Tracking and Counting Apples in Orchards Under Intermittent Occlusions and Low Frame Rates

Gonçalo P. Matos[1,2], Carlos Santiago[2], João P. Costeira[2], Ricardo L. Saldanha[1], Ernesto M. Morgado[1]
[1] SISCOG – Sistemas Congitivos, SA, Lisbon, Portugal
[2] Institute for Systems and Robotics (ISR/IST), LARSyS, Instituto Superior Técnico, Univ. of Lisbon, Portugal
{goncalo.p.matos, carlos.santiago, jpcosteira}@tecnico.ulisboa.pt, {rsaldanha, emorgado}@siscog.pt

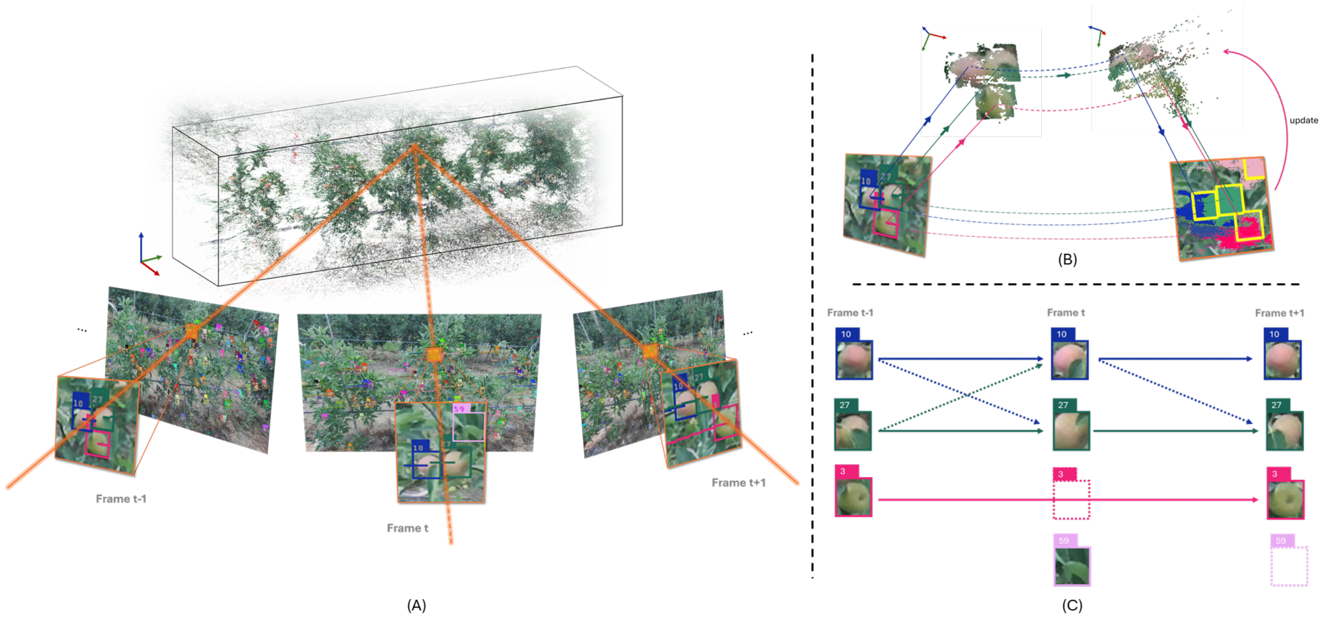

Figure 1. Overview of the proposed method. (A) Fruits are tracked throughout the image sequence by projecting them onto a 3D point cloud and re-projecting them back onto the following 2D frames. (B) 3D projection of three fruits. Rotating the point cloud, the noise in the depth direction is noticeable. (C) A linear assignment is used to match the most probable 3D objects to 2D detections. The method is robust to occlusions (e.g. fruit 3 is occluded in frame t) and false positives (e.g. "fruit" 59 in frame t is actually a leaf).

Abstract

Estimating what will be the fruit yield in an orchard helps farmers to better plan the resources needed for harvesting, storing, and commercialising the crop, and also to take some agricultural decisions (like pruning) that may increase the quality of the yield and increase profits. Therefore, over the last years, several methods based on computer vision were proposed to automate this task, by directly counting the fruits on trees using a video camera. However, existing works and methods usually assume ideal conditions, and may fail under more challenging scenarios with unconstrained camera motion and intermittent occlusions of fruits. Here we show that combining Structure-from-Motion (SfM) with a bipartite graph matching has the potential to address those challenges. We found that our approach applied to real-world datasets, with unconstrained camera motion and low frame rates, outperforms existing methods by a large margin. Our results demonstrate that the proposed method is robust to multiple intermittent occlusions under challenging conditions, and thus suitable to be used in diverse real-world scenarios in orchards, either with a camera operated by hand or mounted on an agricultural vehicle. Although not shown here, we believe that the proposed method can also be applied to other object tracking problems besides counting fruits, under similar settings — i.e. static objects and a freely moving camera.

1. Introduction

Estimating crop yield is a crucial task in modern agriculture as it conditions the demand for resources (e.g. workers for harvesting, packaging) and the planning of activities (e.g. pruning) which, ultimately, affect the quality of the product. Due to its flexibility and breadth, computer vision systems are becoming a key technology in monitoring crop growth.

In this article we propose a methodology to estimate fruit counts in orchards from sequences of images captured by a moving camera. The distinguishing feature stands on the resilience to occlusions and misdetections, conferring our method a surprisingly high precision of the counts, specially in such a complex visual context. The complexity stems from the shape of the tree canopy, its porosity, the intricate patterns of leaves and branches, and how the fruits themselves develop (close to other fruits). Furthermore, we deliver precise counts in long extensions that include multiple trees, often disposed in a continuum.

Figure 1A) shows part of a reconstructed orchard and the images illustrate the complexity of the scene and type of detections. Our method proposes to keep track of detections (Figure 1C) by a virtuous combination between 2D and 3D data, schematically represented in Figure 1B). Representing fruits as 3D (very noisy) point clouds, we track the fruits across images by solving a sequence of linear assignment problems between labelled 3D points and the detections in each image. The robustness of the search lies on the assignment process that easily handles occlusions (e.g. pink path in Figure 1C) and to an incremental update to the 3D representation that reinforces the geometrically correct solution. As we document in the experiments, our method is quite reliable to both missing and false detections as well as to assignment errors due to point cloud noise or proximity between fruits.

We do not impose restrictions on motion neither on the scene. The images can be acquired closely (e.g. video) or largely spaced, as long as they overlap enough to form a connected pose-graph allowing a global reconstruction. Fruits are detected by some detector (e.g. YOLO [19]) trained for high precision, but we admit a low recall. Intrinsic parameters are known and the camera matrices are estimated, for example, by some sparse Structure-from-Motion (SfM) algorithm. Finally we assume that some device (e.g. depth camera) or Multi-View Stereo (MVS) algorithm provides a dense depth estimation just of the fruit patches (detection boxes).

The main contributions of this paper are as follows:

- An **object tracking algorithm** that is robust to multiple intermittent occlusions, and can be applied to any scenario with fixed objects and a freely moving camera;
- Four **datasets** for fruit detection and tracking/counting, one of them synthetic.

2. Related work

Regarding the problem of fruit tracking, the approaches that can be seen in the literature usually rely either on multiple-object tracking algorithms that work on the image domain (2D), and are mostly based on finding similarity between detections in consecutive video frames, or on some kind of 3D data, such as sparse or dense reconstructions achieved by SfM algorithms or specialised depth cameras.

Most works that fall into the first category generally use Kalman filters combined with some other computer vision technique, like the optical flow, to try to predict what will be the position of current frame's detections on the next frame and hence match detections between consecutive frames. This is the approach followed, for example, in [14, 15]. Other works in this category rely on affine transformations to approximate the relationship between frames, under the assumption that the camera movement is small on consecutive frames — this is the case, for example, in [20, 21]. However, this assumption does not hold for all use cases — in particular, due to fast camera motion and low frame rates, finding similarities between consecutive video frames may be a challenging task. Moreover, these methods are not usually robust under intermittent occlusions, because they lose track of objects if they are not visible on the next frame.

If we move out of the literature specialised in fruit tracking and look at the more general problems of multi-object tracking or optical flow, several algorithms have been proposed to track multiple objects or pixels in video sequences based on their 2D appearance. These include FlowNet [4], SORT [2], DeepSORT [27, 28], ByteTrack [29], Omnimotion [26], and CoTracker [12]. The most reported difficulties with this kind of approaches — for example, in [18, 25] — are usually the lack of robustness to occlusions and, in the case of tracking to count objects, the possibility of missing some objects if they appear only after the Region of Interest (ROI)[1]. These methods usually do not work well with low frame rates, either. Moreover, the kind of fruit counting followed by [18] assumes that the camera moves with a relatively constant speed and always in the same direction, and hence that each object crosses the ROI only once. This is not the case if we allow the camera to freely move, and to be operated by hand or mounted on an agricultural vehicle that may accelerate/decelerate, for example. Finally, some of these methods require training data, and may not generalise well for new scenarios without training.

Another family of approaches rely on the use of 3D data that is either produced by depth cameras or stereo cameras, or that is estimated by running a SfM pipeline such as COLMAP [22]. In this category fall the works [15, 23], which rely on a sparse 3D reconstruction and on epipolar

[1]The ROI is usually a line that crosses the video frame and that is used by counting algorithms to count objects only once, as they cross the line.

geometry to match fruit detections. On the other hand, other authors [7, 8, 14, 17, 24] actually relied on a dense depth map to match their fruit detections in 3D. While being computationally more demanding than the 2D tracking methods, this category of methods has the advantage of not being so dependent on 2D similarity between consecutive frames to track objects, and hence has the potential to be more robust to intermittent occlusions. However, most authors use the 3D information only to identify fruits that were tracked in 2D (for example, with Kalman filters) and avoid duplicate counts. Since their tracking is actually performed in 2D with Kalman filters, it assumes a dynamic model which is not compatible with certain types of camera motion and/or low frame rates. We propose a different method to assign 3D fruits to 2D detections that takes advantage of this 3D information to perform the actual tracking, and thus imposes less requirements on camera motion. Furthermore, some of the existing solutions require special setups, such as light shields or "tunnels" to block light, that require some manual labour and do not scale well for a whole orchard. We propose a solution that requires only a video camera, and is thus suitable to be mounted on an agricultural vehicle and be used at scale to cover a whole orchard. The interested reader may find other fruit tracking methods in the excellent reviews of [9, 10].

3. Proposed approach

3.1. Problem statement

We may formulate the problem that is the focus of this work in the following terms:

Given a sequence of images and a set of fruit detections in each image, assuming a static scene and a moving camera, assign a unique ID to each fruit and track it along the whole image sequence, regardless of the multiple occlusions and appearances that each fruit may have due to the erratic camera movement and/or natural structures that may intermittently block it from view.

3.2. Overview of the method

First we either collect video frames with a camera or synthetically generate them. Then, we estimate the camera intrinsics and extrinsics, and a depth map for each frame with SfM. For this task, we used COLMAP [22], for convenience. In parallel, we *detect* the objects of interest in each image of the sequence. For this matter, we trained a YOLOv4 [3] for apples detection, but any other object detector would work. We also annotated four datasets that served as "perfect" detections for the results listed in this paper. Finally, our tracking algorithm — which is summarised in Figure 1 — is fed with the 2D detections, the camera intrinsics and extrinsics, and depth maps for each image in the sequence. Having these inputs, we can relate any pixel in any image with a 3D point in a coordinate system that is common to the whole video. In particular, the pixels representing the same fruit in different images should be mapped to roughly the same 3D region, and that is the rationale used to match the 2D detections with the 3D location of each tracked object. From those tracked objects, the total object count in the video can be directly retrieved. This component is the major contribution of this paper, and to it is devoted Subsection 3.3.

Following up on the introduction, Figure 1 shows the main steps of our approach. Despite the perceived accuracy of the global reconstruction shown in sub-figure A), the fruit's shape is poorly reconstructed, delivering very noisy point clouds as shown in B). The 3D points pertaining to pixels in each bounding box are rigidly transformed to another camera frame and then projected back to the image. As it can be easily inferred, the 3D data *per se* does not allow fruit identification or matching because it is extremely noisy and the point cloud contains data corresponding to both the fruit and the background. However, when back-projected, despite the point spread, the majority of the 3D points fall within the area of the correct detection bounding box, thus counterbalancing the error in the depth. After matching the fruits to the boxes, the new detections contribute with new points to update the 3D representation thus reinforcing the "central cloud", if correct, or spreading around, if the matching is wrong. Finally, sub-figure C) illustrates the assignemnt process that pairs corresponding detections along the sequence. Dashed lines symbolise potential matches, full lines the correct match and the bottom path includes an occluded fruit that must be dealt with. False detections must survive 5 frames or are eliminated.

3.3. The fruit tracking algorithm

The fruit tracking algorithm that we propose is actually a multi-purpose object tracker for image sequences that can be applied to any kind of objects, provided that the objects' positions are fixed in world coordinates — i.e. static objects, moving camera. In the case of trees, this is guaranteed by collecting the images in a windless day, and is easier to achieve if the trees are also trained against a flat surface (e.g. trellis), which is the case in our datasets and on many modern orchards. We make no other assumptions, namely with regards to camera trajectory, which can be freely chosen.

The inputs required by the tracking algorithm are the camera intrinsics and extrinsics, dense depth map and bounding boxes of the detected objects, for each image. It outputs a set of objects, that correspond to the tracked fruits' detections along the image sequence, and the 3D coordinates of their centroids in a global coordinate system.

Algorithm 1 summarises how the tracking algorithm works. To make Algorithm 1 easier to read, we encapsulated a few components in functions in the following pseudo-code that we detail in the next Subsections.

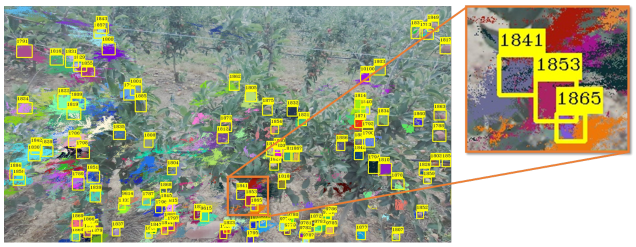

Figure 2. Projection of fruits' 3D points onto 2D detections, on an image of the dataset *Galafab-west*. The re-projected points of each 3D fruit are represented with a unique colour for that fruit. On the right, a closeup over three detections is shown.

Algorithm 1: Fruit tracking algorithm

```
Data: $video, detections, intrinsics, extrinsics, depth\_map$
Result: $fruits$
1  $fruits \leftarrow []$;
2  $frameNr \leftarrow 1$;
3  $newId \leftarrow 1$;
4  while $video$ not ended do
5      $frameNr \leftarrow next\_frame\_number(video)$;
6      $projected\_points \leftarrow$
         $project(fruits, intrinsics, extrinsics[frameNr])$ ;
         /* Project 3D points onto current frame */
7      $votes \leftarrow$
         $count(projected\_points, detections[frameNr])$ ;
         /* Count how many points fall inside each
         detection */
8      $assignment \leftarrow Hungarian(votes)$;
9      for $detection$ in $detetctions[frameNr]$ do
10         if $assignment[detection]$ exists then
               /* Add detection to existing fruit   */
11             $fruit\_id \leftarrow assignment[detection]$;
12             $fruits \leftarrow$
                 $update\_fruit(fruits, fruit\_id, detection,$
                 $intrinsics, extrinsics[frameNr],$
                 $depth\_map[frameNr])$;
13         else
               /* Create new fruit                  */
14             $fruit\_id \leftarrow newId$;
15             $newId \leftarrow newId + 1$;
16             $fruits[fruit\_id] \leftarrow$
                 $create\_fruit(detection, intrinsics,$
                 $extrinsics[frameNr],$
                 $depth\_map[frameNr])$;
17         end
18     end
19 end
20 return $fruits$
```

3.3.1 Project 3D fruits onto the current 2D image

The `project` function takes all the 3D points of the currently tracked fruits and projects them onto a 2D image as observed by the camera in the current frame of the sequence, keeping the ID of the fruit where each point came from. The 3D points that we (re-)project onto the current 2D image correspond to the pixels from all the detections of that same fruit in previous images, projected on a global 3D coordinate space for the whole image sequence. Figure 2 illustrates such a re-projection in one image of the sequence. The coloured patches correspond to the 3D points of previously seen fruits re-projected onto the current image, while the yellow boxes correspond to 2D detections on the current image. There is some distortion of the patches, specially on fruits that were first seen a long time ago, because they were usually first seen in a frontal view and are currently being seen in a considerably different angle, and there is some amount of noise in the depth estimation particularly in the direction perpendicular to the camera.

3.3.2 Match 3D fruits to 2D detections

The `count` function is used to build a score matrix for a Hungarian assignment from 3D fruits to 2D detections. It counts how many re-projected 3D points from a given fruit fall inside the bounding box of a given 2D detection, and puts these results in a 2D matrix.

The `Hungarian` function corresponds to running the bipartite graph matching (Hungarian) algorithm [13] with the score matrix just described to assign existing 3D fruits to 2D detections of the current image, i.e. to assign an ID of a fruit to each detection. As became evident in the closeup detail of Figure 2, there is some overlapping between re-projected points of different fruits onto the same 2D detections. This happens, mainly, on fruits that are physically close to each other, but also on fruits that only *seem* to be close to each other when re-projected in 2D because they are on the same line of sight in the particular point of view of the camera for the current image. Furthermore, some 2D detections also overlap each other and compete for the same fruits. This is the reason why we resort to the Hungarian algorithm, to disambiguate overlapped re-projections and assign them to the most probable detection.

3.3.3 Keep track of fruits

When a 2D detection in the current image is assigned to an existing fruit, we enter the block starting in line 10 of Algorithm 1. In that case, the existing fruit is updated to include the 3D points from the new detection. Function `update_fruit` updates the existing fruit by projecting all the pixels from the new 2D detection onto the corresponding 3D points, using the camera intrinsics and extriniscs and the depth map.

When a 2D detection in the current image cannot not be assigned to an existing fruit we enter the block initiated by line 13 of Algorithm 1. In this case, the function `create_fruit` is used to project all the pixels of the 2D detection onto 3D points, similarly to the function `update_fruit`, but then a brand new object, with a unique fruit ID, is created and added to the `fruits` list, containing these 3D points.

In the end of the algorithm, the list of fruits is returned. This list contains all the 3D objects that were tracked throughout the video and their respective 2D locations on each image. The total number of fruits is simply the number of objects in this list. In our implementation, we discard too short tracks — with less than 5 detections — that usually correspond to false detections. This was experimentally determined to be a good threshold for our use cases.

4. Experimental setup

4.1. Description of the data

In order to test the performance of the proposed algorithm and to elaborate the results presented in Section 5, we collected 4 datasets corresponding to video sequences capturing 3 to 6 trees under different settings. These are summarised in Table 1. For each dataset, we indicate in the last column the ground truth number of apples that are visible in the video, and which will be used later to compute the errors of the estimations. We leave to the supplementary material some figures that illustrate the kind of frames that compose each video, and a representation of the camera trajectories as well.

Table 1. Datasets description.

Dataset	**# trees**	**# frames**	**frame rate (fps)**	**# apples**
Galafab-west	3	110	∼3	233
Schnico-Red-east	5	155	∼3	356
Schniga-Schnico-west	6	200	∼3	373
Synthetic-apples-1	5	250	25	204

The first three datasets correspond to video sequences obtained with a manually operated camera, moved by a person on foot, in an apple orchard from INIAV [11] in Alcobaça, Portugal. The captured trees correspond to three different varieties of apples — named Galafab, Schnico Red and Schniga Schnico — in the final stages of maturity, close to harvest. Trees in these datasets are flat and trained against horizontal wires. These datasets are challenging due to the camera movement, which moves freely on the three axis, panning and tilting to capture the whole tree's canopies that do not fit entirely in one single video frame. Sometimes the camera moves backwards, causing the same fruits to leave and re-enter the video frame a few times. Moreover, there are heavy occlusions caused by leaves and other fruits.

We also synthetically generated a dataset composed of 5 apple trees using the open-source 3D modelling and animation tool Blender [5]. In this dataset, we animated a camera moving horizontally with initial acceleration and final deceleration, simulating the kind of movement one would obtain with a camera mounted on an agricultural vehicle. Apples are red and in their final maturity level, close to harvest. This dataset has perfect data (i.e. no noise) in what concerns the camera intrinsics, extrinsics and depth maps, a higher frame rate (smaller jumps between frames) and softer shadows, compared to the previous datasets. It was meant to be an easier scenario for object tracking.

The datasets are available at `https://www.siscog.pt/en-gb/lp/paper-v4a2024/`.

4.2. Ground truth annotation

Due to the nature of our problem, which can be decomposed into fruit *detection* and fruit *tracking* sub-problems, two kinds of annotations are needed. For the former, we need to annotate with bounding boxes the location of all the fruits that appear in each video frame. For the latter, we have to manually assign a unique fruit ID to each bounding box in the video, thus effectively performing fruit tracking by hand. The two tasks were performed by using a graphical tool that we developed ourselves. For the synthetic dataset, we created a Python script that programmatically generates both the bounding box and the tracking annotations for each fruit present in the images. These annotations are generated at the same time that the images are rendered, inside Blender.

4.3. Comparison procedure

We compared the performance of our algorithm with that of other publicly available tracking algorithms that were proposed in the literature for fruit tracking, on our datasets. We conducted this experiment using the ground truth annotations as input detections for the trackers, to simulate a scenario where the fruit detector was perfect — i.e. Precision and Recall of the detector are both 100%. With this experiment, we assess the performance of each tracker in isolation, under perfect detection conditions.

We took the code of a fruit counter proposed by Gené-Mola et al. [6], and adapted it to receive as input our ground truth detections. It has the ability to run using different multi-object tracking algorithms — namely, SORT [2], DeepSORT [27, 28] and ByteTrack [29] — that the user/programmer may easily switch in the code, which allowed us to compare the performance of several object trackers on our datasets using the same code base. Although the code was prepared to run the three aforementioned trackers, only the source code for ByteTrack was actually included in the repository. Therefore, we added the source code of SORT from the official repository [1] and a PyTorch implementation of DeepSORT [16] to Gené-Mola et al. fruit counting algorithm [6]. For the SORT tracker, we set the parameters `max_age=30` and `min_hits=1` to work better in our datasets — the default `max_age=1` and `min_hits=3` did not work so well in our shaky footage with intermittent occlusions, so the tracker outputted almost no tracks in the majority of the video frames. For the remaining trackers, we used the default parameters, which produced already the best results for them.

4.4. Comparison metrics

In order to compare the performance of the different trackers, given their output tracks and fruit IDs and the ground truth ones, we compute the *Absolute Percentage Error (APE)* defined as:

$$APE = \frac{|Estimated - Actual|}{Actual} \times 100 \quad (1)$$

where *Estimated* is the number of fruits estimated by the algorithm, while *Actual* is the ground truth. Lower APE values are better. We also computed the Multple Object Tracking Accuracy (MOTA), which is commonly seen in the literature related to object trackers in general. It is a metric valued in $]-\infty, 1]$, according to the formula:

$$MOTA = 1 - \frac{FN + FP + IDS}{GT} \quad (2)$$

where *FN* is the total number of False Negatives in the whole sequence, *FP* is the total number of False Positives, *IDS* is the total number of identity switches — i.e. the number of times two objects mistakenly take the identity of one another —, and *GT* is the ground truth number of objects. The closer the MOTA is to 1, the better the tracker.

To count false positives and false negatives and compute the metrics, we considered a minimum Intersection over Union (IoU) of 0.3 for two detection bounding boxes to be considered the same — i.e., if a detection bounding box in a candidate solution overlaps a ground truth detection by at least 30% of their area, it is considered the same detection, hence we tolerate some inaccuracies in the positions and sizes predicted by the tracker that do not affect significantly the task of tracking and counting fruits.

The metrics *Precision* and *Recall* are used in the next Sections to evaluate how well the trackers process the input detections, which are deemed perfect in this scenario. In fact, even with perfect detections, the *Recall* may be lower than 1.0 because many trackers simply omit a detection if for some reason they cannot track it. For example, our tracker fails to process a detection if depth information is unavailable for the pixels contained in it[2]. On the other hand, the *Precision* may also be lower than 1.0, because some trackers try to estimate the future position of a previous detection (e.g. using Kalman filters) and they fail to do so, ending up with a detection box that does not match the ground truth's position and/or dimensions.

On the tables presented in the next Sections, the column *APE % (raw)* refers to the computation of the APE metric on the total number of tracked fruits without any kind of filtering, while the column *APE % (filtered)* corresponds to the actual fruit count estimated and returned by our algorithm after discarding the fruit tracks with less than 5 detections (as explained in Subsection 3.3.3). To make a fair comparison, we apply the same filter to the ground truth data and to the output of the other object trackers. These filtered fruit counts are reported in column *# apples (estimated)* and in the ground truth indicated in the table's captions. The MOTA is always computed over the raw data, to make it more generally comparable with other trackers in the literature without the specificity of our filters for fruit counting.

[2]Sometimes the depth maps have missing values due to the SfM inability to estimate the depth of a particular pixel.

Table 2. Comparison of different trackers on the Galafab-west dataset. The ground truth number of apples is 233.

Tracker	Precision	Recall	# apples (estimated)	APE % (raw)	APE % (filtered)	MOTA
Ours	**1.00**	**0.99**	203	15.93	**12.88**	**0.7591**
ByteTrack	1.00	0.69	365	47.80	56.65	0.2099
SORT	1.00	0.53	79	380.77	66.09	0.0468
DeepSORT	0.53	0.04	40	82.42	82.83	-0.1639

Table 3. Comparison of different trackers on the Schniga-Schnico-west dataset. The ground truth number of apples is 373.

Tracker	Precision	Recall	# apples (estimated)	APE % (raw)	APE % (filtered)	MOTA
Ours	**1.00**	**0.91**	356	27.75	**4.56**	**0.6068**
ByteTrack	0.99	0.70	798	165.04	113.94	0.2079
SORT	1.00	0.30	67	497.03	82.04	-0.1841
DeepSORT	0.63	0.08	162	35.38	56.57	-0.2829

Table 4. Comparison of different trackers on the Schnico-Red-east dataset. The ground truth number of apples is 356.

Tracker	Precision	Recall	# apples (estimated)	APE % (raw)	APE % (filtered)	MOTA
Ours	**1.00**	**0.96**	359	4.44	**0.84**	**0.6982**
ByteTrack	0.99	0.63	418	60.23	17.42	0.1600
SORT	1.00	0.43	49	295.95	86.24	-0.0498
DeepSORT	0.54	0.05	71	78.19	80.06	-0.1684

Table 5. Comparison of different trackers on the Synthetic-apples-1 dataset. The ground truth number of apples is 188.

Tracker	Precision	Recall	# apples (estimated)	APE % (raw)	APE % (filtered)	MOTA
Ours	**1.00**	**1.00**	186	9.90	**1.06**	**0.9375**
ByteTrack	1.00	0.79	180	109.38	4.26	0.5375
SORT	1.00	0.55	81	448.96	56.91	0.2930
DeepSORT	0.94	0.32	74	60.42	60.64	0.1977

5. Results

5.1. Comparison with other trackers under perfect detections

The results are presented in Tables 2, 3, 4 and 5. The results show that our tracker is consistently better than the others in all the four datasets, and in all the aspects that are being evaluated. Our proposed approach achieves a Precision of 1.00 in all datasets, meaning that it never predicts a wrong object location or size, because in fact it never changes the bounding boxes provided as input. DeepSORT, on the other hand, presents significantly lower values for this metric, because it changes considerably the input detections. This suggests that its learned feature descriptor is not very reliable for the particular kind of images in this domain. With respect to the Recall metric, our proposed algorithm achieved values consistently above 0.90 in all datasets with a clear margin over the remaining trackers. This metric reveals that our tracker can process almost all the detections, while other trackers struggle to track certain fruits.

Regarding the total estimated number of apples in the video, our proposed algorithm also achieves the best results, with the lowest APE (filtered) on all datasets. ByteTrack also achieves a remarkable result in the *Synthetic-apples-1* dataset, which has only horizontal movement, however it falls to much higher estimation errors on the other datasets, where the camera moves freely and the frame rate is lower. The *Galafab-west* dataset, though, seems to be particularly challenging for our algorithm. There are frames in this dataset where several fruits have their identities swapped or lost to a new ID, particularly on a set of fruits that are lying on the ground that enter and leave the frame several times. We believe that these issues, while few enough to not affect the MOTA metric too much, are responsible for the higher estimation error in the number of fruits.

When we look at the quality and consistency of the tracks, our proposed algorithm achieves the highest values for the MOTA metric, again with a large margin over the other trackers. This means that it is not simply giving an accurate estimation for the total number of fruits, which could still be achieved with many identity changes between the tracked fruits, but it is actually following each unique fruit with a a high accuracy.

5.2. Ablation studies

5.2.1 Post-processing filter

Another remark that can be derived from the previous results is the relative importance of the filtering procedure. In most cases, the filtered APE exhibits a better value than the raw APE, which means that by discarding fruits with very few observations we are in fact discarding more tracking errors and/or false detections (in case we use a real object detector) than true positives. Taking into account that most false positives correspond to unmatched or spurious detections that usually do not have a following detection in the next video frame, simply imposing a limit of at least 2 observations for each fruit would already filter out the majority of false positives, without sacrificing too much true positives in the final result. The minimum of 5 observations for each fruit used in this paper was determined experimentally for the presented datasets and may need to be adjusted for very different scenarios.

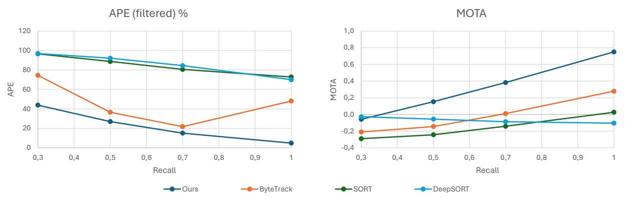

Figure 3. APE (filtered) and MOTA achieved by each tracker, as a function of the Recall of the detector.

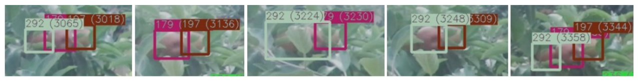

Figure 4. Closeup over three adjacent fruits on frames 50-54 from *Galafab-west* dataset, where YOLO incorrectly merged two fruits in the same detection. Occlusions also occur. Colours and numbers correspond to the tracking output of our algorithm.

5.2.2 Robustness under imperfect detections

To assess the degradation on the fruit counts under imperfect object detections, we randomly removed some detections from the ground truth of each dataset, to simulate an object detector with a *Recall* of 0.7, 0.5 and 0.3. Then, we ran all the fruit trackers with the same "imperfect" detections as input, and we plot the average APE across all the 4 datasets in Figure 3. Results show that our proposed method is consistently more accurate than all the other methods, and it can still provide satisfactory estimations even when paired with an object detector that has a high false negative rate.

The behaviour of the APE curve for ByteTrack is due to its tendency to overestimate the number of fruits. Therefore, by reducing the number of detections, the error reduces. The MOTA curve, however, shows that its performance is actually getting worse. On the other hand, the DeepSORT MOTA curve increases with lower Recalls because it is tracking very few fruits, and hence the number of identity switches is very low. However, the APE curve shows that its real performance is actually very poor.

5.3. Challenging cases

5.3.1 Multiple fruits merged in one detection

For this matter, we consider the proposed fruit tracking algorithm paired with a real object detector — a YOLOv4 model trained to detect apples. Figure 4 illustrates one failure case of the object detector, where two fruits were incorrectly merged into a single detection. Intermittent occlusions also occur in that example. In a situation like this, if the incorrect detection happens consistently on a significant number of images for that pair of fruits, our tracker will unavoidably track those two fruits as one 3D object, resulting in an error in the final count. Even if the fruits were initially detected separately and originated two 3D objects, the in-

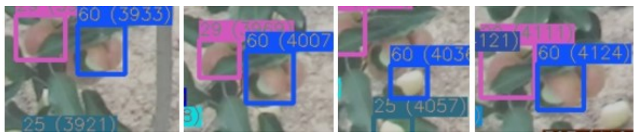

Figure 5. Closeup over two adjacent detections on frames 60-63 from *Galafab-west* dataset. The blue detection is tracking a fruit on a tree, but is switched to a fruit on the ground in the third frame.

correct merge of two fruits over multiple images may contaminate one of the 3D objects with too many points from the other 3D object, up to a point from where the mistake is unrecoverable. However, it may also be the case that the incorrect detection is not persistent between images if, for example, the object detector can correctly resolve the two separate fruits in another image, perhaps from a different viewing angle. In the latter case, sometimes our tracker can recover from the error. When the fruits appear as separate detections, one of them may "attract" the former 3D object in the Hungarian assignment phase and the second detection gets a brand new 3D object — and thus, the final fruit count ends up being correct. Luckily, this was the case in Figure 4, where our tracker recovered the three initial fruits in the last image.

5.3.2 Ambiguities in Z depth

This case happens if two distinct fruits, when re-projected in the point of view of a particular image, end up overlapping the same detection or appear to be roughly on the same line of sight. Figure 5 illustrates one such case, where the detection in blue, corresponding to a fruit hanging on a tree, is mistakenly assigned to a fruit lying on the ground in the background, but on the same line of sight. Luckily, in Figure 5, the ambiguity only lasted 1 frame and our tracker could recover from it. However, several cases like this end up in a wrong track from that point onwards.

5.3.3 Missing depth data

When the depth maps are produced by SfM packages like COLMAP, sometimes it is not possible to estimate the depth for a particular pixel, or the confidence level of that estimation is too low. Therefore, depth maps may contain null values, i.e. missing depths for some pixels. Under some circumstances, such as fast camera motion, low frame rate, or poor distinctive textures in the image to match, the depth map may end up with large areas of missing values. If the area contained in a 2D detection has very few pixels (or no pixel at all) with estimated depth, our tracker cannot project the detection into 3D space, and hence the corresponding fruit cannot be tracked. This situation is responsible for some of the *Recall* values in Tables 2, 3 and 4 that are lower than 1.00 under perfect detections.

6. Conclusions

In this work, we propose a new object tracking algorithm that is applicable to use cases with a camera moving freely in a static scene, and we apply it to count apples in an orchard under a challenging visual context. The results show that the method is robust to intermittent occlusions and, to some extent, to imperfect detections, responding linearly to the decrease in the detection's recall, and outperforming all the other tested trackers on all the datasets. Moreover, the results also suggest that it is robust enough to be applied to diverse real-world use cases, without imposing any constraints to the camera motion nor to the frame rate, and where tree canopies may be dense and thus generate multiple intermittent occlusions of fruits. These characteristics make the solution versatile, in the sense that a farmer may use such a system with virtually any camera, either operated by hand — which generates more erratic camera motions — or mounted on an agricultural vehicle — which, while generating a smoother trajectory, usually implies a higher pace for the video, which may pose a challenge if the frame rate is low.

Acknowledgments

To INIAV [11], namely to Miguel Leão, José Silvestre and Francisco Martinho, for prividing access to their orchards, for collecting images and data. To the colleagues André Leitão, Carolina Resende and Sara Barros from SISCOG, for the help with ground truth annotations. This work was supported by LARSyS funding (DOI: 10.54499/LA/P/0083/2020, 10.54499/UIDP/50009/2020, and 10.54499/UIDB/50009/2020) and 10.54499/2022.07849.CEECIND/CP1713/CT0001, through Fundação para a Ciência e a Tecnologia, and by the SmartRetail project [PRR - C645440011-00000062], through IAPMEI - Agência para a Competitividade e Inovação.

References

[1] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. SORT. `https://github.com/abewley/sort`. Accessed: 2024-03-24. 5

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*, pages 3464–3468, 2016. 2, 5

[3] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection, 2020. 3

[4] Philipp Fischer, Alexey Dosovitskiy, Eddy Ilg, Philip Häusser, Caner Hazırbaş, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, and Thomas Brox. FlowNet: Learning Optical Flow with Convolutional Networks, 2015. arXiv:1504.06852 [cs]. 2

[5] Blender Foundation. Home of the Blender Project. https://www.blender.org/. Accessed: 2024-03-24. 5

[6] Marc Felip Pomés Francesc Net Barnes and Jordi Gené-Mola. AMIGA Fruit Counting. https://github.com/GRAP-UdL-AT/AMIGA_fruit_counting. Accessed: 2024-03-24. 5

[7] Jordi Gené-Mola, Ricardo Sanz-Cortiella, Joan R. Rosell-Polo, Josep-Ramon Morros, Javier Ruiz-Hidalgo, Verónica Vilaplana, and Eduard Gregorio. Fruit detection and 3D location using instance segmentation neural networks and structure-from-motion photogrammetry. *Computers and Electronics in Agriculture*, 169:105165, 2020. 3

[8] A. Gongal, A. Silwal, S. Amatya, M. Karkee, Q. Zhang, and K. Lewis. Apple crop-load estimation with over-the-row machine vision system. *Computers and Electronics in Agriculture*, 120:26–35, 2016. 3

[9] Leilei He, Wentai Fang, Guanao Zhao, Zhenchao Wu, Longsheng Fu, Rui Li, Yaqoob Majeed, and Jaspreet Dhupia. Fruit yield prediction and estimation in orchards: A state-of-the-art comprehensive review for both direct and indirect methods. *Computers and Electronics in Agriculture*, 195: 106812, 2022. 3

[10] Nicolai Häni, Pravakar Roy, and Volkan Isler. A Comparative Study of Fruit Detection and Counting Methods for Yield Mapping in Apple Orchards. *Journal of Field Robotics*, 37(2):263–282, 2020. arXiv:1810.09499 [cs]. 3

[11] INIAV. Instituto Nacional de Investigação Agrária e Veterinária. https://www.iniav.pt/. Accessed: 2024-03-24. 5, 8

[12] Nikita Karaev, Ignacio Rocco, Benjamin Graham, Natalia Neverova, Andrea Vedaldi, and Christian Rupprecht. CoTracker: It is Better to Track Together, 2023. arXiv:2307.07635 [cs]. 2

[13] Harold W Kuhn. The hungarian method for the assignment problem. *Naval research logistics quarterly*, 2(1-2):83–97, 1955. 4

[14] Xu Liu, Steven W. Chen, Shreyas Aditya, Nivedha Sivakumar, Sandeep Dcunha, Chao Qu, Camillo J. Taylor, Jnaneshwar Das, and Vijay Kumar. Robust Fruit Counting: Combining Deep Learning, Tracking, and Structure from Motion, 2018. arXiv:1804.00307 [cs]. 2, 3

[15] Xu Liu, Steven W. Chen, Chenhao Liu, Shreyas S. Shivakumar, Jnaneshwar Das, Camillo J. Taylor, James Underwood, and Vijay Kumar. Monocular Camera Based Fruit Counting and Mapping With Semantic Data Association. *IEEE Robotics and Automation Letters*, 4(3):2296–2303, 2019. 2

[16] ModelBunker. Deep-SORT-PyTorch. https://github.com/ModelBunker/Deep-SORT-PyTorch. Accessed: 2024-03-24. 5

[17] Tien Thanh Nguyen, Koenraad Vandevoorde, Niels Wouters, Erdal Kayacan, Josse G. De Baerdemaeker, and Wouter Saeys. Detection of red and bicoloured apples on tree with an RGB-D camera. *Biosystems Engineering*, 146:33–44, 2016. 3

[18] Addie Ira Borja Parico and Tofael Ahamed. Real Time Pear Fruit Detection and Counting Using YOLOv4 Models and Deep SORT. *Sensors*, 21(14):4803, 2021. 2

[19] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali 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. 2

[20] Pravakar Roy and Volkan Isler. Surveying apple orchards with a monocular vision system. In *2016 IEEE International Conference on Automation Science and Engineering (CASE)*, pages 916–921, Fort Worth, TX, USA, 2016. IEEE. 2

[21] Pravakar Roy, Nikolaos Stefas, Cheng Peng, Haluk Bayram, Pratap Tokekar, and Volkan Isler. Robotic Surveying of Apple Orchards. 2015. 2

[22] Johannes L. Schönberger. *Robust Methods for Accurate and Efficient 3D Modeling from Unstructured Imagery*. PhD thesis, ETH Zurich, 2018. Artwork Size: 291 p. Medium: application/pdf Pages: 291 p. 2, 3

[23] Madeleine Stein, Suchet Bargoti, and James Underwood. Image Based Mango Fruit Detection, Localisation and Yield Estimation Using Multiple View Geometry. *Sensors*, 16(11): 1915, 2016. 2

[24] Yongting Tao and Jun Zhou. Automatic apple recognition based on the fusion of color and 3D feature for robotic fruit picking. *Computers and Electronics in Agriculture*, 142: 388–396, 2017. 3

[25] Juan Villacrés, Michelle Viscaino, José Delpiano, Stavros Vougioukas, and Fernando Auat Cheein. Apple orchard production estimation using deep learning strategies: A comparison of tracking-by-detection algorithms. *Computers and Electronics in Agriculture*, 204:107513, 2023. 2

[26] Qianqian Wang, Yen-Yu Chang, Ruojin Cai, Zhengqi Li, Bharath Hariharan, Aleksander Holynski, and Noah Snavely. Tracking Everything Everywhere All at Once, 2023. arXiv:2306.05422 [cs]. 2

[27] Nicolai Wojke and Alex Bewley. Deep cosine metric learning for person re-identification. In *2018 IEEE Winter Conference on Applications of Computer Vision (WACV)*, pages 748–756. IEEE, 2018. 2, 5

[28] Nicolai Wojke, Alex Bewley, and Dietrich 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. 2, 5

[29] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. ByteTrack: Multi-Object Tracking by Associating Every Detection Box, 2022. arXiv:2110.06864 [cs]. 2, 5