle, the effective receptive field area must cover the complete target, as shown on the right side of Figure 7 (Appendix Figure). The receptive field scale can be defined as:

\[ R_k = 1 + \sum_{j=1}^{k} (F_j - 1) \prod_{i=0}^{j-1} S_i \]  

Here, \( R_k \) denotes the receptive field size of the \( k \)-th convolutional layer, \( F_j \) denotes the convolutional kernel size of the \( j \)-th convolutional layer, and \( S_i \) denotes the step of the \( i \)-th convolutional layer.

The shallow and deep network feature maps have a small and large receptive field size and can detect small-scale and large-scale targets, respectively. To determine the network layers that are more suitable for detecting the field fruit targets, the collected field fruit dataset was annotated, and histogram statistical analysis was performed to explore the characteristics of the field fruit images.

To analyze the characteristics of the fruit scale size distribution, the k-means clustering statistics of the experimental dataset were derived [27]. As shown in Figure 2, k-means clustering produced results of 11 × 12, 14 × 16, 18 × 19, 24 × 16, 30 × 17, and 50 × 31. Each point in the scatter plot in Figure 2(a) represents a labeled fruit target, and the horizontal and vertical coordinates indicate the width and height of the target. Figure 2(b) shows the specific number distribution of the k-means clustering results. Based on all definition criteria pertaining to small targets, such as a target size less than 32 × 32, as defined
by the coco dataset [32], or a target with a scale less than one-tenth that of the original image, most of the fruit targets in the field fruit image dataset can be considered small targets. The results of the receptive field scale and k-means clustering analysis of the fruit dataset from the DarkNet53 network and those obtained through the principle of receptive field scale matching (Equation 1) indicate that:

1) Conv9 feature maps with a receptive field scale of 29 × 29 are the most suitable for detecting fruit targets at scales of 11 × 12, 14 × 16, and 18 × 19.

2) Conv26 feature maps with a receptive field scale of 165 × 165 are the most suitable for detecting fruit targets at scales of 24 × 16, 30 × 17, and 50 × 31.

3) Because certain large-scale targets remain in the image, Conv43 was chosen as the third detection branch and retained the size of the anchor boxes designed by Yolov3 in this layer.

4) The receptive field scale of Conv53 feature maps is not applicable to the in-field fruit detection task. To better utilize these features in the corresponding network layer, they were fused with the shallow network for feature fusion to enrich the semantic features of the shallow network and enhance small-scale fruit detection accuracy.

Dual-attention multiscale fusion module
The receptive field scales and feature information of the feature maps from different convolutional layers are considerably different. For example, the shallow network features have small receptive field scales and focus on the color and textural details of the image, whereas the deep network features have larger receptive field scales and contain abundant semantic features. To exploit the features of different convolutional layers to accomplish vision tasks with high performance, several multiscale fusion strategies have been proposed [33–35]. These fusion strategies, however, generally only up-sample the feature maps of different sizes to ensure that the two...
feature maps are of the same size and directly implement dimensional concatenation.

Although the fused features are derived from different deep convolutional layers, the algorithms do not consider the correlation properties between the different feature maps and thus cannot fully fuse and utilize the effective information from different feature maps. In recent years, the attention mechanism of deep learning has received considerable attention for the enhancement of feature quality. Hu et al. [36] proposed SE-Net to establish the relationship between feature channels, obtain the importance of each channel via the adaptive learning method, and assign different weights according to channel importance to enhance their effectiveness. In contrast to the SE-Net attention module, which focuses only on the weight relationship of each feature channel, the CBAM [37] attention module integrates the spatial features and weight relationship of each channel and can achieve excellent results on other image recognition tasks.

Oranges tend to grow in clusters, and the spatial characteristics of image features must therefore be considered to enhance the discriminative ability of the local information and accurately detect each fruit target. Based on the characteristics of field fruit data, a multiscale fusion module was established based on channel attention and spatial attention mechanisms. First, the channel attention module is separately applied to two feature maps from different convolutional layers, thereby realizing adaptive learning based on the importance of each channel. Next, after channel enhancement, the two feature maps are dimensionally connected and input to the spatial attention module to enhance the response level to each target region.

The structure of this module that contains channel and spatial attention, that is, the dual-attention multiscale fusion module, is shown in Figure 3.

**Channel attention:** The input includes two features from different depth network layers with different dimensions. Shallow feature \( X_1 \) and deep feature \( X_2 \) have dimensions \( W \times H \times C_1 \) and \( \frac{W}{2} \times \frac{H}{2} \times C_2 \), respectively. First, the deep feature \( X_2 \) is up-sampled by a factor of 2 to ensure that it has the same scale as the shallow feature \( X_1 \). Next, the channel attention operation is performed on the two features separately. Let us consider feature \( X_1 \) as an example. First, the global pooling layer compresses each two-dimensional channel of feature \( X_1 \) to a real number. Each real number characterizes the global distribution of the responses over the feature channels, as shown in Equation 2, where \( z_c \) denotes the response value of channel \( c \) in feature \( X_1 \), \( x_c \) represents the \( c \)-th channel of feature \( X_1 \), and \( H \) and \( W \) denote the height and width of feature \( U \), respectively.

After developing the two \( 1 \times 1 \) convolutional layers to establish the correlation between the channels, the sigmoid activation layer can obtain \( c \) weights normalized between 0 and 1. In Equation 3, \( \sigma \) and \( \delta \) denote the sigmoid activation function and the ReLU activation function, respectively, \( W_1 \) and \( W_2 \) denote the weight parameters of the first and second convolutional layers, respectively, and \( S \) is the normalized weight matrix. The scale operation assigns the normalized weights to each channel of feature \( X_1 \) to obtain a channel-weighted feature map \( Y_1 \) with dimensions \( w \times h \times C_2 \). The scale is calculated as shown in Equation 4. \( y \) denotes the feature vector of the \( c \)-th channel in the feature map \( Y_1 \). Finally, the channel attention-enhanced features \( Y_1 \) and \( Y_2 \) are
Table 3. Ablation experiment results for four components. RF (Receptive field) matched: sensory field scale matching and optimized anchor box strategy; Mosaic augmentation: mosaic data augmentation method; Channel attention: channel attention enhancement module; and Dual-attention: dual-attention mechanism fusion module.

| Method     | RF matched | Mosaic augmentation | Channel attention | Dual attention | F1-score | AP  |
|------------|------------|---------------------|-------------------|----------------|----------|-----|
| OrangeYolo | √          | √                   | √                 | √              | 0.873    | 0.905|
|            | √          | √                   |                   | √              | 0.887    | 0.928|
|            | √          | √                   | √                 |                | 0.89     | 0.93 |
|            | √          | √                   |                   |                | 0.892    | 0.933|
|            | √          | √                   |                   |                | 0.897    | 0.938|

Figure 5. Visualization of detection results (blue bounding box). (a), (b), and (c) indicate the visualization results for detection of small-scale targets, medium-scale targets, and large-scale targets, respectively, and (d) indicates the detection results for the standard dataset. Note that orange fruit targets of different scales in the image can be detected accurately.

Dimensionally concatenated to obtain the fused channel-attention-enhanced features $Y$.

$$z_c = \frac{1}{H \times W} \sum_{i=1}^{H} \sum_{j=1}^{W} x_c (i, j)$$

Dimensionally concatenated to obtain the fused channel-attention-enhanced features $Y$.

$$S = F_z (Z, W_1, W_2) = \sigma (W_2 \cdot \delta (W_1 Z))$$

Spatial attention: First, the average pooling and maximum pooling operations are performed on $Y$, respectively. $Y_{\text{concat}}$ represents the feature after the dimensional concatenation of $Y_{\text{avg}}$ and $Y_{\text{max}}$, $\sigma$ represents the sigmoid operation, $f^{7 \times 7}$ represents the convolution operation with a convolution kernel size of $7 \times 7$, and $F_{\text{Scale}}$ represents the dimensional concatenation operation.

$$Y_{\text{fusion}} = F_{\text{Scale}} (\sigma (f^{7 \times 7} ([\text{AvgPool}(Y); \text{MaxPool}(Y)])) , Y)$$

Data augmentation
In the previously presented analysis of small-scale target characteristics, small targets were noted to be challenging to detect. Recently, some researchers have examined techniques for performing image augmentation for small targets in the data pre-processing stage to enhance the recognition ability of detectors for small targets [40, 41]. The Mosaic data augmentation method can combine multiple training images into one image for training according to the set mini-batch, thereby diversifying the target scale and enriching the image context. Figure 8 (Appendix Figure) shows the results of the mosaic data augmentation.

OrangeSort
In this study, we fully analyze the fruit distribution characteristics in the orchard scene and propose the OrangeSort algorithm for citrus fruit tracking. First, we improve the algorithm based on the mainstream multi-target tracking algorithm “sort” and proposed a motion displacement estimation algorithm that can significantly reduce the errors caused by the double-counting problem. Second, we propose a region counting strategy that can solve the problem of counting errors caused by complex occlusion problems in orchard scenes.

Motion displacement estimation
In real orchards, fruit grows densely and overlaps heavily. This is a significant problem for tracking algorithms in computer vision. Although the mainstream sort algorithm [28] performs satisfactorily in many areas of application, double-counting occurs in practice. This problem can have a significant impact on the performance of the algorithm in fruit counting. When a tracking target is
The current and previous frames of the average displacement in the x- and y-axis directions of estimation and 2) Tracking target scale estimation.

Involves two steps: 1) Tracking target center position estimation. To track a particular target correctly, it can consider the average motion displacement over the correctly tracked target to estimate the position of that target in the current frame.

The motion displacement estimation tracking method involves two steps: 1) Tracking target center position estimation and 2) Tracking target scale estimation.

(1) Tracking target center position estimation: The displacement of the correct tracking target center point in the previous frame is used to estimate the corresponding center point in the current frame. The coordinates of the center point of the tracked target are calculated using Equation 6. $x_i$ and $x'_i$ denote the x-axis coordinate of the center point of the i-th correctly tracked target in the current frame and previous frame, respectively. $n$ denotes the number of correctly tracked targets in the current frame, and $Δx$ and $Δy$ denote the average displacement in the x- and y-axis directions of the current and previous frames of the $n$ correctly tracked targets. The position ($x', y'$) of the center point ($x, y$) of the unmatched tracking target in the current frame can be calculated using $Δx$ and $Δy$.

$$ (x', y') = (x + Δx, y + Δy) = \left( x + \frac{\sum_{i=1}^{n} x_i - x'_i}{n}, y + \frac{\sum_{i=1}^{n} y_i - y'_i}{n} \right) \quad (6) $$

The accuracy of multi-target tracking algorithms based on frame-by-frame detection is considerably affected by the detection results, which can easily lead to the loss of tracked targets in the following two cases: 1) the target is occluded and cannot be detected by the detector; 2) the target appears clearly in the video but is missed by the detector in certain frames.

If the detector cannot provide the location of the target to the tracking algorithm, the corresponding tracking target in the current frame is identified as an unmatched tracking target. Compared to the case in which the tracking target appears clearly in the video but is missed by the detector, the tracking target is more likely to be lost due to occlusion. In general, the occlusion of the tracking target in the video is a gradual process from slight to heavy occlusion. At this time, the pixel area of the blocked fruit in the image will gradually decrease, as shown in Figure 9 (Appendix Figure). Therefore, the following tracking target scale estimation method was established to manage the situation in which the tracking target is lost owing to the occlusion condition.

(2) Tracking target scale estimation: After determination of the center position of the tracking target, the width and height dimensions of the tracking target are calculated to obtain the tracking target position area in the current frame. The width and height of the tracking target in the current frame image are estimated. After determination of the location of the center point of the tracking target, the scale size of the target in the previous frame is considered as the base. The scale decay factor $α$ is used to estimate the scale size of the target in the current frame, as shown in Equation 7, where $w_1$ and $h_1$ denote the width and height of the tracking target in the current frame, and $w$ and $h$ denote the width and height of the tracking target in the previous frame, respectively.

$$ (w_1, h_1) = (w * α, h * α) \quad (7) $$

Because the occlusion of the target in the video sequence is a gradual change, the implementation of the motion displacement estimation operation on an unmatched tracking target may cause the estimated tracking target to exhibit a scale decay. The motion displacement estimation tracking strategy is formulated as follows: 1) If the scale of the tracking target decays continuously after a certain level, the target is considered to not appear in the video owing to complete occlusion, and the tracking target is removed from the tracking list. 2) If the target is not matched by the tracked target owing
Table 4. Counting results of OrangeSort, Sort [28], and DeepSort [29] tracking algorithms.

| Video sequence ID | Counting method | Manual counting | Number of fruit counts | Number of correct counts | Number of false counts | Number of double-counting instances | Counting error |
|-------------------|-----------------|-----------------|------------------------|--------------------------|-----------------------|-------------------------------------|----------------|
| 1 Sort            | 165             | 277             | 147                    | 54                       | 76                    | 0.6788                              |
| Deepsort          | 252             | 142             | 14                     | 14                       | 96                    | 0.5273                              |
| OrangeSort        | 189             | 145             | 22                     | 22                       | 22                    | 0.1555                              |
| OrangeSort*       | 166             | 132             | 27                     | 7                        | 7                     | **0.0061**                          |
| 2 Sort            | 72              | 40              | 26                     | 6                        | 125                   | 0.125                               |
| Deepsort          | 64              | 39              | 33                     | 33                       | 33                    | 0.6406                              |
| OrangeSort        | 70              | 40              | 25                     | 5                        | 5                     | 0.0938                              |
| OrangeSort*       | 57              | 40              | 13                     | 4                        | 4                     | **0.1094**                          |
| 3 Sort            | 113             | 66              | 26                     | 21                       | 61                    | 0.6143                              |
| Deepsort          | 70              | 65              | 49                     | 98                       | 2.0286                | 0.3714                              |
| OrangeSort        | 96              | 66              | 16                     | 14                       | 14                    | 0.3714                              |
| OrangeSort*       | 87              | 61              | 14                     | 12                       | 12                    | **0.2429**                          |
| 4 Sort            | 270             | 114             | 111                    | 111                      | 45                    | 0.4754                              |
| Deepsort          | 515             | 115             | 82                     | 318                      | 1.8142                | 0.1265                              |
| OrangeSort        | 211             | 114             | 66                     | 31                       | 31                    | 0.1265                              |
| OrangeSort*       | 170             | 93              | 58                     | 19                       | 19                    | **0.071**                           |
| 5 Sort            | 117             | 66              | 33                     | 18                       | 2.271                 | 0.8913                              |
| Deepsort          | 174             | 64              | 31                     | 79                       | 0.8913                | 1.522                               |
| OrangeSort        | 106             | 65              | 27                     | 14                       | 14                    | 0.1522                              |
| OrangeSort*       | 95              | 65              | 19                     | 11                       | 11                    | **0.0326**                          |
| 6 Sort            | 381             | 195             | 152                    | 34                       | 34                    | 0.5363                              |
| Deepsort          | 588             | 202             | 104                    | 282                      | 282                   | 1.371                               |
| OrangeSort        | 327             | 190             | 115                    | 22                       | 22                    | 0.3185                              |
| OrangeSort*       | 242             | 170             | 59                     | 13                       | 13                    | **0.0242**                          |

Note: OrangeSort* represents the counting results of the OrangeSort algorithm using the specific counting region strategy.

Table 5. Counting standard deviations of OrangeSort, Sort [28], and DeepSort [29] tracking algorithms.

| Counting method | Sort | DeepSort | OrangeSort | OrangeSort* |
|-----------------|------|----------|------------|-------------|
| Standard deviation | 0.4741 | 1.3975 | 0.2555 | 0.08 |

to slight occlusion or misdetection, OrangeYolo can accurately detect the target again in several consecutive frames, thereby providing data support for the tracking algorithm to enable the tracker to track the target again.

Based on the analysis of the motion characteristics of the target in the field fruit video sequence, the steps involved in OrangeSort can be enumerated as follows, as illustrated in Figure 4, where the blue boxed portion is the improvement point concerning the motion displacement estimation.

1) First, the proposed detection algorithm OrangeYolo is used to detect the fruit in each frame of the image. Then a detection box is generated for each fruit target and input to the tracking algorithm OrangeSort.

2) The proposed tracking algorithm OrangeSort determines whether the current image is in the first frame. If it is, a new ID is assigned to each fruit detection box, which is the tracking result of the current frame. If it is not, the detection result of the current frame is matched with the tracking result using the Hungarian algorithm. The boxes are divided into three categories, “unmatched detection box”, “correctly matched detection box”, and “unmatched tracking box”.

3) The three categories are processed as follows.

In the unmatched detection box case, a particular detection result for the current frame does not match any of the tracking results from the previous frame. This indicates that the target is appearing in the image for the first time or is being detected for the first time; therefore, a new tracker ID is created directly.

In the correctly matched detection box case, the detection result is correctly associated with a tracking target. First, the average displacement of all correctly matched detection frames in the coordinates of the current frame and the previous frame is calculated, and the state and tracking list of the corresponding Kalman trackers is updated using the tracking results of the current frame.

In the unmatched tracking box case, a particular tracking result from the previous frame cannot be matched with any of the detected results from the current frame. At this point, the algorithm steps are as follows. First, the coordinates of the center point of the unmatched tracking box in the current frame are calculated, considering the average displacement of all correctly matched detection frames. Then, the width and height of the tracking target in the current frame are calculated based on the target scale of the previous frame and from $\alpha$. If the scale is smaller than the threshold,
the tracking target is considered to be possibly heavily occluded and not present in the image for a long time, and the tracker is therefore deleted.

4) The algorithm determines if the result pertains to the last frame of the video. If not, it loops through steps 2 and 3. If so, it outputs all tracking results.

**Tracking strategy determination**

To alleviate the problems of existing video-based methods, which cannot address complex occlusion situations that may exist in global video sequences, an effective region tracking counting strategy is established by analyzing the characteristics of video sequences of fruit in the field.

Frequent cases of complex and variable fruit occlusion in the video sequences have a considerable influence on tracking performance and are one of the key factors that lead to double counting, as shown in Figure 10 (Appendix Figure) and Figure 11 (Appendix Figure).

**Fruit position classification:** Fruit position states were classified into three categories according to their different positions in the image, as shown in Figure 11 (Appendix Figure): initial position, intermediate position, and leaving position.

Initial position: In frame 29, the fruit indicated by the green arrow is only slightly in the field of view of the camera.

Intermediate position: In frame 71, the fruit indicated by the green arrow is approximately in the central area of the image.

Leaving position: In frame 100, the fruit indicated by the green arrow is about to leave the field of view.

**Counting region determination:** Based on the fact that oranges may have different occlusion states at each position, all possible complex occlusion cases of fruits in the global video sequence were analyzed. Table 1 lists all possible occlusion conditions of the fruit in the global video sequence from entering to leaving the camera view. According to the statistical analysis of fruit occlusion status for the global video sequences of certain trees, conditions A, B, C, and D account for the largest proportion of all global video sequence occlusion conditions. Therefore, the intermediate area of the video was considered as the core tracking counting region. Conditions E, F, and G were considered to maximize the advantages of the video-sequence-based tracking counting method and to reduce the influence of complex occlusion in the global video sequence.

In “Fruit occlusion status”, Y indicates that there is occlusion in this region, and N indicates that there is no occlusion in this region. The final tracking counting region contains part of the initial area and leaving area. In cases A, B, C, and D, the intermediate positions do not exhibit any occlusion. Consequently, it is only necessary to count the fruits in this position, thus reducing the computational resources required. In conditions E, F, and G, the fruits are occluded in the intermediate position and not occluded in either the initial or leaving position.

To count the fruits, the counting area must include part of the initial position area and leaving position area.

The width of the tracking count area is $3/5$th of the original video sequence width. The regions $1/5$th after the initial position and $1/5$th prior to the leaving position are invalid regions. Experimental analysis indicates that the set specific region counting strategy can effectively overcome the influence of occlusion on tracking counting, with reduced tracking computation and enhanced operation speed.

**Experiment**

**Data acquisition**

The data acquisition is performed in two orange orchards in Meishan city, Sichuan Province, China, (29°55′25″N 104°17′23″E, 30°14′26″N 104°06′04″E), and the video was acquired using a field rover equipped with a DJI Osmo Action camera.

The rationality of the data set collection method can be referred to our previous research work [39]. The field rover moves at an even speed along the direction of the tree rows. The camera is perpendicular to the tree row direction when acquiring the video. To observe a sufficient number of fruits, the full view of each fruit tree must be captured, and the vertical direction of each frame must cover the top and bottom of the fruit tree, as shown in Figure 10 (Appendix Figure).

To ensure a diversity of data when selecting the image dataset for the fruit detection algorithm, data acquisition was randomly performed at different locations in the orchard. A random cropping strategy was also implemented on the original video sequences to produce 1465 samples of image datasets with different target scales and different shooting angles for training and testing the target detection algorithm. Among these, the test set of 182 images is the standard dataset. During acquisition, the shooting distance was kept consistent, and the shooting angle was kept constant, ensuring that the vertical height just encompassed the top of the tree canopy and the bottom of the tree trunk.

**Experimental details**

A computer with a GeForce GTX 1080 Ti GPU and an Intel i7 8th generation CPU was used to train and test OrangeYolo. Specifically, 1465 data samples were...
randomly divided into training and testing sets in a ratio of 7:3 (for more details, please refer to our published dataset: https://github.com/I3-Laboratory/orange-dataset). The SGD optimizer was used to optimize the gradient, with a momentum, weight decay, and initial learning rate of 0.9, 0.0005, and 0.01, respectively.

Five evaluation metrics were used to verify the effectiveness of OrangeYolo: precision, recall, F1-score, average precision (AP), and detection speed (FPS). The detection results were divided into true positives (TPs) and false positives (FPs). TP indicates that a manually marked orange target is correctly detected, and FP indicates that a non-target is incorrectly detected as a target; for example, a leafy branch is mistakenly detected as an orange. Precision indicates the proportion of TPs in the detection results relative to TP + FP of all detection results, and it is calculated as shown in Equation 8. Recall indicates the proportion of TPs in the detection results relative to the number of manually marked samples, GroundTruth, and it can be calculated using Equation 9. The F1-score, calculated using Equation 10, takes into account both precision and recall to evaluate detection performance. A larger F1-score corresponds to higher detection performance. AP, which corresponds to the area under the precision and recall curves and the average of all precision values with recall between 0 and 1, indicates the overall performance of the detection model. AP can be calculated using Equation 11, where $p$ and $r$ indicate the precision and recall values, respectively.

\[
\text{Precision} = \frac{TP}{TP + FP} \quad (8)
\]
\[
\text{Recall} = \frac{TP}{\text{GroundTruth}} \quad (9)
\]
\[
\text{F1-score} = \frac{2 \times \text{Precision} \times \text{Recall}}{\text{Precision} + \text{Recall}} \quad (10)
\]
Figure 10. Visible state of the fruit. In frame 50, the marked yellow area enters the camera view, and the fruit in the area is completely occluded by leaves. However, with a change in acquisition perspective, the fruit in the yellow area in frame 70 can be clearly observed.

Figure 11. Video sequence occlusion analysis. The orange indicated by the green arrow can be clearly observed in frame 29; however, as the acquisition device moves, the orange cannot be observed in frame 71 because it has been occluded by an orange next to it. Furthermore, immediately before the fruit leaves the field of view of the camera, the fruit indicated by the green arrow appears clearly in frame 100, owing to a change in acquisition viewpoint.

\[ AP = \int_0^1 \text{Precision}(r) dr \]  

To verify the effectiveness of OrangeSort, tracking counting experiments were performed using six video sequences containing 22 trees, and the results were compared with manual counting results for the video sequences. The accuracy of the count results is also expressed using the mean absolute error (MAE), which can be expressed by Equation 12.

\[ \text{MAE} = \frac{1}{m} \sum_{i=1}^{m} \left| \frac{y_{\text{test}}^{(i)} - \hat{y}_{\text{test}}^{(i)}}{\hat{y}_{\text{test}}^{(i)}} \right| \]  

where \( y_{\text{test}}^{(i)} \) denotes the fruit counting result of the tracking algorithm in the ith video sequence, \( \hat{y}_{\text{test}}^{(i)} \) denotes the artificial fruit counting result in the ith video sequence, and \( m \) denotes the total number of video sequences. As shown in Figure 12 (Appendix Figure), the video sequences contain a variety of complex occlusions, scale variations, and different shooting conditions that are frequently encountered in practical applications.

Comparative analysis of experimental results

(1) Performance analysis of OrangeYolo: Table 2 summarizes the various performance indicators of OrangeYolo. OrangeYolo outperforms the comparison algorithms \([V3, V4]\) \([24, 40]\) in terms of all five indicators, with an F1-score, AP, and FPS of 0.897, 0.938, and 83.3, respectively. The last row of Table 2 shows the test results on 182 images from the standard dataset. The proposed algorithm achieves an average accuracy of 0.957 on standard fruit images and an F1 value of 0.916. This demonstrates that the standard dataset obtained with a standardized shooting method can better help the detection algorithm to detect citrus fruits accurately. The inference time FPS is the same as that of Yolov3, and both are better than other comparison algorithms, demonstrating the practical application value of the proposed method. Next, ablation comparison experiments were performed to verify the effectiveness of the proposed receptive field scale matching strategy. As shown in Table 3, the detection performance could be effectively improved by using the receptive field matching strategy, and the F1-score and AP values were enhanced by 1.4% and 2.3%, respectively. The highest detection performance was achieved after using the dual-attention multiscale fusion mechanism,
and the F1-score and AP values were enhanced by 0.7% and 0.8% from 0.89 and 0.93, respectively. Figure 5 shows the visualization results of the detection algorithm. It can be noted that orange fruit targets of different scales can be detected accurately in the image.

2) Performance analysis of OrangeSort: Figure 6 shows the OrangeSort results obtained using the specific counting region tracking strategy. Table 4 presents the counting results of the OrangeSort, Sort [28], and DeepSort [29] tracking algorithms tested on six video sequences. Six metrics are considered: number of fruit counts, number of correct counts, number of missed counts, number of false counts, number of double-counting instances, and counting error, where the counting error is the MAE of a single video sequence, which can be expressed as $\text{MAE}(m = 1)$. The OrangeSort algorithm exhibits a significantly lower number of double-counting instances compared with the Sort and DeepSort algorithms. The total number of fruit counts is also closer to the number of manual counts, with counting errors of 0.1455, 0.0938, 0.3714, 0.1265, 0.1522, and 0.3185. OrangeSort* indicates the counting results obtained using the specific counting region strategy. Although OrangeSort* has slightly fewer correct counts, it effectively overcomes the ID switch problem in the tracking process and reduces the number of double-counted fruits. The counting results are close to the manual counting results, with counting errors of 0.0061, 0.1094, 0.2429, 0.071, 0.0326, and 0.0242. The data of video sequence 1 were collected in the orchard using standard lighting conditions, and the algorithm therefore gives the smallest counting error for this sequence. Video sequences 2–4 were collected in a poorly lit environment, and the overall counting error is larger. The light is worst in video sequence 3; it is almost impossible to distinguish the categories of fruits and leaves, and the counting error is therefore the largest. Video sequences 5–6 were collected in a strong light environment and were overexposed, with bumps and dramatic camera shakes. Most of the fruit targets could be distinguished, but there were still a few fruit targets that could not be tracked accurately, and the counting error was relatively high. In video sequence 2, OrangeSort counted 70 fruits, with 40 correct counts, 5 double-counting instances, and 25 error counts. OrangeSort* counted 57 fruits, with 40 correct counts, 4 repeat counts, and 13 error counts. The counting errors of OrangeSort and OrangeSort* were similar, with values of 0.0938 and 0.1094, respectively. However, OrangeSort counted more error samples by performing fruit tracking counts in the global region, resulting in nearly twice as many error counts as OrangeSort*. Consequently, the result obtained using OrangeSort* was closer to the true value of the manual counts.

By computing the mean absolute error (MAE) of the six video sequences with the results in Table 5, we can
conclude that OrangeSort\textsuperscript{*} produces the smallest counting error of 0.081 and a standard deviation of 0.8 compared with the Sort and OrangeSort algorithms (without implementing the counting region strategy). The counting results of OrangeSort\textsuperscript{*} are clearly closest to the manual counting results.

### Conclusion

This work improves the accuracy of fruit counting in terms of fruit detection and double-counting. Taking into account the small-scale characteristics of field orange data and the different occlusion statuses in the video sequence, a video-sequence-based fruit counting algorithm was established, including two sub-algorithms, OrangeYolo for fruit detection and OrangeSort for fruit tracking.

The network structure, designed based on the receptive field matching strategy and the dual attention fusion module, constitutes the fruit detection algorithm OrangeYolo, which achieves an AP value of 0.938 on the field orange detection dataset. A specific tracking region counting strategy and tracking algorithm based on motion displacement estimation constitute the OrangeSort tracking algorithm. Using six video sequences taken from two fields containing 22 trees as the validation dataset, the proposed method showed the best performance (MAE = 0.081, SD = 0.08) relative to video-based manual counting. These results demonstrate the practical value of the proposed method compared with other existing algorithms. Future work can be aimed at using 3D technology to locate fruit spatial coordinates to enable more accurate counting, and line tail turns will be explored further in subsequent research work. Furthermore, a lightweight network model on edge devices can be designed and deployed to accomplish lightweight real-time field fruit counting and to link the actual fruit number by considering the unseen part of the tree to achieve actual yield estimation.

### Author contributions

W.Z., J.W., Y.L. and W.G. conceived the ideas and designed methodology; H.L., Y. D., W.W., and Y. S. collected field data with self-designed field rover. All authors discussed, wrote the manuscript, and gave final approval for publication.

### Data availability

The dataset used during this study is available in a repository in accordance with funder data retention policies. We have published the dataset at GitHub (https://github.com/I3-Laboratory/orange-dataset).

### Conflict of interest statement

The authors declare that they have no conflicts of interest.

### Supplementary data

**Supplementary data** is available at Horticulture Research online.

### References

1. Anderson NT, Walsh KB, Wulfsohn D. Technologies for Forecasting Tree Fruit Load and Harvest Timing—from Ground. *Agronomy (Basel)*. 2021;11:1409.
2. Koirala A, Walsh KB, Wang Z. Attempting to estimate the unseen—correction for occluded fruit in tree fruit load estimation by machine vision with deep learning. *Agronomy*. 2021;11:347.
3. Koirala A, Walsh KB, Wang Z et al. Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of ‘MangoYOLO’. *Precis Agric*. 2019;20:1107–35.
4. Apolo-Apolo OE, Martínez-Guanter J, Egea G et al. Deep learning techniques for estimation of the yield and size of citrus fruits using a UAV. *Eur J Agron*. 2020;115:126030.
5. Mu Y, Chen TS, Ninomiya S et al. Intact detection of highly occluded immature tomatoes on plants using deep learning techniques. *Sensors*. 2020;20:2984.
6. Bargoti S, Underwood J. Deep fruit detection in orchards[C]. In: *2017 IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2017;3626–33.
7. Sa I, Ge Z, Dayoub F et al. Deepfruits: a fruit detection system using deep neural networks. *Sensors*. 2016;16:1222.
8. Gan H, Lee WS, Alchanatis V et al. Active thermal imaging for immature citrus fruit detection. * Biosyst Eng*. 2020;198:291–303.
9. Wan S, Goudos S. Faster R-CNN for multi-class fruit detection using a robotic vision system. *Comput Netw*. 2020;168:107036.
10. Gao F, Fu L, Zhang X et al. Multi-class fruit-on-plant detection for apple in SNAP system using faster R-CNN. *Comput Electron Agric*. 2020;176:105634.
11. Kestur R, Meduri A, Narasipura O. MangoNet: a deep semantic segmentation architecture for a method to detect and count mangoes in an open orchard. *Eng Appl Artif Intell*. 2019;77:59–69.
12. Akiva P, Dana K, Oudemans P et al. Finding berries: segmentation and counting of cranberries using point supervision and shape priors. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops*. 2020;2020:50–1.
13. Gené-Mola J, Sanz-Cortiella R, Rosell-Polo JR et al. Fruit detection and 3D location using instance segmentation neural networks and structure-from-motion photogrammetry. * Comput Electron Agric*. 2020;169:105165.
14. Kang H, Chen C. Fruit detection, segmentation and 3D visualisation of environments in apple orchards. *Comput Electron Agric*. 2020;171:105302.
15. Chen SW, Shivakumar SS, Dcunha S et al. Counting apples and oranges with deep learning: a data-driven approach. *IEEE Robotics and Automation Letters*. 2017;2:781–8.
16. Sun G, Wang X, Yang H et al. A canopy information measurement method for modern standardized apple orchards based on UAV multimodal information. *Sensors*. 2020;20:2985.
17. Wang Z, Walsh K, Koirala A. Mango fruit load estimation using a video based MangoYOLO—Kalman filter—hungarian algorithm method. *Sensors*. 2019;19:2742.
18. Roy P, Kislay A, Pionski PA et al. Vision-based preharvest yield mapping for apple orchards. *Comput Electron Agric*. 2019;164:104897.
19. Liu X, Chen SW, Aditya S et al. Robust fruit counting: Combining deep learning, tracking, and structure from motion. In: 2018 IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS). IEEE, 2018, 2018, 1045–52.

20. Liu X, Chen SW, Liu C et al. Monocular camera based fruit counting and mapping with semantic data association. IEEE Robotics and Automation Letters. 2019; 4:2296–303.

21. Gené-Mola J, Gregorio E, Cheein FA et al. Fruit detection, yield prediction and canopy geometric characterization using LiDAR with forced air flow. Comput Electron Agric. 2020; 168:105121.

22. Stein M, Bargoti S, Underwood J. Image based mango fruit detection, yield and canopy geometric characterization. Sensors. 2016; 16:1915.

23. Gan H, Lee WS, Alchanatis V. A prototype of an immature citrus fruit yield mapping system. 2017 ASABE Annual International Meeting. American Society of Agricultural and Biological Engineers. 2017; 2017:1.

24. Redmon J, Farhadi A. Yolov3: an incremental improvement arXiv preprint arXiv:1804.02767. 2018.

25. Luo W, Li Y, Urtasun R et al. Understanding the effective receptive field in deep convolutional neural networks. Proceedings of the 30th International Conference on Neural Information Processing Systems. 2016; 2016:4905–13.

26. Zhang S, Zhu X, Lei Z et al. S3fd: single shot scale-invariant face detector. Proceedings of the IEEE international conference on computer vision. 2017; 2017:192–201.

27. Redmon J, Farhadi A. YOLO9000: better, faster, stronger. IEEE Conference on Computer Vision & Pattern Recognition. 2017; 2017:6517–25.

28. Bewley A, Ge Z, Ott L et al. Simple online and realtime tracking. In: 2016 IEEE International Conference on Image Processing (ICIP). IEEE. 2016, 2016:3464–8.

29. Wojke N, Bewley A, Paulus D. Simple online and realtime tracking with a deep association metric. 2017 IEEE international conference on image processing (ICIP) IEEE. 2017; 2017:3645–9.

30. Liu S, Huang D, Wang Y. Learning spatial fusion for single-shot object detection arXiv preprint arXiv:1911.09516. 2019.

31. Zhang Z, He T, Zhang H et al. Bag of freebies for training object detection neural networks arXiv preprint arXiv:1902.04103. 2019.

32. Lin TY, Maire M, Belongie S et al. Microsoft coco: Common objects in context. In: European Conference on Computer Vision. Springer. Cham, 2014, 2014, 740–55.

33. Lin TY, Dollár P, Girshick R et al. Feature pyramid networks for object detection. Proceedings of the IEEE conference on computer vision and pattern recognition. 2017; 2017:2117–25.

34. Liu S, Qi L, Qin H et al. Path aggregation network for instance segmentation. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018; 2018:8759–68.

35. Tan M, Pang R, Le QV. Efficientdet: scalable and efficient object detection[C]. Proceedings of the IEEE/CVF conference on computer vision and pattern recognition. 2020; 2020:10781–90.

36. Hu J, Shen L, Sun G. Squeeze-and-excitation networks[C]. Proceedings of the IEEE conference on computer vision and pattern recognition. 2018; 2018:7152–41.

37. Woo S, Park J, Lee JY et al. Cbam: convolutional block attention module[C]. Computer Vision – ECCV 2018: 15th European Conference, Munich, Germany, September 8–14, 2018, Proceedings, Part VII. 2018; 11211:3–19.

38. Milan A, Leal-Taixé L, Reid I et al. MOT16: a benchmark for multi-object tracking arXiv preprint arXiv:1603.00831. 2016.

39. Wang J, Zhang W, Chen K et al. A deep learning-based in-field fruit counting method using video sequences https://www.plant-phenotyping.org/CVPPP2020-Programme. 2020.

40. Bochkovskiy A, Wang CY, Liao HYM. Yolov4: optimal speed and accuracy of object detection arXiv preprint arXiv:2004.10934. 2020.

41. Yun S, Han D, Oh SJ et al. Cutmix: regularization strategy to train strong classifiers with localizable features. Proceedings of the IEEE/CVF International Conference on Computer Vision. 2019; 2019:6023–32.