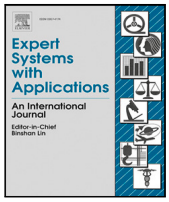
Expert Systems With Applications 255 (2024) 124587

Contents lists available at ScienceDirect

Expert Systems With Applications

journal homepage: www.elsevier.com/locate/eswa



Advancing tracking-by-detection with MultiMap: Towards occlusion-resilient online multiclass strawberry counting

Xuehai Zhou [a], Yuyang Zhang [a], Xintong Jiang [a], Kashif Riaz [b], Phil Rosenbaum [c], Mark Lefsrud [a], Shangpeng Sun [a,*]

[a] McGill University, 21111 Lakeshore Rd, Sainte-Anne-de-Bellevue, H9X 3V9, Quebec, Canada
[b] Ferme d'Hiver Inc., 4255 Boulevard Lapinière, Bureau 206, Brossard, J4Z 0C7, Quebec, Canada
[c] Gush Inc., 17003 Trans-Canada Highway, Kirkland, H9H 5J1, Quebec, Canada

ARTICLE INFO

Keywords:
Yield prediction
Fruit counting
Object detection
Online tracking
Attention mechanism

ABSTRACT

Despite the economic importance and research relevance of strawberries, advances in agricultural engineering for this crop have been hampered by pervasive occlusion challenges. Accurate fruit counting is crucial for both yield prediction and genotype selection; however, current fruit counting techniques fall short in large-scale strawberry farms. Leveraging computer vision, a refined strawberry detection network that integrates YOLOv5s with the attention mechanism identifies and locates both ripe and unripe strawberries in videos from strawberry farms. By utilizing tracking-by-detection algorithms, this system is able to trace multiple fruits from their first appearance to their final disappearance in real-time footage. To strengthen its resilience against occlusion, we have enhanced the tracking-by-detection algorithms with a multiple mapping algorithm. Without compromising its real-time performance, this implementation not only strengthens the robustness to occlusion but also segregates and tallies each strawberry class, presenting multiclass counts to the users. Within prevalent tracking frameworks, our approach records a mere 6.7% relative counting error rate. Furthermore, the lowest error rates for the classes of ripe and unripe strawberries were 8.7% and 9.9%, respectively. We contend that our method furnishes accurate counting data for digital agriculture, with the potential for broader applications. The code is open-source at https://github.com/Xuehaiz/MultiMap.

1. Introduction

In the contemporary landscape of big data, there has been a significant surge in the demand for instantaneous decision-making within the realm of digital agriculture (Wolfert et al., 2017). Precise and real-time information on fruit counts and ripening ratios is indispensable in digital farming, facilitating dynamic adjustments in lighting, irrigation, and fertilization strategies as well as providing vital indication for yield predictions. Subsequently, these metrics guide the strategic introduction of products to the market. Given this context, the integration of an automated fruit counting system emerges as an invaluable asset for modern agricultural practices. In particular, the strawberry, a crop of substantial economic importance, illustrates the need for precise fruit counting. The fragility and short shelf life of strawberries require meticulous planning in harvesting and distribution, while their susceptibility to rapid overripening calls for tailored cultivation practices to optimize yield and quality. In this context, an accurate count of strawberries is not only a convenience but also a necessity for strategic agricultural management. However, the precise counting of strawberries is complicated by their propensity to grow in overlapping clusters, a condition that introduces substantial occlusional challenges, which significantly hinders advances in this domain of agricultural research.

In agricultural engineering, the reconstruction and analysis of 3D data have been recognized as effective approaches to address the problem of occlusions (Sun et al., 2020). However, current 3D methods are still challenging for large-scale applications because it is difficult to obtain high-resolution 3D imaging data at field or regional scales and requires high computational resources. As a result, we need to revisit 2D methods to resolve concerns related to occlusions and clustering. Primarily, convolution-based 2D object detection has been a relatively mature and refined solution in the current stage (Ren et al., 2015). With the emergence of convolution-based object detection, a large number of image-based counting methods appeared (Ghosal et al., 2019; Zhang, Zhang et al., 2022; Zhao et al., 2023). Secondly, the emergence of the attention mechanism as a

* Corresponding author.
E-mail addresses: xuehai.zhou@mail.mcgill.ca (X. Zhou), yuyang.zhang@mail.mcgill.ca (Y. Zhang), xintong.jiang@mail.mcgill.ca (X. Jiang), kriaz@fermedhiver.com (K. Riaz), phil@gush.farm (P. Rosenbaum), mark.lefsrud@mcgill.ca (M. Lefsrud), shangpeng.sun@mcgill.ca (S. Sun).

https://doi.org/10.1016/j.eswa.2024.124587
Received 25 January 2024; Received in revised form 7 June 2024; Accepted 22 June 2024
Available online 26 June 2024

foundational algorithm (Bahdanau et al., 2014) has facilitated a significant enhancement in the precision of computer vision tasks. In recent years, there has also been a proliferation of exemplary works in the domain of plant science that are based on the attention mechanism (James et al., 2024; Zheng et al., 2023). Single-shot image-based detection and counting methods offer numerous advantages such as rapid data processing speeds and minimal storage demands. However, such singular perspective approaches exhibit diminished robustness for occlusions. Furthermore, in large-scale farming operations, there is often a reliance on stitching multiple snapshots together to achieve global counting, thereby sacrificing real-time capability.

Tracking-by-detection has become a paradigm that combines object detection in individual frames with the association of the identical detections over time to track objects in video sequences (Luo et al., 2021). Thanks to the advancement of object detection algorithms, tracking-by-detection algorithms have also seen significant progress in recent years (Bewley et al., 2016). With these tracking algorithms, we can identify and associate the same object across frames in videos. With the motion of cameras, video data capture multi-angle information of the objects, thereby alleviating the occlusion issue of object detection. Furthermore, the online attribute of these tracking algorithms facilitates immediate decision-making, meaning video frames can be processed as they come without the need to access future frames. However, compared to tracking algorithms that prioritize recording the motion trajectory of objects, the task of object counting is often overlooked. Consequently, it becomes necessary to customize a counting module for tracking algorithms to address the counting problem in our application context.

A large volume of counting methods have been proposed and tailored for agricultural scenarios. An outstanding series of contributions to fruit counting in apple orchards has been established by Wu et al. through the development of advanced automatic fruit counting systems, emphasizing the accuracy and reduction of abnormal fruit detections (Wu, Sun, Jiang, Gao et al., 2023; Wu, Sun, Jiang, Mao et al., 2023), while Gao et al. have introduced a novel methodology combining deep learning and trunk tracking for precise apple detection and counting (Gao et al., 2022). Among these approaches, employing tracking-by-detection for counting in 2D videos represents a relatively novel area of research, within which most contributions exhibit varying degrees of limitations. In the context of seedling counting, given the uniformity of the scene and the low-occlusion environment, tracking-by-detection methods were applied and generally did not present substantial challenges in both detection and tracking (Cui et al., 2023; Jiang et al., 2019; Yang et al., 2022). Shen et al. (2023) integrated a cross-line with tracking-by-detection for grape cluster enumeration, providing an additional filtering mechanism for object tallying in frames. However, the display of the cross-line to the user end is extraneous and appears counterintuitive. Moreover, their method presented the count information in the corner of each frame rather than juxtaposing it with the grape clusters, potentially compromising verifiability. In the study by Rong et al. (2023), they introduced a window overlay on the input video to filter the counting numbers to address the counting problem inherited in tracking-by-detection. The rationale behind the window overlay bears a notable resemblance to the cross-line method proposed by Shen et al. (2023), thus implying a shared set of deficiencies across such approaches. Unfortunately, none of the above methods offered the capability for multiclass counting or the evaluation of their online efficiency, revealing a notable gap in current research paradigms.

The overall objective of this study is to develop an automated end-to-end strawberry fruit counting system that adapts to contemporary digital farming operations. To achieve this goal, the task is structured into three phases: (1) developing a strawberry fruit detection deep learning network by combining YOLOv5s and the attention mechanism which prioritize both detection accuracy and computational efficiency. (2) designing a customized tracking-based object counting module, robust against occlusions, proficient in multiclass enumeration, and maintaining real-time capability; and (3) validating the proposed method using video footage sourced from two strawberry farms.

Table 1
Validation dataset overview.

Keys	Ferme d'Hiver	Ferme gush
Video number	30	10
Duration	15 min	5 min
Motion speed	0.8 m/s	0.15 m/s
Ripe average	49	28
Unripe average	139	86
FPS	30	
Resolution	1920 × 1080	

2. Materials and methods

2.1. Data acquisition

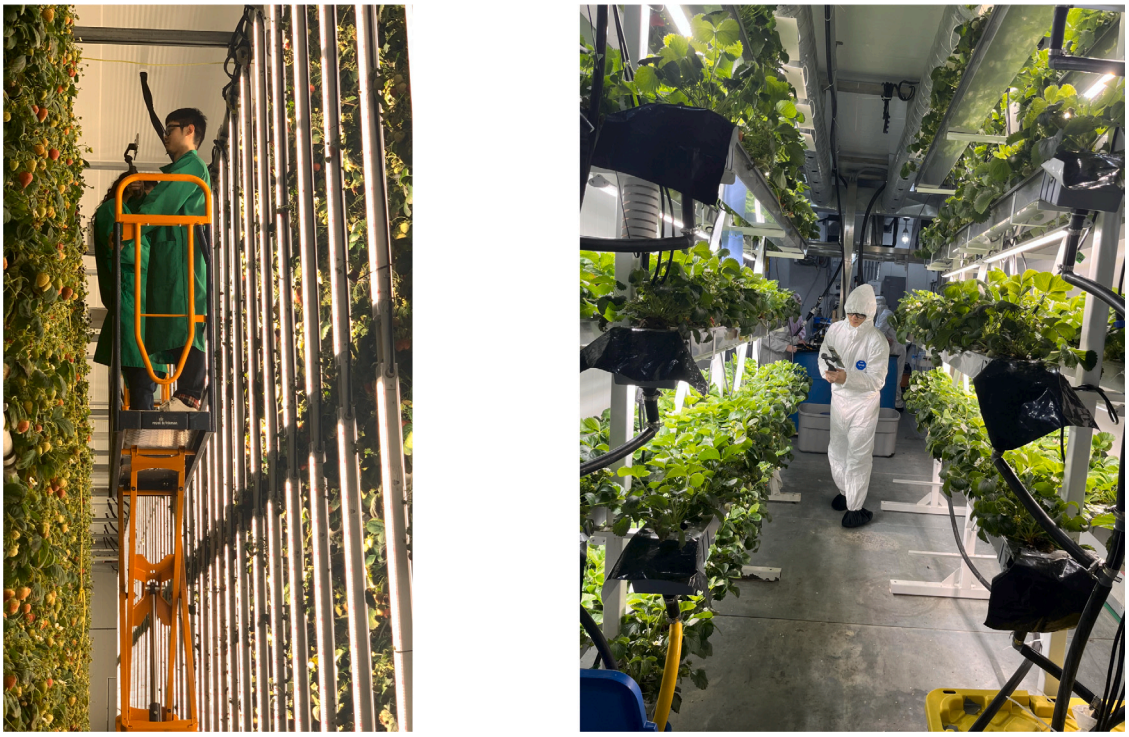
Video data for counting validation. The collection of video data took place in two distinct agricultural environments, both located in the Greater Montréal Area, Canada. The first environment, Ferme d'Hiver, operates as a large-scale commercial vertical farming system (Delden et al., 2021). The planting rack in this farm spans 13.4 meters in length and reaches 4.9 meters in height. Fig. 1a illustrates that we placed ourselves on a lift platform, regulating its vertical and horizontal movements to facilitate video recording. The second environment, Ferme Gush, is an experimental indoor farm (Fig. 1b). The planting rack measures 2.9 meters high and extends 3.0 meters in length. In this setting, we hold the recording device and operate it to collect data at a uniform walking speed.

We utilized a Google Pixel 7 Pro smartphone, mounted on a DJI OSMO Mobile 6 gimbal, for the data collection process. Detailed information on data collection is presented in Table 1. All videos were captured at a resolution of 1920 × 1080 at 30 FPS. In the Ferme d'Hiver, we gathered thirty videos, while in the Ferme Gush, we took ten videos. On average, each video spans a duration of 30 s. At Ferme d'Hiver, a lift platform was utilized, with its speed adjusted to the minimum of approximately 0.8 meters per second, whereas at Ferme Gush, filming equipment was manually operated and transported at a walking speed of about 0.15 meters per second. Within the videos collected from Ferme d'Hiver, an average of 49 ripe fruits and 139 unripe fruits were observed per video. In videos sourced from Ferme Gush, there was an average of 28 ripe fruits and 86 unripe fruits per video.

Image data for detection training. Due to the widespread cultivation of strawberries and the availability of abundant open-source annotated datasets for this crop, we were spared the need for extensive annotation efforts. We selected 440 high-quality, augmented annotated images from these datasets to serve as our training set (Shorten & Khoshgoftaar, 2019). Four images from our training dataset have been randomly selected for demonstration (Fig. 2). Furthermore, we extracted 20 images from the collected video data, which, after annotation, were used as our validation set. To ensure the generalizability of the network we trained, our validation set also incorporated 80 annotated images of strawberries from diverse open-source scenarios.

2.2. Method overview

The overarching structure of our counting method is depicted in Fig. 3. The process unfolds in the following four steps in a systematic manner: (1) Videos sourced from strawberry farms serve as input. (2) A fine-tuned strawberry fruit detection network combining YOLOv5s and the attention mechanism processes these videos, revealing location information in the form of bounding boxes for both ripe and unripe classes. (3) A tracking algorithm discerns individual strawberry fruit identities across consecutive frames, assigning each identity with a unique Track ID. (4) A multiple mapping algorithm extracts the counting information from the tracking algorithm for both classes separately and then presents them to the user end.



(a) (b)

Fig. 1. Video data collection was conducted in two strawberry farms. A cell phone mounted on a gimbal was used for data collection. (a) Thirty video data were collected by standing on the mobile lift platform in the Ferme d'Hiver. (b) Ten video data were collected by walking and shooting in the Ferme Gush.

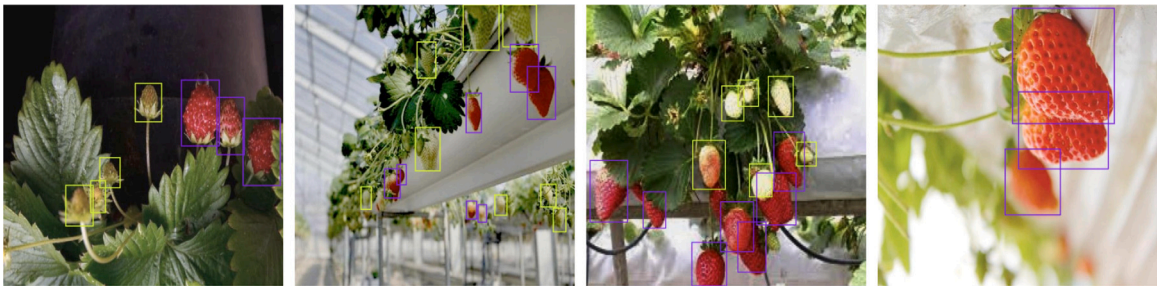


Fig. 2. Four sample images from our training dataset are shown. Ripe strawberries are annotated with purple bounding boxes, while unripe ones are labeled with bright yellow bounding boxes.

(a) (b) (c) (d)

Fig. 3. There are four stages in our strawberry fruit counting method overview. (a) Inputs are videos taken on strawberry farms. (b) Location information in the form of bounding boxes is generated by an object detection network. (c) Track IDs are generated by a tracking algorithm. (d) Track IDs are replaced by counting indicators utilizing our developed MultiMap algorithm.

2.3. Detection module

Object detection is a cornerstone of computer vision research. Among the plethora of frameworks, the YOLO series has earned significant recognition due to its unified architecture (Redmon et al., 2016). Contrary to conventional approaches that distinguish between region proposal and object classification (Ren et al., 2015), YOLO obviates these steps, leading to real-time processing while maintaining satisfactory detection precision. In the evolution of YOLO, the fifth generation, known as YOLOv5, emerged as an object detection framework suitable for downstream industrial applications. This framework incorporates influential work in the field of object detection, such as the path aggregation network (PANet) (Liu et al., 2018). Among its variants, the YOLOv5s model, distinguished by its compactness, is an optimal choice for real-time detection tasks and edge deployment scenarios. Consequently, YOLOv5s has been chosen as our baseline network for strawberry detection.

To augment the efficacy of YOLOv5s, we incorporated various attention mechanisms and transformer modules (Fig. 4):

- **Channel/Spatial Attention Modules:**
  - **CBAM (Convolutional Block Attention Module):** CBAM infers attention maps along both channel and spatial dimensions, refining feature representations accordingly (Woo et al., 2018).
  - **SA-Net (Shuffle Attention Network):** SA-Net provides a light weight and efficient way to model cross-channel relationships and exchange information, improving representational power (Zhang & Yang, 2021).
  - **SimAM (Simplified Attention Module):** SimAM is a simple, parameter-free attention module designed to avoid potential degradation while offering marginal performance gains (Yang et al., 2021).
- **Vision Transformers:**
  - **Vision Transformer (ViT):** ViT repurposes the transformer architecture — originally conceived for natural language processing — for image recognition. It segments images into patches and processes them as a sequence of tokens for parallel computation (Dosovitskiy et al., 2020).
  - **Swin Transformer:** The Swin Transformer builds on ViT by introducing a hierarchical and shifted window approach, optimizing features at varying scales and reducing computational demands (Liu et al., 2021).

These attention and transformer modules can strategically enhance the feature extraction capabilities and performance of YOLOv5s. While some attention mechanisms, such as transformers, can be computationally expensive, others offer a lightweight alternative. Strategically integrating a single attention layer into YOLOv5 can balance the benefits of attention with computational efficiency.

2.4. Tracking module

Upon acquisition of real-time object detection bounding boxes, we want to associate the detected objects with identical objects across frames, while this procedure can be performed using tracking algorithms. In our study, we adopted three recent tracking-by-detection algorithms: DeepSORT (Wojke et al., 2017), ByteTrack (Zhang, Sun et al., 2022), and StrongSORT (Du et al., 2023). Contrary to conventional tracking algorithms, DeepSORT incorporates a learning-based feature extractor to consider object appearance, and StrongSORT further refines this extractor, resulting in state-of-the-art tracking performance on the Multiple Object Tracking (MOT) benchmark dataset (Milan et al., 2016). On the other hand, ByteTrack filters out objects with lower scores from backgrounds with the rationale that objects with lower scores may be occluded or blurred, thereby improving tracking

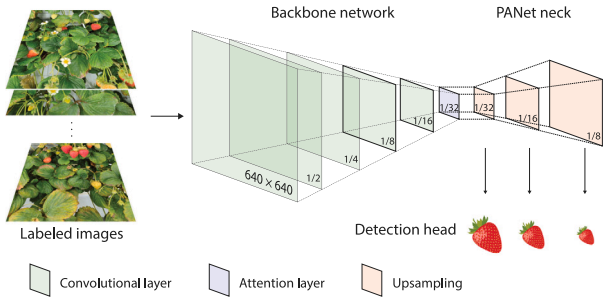


Fig. 4. A brief architecture of a refined detection neural network is presented. Various attention layers were deployed and evaluated.

performance in complex situations. The core of these tracking algorithms is the synergy between the tracks derived from the Kalman filter and the association of detections using the Hungarian algorithm with their respective tracks. To facilitate conceptual clarity in the following discourse, let d_i represent the state of the ith detection in the set of detections $\mathcal{D}$ in the current frame, and t_j signify the jth track in the set of tracks $\mathcal{T}$ from predicted states of the previous frame.

Kalman filter prediction. The Kalman filter is widely used to predict the internal state of a linear dynamic system based on a series of noisy measurements. In our dynamic system, the state and motion space are represented as $(x, y, \gamma, h, \dot{x}, \dot{y}, \dot{\gamma}, \dot{h})$ where (x, y) denotes the centroid of the bounding box, γ signifies the aspect ratio, h represents height, and their corresponding velocities are $\dot{x}$, $\dot{y}$, $\dot{\gamma}$, and $\dot{h}$. In general, the Kalman filtering process bifurcates into two stages: state prediction and state update. During the prediction phase, on the basis of the object position and velocity from the previous frame, the object position and velocity of the current frame are predicted. The update phase leverages the Kalman gain to weigh the importance of the new observation relative to the prediction and corrects the predicted state of an object using the latest measurements. Kalman gain at the time step k during the update phase is derived as

$$K_k = P'_k H_k^T (H_k P'_k H_k^T + R_k)^{-1}, \tag{1}$$

where P'_k and R_k are the covariance of the estimated state and the observation noise, and H_k designates a transition model that maps the state distribution into the measurement space. In the tracking-by-detection paradigm, the noise covariance R_k can be modulated by the detection confidence level c_k, as expressed in

$$\tilde{R}_k = (1 - c_k) R_k, \tag{2}$$

given that elevated confidence levels correlate with diminished observation noise magnitudes (Du et al., 2021).

Association. In tracking algorithms, the association between $\mathcal{D}$ and $\mathcal{T}$ is another critical component. The Hungarian algorithm, which uses a cost matrix to represent the weight associated with each potential assignment, is widely applied to address assignment problems between two sets of entities; in our context, this matrix is used to describe the relationship between d_i and t_j. The Hungarian algorithm we adopted is a variation of the Jonker-Volgenant algorithm, as detailed in Crouse (2016). The methods behind constructing the cost matrix often mark the point of divergence among various tracking algorithms. Therefore, in the following discussion we will use the classical DeepSORT as a representative example to elucidate the concept of association. The measure of distance between d_i and t_j is determined by the following cost function

$$C_{d_i,t_j} = \lambda C^m_{d_i,t_j} + (1 - \lambda) C^a_{d_i,t_j}, \tag{3}$$

where C^m embodies the motion information through computation of the physical Mahalanobis distance. C^a captures the appearance information by calculating the cosine distance between the deep-learned

feature of d_i and the stored features of t_j. The parameter λ, in the range of $[0, 1]$, serves as a trade-off factor that integrates motion and appearance information.

The results of the association can be categorized into matches, where detections are successfully paired with existing tracks; unmatched tracks, which may arise due to occlusions, false detections, or the tracked object leaving the scene; and unmatched detections, which occur when a new object enters the scene or when an existing object is not correctly associated with a track. Following the initial association, an IoU matching offers a second chance for both unmatched tracks and detections to capture runaway matches, thereby enhancing the robustness of the tracking algorithm. If

$$IoU(d_i, t_j) \geq IoU\ threshold, \tag{4}$$

the pair (d_i, t_j) will also be included in the set of matches.

Track ID and ID switch. Tracking algorithms assign a unique Track ID, often an incremental integer, to each individual track. This Track ID serves as an essential tool to preserve the identity of objects, thereby reducing potential ambiguities. In scenarios where multiple objects of similar appearance are present, the implementation of Track IDs ensures clear differentiations between these entities. Without these distinct identifiers, there is an increased likelihood of misidentifying one object with another. Moreover, the provision of the Track ID facilitates a more intuitive interpretation for human analysts. This empowers them to undertake comprehensive analytics based on the trajectory or behavior of specific objects, encompassing aspects like movement patterns and interactions with surrounding entities.

However, the ID switch refers to the incorrect assignment or swap of identities between tracked objects in the subsequent frames, and the phenomenon of the ID switch presents a significant challenge in MOT problems. Fig. 5 illustrates the concept of the ID switch, which occurs when the tracking algorithm misidentifies the tracked object. The figure is a 3D graph with three axes of time, X position and Y position, showing two trajectories: the real trajectory and the trajectory estimated by the Kalman filter, with the ID switch highlighted in blue at time 5. Several factors contribute to this complication: (1) Occlusions, where one object overlaps with another or is concealed behind it, may attribute the identity of one object to another. (2) In scenarios characterized by the presence of multiple objects with similar appearances, such complexities may lead the tracker to inadvertent interchange of their respective IDs. (3) Fast and unpredictable object movements can cause the tracker to lose an object and later misidentify it. (4) If the detector provides inaccurate bounding boxes or misses some detections, it can also lead to ID switches in subsequent tracking phases.

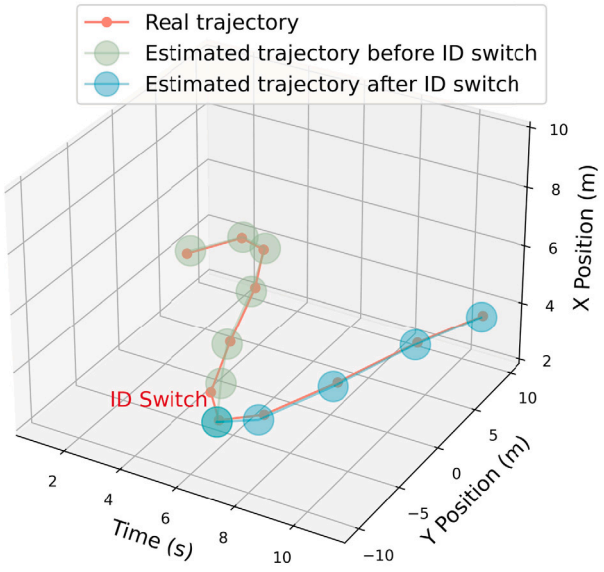


Fig. 5. Real trajectory describes the actual movement path of an object, while estimated trajectory describes the movement path of the object as predicted by the tracking algorithm. The highlighted ID switch event illustrated the moment where the tracking algorithm misidentifies the tracked object.

2.5. Multiple mapping

Before delving into the proposed Multiple Mapping (MultiMap) solutions, it is crucial to clarify the underlying causes of inaccuracies in counting tasks when applying native implementations of tracking algorithms.

State transition. Besides a unique Track ID, each track is also associated with a state indicator, which alternates between Tentative, Confirmed, and Deleted states. Let $\mathcal{S}_T$, $\mathcal{S}_C$, and $\mathcal{S}_D$ symbolize the sets of Tentative, Confirmed, and Deleted tracks, respectively. Parameters n and m are defined as the consecutive hit threshold for track confirmation and the disappearance threshold for track deletion, respectively. The number of consecutive hits for track t is denoted as $H(t)$, while $M(t)$ stands for the number of consecutive frames in which track t has been missed. The state transitions are as follows:

Initialization: For each $d \in D$ not matched with any $t \in \mathcal{T}$, initialize a new track with a unique Track ID and add it to $\mathcal{S}_T$.

From Tentative to Confirmed: For each matched $t \in \mathcal{S}_T$, increment $H(t)$ by 1. Move t from $\mathcal{S}_T$ to $\mathcal{S}_C$ if $H(t) = n$.

From Confirmed to Deleted: For each unmatched $t \in \mathcal{S}_C$, increment $M(t)$ by 1. Move t from $\mathcal{S}_C$ to $\mathcal{S}_D$ if $M(t) = m$.

From Tentative to Deleted: For each unmatched $t \in \mathcal{S}_T$, increment $M(t)$ by 1. Move t from $\mathcal{S}_T$ to $\mathcal{S}_D$ if $M(t) = m$.

The incorporation of state transitions significantly enhances the robustness of the tracking algorithm, especially considering the substantial number of executions on the detection network in noisy video scenes. The threshold of n consecutive hits not only minimizes false positive detections (cases where detections are not the desired objects) but also provides additional data points for the motion model (e.g., the Kalman filter), resulting in a refined motion estimation. Meanwhile, the threshold of m missed tracks curtails false negative cases (situations where desired objects present undetected in a frame) and grants the tracker a grace period to re-identify an object through temporary occlusions, maintaining its original ID.

Typically, setting n to 3 proves effective in noisy environments. However, adjustments may be necessary depending on the complexity of the scene. In more complex scenes, where distinguishing between objects becomes challenging, increasing n helps in ensuring more reliable track confirmation. The value of m, on the other hand, is influenced by the frame rate and the velocity of object movement. A higher m is recommended for scenes with a high frame rate to maintain a consistent duration for track deletion. In cases of slow motion of cameras, it is advisable to increase m, ensuring that the track deletion occurs only after the camera has covered a substantial area.

Problem statement. At this point, an inherent contradiction arises between the Track ID and the state transition when the Track ID is naively considered as a direct counting result. The Track ID is initialized prior to the state transition, which implies that the Track ID might account for detections even before they are confirmed by the state transition. Although the consecutive hit threshold n effectively filters out false positive detections and association failures, Track ID has already incorporated these inaccuracies. Thus, when Track ID is

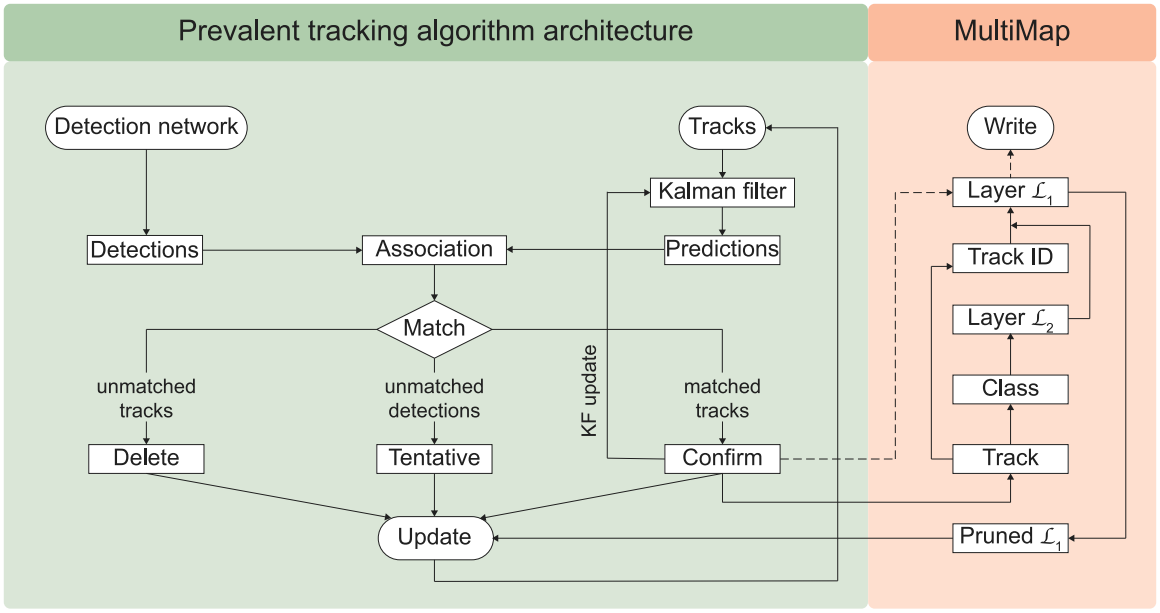


Fig. 6. In this flowchart, we have encapsulated the workflow of the tracking algorithm and provided a visualization detailing our MultiMap algorithm. Within this representation, solid lines delineate the construction of the tracker and reference, while dashed lines indicate the inference and writing stages.

displayed at the user end, it demonstrates a discontinuity in numbering. We define this phenomenon as *ID jumps*. In low-noise scenarios, characterized by minimal occlusions and consistent motion speeds, false positive detections and failure of the association are infrequent, making the error from using Track ID as a counting indicator relatively minor. However, when applying tracking algorithms naively in our contexts, there are significant instances of ID jumps, causing the Track ID to consistently exceed the desired counting result.

It is essential to distinguish between the issues of ID jump and ID switch, as they bear different implications in tracking algorithms. Specifically, ID jump is not perceived as a problem in tracking tasks. The primary purpose of incorporating a Track ID is to serve as an identifier rather than a counting indicator. Provided that the Track ID for an object remains unique and consistent from its appearance to its disappearance, the tracking algorithm remains unaffected by whether the Track ID is continuous when presented to the user. In contrast, tasks focused on counting necessitate strict continuity in their numbering sequence. In essence, the ID switch refers to a scenario where a single object, due to factors such as occlusion or false association, is misidentified as two distinct entities. On the contrary, ID jump represents a phenomenon in which consecutively appearing Track IDs may not exhibit numerical continuity.

MultiMap. Thus, we propose the MultiMap scheme, an effective online solution to the ID jump issue in MOT problems. This scheme, a successor to the Dynamic Mapping algorithm as detailed in Zhou et al. (2023), is both easy to implement and effective in application. While Dynamic Mapping primarily supports single-class counting, MultiMap extends its functionality to enable the counting of objects across multiple distinct classes separately. The intuition behind MultiMap is to concentrate on accounting for tracks within $\mathcal{S}_C$, rather than spanning across all initialized Track IDs. Given that the tracker is updated for tracks in the most recent frame but absent from historical tracks, it is advantageous to employ a buffer to maintain a record of all confirmed tracks. To accommodate our multi-class counting requirements, we utilize a reference to account for all confirmed tracks present up until the current frame while providing a distinction by class. The reference is structured in two layers, denoted by $\mathcal{L}_1$ and $\mathcal{L}_2$. Let $\mathcal{O}$ represent the ensemble of object classes and $\mathcal{L}_2(o)$ designate the mapping for a class o under $\mathcal{L}_2$. In addition, we introduce $\mathcal{L}_2$ as a function to map each confirmed track to a distinct integer. Formally, for $t \in \mathcal{S}_C$ will be assigned to a cardinal number $c \in \mathcal{C}$. We have

$$\begin{cases} \mathcal{L}_1 : \mathcal{O} \to \mathcal{L}_2(o), \\ \mathcal{L}_2 : \mathcal{S}_C \to \mathcal{C}. \end{cases} \tag{5}$$

Within this framework, each confirmed Track ID $t \in \mathcal{S}_C$ is mapped to a unique counting number in the sequence by class.

Algorithm 1 A multi-object mapping scheme that projects Track IDs to counting numbers.

```
procedure MultiMap(S_C, L_1)
    for t in S_C do
        if t.class in L_1 then
            if t.id in L_2 then
                continue
            prev ← max(L_1(t.class).values())
            L_2(t.id) ← prev + 1
        else
            L_2 = init()    # Initiate L_2
            L_2(t.id) ← 1
    return L_1
```

An illustrative overview of the integration of MultiMap with prevalent tracking algorithms is presented in Fig. 6, while Algorithm 1 provides a breakdown specification of our MultiMap scheme. The constructed reference retains historical data from $\mathcal{S}_C$ and synchronizes updates with the tracker. In the posterior stage, rather than displaying the Track ID on each frame, we retrieve the corresponding counting number from the reference using the Track ID and its associated class name. Subsequently, the counting number and class name are displayed along with the bounding boxes on the screen. Through our MultiMap scheme, we manage the counting task in complex scenes.

Having addressed the primary challenge of counting, we want to revisit our method in large-scale application scenarios. As we continuously accumulate historical data in our reference and synchronize it with the tracker, the size of this reference can become cumbersome, particularly when counting tasks in contemporary agricultural practices involve thousands or more objects. Travesing such long lists for each input frame risks compromising our algorithm's real-time efficiency. To address this, we introduce a threshold s within the MultiMap algorithm.

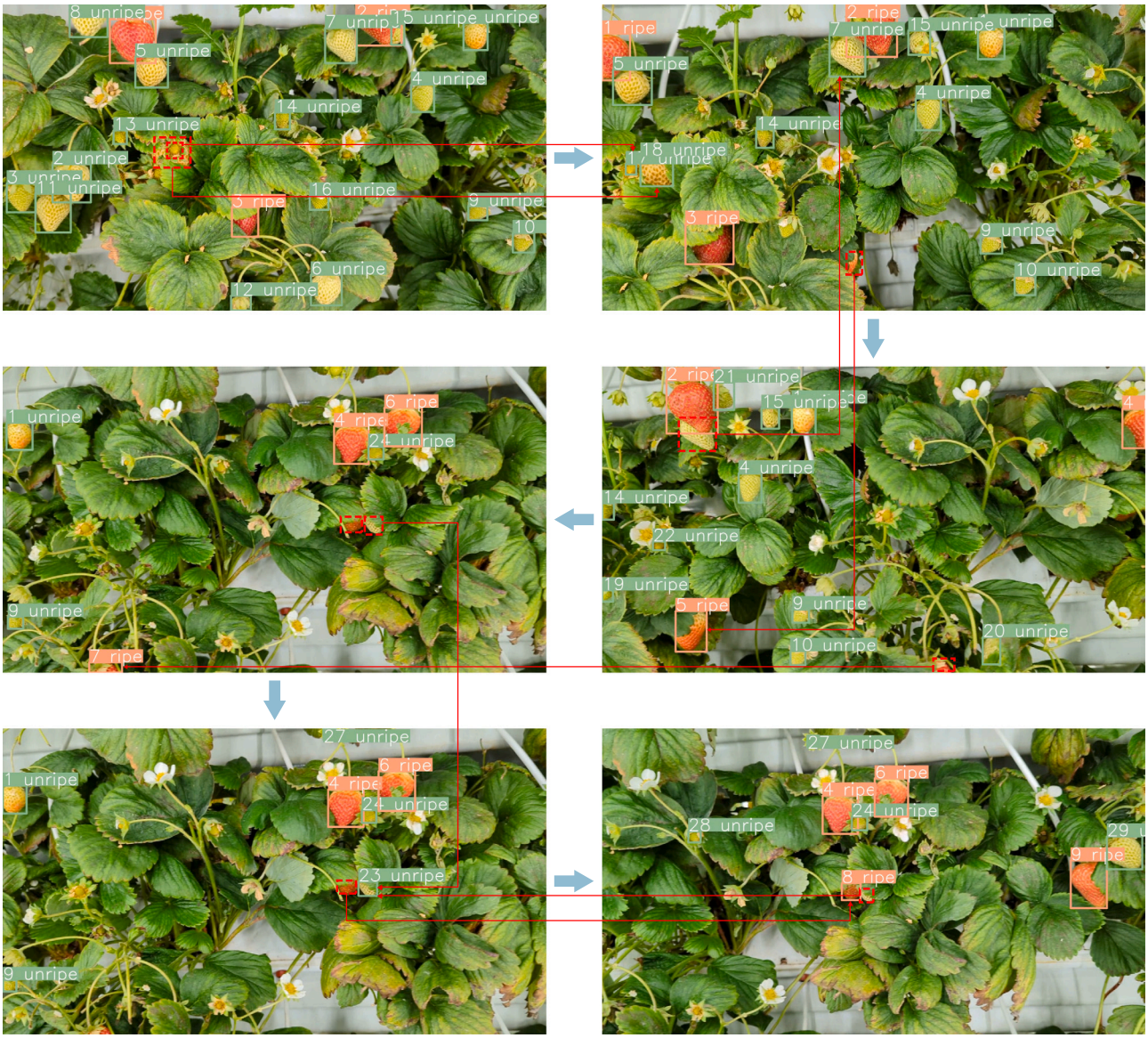


Fig. 7. A randomly selected output video serves as an illustrative example of the counting results. Alongside each bounding box are the counting numbers and the object class. Any false negative detections are highlighted with red dashed bounding boxes. The camera moved from left to right, and the images represent a sequence that spans six consecutive seconds, arranged in the order indicated by the arrows.

We assume that all objects will be present for a maximum of s frames in the input video data. Consequently, any object that persists in our reference beyond the sth frame can be pruned. The value of s can be fine-tuned based on environments such as the approximate number of objects and the relative speed of camera motion in specific contexts. By regulating the size of our reference and ensuring that the MultiMap inference time remains optimal, we optimized our method for real-time counting in large-scale agricultural settings.

2.6. Experimental platform and validation metrics

Computing hardware and settings. All experiments were performed on a workstation equipped with a GeForce RTX 3060 graphics card that possesses 12 GB of memory. For powering these operations, the CPU employed was an Intel Core i7-12700KF. The environment for algorithm development and validation was set up on the Ubuntu 22.04 LTS operating system. As for the software specifications, our system utilized Python version 3.7.16. Complementing this, deep learning models and associated tasks were developed using PyTorch version 1.13.1, along with its auxiliary library, torchvision, version 0.14.1. Furthermore, image processing tasks were facilitated through the OpenCV library, which was version 4.7.0. Parallel computation support was enabled using the NVIDIA CUDA toolkit, version 11.5. Our object detection framework employs Ultralytics YOLO as detailed in Jocher et al. (2023). The implementation of our tracking-by-detection algorithm is presented in Broström (2022).

Validation metrics. In evaluating the detection network, the primary metric adopted was the mean Average Precision (mAP). In addition to mAP, we also assessed ancillary metrics, including GPU memory occupation and total training duration under various parameter configurations. To quantify the counting performance, we employed metrics such as the adjusted coefficient of determination, denoted as R^2_{adj} and the Root Mean Squared Error (RMSE). To enhance the scrutiny of counting accuracy, both the Mean Absolute Error (MAE) and the Mean Relative Error (MRE) were also computed. Furthermore, given that our MultiMap maintains counting in real-time order, we also evaluated the counting method in terms of average frames per second (FPS).

Table 2
An examination of different backbone networks, assessing their GPU memory allocation per epoch and their resultant mAP scores, all tests were conducted under a uniform batch size of 16.

Backbone network	Parameter volume	Attention head	Memory usage	Training duration	mAP50	mAP50:95
YOLOv5s	7 027 720	–	4.69*G*	0.375*hr*	0.836	0.459
YOLOv5s + CBAM	5 870 443	–	**4.19G**	0.355*hr*	0.899	0.545
YOLOv5s + SA-Net	7 018 984	–	4.34*G*	0.376*hr*	0.917	0.559
YOLOv5s + SimAM	7 203 848	–	4.67*G*	0.393*hr*	0.854	0.476
YOLOv5s + ViT	7 018 984	4	4.75*G*	0.389*hr*	0.878	0.505
YOLOv5s + ViT	7 018 984	8	4.87*G*	0.397*hr*	0.883	0.515
YOLOv5s + ViT	7 018 984	16	4.97*G*	0.400*hr*	**0.929**	**0.563**
YOLOv5s + Swin	7 153 904	8	4.71*G*	0.377*hr*	0.860	0.478
YOLOv5s + Swin	7 155 704	16	4.75*G*	0.385*hr*	0.904	0.560
YOLOv5s + Swin	7 159 304	32	4.95*G*	0.403*hr*	0.880	0.536

3. Results and discussion

3.1. Method representation

In Fig. 7, we present the initial few frames in a random output of our method. Although false negative instances might occur periodically across video data frames, as exemplified by the red dashed bounding box cases, the nature of our method significantly mitigates this issue: it is uncommon for a detection network to consistently overlook an object across all appeared frames. As evidenced in this figure, most false negative detections from a particular frame are eventually detected and integrated into the count in the previous or subsequent frames. For example, the fifth ripe fruit, absent in the second frame, is taken into account in subsequent frames in the third frame.

The primary challenge in employing native tracking algorithms for counting tasks is illustrated in Fig. 9. Within complex application contexts like ours, the issue of ID jumps becomes nearly inevitable. The predominant sources of ID jumps arise from false positive detections and failures in the association between tracks and detections. To provide a more intuitive demonstration of the ID jump issue, we focus exclusively on the channel for the ripe fruit class. As depicted in Fig. 9a, although only four ripe fruits are present in the image, StrongSORT produces Track IDs numbering up to eight. Importantly, this ID jump issue is not unique to StrongSORT but pervades many mainstream tracking algorithms. Our MultiMap function associates these Track IDs with an internal counter, enabling our method to retrieve the corresponding counting number and display it as shown in Fig. 9b.

3.2. Strawberry detection result

The default YOLOv5s is used as a baseline. The ninth convolutional layer of YOLOv5s was substituted with the ViT and Swin Transformer layers respectively, which possess varying numbers of attention heads. The feature map of this ninth layer has dimensions of 20 × 20 and consists of 1024 channels. This replacement strategy was motivated by reducing the high computational demand inherent in attention mechanisms. We utilized the Adam optimizer with an initial learning rate set at 10^{-3}, running 300 epochs for each backbone network. The batch size was set to 16, and the window size of the Swin Transformer was set to 8. The confidence threshold and the non-maximum suppression (NMS) IoU threshold were both set at 0.85. Given the inherent multiple normalization layer of YOLOv5s, the Layer Normalization from both the ViT and Swin Transformer layers was removed, resulting in slight savings in computational resources and a minor improvement in model accuracy (Ba et al., 2016; Ioffe & Szegedy, 2015).

As illustrated in Table 2, the incorporation of attention mechanisms notably enhances detection performance across all networks when compared to the default YOLOv5s baseline. Specifically, the integration of YOLOv5s with CBAM and SA-Net modules has significantly improved detection accuracy while reducing GPU memory consumption. In particular, mAP50 has achieved scores of 0.899 and 0.917, respectively, for the integration of CBAM and SA-Net. In contrast, the improvement in precision with mAP50 at 0.854 attributed to the integration of SimAM has not been substantial in our context. Moreover, the ViT layer with 16 attention heads produced superior results, registering mAP50 and mAP50:95 scores of 0.929 and 0.563, respectively. Importantly, the integration of Swin Transformer with 16 attention heads yielded impressive detection results, evidenced by mAP50 scores of 0.904 and mAP50:95 scores of 0.560. Additionally, it proved to be more memory-efficient, allocating 4.75G, in contrast to the 4.97G allocated by the integration of ViT. However, increasing the number of attention heads to 32 in the Swin Transformer led to a decrease in performance, suggesting that the benefits of attention mechanisms are not strictly linearly correlated with the number of heads. This may be attributable to the greed of attention mechanism for a large volume of high-quality labeled data; an excessive number of attention heads without a corresponding increase in training data may, in fact, degrade network performance.

Due to the opaque nature of deep learning models, we selectively exhibit feature maps from various layers of the YOLOv5s + ViT model to clarify and enhance the understanding of our deep learning application in the strawberry detection task. By presenting these feature maps as heatmaps, these maps enable us to discern which regions and features the model prioritizes and assigns significance to during the detection process. Additionally, Grad-CAM (Gradient-weighted Class Activation Mapping) is used to highlight important regions in the input image that influence the model's predictions (Selvaraju et al., 2017). As shown in Fig. 8, the figure includes a sample input image, Grad-CAM result, and feature maps across different stages (0, 4, 8, 9, 12, 15) with layer Conv, C3, ViT, SPPF, Concat, and Upsample, providing a comprehensive insight into the internal workings and decision-making process of the model.

3.3. Strawberry counting result

To assess the scalability and effectiveness of our MultiMap algorithm in counting tasks, we conducted experiments on DeepSORT, ByteTrack, and StrongSORT, as presented in Table 3. Parameters n and m were set to 3 and 15, respectively, while the IoU threshold for tracking determined at 0.7. Our ground truth was established by averaging the independent counts of two people. Given that native tracking algorithms are not inherently designed for counting tasks and do not offer functionality for distinct class counts, we aggregated the total counts across both ripe and unripe classes and compared them with the ground truth. Assuming that the final counting result for the strawberries within the video is represented by the largest Track ID obtained at the end of each run of the native tracking algorithms, it is evident that the counting accuracy of DeepSORT, ByteTrack, or StrongSORT is considerably low. However, with the integration of our MultiMap algorithm, all three tracking algorithms exhibited a substantial increase in counting accuracy. The MRE witnessed decreases from 209.8% to 6.7%, 356.5% to 23.8%, and 110.1% to 10.0% for DeepSORT, ByteTrack, and StrongSORT, respectively. Furthermore, integration with MultiMap not only led to a significant improvement in

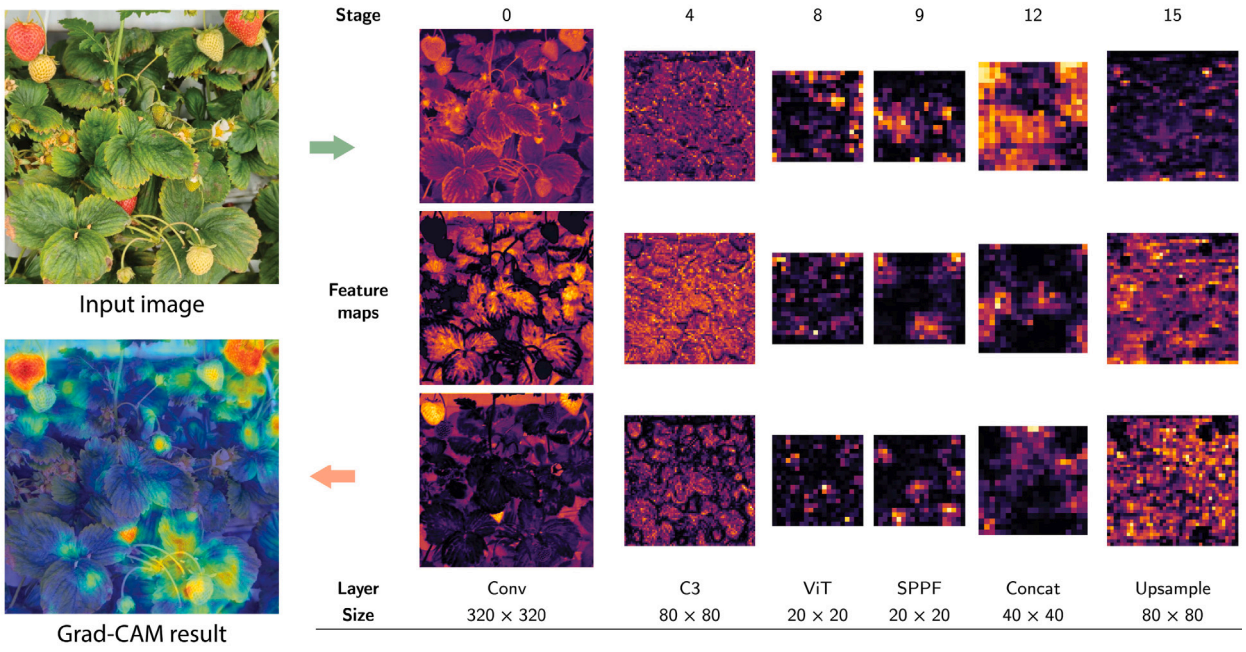


Fig. 8. To enhance human understanding and provide insights into the imagery analysis of strawberry farms, feature maps and a Grad-CAM result for the interpretation of a single input image frame from the YOLOv5s + ViT model are demonstrated. The feature map selection involves the visualization of six distinct stages (Convolution, C3, ViT, SPPF, Concat, Upsample) out of the total twenty-three stages of the network. At each selected stage, three feature maps are randomly chosen from the multitude of channels within each specific layer.

(a) (b)

Fig. 9. We present a side-by-side comparison of the outputs for the same frame generated by StrongSORT and MultiMap. For brevity, we only displayed the channel corresponding to the ripe class. (a) depicts the StrongSORT output. (b) shows the MultiMap output. An example of the ID jump problem is evident in the StrongSORT output.

Table 3
A table demonstrates the effectiveness of the MultiMap function. Thirty data points under Ferme d'Hiver scenarios were examined. In the regression equation, x denotes the method count, whereas y signifies the ground truth.

Method	FPS	Regression equation	R^2_{adj}	RMSE	MAE	MRE (%)
DeepSORT (Wojke et al., 2017)	23.7	$y = 0.18x + 81.75$	0.61	458.1	394.7	209.8
DeepSORT + MultiMap	22.3	$y = 1.12x - 12.85$	0.95	**17.8**	**12.9**	**6.7**
ByteTrack (Zhang, Sun et al., 2022)	39.8	$y = 0.09x + 107.48$	0.59	865.8	699.9	356.5
ByteTrack + MultiMap	**37.2**	$y = 0.78x + 6.88$	0.88	51.3	43.2	23.8
StrongSORT (Du et al., 2023)	26.4	$y = 0.45x + 11.38$	0.77	214.2	201.0	110.2
StrongSORT + MultiMap	25.8	$y = 0.94x - 4.23$	**0.99**	18.8	17.4	10.0

MRE, but also manifested enhancements in the adjusted coefficient of determination R^2_{adj}, RMSE and MAE.

In the analysis, integration with DeepSORT witnessed the highest counting accuracy of 6.7% MRE observed among all online countings, while integration with StrongSORT also demonstrated promising performance, achieving an online counting MRE of 10.0%. Without any parameter fine-tuning, the counting results of ByteTrack were not optimal. However, in terms of data processing speed, ByteTrack demonstrated significant potential, achieving an average of 37 FPS. Upon further investigation, ByteTrack and its variants might possess substantial promise for tasks necessitating high responsiveness. To provide a more nuanced understanding, the distribution of the data along with the respective regression lines for three trackers integrated with MultiMap is represented in Fig. 10. When considering counting accuracy, data correlation and processing speed, the integration of MultiMap with DeepSORT and StrongSORT constitutes viable solutions for real-time counting tasks in sophisticated agricultural applications.

Table 4
Utilizing video data collected from two farms, forty data points in total, we conducted an analysis of the MultiMap algorithm based on DeepSORT and StrongSORT, evaluating its online counting accuracy for both ripe and unripe fruit classes.

Method	Class	Regression equation	R^2_{adj}	RMSE	MAE	MRE (%)
DeepSORT + MultiMap	Ripe	$y = 1.08x - 0.89$	0.94	**5.9**	**4.4**	**8.7**
	Unripe	$y = 1.15x - 17.37$	0.94	16.1	12.4	10.4
StrongSORT + MultiMap	Ripe	$y = 0.90x + 1.17$	**0.97**	5.8	4.9	9.9
	Unripe	$y = 0.93x - 0.90$	**0.99**	**13.0**	**11.5**	**9.9**

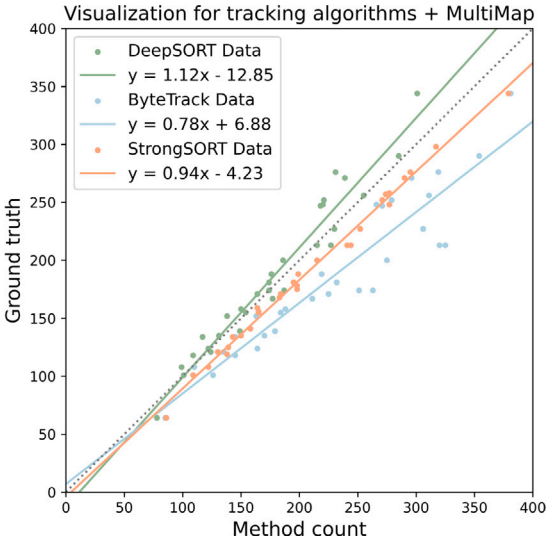


Fig. 10. A visualization of the results obtained from three prevalent tracking algorithms when integrated with MultiMap.

Following a preliminary evaluation of the efficacy of our MultiMap algorithm in three trackers, we decided to delve deeper into the integration of MultiMap with DeepSORT and StrongSORT. We were particularly interested in the counting accuracy for both ripe and unripe fruit classes. Using video datasets obtained from two different farms, our data analysis is cataloged in Table 4 and portrayed in Fig. 11. Overall, the counting performance for ripe fruits was slightly better than for unripe fruits, as expected. As articulated in the table, integration with DeepSORT achieves the best counting accuracy for ripe fruits with an error rate of 8.7%. In contrast, integration with StrongSORT manifests the best counting accuracy for unripe fruits with an error rate of 9.9%. On examining the figure, the results of DeepSORT integration exhibit a greater degree of discreteness in comparison to that of StrongSORT integration. This finding is consistent with the table, where DeepSORT integration's adjusted R^2 value is demonstrably lower. While the integration with DeepSORT may exhibit a lower error rate, the discreteness of its data points compromises its reliability in applications. This is significant when considering the potential calibration of the method results during the post-processing stage, where higher R^2_{adj} values are particularly desirable.

3.4. Discussion

Advantages. Our method improves on the common practice of adding a window overlay to tracking algorithms to fix counting problems (Shen et al., 2023). Ours delves into the foundational structure of the tracking algorithms to pull out the counting information, which is then shown to the user. The method proposed in this study is characterized by several distinctive advantages. In terms of *scalability*, it is adaptive to accept inputs as elementary as videos captured with a smartphone. By harnessing the intrinsic multiclass attributes of tracking algorithms, we have addressed *multiclass counting* challenges, allowing object counting within distinct classes. The implementation of our MultiMap algorithm introduces a minimum computational overhead, maintaining its *real-time capability*. A comprehensive theoretical discussion emphasizing its *robustness to occlusions* can be found in Section 2.5. The representation of the output facilitates user-end *verifiability* by displaying the counting information next to the bounding boxes of the strawberry fruit objects.

Limitations and future works. We have successfully addressed the problem of innate ID jump in counting, but the problem of ID switch inherent in the tracking algorithms remains a primary source of error in our approach. To better cater to real-world application needs, there remains a large room for refinement from both the theoretical and practical perspectives. Theoretically, we have yet to initialize the velocity for our Kalman filters, while prior knowledge about the motion speed of the camera can result in improved tracking performance. Moreover, rather than using default ReID models (Zheng et al., 2019), a customized strawberry feature extractor may capture more information on strawberry appearance so that further improves tracking performance in association. Practically, comparing data collection via smartphones that introduces inevitable jittering and intermittent defocusing issues, cameras mounted on unmanned aerial vehicle (UAV) platforms usually source steadier video footage in given motion speeds, thereby enhancing the counting accuracy. Lastly, although our method was validated only within our strawberry farm contexts, our method can be extended to more generalized scenarios with minimal adaptations. Furthermore, our work does not differentiate in counting between target and non-target rows in the current phase. In practical applications, it is possible to adjust the camera's focal length and the distance from the plants, as well as employ data preprocessing methods such as cropping, to ensure that the field of view encompasses only the target row. In response to the practical needs of applications, we will incorporate a module to distinguish between foreground and background in our future work.

4. Conclusion

Our method effectively formulates tracking algorithms into counting algorithms. The multiclass strawberry detection network we trained attained optimal detection accuracy, with mAP50 and mAP50:95 scores registering at 0.929 and 0.563, respectively. The effectiveness of our developed MultiMap algorithm is demonstrated by the significant reduction in error rate from 209.8% to 6.7%, 356.5% to 23.8% and 110.2% to 10.0%, when integrated with different tracking algorithms. Regardless of the class of ripe and unripe, the synergy between DeepSORT and MultiMap achieved the highest counting accuracy with an error margin of 6.7%. Specifically, the DeepSORT and MultiMap combination achieves the best counting accuracy for ripe fruits with an error rate of 8.7%, whereas the StrongSORT and MultiMap combination manifests the best counting accuracy for unripe fruits with an error rate of 9.9%. After MultiMap integration, execution speeds are maintained at 22 FPS for DeepSORT, 25 FPS for StrongSORT, and 37 FPS for ByteTrack. To summarize, our method is proficient in addressing real-time counting challenges that are robust to occlusions in modern strawberry farms.

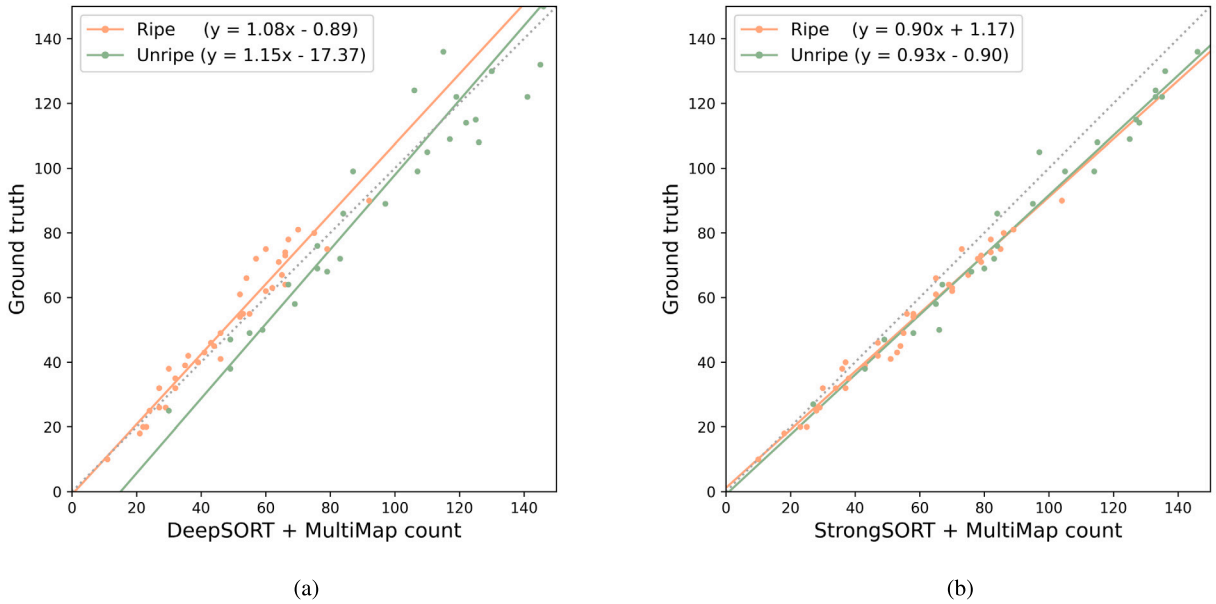


Fig. 11. This figure visualizes the data points referenced in Table 4. (a) DeepSORT + MultiMap. (b) StrongSORT + MultiMap.

CRediT authorship contribution statement

Xuehai Zhou: Developed the methods, Wrote and optimized the code, Conducted examinations, Collected video data from Ferme d'Hiver and Ferme Gush, Led the paper writing. **Yuyang Zhang:** Collected training data for the detection model, Trained detection networks, Performed network examinations. **Xintong Jiang:** Provided preliminary data collected from the Ferme Gush and suggestions on method development. **Kashif Riaz:** As the Director of Smart Farming and Chief Agronomist of Ferme d'Hiver Technologies (FHT), he provided invaluable expertise, guidance, and access to the infrastructure necessary for video data collection. **Phil Rosenbaum:** As the cofounder of Ferme Gush, he played a pivotal role in facilitating video data collection at Ferme Gush. **Mark Lefsrud:** As a member of the committee board, he offered co-supervision guidance, Resource reallocation. **Shangpeng Sun:** As Principal Investigator, he provided supervision advice on manuscript revision, and general direction for the project.

Declaration of competing interest

The authors declare the following financial interests/personal relationships which may be considered as potential competing interests: Shangpeng Sun reports financial support was provided by Weston Family Foundation. Shangpeng Sun reports financial support was provided by FRQNT. If there are other authors, they declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Data availability

Data will be made available on request.

Acknowledgments

This work was supported by Weston Family Foundation the Homegrown Innovation Challenge-Spark Award project (SA-10258) and FRQNT & MAPAQ Partnership Research Program-Sustainable Agriculture (259806).

References

Ba, J. L., Kiros, J. R., & Hinton, G. E. (2016). Layer normalization. arXiv preprint arXiv:1607.06450.

Bahdanau, D., Cho, K., & Bengio, Y. (2014). Neural machine translation by jointly learning to align and translate. arXiv preprint arXiv:1409.0473.

Bewley, A., Ge, Z., Ott, L., Ramos, F., & Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)* (pp. 3464–3468). IEEE.

Broström, M. (2022). Real-time multi-object tracker. https://github.com/mikel-brostrom/yolo_tracking/releases/tag/v8.0, GitHub repository.

Crouse, D. F. (2016). On implementing 2D rectangular assignment algorithms. *IEEE Transactions on Aerospace and Electronic Systems*, *52*(4), 1679–1696. http://dx.doi.org/10.1109/TAES.2016.140952.

Cui, J., Zheng, H., Zeng, Z., Yang, Y., Ma, R., Tian, Y., Tan, J., Feng, X., & Qi, L. (2023). Real-time missing seedling counting in paddy fields based on lightweight network and tracking-by-detection algorithm. *Computers and Electronics in Agriculture*, *212*, Article 108045.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., & Gelly, S. (2020). An image is worth 16x16 words: Transformers for image recognition at scale. arXiv preprint arXiv:2010.11929.

Du, Y., Wan, J., Zhao, Y., Zhang, B., Tong, Z., & Dong, J. (2021). Giaotracker: A comprehensive framework for mcmot with global information and optimizing strategies in visdrone 2021. In *Proceedings of the IEEE/CVF international conference on computer vision (CVPR)* (pp. 2809–2819).

Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., & Meng, H. (2023). Strongsort: Make deepsort great again. *IEEE Transactions on Multimedia*.

Gao, F., Fang, W., Sun, X., Wu, Z., Zhao, G., Li, G., Li, R., Fu, L., & Zhang, Q. (2022). A novel apple fruit detection and counting methodology based on deep learning and trunk tracking in modern orchard. *Computers and Electronics in Agriculture*, *197*, Article 107000.

Ghosal, S., Zheng, B., Chapman, S. C., Potgieter, A. B., Jordan, D. R., Wang, X., Singh, A. K., Singh, A., Hirafuji, M., & Ninomiya, S. (2019). A weakly supervised deep learning framework for sorghum head detection and counting. *Plant Phenomics*.

Ioffe, S., & Szegedy, C. (2015). Batch normalization: Accelerating deep network training by reducing internal covariate shift. In *International conference on machine learning* (pp. 448–456). pmlr.

James, C., Smith, D., He, W., Chandra, S. S., & Chapman, S. C. (2024). GrainPointNet: A deep-learning framework for non-invasive sorghum panicle grain count phenotyping. *Computers and Electronics in Agriculture*, *217*, Article 108485.

Jiang, Y., Li, C., Paterson, A. H., & Robertson, J. S. (2019). DeepSeedling: Deep convolutional network and Kalman filter for plant seedling detection and counting in the field. *Plant Methods*, *15*(1), 141.

Jocher, G., Chaurasia, A., & Qiu, J. (2023). YOLO by ultralytics. URL: https://github.com/ultralytics/ultralytics, GitHub repository.

Liu, Z., Lin, Y., Cao, Y., Hu, H., Wei, Y., Zhang, Z., Lin, S., & Guo, B. (2021). Swin transformer: Hierarchical vision transformer using shifted windows. In *Proceedings of the IEEE/CVF international conference on computer vision (CVPR)* (pp. 10012–10022).

Liu, S., Qi, L., Qin, H., Shi, J., & Jia, J. (2018). Path aggregation network for instance segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition* (pp. 8759–8768).

Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., & Kim, T.-K. (2021). Multiple object tracking: A literature review. *Artificial Intelligence*, *293*, Article 103448.

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., & Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831.

Redmon, J., Divvala, S., Girshick, R., & Farhadi, A. (2016). You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition (CVPR)* (pp. 779–788).

Ren, S., He, K., Girshick, R., & Sun, J. (2015). Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in Neural Information Processing Systems (NeurIPS)*, *28*.

Rong, J., Zhou, H., Zhang, F., Yuan, T., & Wang, P. (2023). Tomato cluster detection and counting using improved YOLOv5 based on RGB-D fusion. *Computers and Electronics in Agriculture*, *207*, Article 107741.

Selvaraju, R. R., Cogswell, M., Das, A., Vedantam, R., Parikh, D., & Batra, D. (2017). Grad-cam: Visual explanations from deep networks via gradient-based localization. In *Proceedings of the IEEE international conference on computer vision* (pp. 618–626).

Shen, L., Su, J., He, R., Song, L., Huang, R., Fang, Y., Song, Y., & Su, B. (2023). Real-time tracking and counting of grape clusters in the field based on channel pruning with YOLOv5s. *Computers and Electronics in Agriculture*, *206*, Article 107662.

Shorten, C., & Khoshgoftaar, T. M. (2019). A survey on image data augmentation for deep learning. *Journal of Big Data*, *6*(1), 1–48.

Sun, S., Li, C., Chee, P. W., Paterson, A. H., Jiang, Y., Xu, R., Robertson, J. S., Adhikari, J., & Shehzad, T. (2020). Three-dimensional photogrammetric mapping of cotton bolls in situ based on point cloud segmentation and clustering. *ISPRS Journal of Photogrammetry and Remote Sensing*, *160*, 195–207.

Van Delden, S., SharathKumar, M., Butturini, M., Graamans, L., Heuvelink, E., Kacira, M., Kaiser, E., Klamer, R., Klerkx, L., & Kootstra, G. (2021). Current status and future challenges in implementing and upscaling vertical farming systems. *Nature Food*, *2*(12), 944–956.

Wojke, N., Bewley, A., & Paulus, D. (2017). Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)* (pp. 3645–3649). IEEE, http://dx.doi.org/10.1109/ICIP.2017.8296962.

Wolfert, S., Ge, L., Verdouw, C., & Bogaardt, M.-J. (2017). Big data in smart farming-a review. *Agricultural Systems*, *153*, 69–80.

Woo, S., Park, J., Lee, J.-Y., & Kweon, I. S. (2018). Cbam: Convolutional block attention module. In *Proceedings of the European conference on computer vision* (pp. 3–19).

Wu, Z., Sun, X., Jiang, H., Gao, F., Li, R., Fu, L., Zhang, D., & Fountas, S. (2023). Twice matched fruit counting system: An automatic fruit counting pipeline in modern apple orchard using mutual and secondary matches. *Biosystems Engineering*, *234*, 140–155.

Wu, Z., Sun, X., Jiang, H., Mao, W., Li, R., Andriyanov, N., Soloviev, V., & Fu, L. (2023). NDMFCS: An automatic fruit counting system in modern apple orchard using abatement of abnormal fruit detection. *Computers and Electronics in Agriculture*, *211*, Article 108036.

Yang, H., Chang, F., Huang, Y., Xu, M., Zhao, Y., Ma, L., & Su, H. (2022). Multi-object tracking using deep SORT and modified CenterNet in cotton seedling counting. *Computers and Electronics in Agriculture*, *202*, Article 107339.

Yang, L., Zhang, R.-Y., Li, L., & Xie, X. (2021). Simam: A simple, parameter-free attention module for convolutional neural networks. In *International conference on machine learning* (pp. 11863–11874). PMLR.

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., & Wang, X. (2022). ByteTrack: Multi-object tracking by associating every detection box. In *Proceedings of the European conference on computer vision (ECCV)* (pp. 1–21).

Zhang, Q.-L., & Yang, Y.-B. (2021). Sa-net: Shuffle attention for deep convolutional neural networks. In *ICASSP 2021-2021 IEEE international conference on acoustics, speech and signal processing* (pp. 2235–2239). IEEE.

Zhang, Y., Zhang, W., Yu, J., He, L., Chen, J., & He, Y. (2022). Complete and accurate holly fruits counting using YOLOX object detection. *Computers and Electronics in Agriculture*, *198*, Article 107062.

Zhao, J., Kaga, A., Yamada, T., Komatsu, K., Hirata, K., Kikuchi, A., Hirafuji, M., Ninomiya, S., & Guo, W. (2023). Improved field-based soybean seed counting and localization with feature level considered. *Plant Phenomics*, *5*, 0026.

Zheng, Z., Hu, Y., Guo, T., Qiao, Y., He, Y., Zhang, Y., & Huang, Y. (2023). AGHRNet: An attention ghost-HRNet for confirmation of catch-and-shake locations in jujube fruits vibration harvesting. *Computers and Electronics in Agriculture*, *210*, Article 107921.

Zheng, Z., Yang, X., Yu, Z., Zheng, L., Yang, Y., & Kautz, J. (2019). Joint discriminative and generative learning for person re-identification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition* (pp. 2138–2147).

Zhou, X., Zhang, Y., Sun, S., & Rosenbaum, P. (2023). A dynamic object counting method for strawberry fruits using vision transformer networks and Kalman filter tracking. In *ASABE annual international meeting* (p. 1). American Society of Agricultural and Biological Engineers.