Universiteit Leiden

Master Computer Science

Increasing Visual Tracking Algorithm Robustness to the Challenges of Agricultural Datasets

Name:	Irina-Mona Epure
Student ID:	3373541
Date:	30/06/2023
Specialisation:	Artificial Intelligence
1st supervisor:	Dr. Erwin M. Bakker
2nd supervisor:	Prof. dr. Michael S. K. Lew

Master's paper in Computer Science

Leiden Institute of Advanced Computer Science (LIACS)
Leiden University
Niels Bohrweg 1
2333 CA Leiden
The Netherlands

Abstract

Due to the accelerating rise in the global population, it is more important than ever to increase the efficiency of agricultural processes. To facilitate downstream applications such as yield forecasting, we must first improve computer vision techniques on agricultural datasets, which present their own unique set of challenges. In this paper, we focus on the use case of monitoring strawberry crops, and we analyze specific challenges for the multi-object tracking task on the Growing Strawberries Dataset (GSD). Namely, issues such as camera motion, low frame rates, the prevalence of non-linear movements, and the appearance evolution of the objects increase the difficulty of tracking the strawberries in this dataset. We replicate the characteristics of GSD through a series of transformations applied to the popular multi-object tracking benchmark MOT20. We give details on how data enhancements can be carefully curated to achieve similar properties as found in GSD, and then check the robustness of state-of-the-art tracking algorithms DeepSORT and StrongSORT to these particularities, by benchmarking them on GSD and ModMOT20. We also propose additional features for these algorithms, which lead to a considerable improvement in terms of tracking performance. Specifically, the HOTA performance of StrongSORT has improved on GSD from 27.9 to 42.8, and on the ModMOT20 from 34.9 to 39.0. We explain how the performance enhancing features such as camera shift compensation, immediate track initialization, aspect ratio verification, relaxed location matching constraints, and appearance-only matching can be implemented and applied on other datasets similar to GSD.

1 Introduction

The current rapid rate of population increase is causing enormous concerns about the human species' survival on food resources that are naturally limited on planet Earth. It is estimated that the total food production will need to increase by 51% before 2050 in order to suffice for the predicted population of that year [70]. The scientific consensus is that the best approach for sustaining this swift population growth is to optimize agricultural activities through the use of existing and emerging technologies. This is hardly a new initiative, as innovative agriculture projects started being developed in the 1980s, and include the use of Artificial Intelligence (AI) techniques, wireless networks, robots, and drones to manage and automate the care of crops such as corn, cotton, citrus, wheat, apple, and soybean [3]. One of the main goals that technology can help achieve is the reduction of food waste through crop monitorization, and yield predictions, which can help optimize the planning of employee shifts and supply deliveries to clients.

In this paper, we aim to facilitate the further advancement of crop monitorization through multi-object tracking (MOT) geared specifically towards the agricultural sector. We focus on the use-case of the Growing Strawberries Dataset (GSD) [77], which contains time-sequential images of strawberries as they grow in a greenhouse. This dataset was created to facilitate a downstream application in which the total number of strawberries, their associated ages, and their progressively changing appearances are used to forecast the yield obtained at a specific time after the last recorded image. However, it presents several particularities that make tracking challenging, including the evolving appearances of the fruits, and the non-linear movements caused by environmental factors.

Although the plants grow under controlled conditions in terms of humidity and temperature, the light conditions are natural. This leads to colors being distorted by shadows and reflections due to the varying lighting angles from the sun, as well as nightly periods when the captured images are dark. In addition, the strawberries sometimes present non-linear movements when the greenhouse workers reposition them to help them grow or accidentally move them when passing by. They also pick the fruits when they are ripe, and regularly spray the plants, during which time the cameras are covered in order to protect them. The strawberries themselves often push other fruits away while growing, due to occupying more space. Sometimes, the cameras present vertical shifts downwards due to humans moving them accidentally or due to gravity itself. All of these irregular changes and gaps in usable images lead to an increase in the difficulty of tracking individual strawberries across sequences of frames. While these issues hinder our ability to easily track the fruits, their existence is generally to be expected in real agricultural applications involving tracking. Hence, it is important to create a tracking system that functions well when faced with difficult impediments such as these.

We advocate for the evaluation of the robustness of popular tracking algorithms against some of the particularities of industrial and agricultural datasets such as GSD. We encourage the use of data transformation techniques to replicate difficulty-inducing characteristics like camera motion, non-linear movements, and appearance evolution. An example is offered of how this can be achieved by using shifting to simulate camera movements, and altering color values on the MOT20 dataset. This has resulted in a new dataset, ModMOT20, which closely resembles the characteristics of GSD, in terms of properties like the Intersection-over-Union (IoU) of

consecutive bounding boxes (bboxes), and the color evolution of the subjects and images. We provide further insight on how to create such a dataset in order to evaluate the robustness of trackers in the presence of amplified or diminished degrees of the presented challenges.

Finally, we offer implementations of several features that can be added to state-of-the-art tracking algorithms DeepSORT and StrongSORT in order to increase performance on datasets presenting similar challenges as GSD and ModMOT20. The provided features include camera motion compensation, immediate track initialization, aspect ratio verification, less strict location matching constraints, and appearance-only matching. We benchmark the original and customized algorithms on GSD and ModMOT20, obtaining a higher performance when making use of the aforementioned improvements. Our methods give important insights about the kinds of features that are suitable for use with complex agricultural datasets, and they could inspire further work on the topic.

2 Related Research

Object Detection

Deep learning object detection techniques have already been used in numerous agriculture-related projects. The tasks in question present unique challenges due to the high variety and changing states of the objects, as well as exposure to different lighting conditions. Particularly, strawberry fruit and flower detection have been used for automated harvest picking with robots [83, 80], yield estimation [39], plant disease detection [51, 32], and distinguishing mature and immature fruit [26]. Similar projects have been implemented to enhance the cultivation techniques of other plants. These include the counting of grape clusters [62], sweet-peppers [74], wheat spikes [79], oranges [86], mangoes, almonds, and apple fruits [4] and flowers [18].

The techniques used in accomplishing these tasks include Faster R-CNN [39, 40, 51, 62, 61, 4] as the most popular one, accompanied by Mask R-CNN [83], YOLO [80, 86], and other Convolutional Neural Network (CNN) architectures [26, 79, 18], as well as classical machine learning algorithms applied to RGBD data [74]. In all of these situations, a high detection accuracy was achieved despite the vast variety of appearances and the changing illumination conditions associated with footage coming from outdoor gardens, fields, and farms.

Faster R-CNN, Mask R-CNN, and YOLO are among the numerous neural network architectures that comprise the state-of-the-art landmark on popular object detection becnhmarks such as Common Objects in Context (COCO) [41], Pascal VOC [29], and CrowdHuman [63]. Other object detection architectures that achieve the overall highest performances on these datasets are Co-DETR [90] for COCO, and InternImage-H [75] on Pascal VOC 2007 and CrowdHuman. However, these are part of a large number of architectures that closely compete to achieve the highest mean Average Precision (mAP) score on these tasks, their performances often being separated by just a few decimal points.

Another relevant aspect to consider when choosing a model architecture is the ease with which it can be integrated within a new project. The YOLO series is very commonly used in industry projects due to both its high performance on object detection tasks, and to its

open source availability. As a result, we opted to make use of the newest release in the YOLO network architecture series, YOLOX [22]. It has a straight-forward training pipeline which is easily adapted to various use cases and datasets.

Multi-Object Tracking (MOT)

Object detection in agricultural settings often precedes the application of tracking algorithms to the obtained predictions [80, 62, 86]. GSD has previously been used to train Faster R-CNN [24] and YOLOX [59] models for strawberry detection [77]. The mAP score achieved using these architectures is 55.7 for YOLOX, and 55.8 for Faster R-CNN. Given the similar performance of these two architectures, it was concluded that the choice of detection model among these would have little influence over the performance of the tracking pipeline. The resulting detections were then used to benchmark the tracking performance of several MOT algorithms: SORT [7], OC-SORT [9], ByteTrack [88], DeepSORT [78], and StrongSORT [15]. SORT was published in 2016, and DeepSORT in 2017, while ByteTrack and OC-SORT both appeared in 2022. StrongSORT [15], which was released at the end of 2022 is a new online tracking algorithm which positions itself near the state-of-the-art on benchmark datasets like MOT17 and MOT20.

Algorithms based solely on location matching, such as SORT, OC-CORT, and ByteTrack, fail to leverage appearance as a way to perform matching despite the non-linearity of object movements in GSD. Our experiments thus focus on DeepSORT [78] and StrongSORT [15], two online tracking algorithms that take both the location and appearance of detections into consideration when matching them together to form a trajectory. Both of these algorithms use a Kalman Filter framework to perform location matching based on velocity vectors, assuming a consistently linear motion of the tracked objects. Appearance information is used to achieve a higher matching performance. DeepSORT was the first tracking algorithm to propose appearance matching, based on features extracted using a deep neural network. To solve the association problem, DeepSORT uses a Mahalanobis gating distance to verify whether an association is admissible. Then, matching is performed solely based on the appearance cost, which is measured using the cosine distance. Meanwhile, StrongSORT is an enhanced version of DeepSORT, which proposes certain improvements, such as abandoning the use of the gating threshold, and combining the motion cost (A_m) and the appearance cost (A_a) in one equation for the overall association cost (C):

$$C = \lambda A_a + (1 - \lambda) A_m. \tag{1}$$

Typically, a much bigger weight is placed on the appearance cost, as the standard value for λ is 0.98. StrongSORT also includes an object representation updating strategy using Exponential Moving Average (EMA), and camera motion compensation with Enhanced Correlation Coefficient (ECC) maximization [17].

Agricultural Applications of Visual Detections and Tracking

There are numerous implementations of other agriculture-related applications of object detection and MOT. For example, a simple solution is used for locating highly occluded citruses [27] by processing images to find clusters of pixels within manually defined valid fruit color ranges. More novel solutions are used for the three-dimensional (3D) tracking of tomatoes

[58], including object detection with Mask R-CNN over two-dimensional (2D) images captured at multiple angles, which are then combined to create a 3D representation of the space. Then, tracking is performed using a customized algorithm. Another relevant application is the prediction of the future paths of defective citrus fruits on a conveyor belt. These are detected using a CNN, and tracked with SORT. In this case, tracking is performed with the aim to then pick out the undesirable fruits using a robot arm [11]. Tracking has also previously been used as part of a citrus classification application which identifies the quality grade of each fruit [84]. In this case, a personalized tracking algorithm was used, created using a Program Logic Controller (PLC).

A number of other projects also use object detection with the purpose of automatically counting fruit in videos. Such applications exist for counting avocados, lemons, and apples using Faster R-CNN, MobileNet [72], and YOLOv4 [21]. A visual counting application has also been implemented for cotton seedlings, involving detection with CenterNet and tracking with DeepSORT [81]. Lastly, strawberries have also been a target for such an application [33], which involves detection with a CNN, and tracking with a DeepSORT inspired algorithm.

Fruit Quality Prediction

Notably, GSD has also been used to create a strawberry quality prediction model in [76]. Several approaches were benchmarked for this application, including deep learning methods such as CNN, Variational Autoencoders (VAE) and Multilayer Perceptron (MLP). Other machine learning techniques were also used, inluding Principal Component Analysis (PCA), Kernelized Ridge Regression (KRR), and Support Vector Regression (SVR). The segmentation images of strawberries at the moment of harvest were used as input, alongside with plant load data and environmental parameters, including temperature, humidity, and lighting. Based on these features, the model would output a prediction of the Brix degree of a strawberry at harvest time, referring to the Total Soluble Solid (TSS) in the fruit. This value is in direct correlation with the sweetness of a strawberry, and usually ranges between 6 and 12 for strawberry fruits. The model achieved a Root Mean Squared Error (RMSE) value of 1.10 for the predicted Brrix degree, and it was stipulated that a better performance may be reached by using segmentation images from the fruit's entire life cycle instead of a single image at the moment of harvest.

Various previous papers have described how deep learning can be used to predict the ripeness of fruit. One example is using classification and regression on images of bananas to predict their ripeness class and expected shelf life with CNN and AlexNet [1]. Another paper has presented a statistical method for predicting the optimal picking date for all apples in an orchard [56]. However, in this case, all fruits in the orchard become ripe during the same period of time. In addition to visual data on fruit size, and color of the ground and background, some data is collected through methods that involve destroying a part of the fruits, by measuring the soluble solids content, acidity, firmness, and stage of starch transition. Other methods use environmental data to predict the optimal harvest date, for example by using a statistical plant growth simulation model for tomatoes [68]. In such methods, special attention is generally given to the temperature factor [38]. Fruit maturity has also previously been modeled as a time series for grapes [5]. In this case, RGB images were used to chromatically study the fruit, particularly in order to determine the color intensity for the green color channel. This value has been found to have a high correlation to the fruits' ripeness. Using an optimizable regressor

that can be implemented with a simple neural network, the ripeness interval of a fruit can be determined based on its color.

Yield Estimation

Predicting yield is a very popular application of AI in agriculture. The machine learning models built for this scenario generally use environmental data, such as climatic parameters [73, 55, 34, 31, 30, 2, 69, 13], soil data [55, 43, 31, 2, 13], plant load and vegetation index observations [16, 50, 35, 34], and data about human agricultural activities [64, 52, 65]. Some of these applications use visual data, such as Normalised Difference Vegetation Index (NDVI) values derived from a Moderate Resolution Imaging Spectroradiometer (MODIS) sensor data [50] or from satellite images [35]. However, none of these yield forecasting models used images captured over time with regular photographic cameras.

Robustness Through Data Augmentation

Agricultural applications are in dire need of potent tracking algorithms, that will enhance the usability of the aforementioned applications. Currently, however, standard evaluation datasets for MOT algorithms are carefully curated, and only contain short, high-quality videos, with high frame rates, and homogeneous lighting conditions. In addition, the most popular MOT datasets have been designed to encourage advances in the field of autonomous driving. As such, the objects tracked in these datasets are usually humans, as in the MOT Challenge, [36], DanceTrack [67], SportsMOT [12], HiEve [42], and SeaDronesSee [71], or vehicles, as in KITTI [23], BDD100K [82], and Synthehicle [28]. Increasing robustness to the agricultural industry's challenges is an important goal in the development of AI techniques, including the field of computer vision. This has previously been done for tasks such as image classification and object detection. Several augmentation techniques have been employed to obtain more varied training data for neural networks, leading to an overall improved performance [14, 48]. Such methods include simple alterations like flipping, rotating, shifting, and cropping the image, as well as more advanced ones like adding noise, PCA jittering General Adversarial Networks (GANs), and Wasserstein GANs (WGANs).

Certain tracking algorithms have also been developed with specific dataset challenges in mind, which has required evaluation data to contain a high amount of these specific challenges. This was generally achieved through curating a data collection to contain as much of the studied characteristic as possible. Such difficulties include occlusion occurrence [85, 53], varying light conditions [57, 89], and busy backgrounds [10, 47]. One study has observed the importance of gradual and controllable data augmentation, by studying how robust tracking algorithms are when faced with incremental noisiness added to the evaluation data [20, 53]. No existing MOT benchmark datasets present the same challenges as GSD in terms of non-linear motion and evolving object appearances. However, carefully modifying a popular benchmark is a promising strategy for obtaining such a dataset.

3 Fundamentals

3.1 Neural Networks

The use of Neural Networks (NNs) is a fundamental building block for the methods used in this paper. Specifically, **CNNs** are a particularly popular type of NN for processing images, which are arrays of pixel color values [37]. For example, RGB images are composed of three two-dimensional arrays, one for each individual color channel: red, green, and blue. CNNs contain a series of convolutional and pooling layers. Convolutional layers are composed of multiple feature maps, in which each unit is associated to a patch of units in the previous layer through a weighted sum. An activation function is then applied to the result of this weighted sum. Thus, convolutional layers are extremely good at detecting the occurence of motifs in the previous layer, which can then be used in the subsequent layer to detect a motif at a larger scale, formed by the co-occurrence of a number of motifs in the current layer. This is how a CNN can go from locating simple features in an image, like straight lines and corners, to classifying, for example, an image of a dog or a cat.

Meanwhile, pooling layers typically replace the values in a feature map with the local maximum values of a small neighbourhood of units, making the network more robust to small variations in the identified motifs. Training a CNN is done by tweaking the weights associated to the units in its feature map, within any convolutional layer. This is performed through supervised learning, by using a series of labeled images as input for the network. The difference between the model's output and the image's label is treated as an error. Specifically, if the model predicts that a dog-labelled image is a dog image with a confidence of 0.6, the error is equal to 0.4. This classification error is used to modify the weights in the network in a process called backpropagation.

The CNN architecture is a great solution to image classification problems, and it has been adapted to detection tasks by using a sliding window [19] or region proposal methods [25]. However, this approach results in pipelines that are difficult to train, as each component has a separate training process. By far the fastest architecture in terms of training and inference is **YOLO** [59]. As its name suggests, running inference on an image is done after processing the image only one time, as opposed to other existing models. This is achieved by treating the task as a regression problem, receiving the image pixels as input for a CNN, and returning the bbox locations. Since the introduction of YOLO, there has been a series of improvements added to the original network. In this paper, we are using the latest version, YOLOX [22].

Convolutional layers are also used to build **Convolutional Auto-Encoder (CAE)** models [49, 87]. Auto-Encoders (AEs) are unsupervised models whose input is almost identical to their output. Their goal is to preserve the meaning of their input, but improve certain characteristics of it, such as lowering its storage requirements, and removing noise and redundant content. AEs have two main components: the encoder, and the decoder. In regular AEs, these components are formed using fully-connected layers, organized in a feed-forward network. However, these layers can be swapped for convolutional layers to create CAE models, which are better at using images as their input and output. These networks have input and output layers with the same size, while the middle layers, which compose the bottleneck of the model, are much smaller, as shown in Figure 1. The outputs of the bottleneck layers of the CAE are

latent featuress, which are compressed encodings of the original images, without losing any of the important information stored in them. Typically, a CAE arechitecture is used either for image compression, or for generating de-noised images based on original ones. CAEs can also be used to extract the latent features of detected objects in image datasets. In this paper, we use feature encodings generated in this way for GSD.

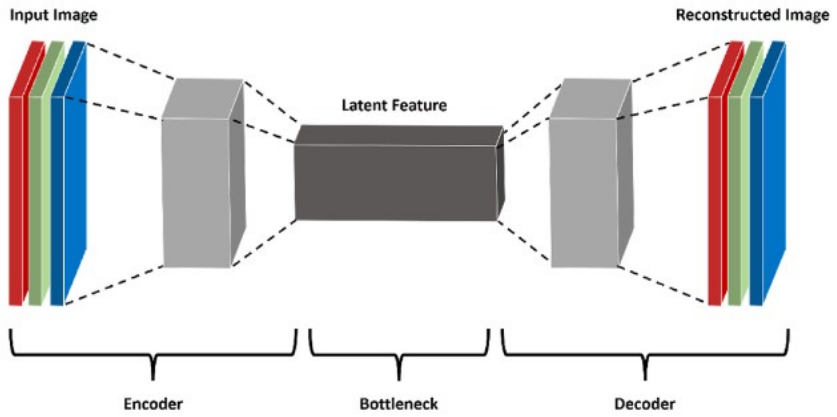


Figure 1: Representation of a typical CAE architecture. This image is from [54].

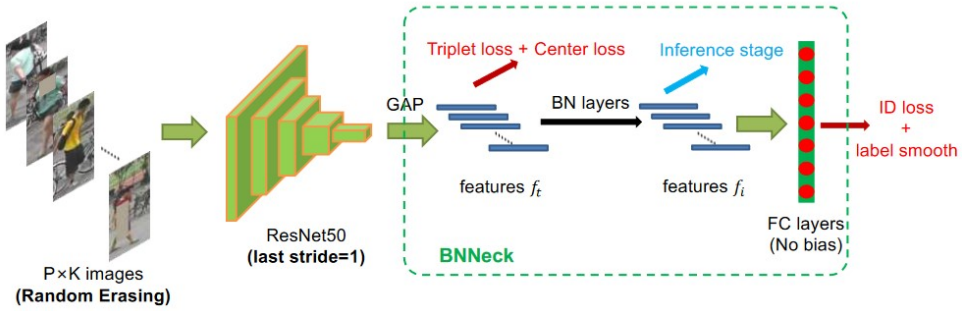


Figure 2: Representation of the BoT feature extractor architecture and pipeline. This image is from [45].

Another popular strategy for feature extraction is the use of a CNN specifically trained for this task. One of the best performing models that does this is **Bag of Tricks (BoT)**, which uses a 50-layer CNN named ResNet50 trained according to several best practices for this task [45]. These practices refer to various training aspects, such as the warm-up learning rate, augmentation through random image region erasure, label smoothing, reducing the size of the last stride, and adding a Batch Normalization (BN) layer after the original final layer of ResNet50, using a structure called BNNeck. The resulting pipeline is shown in Figure 2, and the BoT architecture is used in Section 5.1.

3.2 Evaluation Metrics

For the purpose of evaluating the performance of the tracking algorithms used on our data, we chose several established metrics which are also used in the MOT Challenge. As MOT is a complex task, these metrics focus on different combinations of aspects related to MOT performance. Firstly, the detection performance refers to finding the objects of interest in each frame. Meanwhile, localization performance shows how well the location of the objects is reported in each frame. Lastly, the association performance describes how accurately objects from different frames are found to belong to the same or different tracks.

In recent years, **High Order Tracking Accuracy (HOTA)** has been preferred as a tracking metric, as it encompasses detection, association, and localization performance in a single formula [44]. This metric has multiple components, and starts with finding a one-to-one mapping between detections and ground truth annotations by using the Hungarian algorithm based on IoU values calculated between each detection-annotation pair. IoU is the ratio between spatial overlap for two bboxes (intersection) and the total space they collectively occupy (union). Different $HOTA_\alpha$ values can be calculated based on the IoU matching threshold, α, used to generate this mapping.

Based on the one-to-one mapping obtained, correctly mapped objects that have an IoU higher than α, are called True Positive (TP_α). Meanwhile, extra detections are considered False Positive (FP_α), and missing ones are considered False Negative (FN_α). Thus, the **Detection Accuracy (DetAα)** is calculated within a HOTA based ratio of accurate detections in the following way:

$$DetA_\alpha = \frac{|TP_\alpha|}{|TP_\alpha| + |FP_\alpha| + |FN_\alpha|} \tag{2}$$

The **Association Accuracy (AssAα)** is the average accuracy of trajectory matching over all detections, when using an IoU threshold α. To calculate it, the set of True Positive Associations (TPA_α) for a given TP_α instance, c, are described as the set of all pairs in TP_α that have the same ground truth ID (grID) and predicted ID (prID) as c. Similarly, the set of False Negative Associations (FNA_α) for c is formed of all elements in TP_α with the same grID as c, but a different prID. Lastly, the set of False Positive Associations (FPA_α) for c is comprised of the TP_α pairs with the same prID as c, but which have a different grID.

$$AssA_\alpha = \frac{1}{|TP_\alpha|} \Sigma_{c \in TP_\alpha} \frac{|TPA_\alpha(c)|}{|TPA_\alpha(c)| + |FPA_\alpha(c)| + |FNA_\alpha(c)|} \tag{3}$$

The final component of $HOTA_\alpha$, the **Location Accuracy (LocAα)**, is calculated as the average IoU of all TPs in the dataset:

$$LocA_\alpha = \frac{1}{|TP_\alpha|} \Sigma_{c \in TP} IoU(c) \tag{4}$$

Detection and association evaluation are then combined to measure $HOTA_\alpha$ in the following way:

$$HOTA_\alpha = \sqrt{DetA_\alpha \cdot AssA_\alpha} \tag{5}$$

Finally, a comprehensive value for $HOTA$ is calculated over 19 different α values within (0, 1) with the formula below. Although location is not explicitly included in the $HOTA_\alpha$ formula, location accuracy is automatically included in the value of $HOTA$ as it stands as the base for the one-to-one mappings obtained each α value.

$$HOTA = \int_{0<\alpha\leq 1} HOTA_\alpha \approx \Sigma_{\alpha=0.05,\alpha+=0.05}^{0.95} HOTA_\alpha \tag{6}$$

Another popular MOT metric is **MOT Accuracy (MOTA)**, which emphasizes detection rather than association performance, and does not account for localization at all [6]. Just like $HOTA$, $MOTA$ relies on a one-to-one mapping between ground truth and predicted bboxes, obtained by using IoU scores and the Hungarian algorithm. Then, the frequency of three types of errors is evaluated: False Positive (FP), False Negative (FN), and **Identity Switch (IDSW)**. An $IDSW$ refers to an object's track identity mistakenly being swapped with the identity of a different object, or to a track being considered lost, and then being reinitialized with a different track ID. Given a set of ground truth Detections ($gtDet$), the formula of $MOTA$ is:

$$MOTA = 1 - \frac{|FN| + |FP| + |IDSW|}{|gtDet|} \tag{7}$$

Finally, the **Identification (IDF1)** metric focuses on measuring association rather than detection performance [60], in contrast with $MOTA$. Based on a one-to-one mapping between ground truth Trajectories ($gtTraj$), and predicted Trajectories ($predTraj$), once again based on the Hungarian algorithm. The correctly identified objects are called Identity TP ($IDTP$). Similarly, the occurrences of Identity FP ($IDFP$) and Identity FN ($IDFN$) are counted as well. This gives the following formula for calculating $IDF1$:

$$IDF1 = \frac{|IDTP|}{|IDTP| + 0.5|IDFP| + 0.5|IDFN|} \tag{8}$$

4 Data

4.1 The Growing Strawberries Dataset

Overview

The industry dataset studied in this paper, GSD, contains image sequences of strawberry plants in a greenhouse. Data was collected with the use of six fixed cameras. Three of these cameras capture images in a Red-Green-Blue color scheme (RGB), while the other three cameras do so in Orange-Cyan-Near-infrared (OCN). The experiments reported in this study focus on the RGB camera image sequences: RGBCAM1, RGBCAM3, and RGBCAM5. The plants grow under natural light conditions, with alternations between day and night. As a result, the images captured have cyclically varying light conditions, which can be observed in Figure 4. The strawberry fruits are in constant but slow change due to their growth, from their first appearances until they are ripe and ready for picking. This can take between 3 and 7 days. Due to the slow growth process of the fruit, images are captured with a frame rate of one per

hour. It is not useful to capture videos at the higher frame rates commonly used by popular multi-object tracking benchmarks. For example, the MOT Benchmark contains camera sequences with frame rates ranging between 14 and 30 frames per second [36]. These values are useful, for example, when tracking pedestrians that move in and out of view within a matter of seconds. However, in the case of mostly static objects, this would result in an unnecessary amount of data being collected.

The goal of this dataset is to be used for harvest estimation by automating fruit growth observation. The first step in achieving this goal is performing multi-object tracking on the strawberry fruits. In this way, the appearance of each fruit can be monitored over time in order to inform a prediction of the time when the fruit will be ripe. The images are accompanied by COCO annotation files. These facilitate the use of GSD for training detection models, and for evaluating detection and tracking performance.

(a) (b)

Figure 3: Consecutive frames from GSD, which present lighting changes and camera motion: a) RGBCAM1, image ID 818; b) RGBCAM1, image ID 819.

However, the task of tracking on GSD presents many challenging characteristics, as one can expect from an industry dataset. Firstly, **strawberries often do not look like previous images of themselves**. The appearance changes caused by the growth of the fruits make matching and tracking based on appearance very challenging. In addition, the changes in light conditions cause the strawberries to appear different from one frame to the next, amplifying the challenge of appearance matching even further. **Strawberries also look like each other**. The variety in the appearance of strawberry fruits is much more limited than, for example, that of human pedestrians. Additionally, although the low frame rate is reasonable for this use case, it also causes further challenges. For example, agricultural activities are performed by humans in the greenhouse, which sometimes involve moving the fruits. This causes sudden **non-linear movements** to occur within a trajectory between consecutive frames. The setup of the cameras in the greenhouse also makes these susceptible to regularly occurring camera movements on the vertical axis. These vertical motions displace the camera's field of view by varying amounts of pixels. This phenomenon is another cause of non-linear shifts of the annotated strawberries within series of consecutive frames. Finally, a lot of the frames do not contain clear images of the strawberries, due to being captured during nighttime, when visibility is very low. There are also frames captured while cameras are covered to be protected during pesticide spraying, resulting in completely occluded images.

Some of the presented challenges can be bypassed with simple preprocessing steps. For example, night-time and completely occluded images can be filtered out, as they do not contain valuable annotations of strawberries. Meanwhile, to tackle more difficult characteristics, such as non-linear fruit shifts, the solution often lies in improving the robustness of the tracking algorithms themselves.

Preprocessing Steps

Nighttime images from GSD were not used in the experiments described in this paper, as they would hinder the performance of detection models on the dataset, due to containing barely visible strawberries. Frames captured during the night were filtered out by setting a minimum luminance threshold of 50, and only using images with an average pixel luminance higher than that. As nighttime images only occured on average twice a day, we decided that this method of filtering them out is favorable to performing image normalization. The luminosity of each pixel, Y, was calculated based on the RGB values of that pixel according to the ITU-R BT.601 standard [8], using the formula:

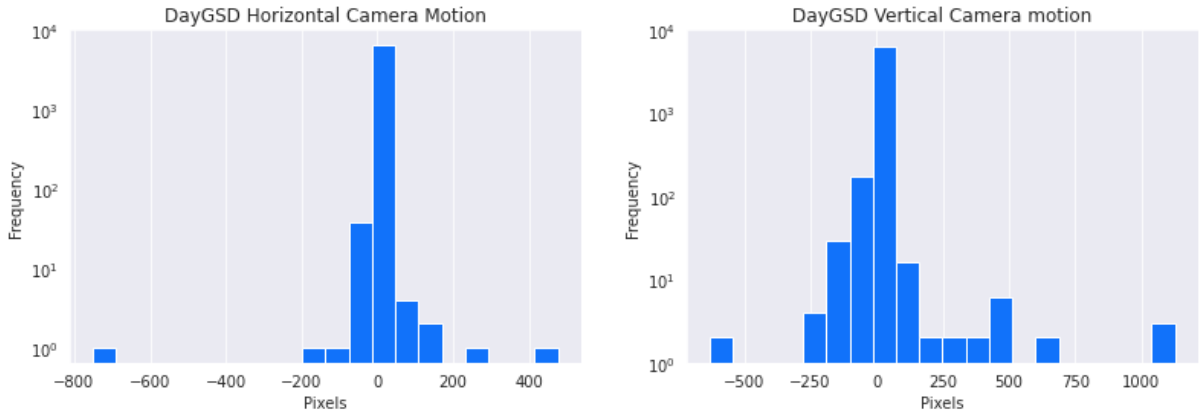
$$Y = 0.299R + 0.587G + 0.114B. \tag{9}$$

Furthermore, the frames where the field of view is completely or partially **occluded** due to human activities were also filtered out. Such images were identified by calculating the standard deviation of the image RGB array. A standard deviation less than 40 was interpreted as a signal that the image has low variance in colors, and is most likely occluded. The process of filtering out night time and occluded images was also described in [77]. The remaining frames shall hence be referred to as DayGSD.

Further Analysis

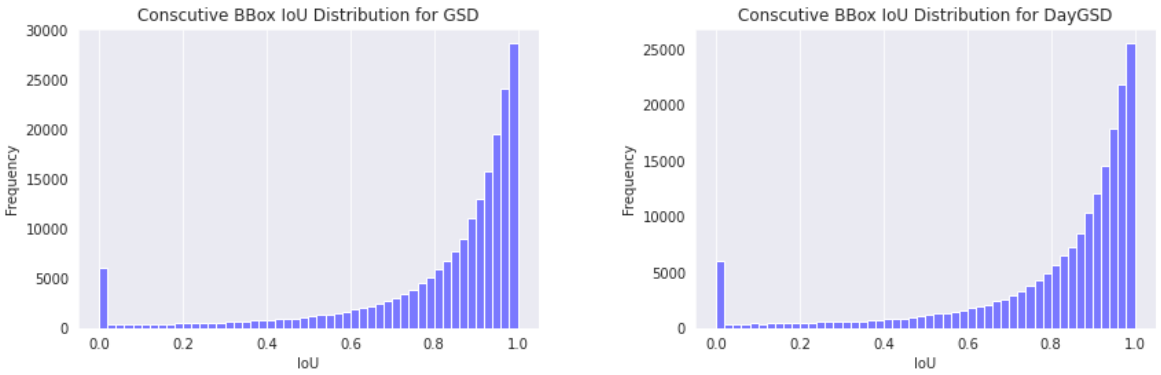
To quantitatively determine what makes GSD a highly complex tracking dataset, certain characteristics were examined in more detail. Particularly, we measured and stored the camera motion occurring between every consecutive pair of images. We used Local Feature Matching with Transformers (LoFTR) [66] to find pixel-wise matches between each pair of consecutive frames. The difference between the horizontal and vertical coordinates of the pairs of pixels matched between two consecutive frames was averaged to compute the camera motion occurring between one frame and the next. The resulting estimated camera motions were compiled into Figure 4 for DayGSD. It appears that vertical camera displacements are much more prevalent than horizontal ones. This is a known issue, arising from the setup of the cameras in the greenhouse. As they are placed on a horizontal cylindrical bar, they can easily rotate forward due to gravity. They often require repositioning by the employees of the facility. Camera motion, together with non-linear movements of some annotated strawberries, leads to difficulties when performing location-based matching, as the IoU is often even 0 for consecutive-frame annotations belonging to the same track. This is shown in Figures 5a and 5b for GSD, and DayGSD, respectively. This is an important observation, as most tracking algorithms use IoU distance as a performance measurement for location-based matching.

In addition to the location matching difficulties imposed by GSD, the appearance-based matching is also made more complicated by the strawberries evolving over time. Figure 6 shows the



(a) (b)

Figure 4: Histograms of the horizontal (a) and vertical (b) camera motion values registered for frames in DayGSD.



(a) (b)

Figure 5: Histograms of the IoU values between consecutive locations of bboxes belonging to the same strawberry for GSD (a), and DayGSD (b).

Figure 6: Average RGB values of the pixels inside bboxes according to their order inside a track. Representation includes bboxes up to position 300.

evolution of the color composition of bounding boxes belonging to the same strawberry over time. The average track length in GSD is 155, and there are few tracks with a length greater than 300, which is the reason why this limit was chosen for the plot's horizontal axis. It is evident that, as strawberries mature, their color becomes less green and more red. Namely, the average R value of the pixels in a bbox grows from 125 to 150 over 300 frames. During that time, the B value also increases from 50 to 75, and the G value decreases from 125 to 75.

4.2 Modification of MOT20

Original MOT Dataset

The MOT Benchmark is a collection of video sequences that is widely used to evaluate the performance of tracking algorithms. Several versions of this dataset have been released in 2015, 2016, 2017, and 2020. All releases contain a train set and a test set. Each of them contains several surveillance-style videos recorded in areas with heavy pedestrian traffic, such as street corners, markets, and train stations. The datasets present multiple difficulties for the tasks of detection and tracking, including occlusions of pedestrians and distance to camera. However, they do not present some of the challenges that often occur in less carefully curated datasets, such as the datasets used for agricultural industry applications, like GSD.

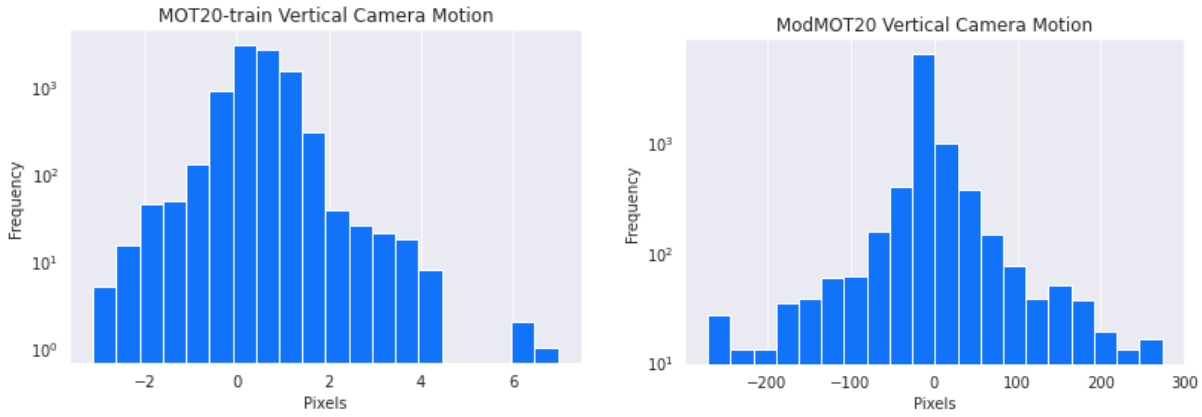
MOT Modification Steps

To reproduce some of the challenges present in GSD, we have modified the newest release of the MOT Benchmark, MOT20, to include videos with camera motion and chromatic evolution. The purpose of these modified versions of MOT20 is to benchmark different tracking algorithms and observe their performance when faced with non-linear pedestrian movements and changing appearances. The Modified MOT20 (ModMOT20) dataset was created based on the train set of the original MOT20. This might seem unusual, as we intend to use it for evaluating trackers, just as the test set is typically used. However, we need to be able to generate ground truth files for the modified dataset. As the ground truth annotations are not public for the MOT20-test set, we had to resort to using the original train set as a basis for our transformed dataset.

The first modification performed was the **addition of camera motion**. Since GSD presents camera motion mostly on the vertical axis, only vertical movements were artificially added to MOT20. To achieve this, the images of MOT20-train were cropped down to 60% of their original height. The camera motion value added to each cropped frame with global image ID i, $y^i_{ModMOT20}$, was calculated based on the value obtained during the camera motion analysis of GSD, y^i_{GSD}, following the formula:

$$y^i_{ModMOT20} = y^i_{GSD} \cdot \frac{h^i_{ModMOT20}}{h^i_{GSD}} \tag{10}$$

This formula includes scaling the added camera motion value based on the height of the images in ModMOT20, $h^i_{ModMOT20}$, and in GSD, h^i_{GSD}. The distributions of camera motions on the vertical axes before and after this modification are shown in Figure 7. These graphs show how the camera motion initially falls within a very restricted range, between -6 and 8 pixels. After



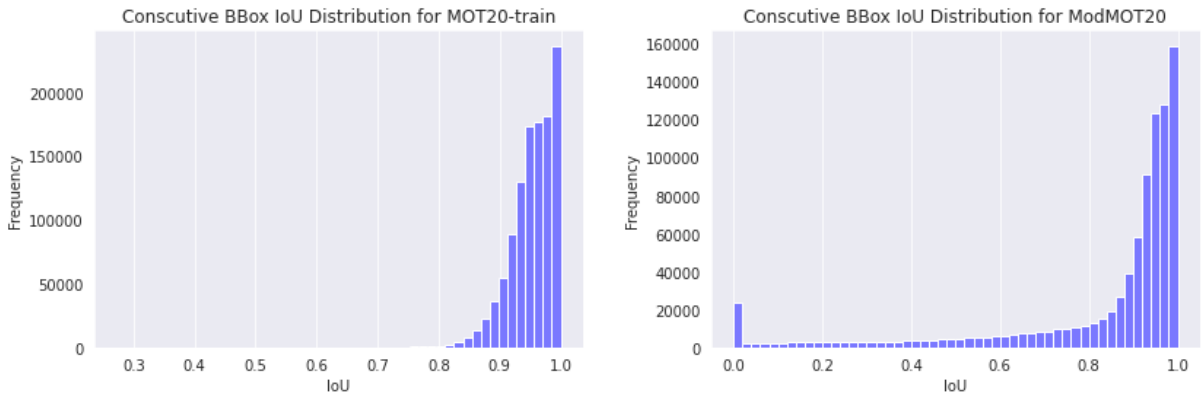
(a) (b)

Figure 7: Histograms of the vertical camera motion values registered for frames in the original MOT20-train (a), and the new ModMOT20 (b).

the modification, there are numerous instances of images affected by a high degree of camera movement, with values varying between -300 and 300 pixels. This modification has caused the distribution of IoU values between consecutive bbox locations in ModMOT20 to become much more similar to that in GSD and its filtered version. Figure 8a shows the distribution before adding camera motion, in which all such IoU values are above 0.8, and most of them are equal to 1.0. After this modification step, there appear to be lower values as well, with a considerable proportion of them being equal to 0.0, as depicted in Figure 8b.

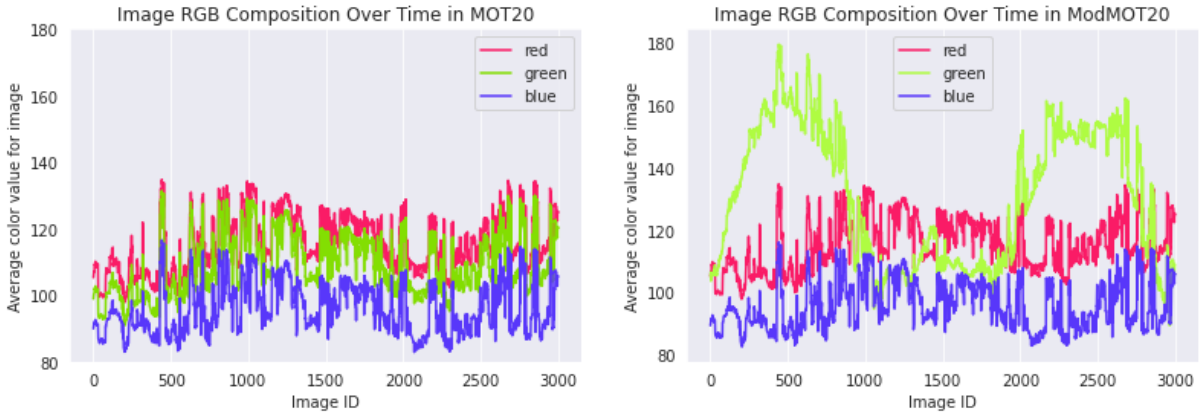
We have also generated a number of different modified datasets with a diminished frame rate, by sampling every 5^{th}, 10^{th}, 20^{th} or 50^{th} frame from the original dataset, and adding camera motion to the resulting subset of frames. The IoU between consecutive bboxes in these modified variants can be seen in the plots included in Appendix A. We can observe that the degree to which the frame rate is lowered is directly linked to a bigger shift of the distribution towards lower IoU values. Meanwhile, adding camera motion produces a new peak of the distribution around the IoU = 0.0 value. These insights could be helpful in finding the right modification parameters to reproduce the IoU distributions of other high complexity industry and agricultural datasets in the future.

The second modification step involves inducing **progressive color changes** into the MOT20-train dataset, which makes it more similar to the case of growing strawberries. We decided to alter the green channel of the images over time, by setting the green channel equal to the red one, but then adding values from a sinusoidal distribution to it. As in GSD, the progression of the red and green channels starts with the two having the same value, but after 300 frames, the values differ by roughly 75, on a scale from 0 to 255. As such the sinusoidal distribution used for changing MOT20 has an amplitude of 75, and a step of 0.0033 units, resulting in the same green and red channel value difference being achieved in the same number of frames as in GSD. These alterations were made at an image level, over the entire dataset. Figure 9 shows the evolution of the average color levels across the first 3000 frames of MOT20, and ModMOT20. The values of the other color channels, red and blue, were kept as in the original images. Some examples of the color and camera motion variety present in ModMOT20 are shown in Figure 10.



(a) (b)

Figure 8: Histograms of the IoU values between locations of consecutive bbox locations for objects in the original MOT20-train (a), and the new ModMOT20 (b).



(a) (b)

Figure 9: Image color composition analysis for the first 3000 frames of the original MOT20-train (a), and the new ModMOT20 (b).

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

(e) (f) (g) (h)

Figure 10: Example of frames in the original MOT20-train: (a) MOT20-01, image ID 1, (b) MOT20-01, image ID 150, (c) MOT20-01, image ID 300, and (d) MOT20-02, image ID 21, and their new ModMOT20 versions: (e)-(h), respectively.

5 Methods

5.1 Model Architectures

The object detection models used for GSD were taken from [77], where they were trained and evaluated on GSD. Training new object detection models for GSD is beyond the scope of this paper, which focuses on MOT methods and datasets. However, we are including a detailed overview of the training procedure employed. Three YOLOX-x [22] detection models pre-trained on the COCO dataset [41] were finetuned on GSD. Cross-validation was used by leaving a different camera sequence out during the training of each model. During preprocessing, the images were rescaled to a size of 2133×1600 and afterwards padded to 2174×1600. The model was trained for 100 epochs, with a batch size of 4, a cosine annealing learning rate of $3.125e^{-6}$, and a weight decay of $5e^{-4}$. Several data augmentation processes were involved in the training, including horizontal and vertical flipping with a probability of 50%, rotation between 0 and 90 degrees, brightness amplification with a value between 0.92 and 1.12, and random contrast amplification with a value between 0.92 and 1.12. The detection files further used in the tracking experiments were generated for each camera sequence using the specific YOLOX-x model that had not been trained on that set of images.

As described in Section 3.1, the feature extractor used for GSD was created by taking the encoder part of a CAE model trained on all strawberry segmentations acquired before August 2021. The model has three convolutional layers, interleaved with 2 maxpooling layers of sizes (5, 5), and (4, 4). The dimension of both the input and output layers is 200x200x3. The strawberry instances used to train the feature extractor for GSD are also present in the test sets used for evaluating tracking performance on GSD. However, their effect on tracking performance is negligible, as the CAE model is not trained to make representations more similar for segmentations belonging to the same track.

For the object detector for MOT20, we used the same object detection model as ByteTrack [88], which was also used in the StrongSORT paper to obtain the reported results [15]. Also here, a YOLOX-x model pretrained on the COCO dataset was used. However, this time it was fine-tuned on MOT20-train and CrowdHuman. The object detection files generated with this model on MOT20-train were then processed to obtain detections corresponding to ModMOT20. This operation was performed by clipping the bounding boxes from MOT20 to the new cropped images, and also accounting for the added camera motion.

For both MOT20 and ModMOT20, we extracted features with the same model as proposed by StrongSORT [15]. It is a Bag of Tricks (BoT) [46] model pretrained on the DukeMTMC dataset [60], and finetuned on MOT20-train and CrowdHuman. More details on this technique are given in Section 3.1. As the images in the ModMOT20 dataset are modified versions the MOT20 training set, a slighlty different version of the ModMOT20 objects is captured in the feature extractor model. However, for the same reason as in the case of the GSD feature extractor, this has an insignificant effect on the tracking performance.

5.2 MOT Algorithm Improvements

Two popular tracking algorithms were benchmarked on DayGSD, MOT20 and ModMOT20: DeepSORT, and StrongSORT. After observing the particularities of these two datasets, some additional features were incorporated into DeepSORT and StrongSORT to improve their performance under these conditions. This resulted in customized versions of these algorithms, which we refer to as Upgraded DeepSORT, and Upgraded StrongSORT, respectively. This subsection explains how to replicate this customization to facilitate tracking on any new datasets with similar characteristics as GSD.

Gating Threshold Amplification

Firstly, in the case of DeepSORT, gating threshold amplification was performed using a new parameter, called *alpha* (α). The goal of this parameter is to allow location matching to consider candidates with a Mahalanobis distance higher than the original value, $t^{(1)} = 9.4877$, from the distribution of objects already associated with the track. This threshold value corresponds with a 95% confidence interval based on the inverse chi-squared (χ^2) distribution. Originally, DeepSORT assumes a high degree of linearity in the movements of the tracked objects. However, with the non-linear movements present in GSD and ModMOT20, it is beneficial to relax this requirement. Implementing this feature for DeepSORT is done by simply multiplying the gating threshold with a value α. The condition for excluding a candidate bbox j from being associated with the distribution of a track i was originally:

$$d^{(1)}(i,j) \geq t^{(1)}, \tag{11}$$

where $d^{(1)}(i,j)$ is the Mahalanobis distance between i and j. Amplification with the α parameter will result in a new exclusion condition:

$$d^{(1)}(i,j) \geq \alpha \cdot t^{(1)}. \tag{12}$$

The optimal value for α can be found through a hyperparameter optimization search for any specific dataset. For GSD, this value was found to be 50, as described in Section 5.3.

Camera Movement Compensation

The DeepSORT code can also be altered to account for the camera motion occurring in a given dataset. The vertical camera motion values can be measured with LoFTR. This technique is used to identify pixel-level correspondences between consecutive frames in the dataset, and calculate the horizontal and vertical difference between the pixels' locations within those images. The output of LoFTR is comprised of two lists: one of horizontal distances, $dist_i^x(p,q)$, and one of vertical distances, $dist_i^y(p,q)$, between the corresponding pixel pairs, (p, q), identified for each pair of consecutive images, with IDs $i-1$ and i. The complete set of corresponding pixel pairs between images $i-1$ and i is denoted by P_i. To compute the horizontal ($motion_i^x$) and vertical ($motion_i^y$) camera motion values for each Image i, average horizontal and vertical distances are calculated across the set of corresponding pixels in the following way:

$$motion_i^x = \frac{\Sigma_{(p,q)\in P_i} dist_i^x(p,q)}{|P_i|} \tag{13}$$

$$motion_i^y = \frac{\Sigma_{(p,q)\in P_i} dist_i^y(p,q)}{|P_i|} \tag{14}$$

These values can then be subtracted from the bounding box coordinates, (x_{min}, y_{min}, $width$, $height$), of every detection within each image in the dataset. The following formulas exemplify this update rule for the bounding boxes corresponding to the detections in Image i:

$$x_{min} = x_{min} - motion_i^x \tag{15}$$

$$y_{min} = y_{min} - motion_i^y \tag{16}$$

This leads to a better overlap between consecutive bounding boxes belonging to the same track, despite apparently non-linear movements caused by the camera motion. Given that GSD is affected by mostly vertical camera movements, we only use vertical camera motion compensation in our implementation. This also works well for ModMOT20, where only vertical camera shifts were artificially added to the original MOT20 frames. However, choosing which kind of camera movements to compensate is dependent on the specific characteristics of any MOT dataset.

Immediate Track Initialization

Another addition made is immediate track initialization, which allows a track to be instantiated based on a single detection. This does not happen with the original code of DeepSORT, where even setting the n_init parameter to 1 causes tracks to be considered confirmed only on their second detection. In the original code, track states can have one of three different states: tentative, confirmed, or deleted. Tracks are always initialized with the tentative state, and only acquire the confirmed state when the number of detections associated to them is larger or equal to n_init. However, this condition is never checked when the track is first initialized, and only when additional detections are acquired for it. As only detections associated to confirmed tracks are included in the tracking algorithm's output files, this leads to the occurrence of false negatives. We changed the code to initialize every track in the confirmed state from the very begining.

Aspect Ratio (AR)

Finally, an AR condition was introduced as a method to identify and discard erroneous object detections, in order not to use them as input for the matching procedure. Previously, detections were only checked against a minimum detection height, and disregarded, if they had a lower value than that. We changed the code so that the ratio between the height and the width of each bounding box must fall within a certain range to be considered as legitimate. The following formula shows how to calculate the AR of a bounding box, $BBox_{AR}$:

$$BBox_{AR} = \frac{BBox_{height}}{BBox_{width}} \tag{17}$$

The truth value of the following Boolean expression is then evaluated to decide whether the bounding box's shape corresponds with a legitimate detection:

$$min_{AR} \leq BBox_{AR} \leq max_{AR} \tag{18}$$

The minimum (min_{AR}) and maximum (min_{AR}) AR threshold values depend on the general shapes of the tracked objects. They can be found through statistical analysis of the detection bbox shapes or through hyperparameter optimization on any given dataset. For strawberry

tracking on GSD, the optimal values found are $min_{AR} = 0.6$, and $max_{AR} = 1.6$, as described in Section 5.3.

Immediate track initialization and AR conditions were also added as improvement measures for StrongSORT. However, the α parameter was not used as an upgrade for StrongSORT, as the original algorithm does not use a gating threshold during matching. StrongSORT also has its own camera motion compensation module, which uses ECC. Thus, it was not necessary to perform further compensation based on the values obtained using LoFTR.

Appearance-Only Matching

However, the association method of StrongSORT was altered to perform matching solely based on appearance. In this way, we expect to bypass all the issues caused by non-linear movements existing in the dataset. The original matching cost C evaluation formula was expressed based on the appearance cost, A_a, and the motion cost, A_m, as shown in Equation 1. The λ parameter quantifies the emphasis placed on appearance matching. Its original value is set to 0.98 in StrongSORT, showing that the algorithm already relies strongly on apperance matching over location matching. To achieve appearance-only matching, this formula was changed to:

$$C = A_a \tag{19}$$

This corresponds to changing the value of the *lambda* (λ) parameter from the algorithm's default of 0.98 to 1.

5.3 Hyperparameter Values

Choosing hyperparameter values for the original and upgraded DeepSORT and StrongSORT algorithms can be done using any hyperparameter optimization technique. A grid search was conducted to determine the best hyperparameter values for our new version of DeepSORT on DayGSD. The resulting values are shown in Table 3. Most of these hyperparameters are common to the original DeepSORT and StrongSORT algorithms, as well as to the improved version of StrongSORT. Additionally, as previously demonstrated, the GSD and ModMOT20 datasets have very similar properties. Because of this, and hyperparameter optimization processes being very time consuming, we opted to reuse most of these values when running the remaining algorithms on DayGSD and ModMOT20. The hyperparameter values used for applying the other algorithms are listed in Tables 1, 2, and 4.

However, to accommodate for object specific characteristics, some of the parameters have been tuned for the respective dataset. For example, the minimum and maximum threshold values used for AR were optimized specifically for strawberry tracking, and found to be 0.6 and 1.6, respectively. Meanwhile, ModMOT20 is a pedestrian tracking dataset. Because the expected aspect ratio of a human differs from that of a strawberry, using different values was necessary. To decide on an appropriate values, we looked to the original StrongSORT and DeepSORT state-of-the-art performances on MOT20-test, which are achieved with the comparable parameter $min\_detection\_height$ set to 0. This value essentially signifies that the threshold is not used, because bounding boxes always have a height larger than 0. Thus, we used $min\_AR = 0$, and $max\_AR = 100$ for ModMOT20, all bounding boxes in the dataset fall within these thresholds.

Another case of differing parameter values is that of nn_budget. This parameter corresponds to the number of feature vectors retained in a feature bank for each object, in order to be used when calculating the cosine distance of a new detection to any the existing tracks. For DeepSORT and its customized version, we used the value 10, found through the hyperparameter search. This means that the 10 most recent feature vectors belonging to each trajectory are available always available for use in the matching procedure. However, instead of the feature bank used for DeepSORT, StrongSORT saves each object representation as a single vector of Exponential Moving Average (EMA) values over all previous feature encodings acquired for that object. Due to the use of EMA, an nn_budget value of only 1 is necessary when applying StrongSORT and the upgraded StrongSORT algorithm. Additionally, as previously explained, the value of $lambda$ differs between StrongSORT and upgraded StrongSORT in order to perform the association of the tracked objects only based on appearance.

Tables 1 and 2 also show the hyperparameters used to obtain close to state-of-the-art results on MOT20 in the case of DeepSORT, and StrongSORT. These values will be used to reproduce and validate these performances on the MOT20-test set.

Table 1: Hyperparameter values used for running DeepSORT code on GSD, MOT20 and ModMOT20.

DeepSORT Hyperparameters			
Hyperparameter	MOT20	ModMOT20	DayGSD
min_confidence	0.6	0.1	0.1
nms_max_overlap	1.0	0.7	0.7
min_detection_height	0	0	0
max_cosine_distance	0.3	0.1	0.1
nn_budget	100	10	10
max_iou_distance	0.7	0.3	0.3
max_age	30	3	3
n_init	3	0	0

Table 2: Hyperparameter values used for running StrongSORT code on GSD, MOT20 and ModMOT20.

StrongSORT Hyperparameters			
Hyperparameter	MOT20	ModMOT20	DayGSD
min_confidence	0.6	0.1	0.1
nms_max_overlap	1.0	0.7	0.7
min_detection_height	0	0	0
max_cosine_distance	0.45	0.1	0.1
nn_budget	1	1	1
max_iou_distance	0.7	0.3	0.3
max_age	30	3	3
n_init	3	0	0
lambda	0.98	0.98	0.98

Table 3: Hyperparameter values used for running altered DeepSORT code on GSD and ModMOT20.

Upgraded DeepSORT Hyperparameters			
Hyperparameter	MOT20	ModMOT20	DayGSD
min_confidence	0.6	0.1	0.1
nms_max_overlap	1.0	0.7	0.7
min_AR **(new)**	0	0	0.6
max_AR **(new)**	100	100	1.6
max_cosine_distance	0.3	0.1	0.1
nn_budget	100	10	10
max_iou_distance	0.7	0.3	0.3
max_age	30	3	3
n_init	0	0	0
alpha **(new)**	50	50	50

Table 4: Hyperparameter values used for running altered StrongSORT code on GSD and ModMOT20.

Upgraded StrongSORT Hyperparameters			
Hyperparameter	MOT20	ModMOT20	DayGSD
min_confidence	0.6	0.1	0.1
nms_max_overlap	1.0	0.7	0.7
min_AR **(new)**	0	0	0.6
max_AR **(new)**	100	100	1.6
max_cosine_distance	0.45	0.1	0.1
nn_budget	1	1	1
max_iou_distance	0.7	0.3	0.3
max_age	30	3	3
n_init	0	0	0
lambda	1	1	1

6 Results

6.1 Reproduction of Results on MOT20-test

To ensure correct usage of the open source code of DeepSORT and StrongSORT, the two algorithms were first benchmarked on the MOT20 dataset. The hyperparameter values used are reported in Tables 1 and 2, and they correspond to the values reported in the publications associated with the two algorithms [15]. Table 5 shows the performances achieved in terms of several popular MOT evaluation metrics. Our obtained metric values are very close to the reported ones, demonstrating a correct approach and implementation.

Table 5: Reproduced and reported performance metrics obtained on MOT20-test.

Results on MOT20-test						
Algorithm	HOTA(↑)	IDF1(↑)	MOTA(↑)	AssA(↑)	DetA(↑)	IDSW(↓)
DeepSORT (ours)	57.12	69.62	71.77	55.49	58.97	1418
DeepSORT (official [15])	57.10	69.60	71.80	55.50	59.00	1418
StrongSORT (ours)	61.63	76.31	72.08	63.71	59.77	1058
StrongSORT (official [15])	61.50	75.90	72.20	63.20	59.90	1066

6.2 Benchmarking on DayGSD

The performances achieved with the four tracking algorithms are listed in Table 6. The upgrades made to DeepSORT and StrongSORT led to a considerable improvement in terms of the metrics HOTA, IDF1, AssA, and IDSW. There is a slight lag behind the original algorithms when considering MOTA and DetA performance. However, the improved algorithms generally lead to a better performance. The performance metrics used are detailed in Section 3.2. Of these, HOTA is generally viewed as the most comprehensive MOT performance metric, as it includes all aspects of tracking: detection, association, and localization. A clear improvement of DeepSORT and StrongSORT can thus be observed after applying our upgrades. This translates into an increase of the leading metric, HOTA, from 30.93 to 43.01 for DeepSORT, and from 27.95 to 42.89 for StrongSORT.

Table 6: Performance metrics obtained on DayGSD using DeepSORT, upgraded DeepSORT, StrongSORT, and upgraded StrongSORT.

Results on DayGSD						
Algorithm	HOTA(↑)	IDF1(↑)	MOTA(↑)	AssA(↑)	DetA(↑)	IDSW(↓)
DeepSORT	30.93	26.98	**53.22**	17.33	**56.75**	11141
Upgraded DeepSORT	**43.01**	**44.67**	49.26	**35.29**	53.40	**5715**
StrongSORT	27.95	23.10	**52.12**	14.05	**57.36**	15427
Upgraded StrongSORT	**42.89**	**42.16**	50.24	**33.93**	55.33	**7892**

6.3 Benchmarking on ModMOT20

Similar to the performance overview on GSD, Table 7 shows that the results achieved using the upgraded versions of DeepSORT and StrongSORT surpass those reached using the original algorithms. In this case, MOTA and DetA metrics are also higher when the customized algorithms are used. The only performance metric that is not improved by the upgraded StrongSORT is IDSW, for which the original StrongSORT achieves a lower value. The overall improved performance confirms that the characteristics identified as challenging in both GSD

and ModMOT20 were adequately addressed by the new features added to DeepSORT and StrongSORT.

Table 7: Performance metrics obtained on DayGSD using Deepsort, upgraded DeepSORT, StrongSORT, and upgraded StrongSORT.

Results on ModMOT20						
Algorithm	HOTA(↑)	IDF1(↑)	MOTA(↑)	AssA(↑)	DetA(↑)	IDSW(↓)
DeepSORT	23.19	22.26	64.53	8.54	63.50	70941
Upgraded DeepSORT	**36.53**	**42.68**	**71.72**	**20.33**	**66.19**	**60462**
StrongSORT	34.97	39.63	70.75	18.06	67.89	**53316**
Upgraded StrongSORT	**39.01**	**44.92**	**75.13**	**21.39**	**71.41**	60597

7 Discussion and Conclusions

The results achieved in this paper demonstrate multiple important findings. To begin with, we show that the performance of standard tracking algorithms on challenging datasets can be improved by further adding important customized features. These additions are tailored to face the challenges offered by industrial and agricultural datasets. As seen in the cases of GSD and ModMOT20, challenges can arise, among others, in the form of camera motion, non-linear trajectories, and appearance changes over time. One important contribution of this paper is proposing additional features for existing MOT methods, including camera motion compensation, immediate track initialization, aspect ratio, relaxed location constraints, and appearance-only association. In our case, these features led to clear improvements in terms of all considered metrics on ModMOT20. Meanwhile, on GSD, the performance has vastly improved in terms of HOTA, IDF1, AssA, and IDSW, and is still comparable to that of the original trackers in terms of MOTA and DetA. The procedure used for adding these features to existing MOT algorithms is described in the methods of this paper, and can be used to improve tracking performance on other datasets similar to GSD.

Additionally, we provide a way to verify algorithm robustness to the aforementioned challenges. Given that the most popular MOT datasets present short videos, captured with stable cameras, it is difficult to estimate performance when complications are added. But, by benchmarking on our proposed dataset, ModMOT20, performance on challenging datasets can now be estimated, and potentially improved. We also provide insights regarding how to best imitate the various properties of the datasets we are interested in when transforming a well known dataset like MOT20. The camera motion we induce, the downsampling of the images, and the color variation we add can all be carefully tweaked to closely simulate certain variations in IoU shifts or chromatic evolution that typically appear in agricultural datasets.

The methods and experiments presented demonstrate a huge potential for further advancements in the use of challenging image datasets, and also the need for such datasets to be benchmarked on. The reported improvements give a promising overview of the use of agricultural datasets, despite the challenges of tracking objects whose intrinsic and extrinsic visual

properties are constantly changing. Improved performance on this task has vast implications and ramifications into optimizing the use of technology in the food sector.

References

[1] N. Aherwadi, U. Mittal, J. Singla, N. Jhanjhi, A. Yassine, and M. S. Hossain. Prediction of fruit maturity, quality, and its life using deep learning algorithms. *Electronics*, 11(24):4100, 2022.

[2] R. Alvarez. Predicting average regional yield and production of wheat in the argentine pampas by an artificial neural network approach. *European Journal of Agronomy*, 30(2):70–77, 2009.

[3] G. Bannerjee, U. Sarkar, S. Das, and I. Ghosh. Artificial intelligence in agriculture: A literature survey. *International Journal of Scientific Research in Computer Science Applications and Management Studies*, 7(3):1–6, 2018.

[4] S. Bargoti and J. Underwood. Deep fruit detection in orchards. In *2017 IEEE international conference on robotics and automation (ICRA)*, pages 3626–3633. IEEE, 2017.

[5] C. Bazinas, E. Vrochidou, C. Lytridis, and V. G. Kaburlasos. Time-series of distributions forecasting in agricultural applications: An intervals' numbers approach. *Engineering Proceedings*, 5(1):12, 2021.

[6] K. Bernardin and R. Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing*, 2008:1–10, 2008.

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

[8] S. Bezryadin, P. Bourov, and D. Ilinih. Brightness calculation in digital image processing. In *International symposium on technologies for digital photo fulfillment*, volume 1, pages 10–15. Society for Imaging Science and Technology, 2007.

[9] J. Cao, X. Weng, R. Khirodkar, J. Pang, and K. Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking. *arXiv preprint arXiv:2203.14360*, 2022.

[10] A. Cavallaro, O. Steiger, and T. Ebrahimi. Tracking video objects in cluttered background. *IEEE transactions on circuits and systems for video technology*, 15(4):575–584, 2005.

[11] Y. Chen, X. An, S. Gao, S. Li, and H. Kang. A deep learning-based vision system combining detection and tracking for fast on-line citrus sorting. *Frontiers in Plant Science*, 12:622062, 2021.

[12] Y. Cui, C. Zeng, X. Zhao, Y. Yang, G. Wu, and L. Wang. Sportsmot: A large multi-object tracking dataset in multiple sports scenes. *arXiv preprint arXiv:2304.05170*, 2023.

[13] S. S. Dahikar and S. V. Rode. Agricultural crop yield prediction using artificial neural network approach. *International journal of innovative research in electrical, electronics, instrumentation and control engineering*, 2(1):683–686, 2014.

[14] S. Dodge and L. Karam. Understanding how image quality affects deep neural networks. In *2016 eighth international conference on quality of multimedia experience (QoMEX)*, pages 1–6. IEEE, 2016.

[15] Y. Du, Y. Song, B. Yang, and Y. Zhao. Strongsort: Make deepsort great again. *arXiv preprint arXiv:2202.13514*, 2022.

[16] G. Dunn. Yield forecasting. *Technical Booklet*, page 19, 2010.

[17] G. D. Evangelidis and E. Z. Psarakis. Parametric image alignment using enhanced correlation coefficient maximization. *IEEE transactions on pattern analysis and machine intelligence*, 30(10):1858–1865, 2008.

[18] G. Farjon, O. Krikeb, A. B. Hillel, and V. Alchanatis. Detection and counting of flowers on apple trees for better chemical thinning decisions. *Precision Agriculture*, 21(3):503–521, 2020.

[19] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *IEEE transactions on pattern analysis and machine intelligence*, 32(9):1627–1645, 2009.

[20] M. Fiaz, A. Mahmood, and S. K. Jung. Tracking noisy targets: A review of recent object tracking approaches. *arXiv preprint arXiv:1802.03098*, 2018.

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

[22] Z. Ge, S. Liu, F. Wang, Z. Li, and J. Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021.

[23] A. Geiger, P. Lenz, and R. Urtasun. Are we ready for autonomous driving? the kitti vision benchmark suite. In *2012 IEEE conference on computer vision and pattern recognition*, pages 3354–3361. IEEE, 2012.

[24] R. Girshick. Fast r-cnn. In *Proceedings of the IEEE international conference on computer vision*, pages 1440–1448, 2015.

[25] R. Girshick, J. Donahue, T. Darrell, and J. Malik. Rich feature hierarchies for accurate object detection and semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 580–587, 2014.

[26] H. Habaragamuwa, Y. Ogawa, T. Suzuki, T. Shiigi, M. Ono, and N. Kondo. Detecting greenhouse strawberries (mature and immature), using deep convolutional neural network. *Engineering in Agriculture, Environment and Food*, 11(3):127–138, 2018.

[27] R. Harrell, D. Slaughter, and P. D. Adsit. A fruit-tracking system for robotic harvesting. *Machine Vision and Applications*, 2:69–80, 1989.

[28] F. Herzog, J. Chen, T. Teepe, J. Gilg, S. Hörmann, and G. Rigoll. Synthehicle: Multi-vehicle multi-camera tracking in virtual cities. In *Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision*, pages 1–11, 2023.

[29] D. Hoiem, S. K. Divvala, and J. H. Hays. Pascal voc 2008 challenge. *World Literature Today*, 24(1), 2009.

[30] B. Ji, Y. Sun, S. Yang, and J. Wan. Artificial neural networks for rice yield prediction in mountainous regions. *The Journal of Agricultural Science*, 145(3):249–261, 2007.

[31] M. Kaul, R. L. Hill, and C. Walthall. Artificial neural networks for corn and soybean yield prediction. *Agricultural Systems*, 85(1):1–18, 2005.

[32] B. Kim, Y.-K. Han, J.-H. Park, and J. Lee. Improved vision-based detection of strawberry diseases using a deep neural network. *Frontiers in Plant Science*, 11:559172, 2021.

[33] R. Kirk, M. Mangan, and G. Cielniak. Robust counting of soft fruit through occlusions with re-identification. In *Computer Vision Systems: 13th International Conference, ICVS 2021, Virtual Event, September 22-24, 2021, Proceedings 13*, pages 211–222. Springer, 2021.

[34] F. Kogan, N. Kussul, T. Adamenko, S. Skakun, O. Kravchenko, O. Kryvobok, A. Shelestov, A. Kolotii, O. Kussul, and A. Lavrenyuk. Winter wheat yield forecasting in ukraine based on earth observation, meteorological data and biophysical models. *International Journal of Applied Earth Observation and Geoinformation*, 23:192–203, 2013.

[35] A. Kolotii, N. Kussul, A. Shelestov, S. Skakun, B. Yailymov, R. Basarab, M. Lavreniuk, T. Oliinyk, and V. Ostapenko. Comparison of biophysical and satellite predictors for wheat yield forecasting in ukraine. *International Archives of the Photogrammetry, Remote Sensing & Spatial Information Sciences*, 2015.

[36] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *arXiv preprint arXiv:1504.01942*, 2015.

[37] Y. LeCun, Y. Bengio, and G. Hinton. Deep learning. *nature*, 521(7553):436–444, 2015.

[38] B. Li, J. Lecourt, and G. Bishop. Advances in non-destructive early assessment of fruit ripeness towards defining optimal time of harvest and yield prediction—a review. *Plants*, 7(1):3, 2018.

[39] P. Lin and Y. Chen. Detection of strawberry flowers in outdoor field by deep neural network. In *2018 IEEE 3rd International Conference on Image, Vision and Computing (ICIVC)*, pages 482–486. IEEE, 2018.

[40] P. Lin, W. Lee, Y. Chen, N. Peres, and C. Fraisse. A deep-level region-based visual representation architecture for detecting strawberry flowers in an outdoor field. *Precision Agriculture*, 21(2):387–402, 2020.

[41] T.-Y. Lin, M. Maire, S. Belongie, J. Hays, P. Perona, D. Ramanan, P. Dollár, and C. L. Zitnick. Microsoft coco: Common objects in context. In *Computer Vision–ECCV 2014: 13th European Conference, Zurich, Switzerland, September 6-12, 2014, Proceedings, Part V 13*, pages 740–755. Springer, 2014.

[42] W. Lin, H. Liu, S. Liu, Y. Li, R. Qian, T. Wang, N. Xu, H. Xiong, G.-J. Qi, and N. Sebe. Human in events: A large-scale benchmark for human-centric video analysis in complex events. *arXiv preprint arXiv:2005.04490*, 2020.

[43] G. Liu, X. Yang, and M. Li. An artificial neural network model for crop yield responding to soil parameters. In *International Symposium on Neural Networks*, pages 1017–1021. Springer, 2005.

[44] J. Luiten, A. Osep, P. Dendorfer, P. Torr, A. Geiger, L. Leal-Taixé, and B. Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, 129:548–578, 2021.

[45] H. Luo, Y. Gu, X. Liao, S. Lai, and W. Jiang. Bag of tricks and a strong baseline for deep person re-identification. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition workshops*, pages 0–0, 2019.

[46] H. Luo, W. Jiang, Y. Gu, F. Liu, X. Liao, S. Lai, and J. Gu. A strong baseline and batch normalization neck for deep person re-identification. *IEEE Transactions on Multimedia*, 22(10):2597–2609, 2019.

[47] Y. Mae, Y. Shirai, J. Miura, and Y. Kuno. Object tracking in cluttered background based on optical flow and edges. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 1, pages 196–200. IEEE, 1996.

[48] K. Maharana, S. Mondal, and B. Nemade. A review: Data pre-processing and data augmentation techniques. *Global Transitions Proceedings*, 2022.

[49] J. Masci, U. Meier, D. Cireşan, and J. Schmidhuber. Stacked convolutional auto-encoders for hierarchical feature extraction. In *Artificial Neural Networks and Machine Learning–ICANN 2011: 21st International Conference on Artificial Neural Networks, Espoo, Finland, June 14-17, 2011, Proceedings, Part I 21*, pages 52–59. Springer, 2011.

[50] M. Mkhabela, P. Bullock, S. Raj, S. Wang, and Y. Yang. Crop yield forecasting on the canadian prairies using modis ndvi data. *Agricultural and Forest Meteorology*, 151(3):385–393, 2011.

[51] X. Nie, L. Wang, H. Ding, and M. Xu. Strawberry verticillium wilt detection network based on multi-task learning and attention. *IEEE Access*, 7:170003–170011, 2019.

[52] R. Pahlavan, M. Omid, and A. Akram. Energy input–output analysis and application of artificial neural networks for predicting greenhouse basil production. *Energy*, 37(1):171–176, 2012.

[53] J. Pan and B. Hu. Robust occlusion handling in object tracking. In *2007 IEEE Conference on Computer Vision and Pattern Recognition*, pages 1–8. IEEE, 2007.

[54] J. Park and J. Jeong. Assessment of the effectiveness of a convolutional autoencoder for digital image-based automated core logging. *Geoenergy Science and Engineering*, 227:211802, 2023.

[55] D. Paudel, H. Boogaard, A. de Wit, S. Janssen, S. Osinga, C. Pylianidis, and I. N. Athanasiadis. Machine learning for large-scale crop yield forecasting. *Agricultural Systems*, 187:103016, 2021.

[56] A. Peirs, J. Lammertyn, K. Ooms, and B. M. Nicolaı. Prediction of the optimal picking date of different apple cultivars by means of vis/nir-spectroscopy. *Postharvest Biology and Technology*, 21(2):189–199, 2001.

[57] Y. Raja, S. J. McKenna, and S. Gong. Tracking and segmenting people in varying lighting conditions using colour. In *Proceedings Third IEEE International Conference on Automatic Face and Gesture Recognition*, pages 228–233. IEEE, 1998.

[58] D. Rapado Rincon, E. J. van Henten, and G. Kootstra. Development and evaluation of automated localization and reconstruction of all fruits on tomato plants in a greenhouse based on multi-view perception and 3d multi-object tracking. *arXiv e-prints*, pages arXiv–2211, 2022.

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

[60] E. Ristani, F. Solera, R. Zou, R. Cucchiara, and C. Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II*, pages 17–35. Springer, 2016.

[61] I. Sa, Z. Ge, F. Dayoub, B. Upcroft, T. Perez, and C. McCool. Deepfruits: A fruit detection system using deep neural networks. *sensors*, 16(8):1222, 2016.

[62] T. T. Santos, L. L. de Souza, A. A. dos Santos, and S. Avila. Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association. *Computers and Electronics in Agriculture*, 170:105247, 2020.

[63] S. Shao, Z. Zhao, B. Li, T. Xiao, G. Yu, X. Zhang, and J. Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018.

[64] R. K. Singh et al. Artificial neural network methodology for modelling and forecasting maize crop yield. *Agricultural Economics Research Review*, 21(347-2016-16813):5–10, 2008.

[65] F. Soheili-Fard and S. B. Salvatian. Forecasting of tea yield based on energy inputs using artificial neural networks (a case study: Guilan province of iran). In *Biological Forum*, volume 7, page 1432. Research Trend, 2015.

[66] J. Sun, Z. Shen, Y. Wang, H. Bao, and X. Zhou. Loftr: Detector-free local feature matching with transformers. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 8922–8931, 2021.

[67] P. Sun, J. Cao, Y. Jiang, Z. Yuan, S. Bai, K. Kitani, and P. Luo. Dancetrack: Multi-object tracking in uniform appearance and diverse motion. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 20993–21002, 2022.

[68] L. Teng, Z. Cheng, X. Chen, and L. Lai. Study on simulation models of tomato fruit quality related to cultivation environmental factors. *Acta Ecologica Sinica*, 32(2):111–116, 2012.

[69] K. Thongboonnak and S. Sarapirome. Integration of artificial neural network and geographic information system for agricultural yield prediction. 2011.

[70] M. van Dijk, T. Morley, M. L. Rau, and Y. Saghai. A meta-analysis of projected global food demand and population at risk of hunger for the period 2010–2050. *Nature Food*, 2(7):494–501, 2021.

[71] L. A. Varga, B. Kiefer, M. Messmer, and A. Zell. Seadronessee: A maritime benchmark for detecting humans in open water. In *Proceedings of the IEEE/CVF winter conference on applications of computer vision*, pages 2260–2270, 2022.

[72] J. P. Vasconez, J. Delpiano, S. Vougioukas, and F. A. Cheein. Comparison of convolutional neural networks in fruit detection and counting: A comprehensive evaluation. *Computers and Electronics in Agriculture*, 173:105348, 2020.

[73] S. Veenadhari, B. Misra, and C. Singh. Machine learning approach for forecasting crop yield based on climatic parameters. In *2014 International Conference on Computer Communication and Informatics*, pages 1–5. IEEE, 2014.

[74] E. Vitzrabin and Y. Edan. Adaptive thresholding with fusion using a rgbd sensor for red sweet-pepper detection. *Biosystems Engineering*, 146:45–56, 2016.

[75] W. Wang, J. Dai, Z. Chen, Z. Huang, Z. Li, X. Zhu, X. Hu, T. Lu, L. Lu, H. Li, et al. Internimage: Exploring large-scale vision foundation models with deformable convolutions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 14408–14419, 2023.

[76] J. Wen, T. Abeel, and M. de Weerdt. "how sweet are your strawberries?": Predicting sugariness using non-destructive and affordable hardware. *Frontiers in Plant Science*, 14:1160645, 2023.

[77] J. Wen, C. R. Verschoor, C. Feng, I.-M. Epure, T. Abeel, and M. de Weerdt. The growing strawberries: Tracking multiple objects with biological development over an extended period. NeurIPS 2023 Track Datasets and Benchmarks submission, with dataset available at https://data.4tu.nl/private_datasets/FfG7B9Q5FLDxTPx-4IFPDM6Wwp3Eo0o_4ReTf4zLvhw.

[78] N. Wojke, A. Bewley, and D. Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.

[79] H. Xiong, Z. Cao, H. Lu, S. Madec, L. Liu, and C. Shen. Tasselnetv2: in-field counting of wheat spikes with context-augmented local regression networks. *Plant methods*, 15(1):1–14, 2019.

[80] Y. Xiong, Y. Ge, and P. J. From. An improved obstacle separation method using deep learning for object detection and tracking in a hybrid visual control loop for fruit picking in clusters. *Computers and Electronics in Agriculture*, 191:106508, 2021.

[81] H. Yang, F. Chang, Y. Huang, M. Xu, Y. Zhao, L. Ma, and H. Su. Multi-object tracking using deep sort and modified centernet in cotton seedling counting. *Computers and Electronics in Agriculture*, 202:107339, 2022.

[82] F. Yu, H. Chen, X. Wang, W. Xian, Y. Chen, F. Liu, V. Madhavan, and T. Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pages 2636–2645, 2020.

[83] Y. Yu, K. Zhang, L. Yang, and D. Zhang. Fruit detection for strawberry harvesting robot in non-structural environment based on mask-rcnn. *Computers and Electronics in Agriculture*, 163:104846, 2019.

[84] J. Zhang, Y. Xun, W. Li, and C. Zhang. Double-channel on-line automatic fruit grading system based on computer vision. In *27th International Congress on High-Speed Photography and Photonics*, volume 6279, pages 1034–1040. SPIE, 2007.

[85] T. Zhang, K. Jia, C. Xu, Y. Ma, and N. Ahuja. Partial occlusion handling for visual tracking via robust part matching. In *Proceedings of the IEEE Conference on computer vision and pattern recognition*, pages 1258–1265, 2014.

[86] W. Zhang, J. Wang, Y. Liu, K. Chen, H. Li, Y. Duan, W. Wu, Y. Shi, and W. Guo. Deep-learning-based in-field citrus fruit detection and tracking. *Horticulture Research*, 9, 2022.

[87] Y. Zhang. A better autoencoder for image: Convolutional autoencoder. In *ICONIP17-DCEC. Available online: http://users. cecs. anu. edu. au/Tom. Gedeon/conf/ABCs2018/paper/ABCs2018_paper_58. pdf (accessed on 23 March 2017)*, 2018.

[88] Y. Zhang, P. Sun, Y. Jiang, D. Yu, F. Weng, Z. Yuan, P. Luo, W. Liu, and X. Wang. Bytetrack: Multi-object tracking by associating every detection box. In *European Conference on Computer Vision*, pages 1–21. Springer, 2022.

[89] Z. Zhu, K. Fujimura, and Q. Ji. Real-time eye detection and tracking under various light conditions. In *Proceedings of the 2002 symposium on Eye tracking research & applications*, pages 139–144, 2002.

[90] Z. Zong, G. Song, and Y. Liu. Detrs with collaborative hybrid assignments training. *arXiv preprint arXiv:2211.12860*, 2022.

A Effects of Frame Rate and Camera Motion Modifications

We created multiple modified versions of MOT20, by progressively lowering the frame rate of the original dataset. The effect of this modification on the IoU distribution of consecutive bbox locations belonging to the same object is shown in Figure 11. The effect of further adding camera motion to these modified datasets is shown in Figure 12.

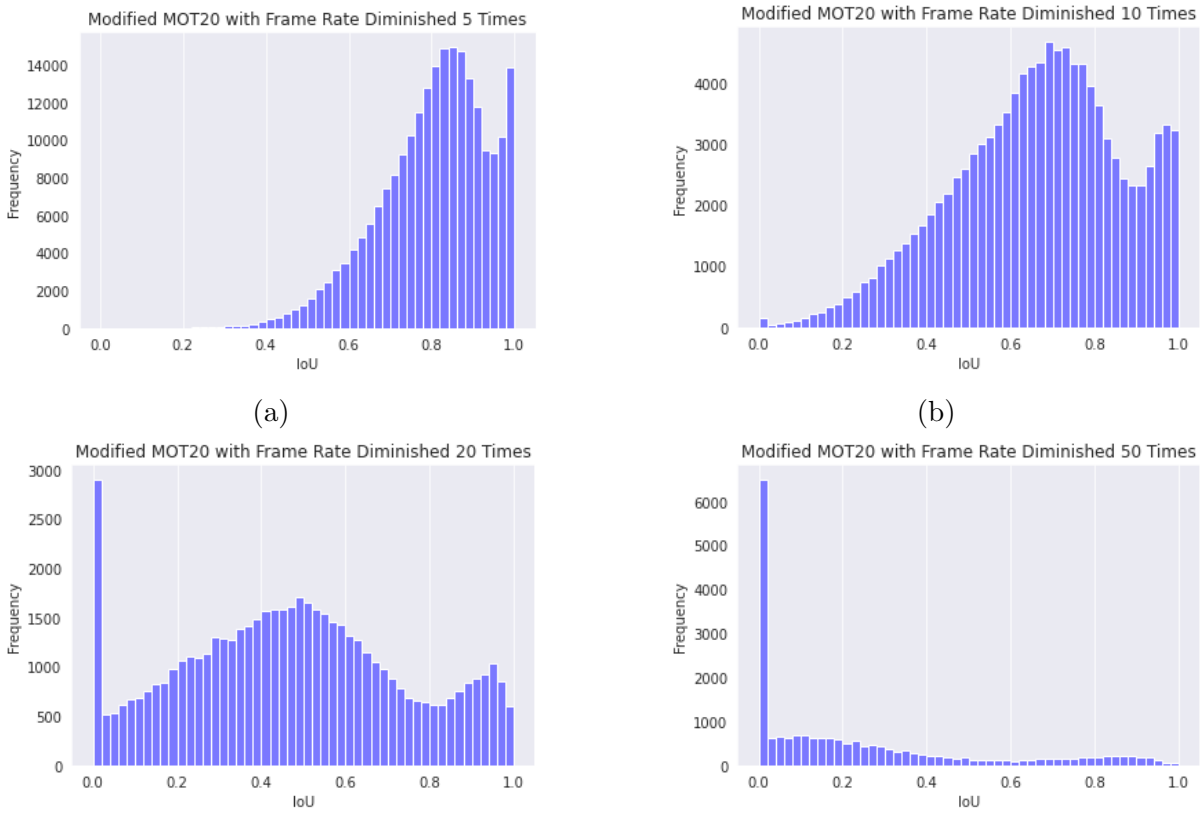


Figure 11: Histograms of the IoU values between consecutive bbox locations of objects in MOT20 modified to have frame rates lowered 5 times (a), 10 times (b), 20 times (c), and 50 times (d).

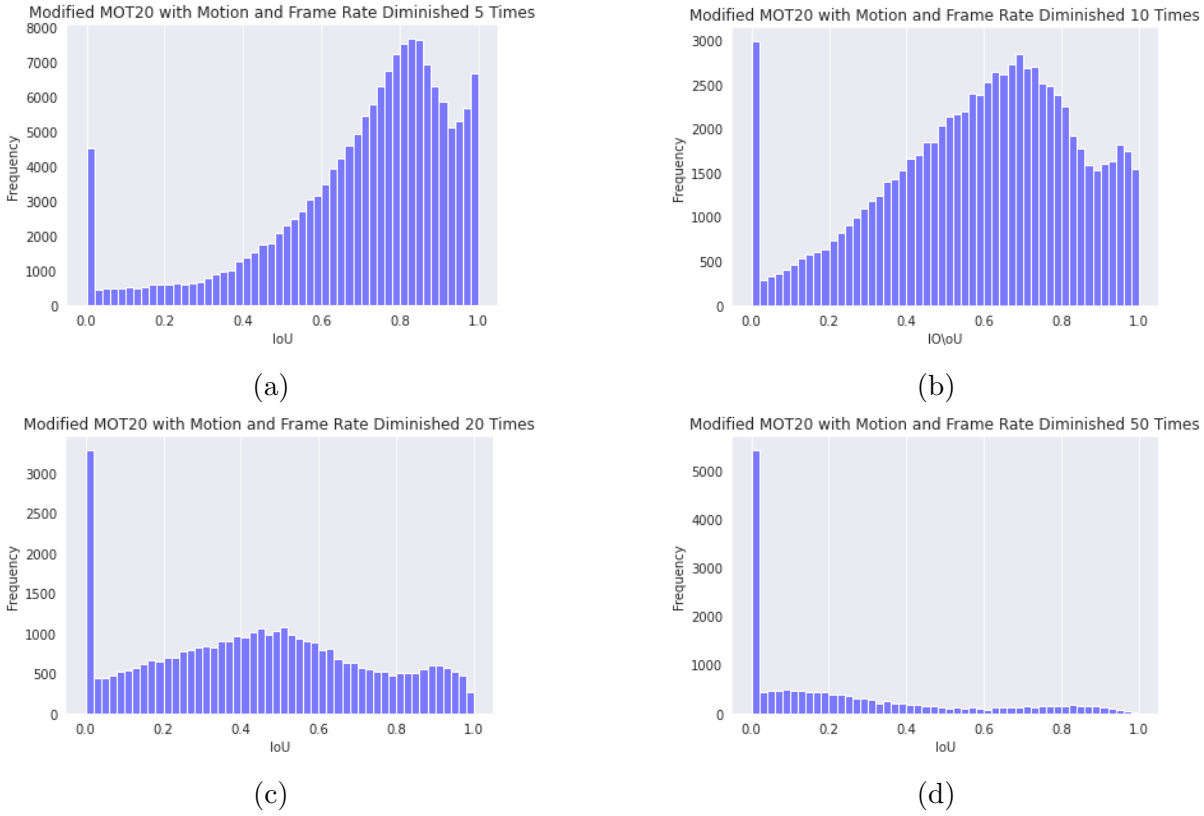


Figure 12: Histograms of the IoU values between consecutive bbox locations of objects in MOT20 modified to have camera motion, and frame rates lowered 5 times (a), 10 times (b), 20 times (c), and 50 times (d).