

Grape Detection and Tracking Using Deep Neural Networks

Bachelor's Thesis of

Jacob Piazolo

at the Department of Mathematics
Institute for Applied and Numerical Mathematics

in cooperation with the
Fraunhofer Institute of Optonics, System Technologies and Image
Exploitation IOSB

Reviewer: PD Dr. Gudrun Thäter

Advisor: M.Sc. Benedikt Fischer

November 02, 2023 – May 02, 2024

Karlsruher Institut für Technologie
Fakultät für Mathematik
Gebäude 20.30
Englerstraße 2
76131 Karlsruhe

I declare that I have developed and written the enclosed thesis completely by myself, and have not used sources or means without declaration in the text.

Karlsruhe – April 30, 2024



.....
(Jacob Piazzolo)

Abstract

Vineyard managers traditionally count grape clusters manually for yield estimation, a process that is both time-consuming and labor-intensive. Recent advancements in computer vision and deep neural networks enable the autonomous tracking and counting of grape clusters using video footage. Filling the gap in the emerging field of grape tracking, robotic precision farming, and yield estimation, this thesis evaluates the real time tracking performance and counting accuracy of the algorithms SORT, DeepSORT, ByteTrack, SORT+ and DeepSORT+. Enhanced versions, SORT+ and DeepSORT+, incorporate a novel matching process that leverages the complementary strengths of different similarity measures, thereby increasing both tracking performance and counting accuracy.

Despite the low detection accuracy by the uniform Mask R-CNN detector, DeepSORT, SORT+ and DeepSORT+ demonstrate robust tracking and counting performances. DeepSORT+ achieves the highest tracking scores with a *MOTA* of 42.7%, closely followed by DeepSORT. SORT+ offers a viable alternative with slightly lower *MOTA* of 41.5%, which may be preferable due to the complex implementation of the classification network required by DeepSORT and DeepSORT+. This work illustrates the feasibility of using unmanned aerial vehicles (UAVs) to autonomously track and count grape clusters in challenging real-world vineyard settings. By potentially improving yield estimation and nondestructive robotic farming, these findings support sustainable farming practices and economic growth.

Zusammenfassung

Weinbauern müssen Weintrauben zur Ertragsvorhersage manuell zählen, ein Prozess, der sowohl zeit- als auch arbeitsintensiv ist. Neue Fortschritte in der Bilderkennung und den tiefen neuronalen Netzwerken ermöglichen das automatische Tracking und Zählen von Trauben in Videoaufnahmen. Diese Studie füllt eine Lücke in dem gerade entstehenden Forschungsgebiet des Traubentrackings, der robotergestützten Präzisionslandwirtschaft und der Ertragsprognose. Als Referenz für zukünftige Studien werden das Echtzeit-Tracking und die Zählgenauigkeit der Algorithmen SORT, DeepSORT, ByteTrack, SORT+ und DeepSORT+ bewertet. Die erweiterten Versionen SORT+ und DeepSORT+ nutzen einen neuen Zuordnungsalgorithmus, der die Trackingleistung und die Zählgenauigkeit durch die komplementären Stärken verschiedener Ähnlichkeitsmaße steigert.

Trotz des schwachen Mask R-CNN Detektors erreichen DeepSORT, SORT+ und DeepSORT+ relativ gute Tracking- und Zählwerte. DeepSORT+ trackt am besten mit einer *MOTA* von 42,7%, dicht gefolgt von DeepSORT. SORT+ bietet eine gute Alternative mit einem niedrigeren *MOTA*-Wert von 41,5%, da es keine komplexe Implementierung eines Klassifikationsnetzwerks benötigt, wie es bei DeepSORT und DeepSORT+ der Fall ist.

Diese Ergebnisse zeigen, dass man mit Drohnen auch Trauben in herausfordernden Weinstöcken realitätsnah automatisiert tracken und zählen kann. Durch die mögliche Anwendung in der Ertragsprognose und der robotergestützten schonenden Landwirtschaft unterstützen diese Erkenntnisse die landwirtschaftliche Nachhaltigkeit und das wirtschaftliche Wachstum.

Contents

Abstract	i
Zusammenfassung	iii
1. Introduction	1
2. Related Works	3
2.1. Literature on Fruit Tracking	4
2.1.1. Tracking of Grape-Clusters	4
2.1.2. Tracking of Various Fruits	5
2.2. Literature on Grape Yield	6
2.3. Literature on Stitching Images	7
3. Materials and Methods	9
3.1. Dataset	9
3.1.1. UAV RGB Video Dataset Specifications	10
3.1.2. UAV RGB Video Dataset Problems	11
3.2. Metrics	12
3.2.1. Detection Metrics	12
3.2.2. Tracking Metrics	14
3.3. Mask R-CNN for Detection	17
3.4. Tracking	18
3.4.1. Simple Online and Realtime Tracking (SORT)	18
3.4.2. DeepSORT	24
3.4.3. ByteTrack	26
4. Concept	29
4.1. Code Environment	29
4.2. Dataset Configuration and Data Transformation	29
4.3. Detector Configuration	31
4.4. Tracker Configurations	33
4.4.1. SORT Configuration	34
4.4.2. SORT+ Configuration	34
4.4.3. DeepSORT Configuration	35
4.4.4. DeepSORT+ Configuration	38
4.4.5. ByteTrack Configuration	39

5. Results	41
5.1. Detection Results	41
5.1.1. Optimizer Results	41
5.1.2. Mask R-CNN Results	42
5.1.3. Classification Network Results	43
5.2. Tracking Results	44
6. Discussion	47
6.1. Evaluation of Tracking Algorithms	47
6.2. Contextualizing the Results	48
6.3. Limitations and Further Research	49
7. Conclusion	53
Bibliography	55
A. Appendix	67
A.1. Proofs	67
A.1.1. S_t is symmetric and positive definite	67
A.1.2. $ S_t^{\frac{1}{2}} \vec{z} = L_t^T \vec{z} $	70
A.2. List of Abbreviations	71
A.3. Mask R-CNN Configuration Table	72

List of Figures

3.1.	Third annotated image from row 7.4.2. Red boxes: Grape clusters that are not annotated on the vine row of interest. Blue boxes: Grape clusters in the background. Orange boxes: Ground Truth data of annotated grape clusters.	11
3.2.	The image from row 8.1 is zoomed in on detected grape clusters in the background.	11
3.3.	This is an example of a Precision-Recall curve with an <i>IoU</i> -threshold of 0.5. Each point on the curve represents the Precision and Recall values for a specific confidence score (e.g., 0.56). [26]	13
3.4.	Mask R-CNN Architecture [Own Figure]	17
3.5.	Simple Online and Realtime Tracking Overview [Own Figure]	19
3.6.	Flowchart of the Hungarian Algorithm. [31]	22
3.7.	DeepSORT as an extension of SORT. [Own Figure]	24
3.8.	ByteTrack Overview [18]	27
4.1.	Learning Rate Scheduler [Own Figure]	33
4.2.	The Classification Network in DeepSORT. [Own Figure]	37
5.1.	Testing of Stochastic Gradient Descent (SGD) [5] [Own Figure]	41
5.2.	Testing of Adam [44] [Own Figure]	42
5.3.	<i>MOTA</i> scores with their corresponding 95% confidence interval. [Own Figure]	45
5.4.	<i>IDF1</i> scores with their corresponding 95% confidence interval. [Own Figure]	45
5.5.	Number of ID switches with their corresponding 95% confidence interval. [Own Figure]	45
6.1.	Three datasets presenting unique challenges, impacting tracking performance: <i>a)</i> Ninth annotated image in Row 4.2.1 in the UAV RGB Video dataset [7], <i>b)</i> Annotated image CDY_2037 showing Chardonnay from the WGSD dataset [74], <i>c)</i> Sixth annotated image from “CloseUp 2” in the Lazio dataset [77].	48
6.2.	The second annotated image in Row 7.4.2 showing “Vitis Vinifera” of the UAV RGB Video dataset [7].	50
6.3.	An annotated image showing a dataset with grape clusters very close to each other. [Internal Fraunhofer IOSB dataset]	50

List of Tables

3.1.	The Table compares datasets available for grape detection, segmentation, and tracking. “Var” denotes Varieties, reflecting the number of grape varieties included in the dataset. “IDs” indicates whether instance ID annotations are used. Datasets with “Masks” are suitable for instance segmentation applications. Datasets with “Occ” for Occlusion feature occluded or partially occluded grape clusters.	9
3.2.	UAV RGB Video Dataset Specifications	10
4.1.	For the training/test split, four vine rows were segmented into subsections and further into sub-subsections, each ranging from 250 to 2000 frames. On average, each video section includes approximately 22 annotated frames.	30
4.2.	Transformations applied to the training data of the Mask R-CNN network [38]	30
4.3.	Transformations applied to the training data of the CNN (classification network) within DeepSORT [88]	31
4.4.	Training Configurations of the Mask R-CNN Network [38]	32
4.5.	Configuration of the SORT Algorithm [12]	34
4.6.	Configurations of the SORT+ Algorithm	35
4.7.	Configurations of the DeepSORT Algorithm [88]	36
4.8.	Configuration of the Classification Network in DeepSORT [88]	37
4.9.	Configurations of the DeepSORT+ Algorithm	38
4.10.	Configurations of the ByteTrack Algorithm [97]	39
5.1.	Mask R-CNN Optimizer Testing	42
5.2.	Mask R-CNN Bounding Box Results	43
5.3.	Mask R-CNN Segmentation Results	43
5.4.	Classification Network Optimizer Testing	43
5.5.	Bootstrapped Tracking Results [60]	44
5.6.	Grape Cluster (Track) Count and Counting Accuracy	46
A.1.	Configuration of the Mask R-CNN Network [38]	73

1. Introduction

Wine not only plays an important role in culinary culture worldwide but also generates approximately 353 billion dollars in revenue annually, with an expected growth rate of 4.72% each year [87]. This market drives technical advancements aimed at optimizing grape monitoring for precision agriculture and disease prevention, as well as improving grape yield estimation to inform logistical and financial decisions for vineyards and the wine industry at large. Traditional methods for estimating grape yield, involve labor-intensive and time-consuming manual counting, which is prone to human error. Emerging technologies, such as unmanned aerial vehicles (UAVs) for fast video capturing and recent advances in computer vision, offer promising solutions to these challenges [42, 41, 83, 35, 36, 6]. Recent advances in deep learning, provide inexpensive methods for grape detection, tracking and counting, making these capabilities accessible to both small businesses and large enterprises. Beyond yield estimation, grape tracking has the potential to be utilized for sustainable robotic precision farming and harvesting, thereby offering alternatives to manual or destructive harvesting.

Tracking grape clusters using a Kernelized Correlation Filter [39] and Faster R-CNN [72] for detection, Jaramillo, Vanden Heuvel, and Petersen (2021) [41] demonstrate that automated counting outperforms manual methods and is more robust across sections with varying grape densities. While the tracking of fruit, such as mangoes and apples has been researched since the mid 2010s [81, 85], automated grape tracking only started in 2020 with the introduction of the WGISD dataset [74] by Santos et al. (2020) [75]. Until 2023, studies on grape tracking and counting primarily served as proof of concept and did not evaluate tracking performance.

Only recently, Ciarfuglia et al. (2023) [22], Ariza-Sentís et al. (2023) [8] and Saraceni et al. (2023) [77] began to evaluate their tracking performance. To date, there has been no comparison of tracking algorithms, evaluating their tracking performance and counting accuracy. The exception is Saraceni et al. (2023) [77], who compare the tracking performance of multiple algorithms, not all of which were fine-tuned for grape tracking.

Which is the best tracking algorithm to achieve high tracking performance and counting accuracy?

This thesis expands the current research by implementing and rigorously evaluating five tracking algorithms, each fine-tuned for the [UAV RGB Video dataset](#) from Ariza-Sentís, Vélez, and Valente (2023) [7]. This work aims to serve as a reference point for the emerging field of grape tracking, counting and yield estimation by making the following contributions:

1. Proposing an enhanced matching process for the trackers [SORT](#) [12] and [DeepSORT](#) [88], which are referred to as [SORT+](#) and [DeepSORT+](#).
2. Presenting a comprehensive evaluation of the tracking performance of [SORT](#) [12], [DeepSORT](#) [88], [ByteTrack](#) [97], [SORT+](#) and [DeepSORT+](#), employing the uniform [Mask R-CNN](#) network [38] and utilizing bootstrapping [60].
3. Evaluating the counting accuracies of the tracking algorithms to determine their suitability for grape yield estimation.

Structure Overview

This thesis includes seven chapters. This introduction chapter motivates the study of grape tracking and outlines the purpose of this research. Chapter 2, “Related Works” familiarizes the reader with the field of fruit and especially grape tracking. It also discusses image stitching as an alternative method to grape tracking for yield estimation.

Chapter 3, “Materials and Methods” explains the dataset ([UAV RGB Video dataset](#) [7]), the metrics, the detector ([Mask R-CNN](#) [38]) and the tracking algorithms ([SORT](#) [12], [DeepSORT](#) [88] and [ByteTrack](#) [97]) used in this study. The “[Concept](#)” chapter specifies the exact configuration of the dataset, networks, and algorithms. It also introduces the enhanced tracking algorithms [SORT+](#) and [DeepSORT+](#).

Without evaluating the findings, chapter 5, “Results” presents the detection results from [Mask R-CNN](#) [38]), along with the tracking performance and counting accuracy of [SORT](#) [12], [DeepSORT](#) [88], [ByteTrack](#) [97], [SORT+](#) and [DeepSORT+](#). In chapter 6, “Discussion” these results are evaluated and contextualized within previous research. The [discussion chapter](#) also highlights limitations to this work and suggests how further research can address these challenges.

The “[Conclusion](#)” chapter summarizes the thesis, examining the results within the specific context of the research goals of this thesis. The “[Appendix](#)” includes a list of abbreviations, Table A.1, specifying the [Mask R-CNN](#) [38] configuration and two mathematical proofs that validate the theorems used in this study.

2. Related Works

This chapter discusses literature related to grape tracking, covering various aspects of this research. Initially, it outlines general topics associated to grape tracking, followed by a detailed description of studies closely related to this work. Given that [grape yield estimation](#) serves as a fundamental motivation for grape tracking, a dedicated [subsection](#) explores important research on this topic. Afterward, [image stitching](#) is compared to grape tracking as an alternative method for counting grape clusters and thus grape yield estimation.

It should be noted that many concepts and abbreviations briefly introduced in this chapter are clarified later in the text. By clicking on the word, a hyperlink will direct the reader to the relevant section for an in-depth explanation.

Tracking, as a fundamental component of computer vision, involves the continuous observation of objects across a series of frames or images to understand their movements or state changes over time. It has broad applications, from surveillance and security to robotic vision and beyond. The origin of tracking in computer vision dates back to the early experiments in the 1960s, aimed at enabling machines to interpret visual data similarly to how humans do. With advancements in computational power and algorithmic sophistication, tracking has evolved significantly, incorporating techniques like optical flow, [Kalman filtering](#) [86], and more recently, deep learning approaches for enhanced accuracy and robustness in various environments.

In contrast, the subset of fruit tracking, and specifically grape tracking, presents unique challenges different from tracking in general. Fruit tracking involves not only detecting and re-identifying multiple fruit across sequences, but also dealing with complexities caused by natural growth. Factors such as variable lighting conditions, ripeness of fruit, occlusions caused by foliage, and the variety of grapes require specialized approaches.

Research in computer vision concerning grapevines predominantly focuses on the detection of grape clusters. Detection, a cornerstone within computer vision, is studied on images, where the entire canopy is visible [55, 49, 3, 10, 59, 24], as well as on close-up images targeting single grape clusters [69, 95, 54, 49, 68, 32, 35]. Besides identifying grape clusters, researchers work on the detection of individual grape berries from varying distances [14, 67, 28, 21, 94, 66, 54, 23, 33, 93, 6, 63].

For robotic applications and yield prediction, a few studies investigate the detection and tracking of grapes. However, most papers only superficially evaluate tracking due to inadequate datasets [79, 9, 41, 75]. Notably, all papers evaluating grape tracking in video-frames were published in 2023 [77, 8, 22], with two out of three published in the second half of the year. This work ties in with the current research on grape tracking that is

still in its early stages. Other than grapes, studies focus on the detection and counting of inflorescence to extrapolate the grape yield [42, 84, 73].

2.1. Literature on Fruit Tracking

2.1.1. Tracking of Grape-Clusters

Several [datasets](#) annotated with bounding boxes (bboxes) and masks are available, while Ariza-Sentís, Vélez, and Valente (2023) [7] and Saraceni et al. (2023) [77] provide the only publicly accessible [datasets](#) that assign instance IDs to grape clusters. Therefore, in the absence of instance ID annotations, many researchers have relied on mathematical motion predictions, such as [Simple-Online-Realtime-Tracking \(SORT\)](#) [12] or geometric similarities, for re-identification.

Santos et al. (2020) [75] for example, compare the detection results of a [Mask R-CNN](#) [38], YOLOv2 [71] and a YOLOv3 network [70]. [Mask R-CNN](#) [38] outperforms the others with a [mean Average Precision \(mAP\)](#) of up to 81%. Santos et al. (2020) [75] are the first to demonstrate grape tracking using Structure-from-Motion (SfM) [37]. SfM [37] extracts the three-dimensional structure of a scene using video frames. The two instances (grape clusters) from different frames that exhibit the highest number of connections to the same 3D points are assigned identical instance IDs. However, SfM [37] is not suitable for real time tracking due to its computational time requirements.

Shen et al. (2023) [79], Jaramillo, Vanden Heuvel, and Petersen (2021) [41] employ mathematical algorithms to predict the motion of the grape clusters and match them with detections in real time. Shen et al. (2023) [79] track instances (grape clusters) with [SORT](#) [12]. [SORT](#) [12] consists of a [Kalman Filter](#) [86] to predict the position of each grape cluster in the subsequent frame. The [Hungarian Algorithm](#) [48] matches predictions with detections by minimizing a cost matrix based on the largest [Intersection over Union \(IoU/Jaccard Index\)](#). Shen et al. (2023) [79] evaluate the tracking by comparing the number of tracks to the number of manually counted grape clusters, resulting in 84.9% counting accuracy.

In Jaramillo, Vanden Heuvel, and Petersen (2021) [41], a Kernelized Correlation Filter [39] correlates the appearance features of an instance (grape cluster) with the next frame, predicting its position. Afterward, predictions are matched with detections using an [Hungarian Algorithm](#) [48].

Arlotta, Lippi, and Gasparri (2023) [9] simulate a virtual vineyard instead of using a dataset to test tracking. In their study, a virtual robot is tasked to track and harvest grapes using an Extended Kalman Filter [86].

Ciarfuglia et al. (2023) [22] found that Structure-from-Motion (SfM) [37] yields better results than [DeepSORT](#) [88]. [DeepSORT](#) [88] extends [SORT](#) [12] by incorporating a re-identification network that compares appearance features. [DeepSORT](#) [88] is suitable for real time application. The Author points out that SfM [37] required approximately 5 hours to compute and limits the video to a few hundred frames.

On the 28th of September 2023 Saraceni et al. (2023) [77] published a paper that com-

compares the tracking results of SORT [12], BoTSORT [4], OC-SORT [19], ByteTrack [97], StrongSORT [29] and a new tracker named AgriSORT [77]. The Dataset, which has been published March 2024, consists of two sequences labeled as “CloseUp” and two as “Overview”. The “CloseUp” sequences exhibit irregular motion, whereas the “Overview” sequences are of consistent movement. OC-SORT [19] and BoTSORT [4] show poor performance. Both BoTSORT [4] and StrongSORT [29] employ a re-identification network that is not fine-tuned for grape applications. SORT [12] and AgriSORT [77] perform similarly on the “Overview” sequences. StrongSORT [29], an improvement of DeepSORT [88] and the self-designed tracker, AgriSORT [77] perform better on the “CloseUp” sequences with AgriSORT [77] outperforming the other trackers in almost every metric. AgriSORT [77] assumes that only the camera moves, while the instances remain stationary. To predict camera movement, a Kalman Filter [86] first predicts the motion of the grape clusters and then “Lucas-Kanade sparse optical flow correspondences” [56] estimates the camera’s motion. This estimated camera motion is then utilized to predict the coordinates of the grape bounding boxes, keeping height and width the same.

Ariza-Sentís et al. (2023) [8] evaluate tracking results using their public dataset [7] collected with an unmanned aerial vehicle (UAV). The authors implement five variants of PointTrack [91] designed to output segmentation masks and instance IDs for detected grape clusters. PointTrack [91] converts instances into point clouds, which it compares to previous objects for instance re-identification. However, the dataset [7] exhibits incomplete annotations, with not every grape cluster being annotated. Furthermore, the area of individual grape clusters ranges from 0.01% to 0.17% of the frame. Sometimes only a single grape cluster gets detected instead of two or three in close proximity. This causes many false positives (FP) and some false negatives (FN), thereby leading to a negative Multi Object Tracking and Segmentation (MOTSA) score. Ariza-Sentís et al. (2023) [8] reach a MOTSA score of -8.2%.

2.1.2. Tracking of Various Fruits

Tracking of fruits other than grapes has been underway since the mid-2010s, whereas research specifically targeting grape tracking emerged in 2020 with the paper published by Santos et al. (2020) [75].

Stein, Bargoti, and Underwood (2016) [81] employ multi-object-tracking to solve the problem of occlusion in mango trees, aiming to accurately estimate yield. To track mangoes in real time, Wang, Walsh, and Koirala (2019) [85] implement SORT [12], achieving mango counting with 2.6% over-count.

Grape species exhibit considerable variation in color and shape. This makes it difficult applying neural networks trained on one variety to another. Bellocchio et al. (2020) [11] address this issue by developing a method to translate datasets, such as converting an olive dataset into an almond dataset. They utilize a Cycle Generative Adversarial Network [98] (C-GAN) to transform fruits from an annotated source dataset to those of a target dataset. Through this process, Bellocchio et al. (2020) [11] manage to train a network capable of detecting and counting apples, olives, and almonds without depending on fruit-specific annotated datasets. Such an approach could be useful in adapting tracking methods from

one grape variety to other varieties or species.

Li, Bao, and Qi (2022) [50] employ an UAV to capture video footage of maize seedlings. Furthermore, the authors prove the ability of SORT [12] to accurately count the seedlings. Instead of rectangular bounding boxes, Liu et al. (2020) [53] use circular bounding boxes for tomatoes to improve the *IoU-overlap* (Jaccard Index) calculation. This can help with Non-Maximum-Suppression (NMS) [64] and matching in multi-object-trackers. Similarly, the use of trapezoid or triangle bounding boxes for grapes could potentially improve tracking accuracy.

Fang et al. (2022) [30] leverage the easily detectable trunks of apple trees as a reference point to calculate movement and subsequently predict the positions of apples in the following frame. This approach achieves significantly better counting accuracy compared to DeepSORT [88], which overcounts the apples. Researchers could similarly install markers in vine rows to adapt this approach for grape tracking.

To ensure accurate yield predictions, it is crucial not to track grapes detected in the background. Wu et al. (2023) [90] use a depth camera and confidence scores to exclude apples located on trees in the background or apples that have fallen to the ground. This removal increases detection results by up to 14% and improves *MOTA* and *MOTP* scores by 4.2% and 3.3% respectively.

2.2. Literature on Grape Yield

Accurately predicting yield depends on several variables. Grapes growing in the background, not on the vine-row of interest, should be excluded from counts. Wu et al. (2023) [90] address this problem in apple orchards by utilizing a depth camera. Alternatively, some researchers collect their dataset at night using flashlights [65, 41, 6, 45, 85], ensuring that the camera captures only the targeted vine-row against a black background.

Another variable for yield estimation is the maturity stage of the studied grape variety. According to Hacking, Poona, and Poblete-Echeverria (2020) [35], the ideal stage for yield estimation occurs during the final stages of berry ripening, specifically during sugar development. For red grape species, the stage of color development can significantly reduce yield estimates due to the differences in grape berry colors, with some remaining green while others turn red. Nonetheless, Khokher et al. (2023) [42] (2023) manage to predict yield with an average error of 10% by counting inflorescence. Several months before the final stages of berry ripening, the early counting of inflorescence could be beneficial.

Research conducted by Olenskyj et al. (2022) [65], Victorino et al. (2019) [83], Lopes and Cadima (2021) [54] explores which attributes most closely correlate with grapevine yield. Lopes and Cadima (2021) [54] demonstrate that, during the final stages of berry ripening, the area of grape clusters is a more accurate predictor of yield than the total number of berries counted. This finding is supported by Victorino et al. (2019) [83] and Olenskyj et al. (2022) [65], where the grape cluster area show a stronger correlation with yield compared to the number of grape clusters. Consequently, a Mask R-CNN network [38] was chosen for detection, due to its capability to output the areas of grape clusters. However, caution is advised when applying findings from one grape species to another, as Victorino et al.

(2019) [83], Lopes and Cadima (2021) [54] observe differences in results depending on the grape species.

2.3. Literature on Stitching Images

To count grape clusters, berries, or inflorescence an alternative method to tracking involves stitching individual frames into a panorama. This technique [17] allows for the detection of instances without the risk of double counting [42, 94, 6].

In 2018, Aquino et al. (2018) [6] were the first to use image stitching [17] counting grape berries by utilizing the 'Auto-Blend' feature in Adobe Photoshop CC 2015 (Adobe Systems Incorporated; San José, California, USA). Although the yield detected (in kg, estimated from the number of grape berries detected) is less than half of the actual yield (kg), a linear correlation between the detected and actual yield has been found. This way, Aquino et al. (2018) [6] are able to predict the yield of 90 vines with a mean error of 12.8%.

In 2023, Khokher et al. (2023) [42] tried image stitching [17] for counting inflorescence, but found tracking methods to be more promising. The authors observe that some inflorescence present in the video disappear in the stitched panorama. This can be improved using reviewed techniques in Lyu et al. (2019) [57]. Nonetheless, Khokher et al. (2023) [42] determine a false positive (FP) ratio ($\frac{FP}{TP}$) of 38% in the stitched panoramas, compared to 24% in the video sequences. The authors suggest that FPs can be filtered out during the tracking process, because it is unlikely for the same FPs to be detected across consecutive frames. In the stitched panorama every FP is counted.

Zabawa et al. (2022) [94] employ a basic form of image stitching for counting grape berries. The authors utilize GNSS coordinates to take images, which edges should ideally touch, but not overlap. With perfect precision every grape would be captured exactly once. Zabawa et al. (2022) [94] achieve a mean average error (MAE) of up to 8% counting berries and a MAE of 26% in yield calculation, primarily due to occluded grapes.

Research on the application of image stitching [17] for yield estimation has been relatively scarce. However, Aquino et al. (2018) [6] and Zabawa et al. (2022) [94] have demonstrated that image stitching [17] can serve as an alternative to tracking for counting grape clusters, berries, or inflorescence. Image stitching [17] could be beneficial in scenarios where only a few frames per instance are available, as significant motion between frames can challenge trackers. Conversely, tracking can be used for counting grapes as well as in robots for fruit harvesting. Both tracking and image stitching [17] are theoretically viable in real time applications, as there are studies on real time image stitching in Yeh and Lai (2017) [92]. However, tracking may be more advantageous, as most trackers maintain a consistent workload, whereas the computational demand for image stitching [17] increases with the number of frames.

Khokher et al. (2023) [42] achieve better results in yield estimation through tracking compared to image stitching [17]. They point out that tracking can minimize the error caused by false positives (FP s). Considering the greater flexibility of tracking across various use cases and its capabilities in solving counting problems, this thesis studies tracking, instead of image stitching [17] for counting grape clusters.

3. Materials and Methods

3.1. Dataset

Machine learning datasets are usually split into training, validation, and testing subsets. The training iteratively fine-tunes the neuronal network to detect grape clusters within its data. After each training epoch, the network’s performance is evaluated on the validation set using [mean Average Precision \(mAP\)](#). Overfitting occurs when the network gets too good at recognizing grape clusters from the training set, impacting its performance on unseen data, such as the validation set. Therefore, the epoch was chosen, where the network is most effective at identifying grape clusters in the unseen validation data. The neural network can now be used in different scenarios, including grape cluster tracking on the test set.

A re-identification network classifies which grape clusters are the same (have the same instance ID) by comparing detected grapes across frames. [DeepSORT](#) [88] utilizes a re-identification network for this purpose. Training such a network requires grape clusters in the training subset to be annotated with instance IDs. While trackers like [SORT](#) [12] or Structure from Motion (SfM) [37] do not require a re-identification network, evaluating their tracking performance necessitates instance ID labels in the test subset. To evaluate tracking on the WGISD dataset [74], Ciarfuglia et al. (2023) [22] label only a small test dataset with instance IDs.

Dataset	Labelled	Vehicle	Var	IDs	Masks	Occ	MOT(S)A
UAV RGB [7]	681	UAV	1	x	x	x	-8.2%
WGISD [74]	300	foot	5		x	x	46.7%
WSU Full Stages [96]	459	foot	2			x	
AI4Agriculture [61]	250	foot	1			x	
GrapeCS-ML [78]	2078	foot	15				
GrapeUNIPD-DL [80]	271	foot	4			x	
CERTH [13]	2502	foot	1		x	x	
Lazio [77]	800	foot	1	x		x	66.1%

Table 3.1.: The Table compares datasets available for grape detection, segmentation, and tracking. “Var” denotes Varieties, reflecting the number of grape varieties included in the dataset. “IDs” indicates whether instance ID annotations are used. Datasets with “Masks” are suitable for instance segmentation applications. Datasets with “Occ” for Occlusion feature occluded or partially occluded grape clusters.

At the start of my bachelor thesis the 'UAV RGB Video Dataset', published by Ariza-Sentís, Vélez, and Valente (2023) [7] was the only available dataset offering instance IDs for every annotated grape cluster. Consequently, it was the only viable choice for training the re-identification network from DeepSORT [88] and to evaluate tracking. On March 6, 2024, Leonardo Saraceni published an alternative grape dataset from southern Lazio, also including instance IDs.

Using this new dataset Saraceni et al. (2023) [77] achieved [Multi Object Tracking Accuracy \(MOTA\)](#) scores ranging from 50% to 66% depending on the video sequences tested. The Lazio Dataset [77] was collected on foot, capturing the lower canopy area where most grape clusters are found. These grape clusters, primarily red, cover 0.3% to 20% of the frame's area. Because the camera does not capture the entire vine row, the dataset of Saraceni et al. (2023) [77] is best suited for applications in robotic detection and tracking, rather than for estimating grape yield. For segmenting and counting grape clusters the UAV RGB Video Dataset, published by Ariza-Sentís, Vélez, and Valente (2023) [7] continues to be the most suitable dataset available.

3.1.1. UAV RGB Video Dataset Specifications

Source	Dataset on UAV RGB videos acquired over a vineyard including bunch labels for object detection and tracking [7]
UAV	DJI Matrice 210 RTK
Flight Speed	0.7 m/s
Flight altitude	3 m above ground level
Frame Width	4096
Frame Height	2160
Frame Rate	59.94 frames/s
Annotated Frames	664
Train Split	528
Test Split	136
Data Collection	"The four flights were executed on June 28th, 2021 over four rows of the vineyards. The flights were carried out on a sunny day with wind velocity lower than 0.5 m/s. The four rows were selected according to the ripening stage of the grape clusters, to have a representative sample of the development status over the four rows. The grape bunch annotations were labelled using the CVAT software and are available in the MOTS style." [7]
Vineyard Location	41°57'18.3"N 8°47'41.9"W Tomiño, Spain

Table 3.2.: UAV RGB Video Dataset Specifications

For the UAV RGB Video Dataset [7] a drone captures the whole vine, therefore the green grape clusters take up merely 0.01% to 0.17% of the frame area. Furthermore, a dense canopy with significant occlusion makes detection a challenging task. Since no pruning

nor leave removal was performed, this dataset reflects real world conditions for fast data collection.

3.1.2. UAV RGB Video Dataset Problems

The UAV RGB Video dataset [7] faces two major issues that hinder detection and consequently tracking, as well as a third problem that presents difficulties specifically for grape yield estimation.

1. The areas of the grape clusters are very small, ranging from 0.01% to 0.17% of the frame.
2. Some grape clusters were not identified and thus not annotated. Refer to Figure 3.1. This negatively affects the detector's performance and, by extension, the tracker's accuracy. Moreover, the missing annotations cause false positives that should have been true positives, substantially lowering the *MOTA* and *MOTSA* scores.
3. Grape clusters present on vine rows in the background (Refer to Figure 3.2) are incorrectly detected as false negatives. This not only further reduces the *MOTA* and *MOTSA* scores, but also falsely inflates yield calculations.



Figure 3.1.: Third annotated image from row 7.4.2. Red boxes: Grape clusters that are not annotated on the vine row of interest. Blue boxes: Grape clusters in the background. Orange boxes: Ground Truth data of annotated grape clusters.



Figure 3.2.: The image from row 8.1 is zoomed in on detected grape clusters in the background.


3.2. Metrics

Up arrows (↑) indicate that higher values represent better performance, whereas down arrows (↓) suggest that lower scores are preferred.

3.2.1. Detection Metrics

↑ **Intersection over Union (IoU):**

The Intersection over Union, also known as Jaccard-Index, measures the overlap between two bounding boxes, typically between a detection and a ground truth. If the *IoU* exceeds a specified threshold (e.g., 0.5), the prediction is classified as a true positive (*TP*) or otherwise as a false positive (*FP*).

$$IoU = \frac{\text{Area of Overlap}}{\text{Area of Union}} = \frac{\text{Area of Overlap}}{\text{Area of Union}} \quad (3.1)$$


↑ **Precision:**

Precision is the model's accuracy in identifying *TPs* among all detections it makes.

$$Precision = \frac{TP}{TP + FP} \quad (3.2)$$

↑ **Recall (Sensitivity):**

Recall is the percentage of detected *TPs* out of the total number of *TPs* that were possible to detect.

$$Recall = \frac{TP}{TP + FN} \quad (3.3)$$

Precision-Recall Curve:

An object detection model generates each detection with a confidence score. Detections are retained if their confidence score surpasses a specified threshold; otherwise, they are discarded. Employing a certain *IoU*-threshold, such as 0.5, calculate [Precision](#) and [Recall](#) for confidence scores from 0.00 to 1.00 in increments of 0.01 ([0.00, 0.01, 0.02...0.98, 0.99, 1.00]). Graph these calculations by plotting [Precision](#) on the y-axis and [Recall](#) on the x-axis.

A Precision-Recall Curve can also be created for a [classification model](#) that extracts appearance features from images in the form of feature vectors. A distance matrix can be computed by calculating the pairwise Euclidean distance between appearance feature vectors of all grape cluster detections, where smaller distances signify greater similarity. For each grape cluster, other detections are ranked by proximity. The closest k% are classified as the same grape cluster and can now be identified as *TPs* or *FPs*. Other true matches with ground truths of the query grape clusters with a distance outside the top k% are classified as *FNs*.

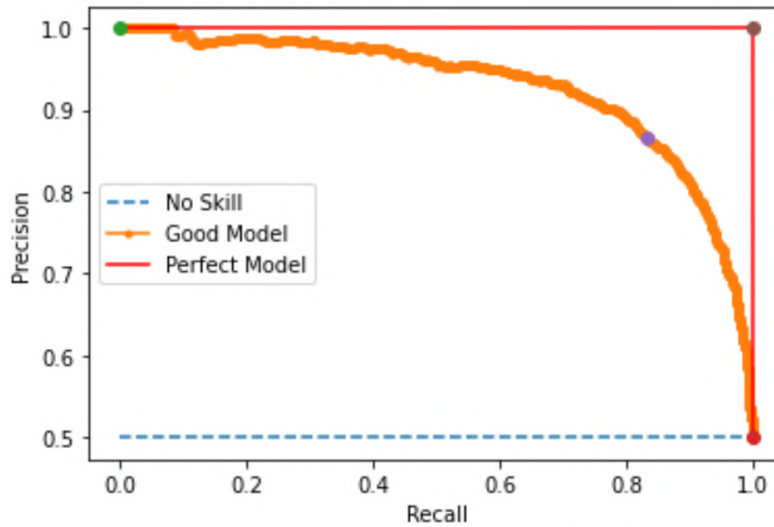


Figure 3.3.:

This is an example of a Precision-Recall curve with an *IoU*-threshold of 0.5. Each point on the curve represents the **Precision** and **Recall** values for a specific confidence score (e.g., 0.56). [26]

By calculating the **Precisions** and **Recalls** for the top $k\%$ closest feature vectors from 1% to 100% in 1% increments, a Precision-Recall Curve can be plotted.

↑ **mean Average Precision (mAP):**

The mean Average Precision for a certain *IoU*-threshold, such as 0.5, equals the area under the **Precision-Recall curve**. This area can be calculated using the trapezoidal rule.

$$mAP_{50} = mAP_{0.50} = \frac{1}{200} \sum_{i=1}^{100} (Precision(Recall_{\frac{i-1}{100}}) + Precision(Recall_{\frac{i}{100}})) \quad (3.4)$$

Traditionally in object detection, mAP refers to the average of the mAP s calculated at the *IoU*-thresholds ranging from 0.50 to 0.95 in increments of 0.05 ([0.50, 0.55, 0.60...0.90, 0.95]). The average mAP usually serves to identify the optimal checkpoint (weights at a certain iteration/epoch) for a model.

$$mAP = \frac{1}{10} \sum_{i=0}^9 mAP_{0.50+0.05i} \quad (3.5)$$

To calculate mAP_{small} , mAP_{medium} and mAP_{large} categorize bounding boxes as small, medium, or large and calculate the mAP of each category. The size classifications follow the COCO standard classifications [52], with dimensions expressed in pixels (px):

$$small < 32^2px = 1024px \leq medium < 96^2px = 9216px \leq large \quad (3.6)$$

The mAP can similarly be calculated for masks using the same methodology.

The mAP for a **classification network** that outputs feature vectors is the area under the unique **Precision-Recall curve**, which can also be approximated using the trapezoidal rule.

3.2.2. Tracking Metrics

Due to high occlusion and the small size of grape clusters in the [UAV RGB Video dataset](#) [7], the resulting detector often detects only parts of a grape cluster. Furthermore, the grape clusters are rarely in close proximity of each other, allowing for the matching of track detections with ground truth tracks even when the *IoU*-overlap is low. Consequently, a minimum threshold of at least 20% *IoU*-overlap between a track detection and a ground truth (*GT*) was chosen to classify a track detection as true positive (*TP*). The [Hungarian Algorithm](#) [48] makes an optimal one-to-one assignment by maximizing the average *IoU*-overlap. If matches exhibit less than 20% *IoU*-overlap, they are unmatched, making the detection a false positive (*FP*) and the ground truth(*GT*) a false negative (*FN*). The 20% *IoU*-threshold strikes a good balance between accurately assigning correct detections to *GT*s and minimizing the number of ID switches due to incorrect assignment.

↓ ID switches (*IDsw*):

IDsw counts the number of times when the tracker incorrectly switches the identity of a grape cluster from one instance ID to another.

↑ Multi Object Tracking Accuracy (*MOTA*):

MOTA is one of the most important metrics to evaluate tracking performance. However, *MOTA* is influenced by the performance of the detector. Poor detection results cause poor *MOTA* scores. Multi Object Tracking and Segmentation Accuracy (*MOTSA*) closely resembles *MOTA*, but also incorporates segmentation masks to calculate *TP*s, *FP*s and *FN*s.

$$MOTA = 1 - \frac{|FP| + |FN| + |IDsw|}{|GT|} = \frac{|TP| - |FP| - |IDsw|}{|GT|} \quad (3.7)$$

↑ Multi Object Tracking Precision (*MOTP*):

MOTP assesses how precisely the locations of the tracked objects are estimated. This happens independently of how accurately instance IDs are assigned to tracks.

$$MOTP = \frac{\sum_t \sum_{i=1}^{c_t} u_{t,i}}{\sum_t c_t} \quad (3.8)$$

In frame t , c_t represents the number of matches, including correct matches and matches after an ID switch. The *IoU*-overlap $u_{t,i} = IoU(Pred_i, GT_i)$ measures how much prediction i ($Pred_i$) and ground truth (GT_i) overlap. The closer *MOTP* is to 1, the more accurate the Multi Object Tracking Precision is. Multi Object Tracking and Segmentation Precision (*MOTSP*) is very similar to *MOTP*, but it also includes segmentation model's masks to calculate the *IoU*-overlap between the masks and their corresponding ground truths (*GT*s).

↓ False Negative Identities (*IDFN*) and ↓ False Positive Identities (*IDFP*):

IDFN represents the number of ground truth detections that were either overlooked by the tracker or assigned incorrectly identities.

$$IDFN = \sum_{\tau \in AT} \sum_{t \in T_\tau} m(\tau, \gamma_m(\tau), t) \quad (3.9)$$

IDFP refers to the number of detections that were either absent in the ground truth or had incorrect identities attributed to them.

$$IDFP = \sum_{\gamma \in AC} \sum_{t \in T_\gamma} m(\tau_m(\gamma), \gamma, t) \quad (3.10)$$

τ : Represents a true trajectory; γ : Represents a computed trajectory; AT : The set of all true trajectories; AC : The set of all computed trajectories; $t \in T_\tau$: A specific frame t within the set of frames T , where the trajectory τ exists; $m(\tau, \gamma, t)$: The number of missed detections in true trajectories τ and computed trajectories γ in frame t ; $\gamma_m(\tau)$: The true trajectory τ that is matched with a computed trajectory; $\tau_m(\gamma)$: The computed trajectory γ that is matched with a true trajectory; $|| \cdot ||$: The count of detections in a given trajectory.

↑ **True Positive Identities (*IDTP*):**

IDTP equals the number of detections accurately matched with the correct object identity.

$$IDTP = \sum_{\tau \in AT} ||\tau|| - IDFN = \sum_{\gamma \in AC} ||\gamma|| - IDFP \quad (3.11)$$

↑ **Identification Precision (*IDP*):**

Identity Precision measures the proportion of correctly identified detections over all computed detections made by the tracker.

$$IDP = \frac{IDTP}{IDTP + IDFP} \quad (3.12)$$

↑ **Identification Recall (*IDR*):**

Identity Recall measures the proportion of correctly identified detections over all *GT* detections.

$$IDR = \frac{IDTP}{IDTP + IDFN} \quad (3.13)$$

↑ **Identification F1 Score (*IDF1*):**

IDF1 is a metric that assesses an algorithm's capability to accurately identify and maintain the instance IDs of objects over time. It offers a single measure that reflects both the accuracy and the robustness of the tracking in maintaining consistent identities.

$$IDF1 = \frac{2 \cdot IDP \cdot IDR}{IDP + IDR} = \frac{2 \cdot IDTP}{2 \cdot IDTP + IDFP + IDFN} \quad (3.14)$$

↑ **Mostly Tracked (*MT*):**

A grape clusters is considered mostly tracked (*MT*) when it is detected by the detector for at least 80% of its ground truth lifespan. The ground truth lifespan refers to all frames in which the grape cluster is both visible and annotated. The Metric *MT* counts the number of annotated grape clusters that are mostly tracked.

Partially Tracked (*PT*):

A grape clusters is classified as partially tracked (*PT*) if detected in more than 20%, but

fewer than 80% of its annotated ground truth frames. The Metric *PT* counts the number of annotated grape clusters that are partially tracked.

↓ **Mostly Lost (*ML*):**

A grape cluster is considered mostly lost (*ML*) when it is detected in less than 20% of its ground truth lifespan. The Metric *ML* counts the number of mostly lost annotated grape clusters.

The values of *MT*, *PT* and *ML* add up to the total number of annotated grapes in the tested dataset.

↓ **Fragmentations (*FM*):**

The metric Fragmentations (*FM*) measures the number of interruptions within all ground truth tracks. An interruption occurs through the detector's failure to identify a grape cluster in an annotated frame. Only after the detector has again identified a corresponding detection, another interruption can occur.

It is important to note that the metrics *MT*, *PT*, *ML* and *MOTP* solely depend upon the detector, the set confidence threshold that dictates which detections are retained and the *IoU*-threshold that defines *TPs*, *FPs* and *FNs*. These metrics do not consider the tracking accuracy concerning the correct identification of instance IDs.

↑ **Counting Accuracy**

Counting accuracy is a metric that evaluates the precision with which a tracking algorithm counts the number of grape clusters in a dataset.

$$\text{Counting Accuracy} = 1 - \left| \frac{AG - CT}{AG} \right| \quad (3.15)$$

AG represents the number of annotated grape clusters in the dataset, while *CT* denotes the number of grape clusters counted by the tracking algorithm.

3.3. Mask R-CNN for Detection

A Mask R-CNN network, as proposed by He et al. (2018) [38] extends the functionalities of Faster R-CNN [72] by introducing an additional branch that predicts segmentation masks for each Region of Interest (RoI). This enables the model to perform both object detection - identifying bounding boxes around objects - and instance segmentation - generating pixel-level masks for each object.

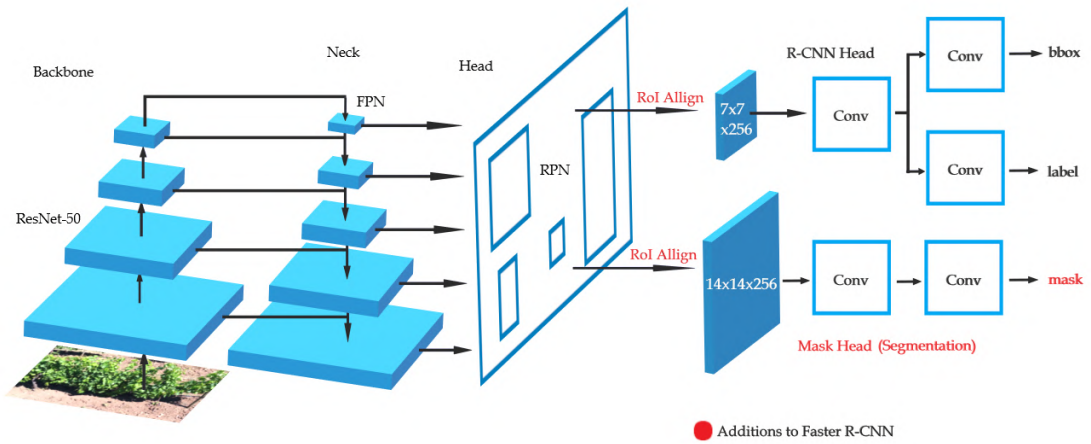


Figure 3.4.: Mask R-CNN Architecture [Own Figure]

The Mask R-CNN [38] framework is build from several components. Each component is customizable or interchangeable to best fit the task at hand. For details on configurations, the reader is referred to Bouraya and Belangour (2021) [15]. The following Mask R-CNN model [38] describes one out of many variations:

- **Backbone:** The backbone extracts features from the input image. Here, a ResNet-50 model [47] was chosen out of several options and pretrained on the COCO dataset [52]. The Common Objects in Context (COCO) dataset is popular for its extensive annotations covering a wide array of common objects, enabling the extraction of basic features. This backbone is divided into four stages, with the initial stage being frozen to preserve pre-learned features. Each stage halves the spatial resolution, while doubling the number of channels (e.g., $1088 \times 2048 \times 256 \rightarrow 544 \times 1024 \times 512$).
- **Neck:** The Feature Pyramid Network (FPN) [51], which is also used by He et al. (2018) [38] is a popular choice for the neck. The FPN uses the hierarchical feature maps of all 4 stages with different depths from the backbone. By integrating these feature maps, which vary in size (forming a pyramid), the FPN captures both high-level semantic information at coarser resolutions and detailed features at finer resolutions. The FPN outputs feature maps of different spatial resolution, but with the same number of channels.

- **Region Proposal Network (RPN) Head:** In conjunction with the Feature Pyramid Network (FPN) [51], the Region Proposal Network (RPN) identifies image areas with high object (e.g., grape cluster) presence probability. The RPN proposes regions that can contain objects (grape clusters) of varying sizes and scales.
- **Region of Interest (RoI) Head:** RoI Align [38], introduced by He et al. (2018) [38] alongside Mask R-CNN, precisely extracts fixed-size feature maps (e.g., 7×7) from variable-sized regions of interest. Given that a single tile in the feature map may not correspond to an integer number of pixels in the region of interest, RoI Align employs bilinear interpolation to calculate exact values at floating point locations. This method ensures the accurate alignment of the information with the objects (grape clusters) in the image, a necessary requirement for segmentation.
- **Bounding Box (bbox) Head:** The bbox head, which includes multiple convolutional layers, classifies each RoI. It generates a confidence score for the presence of an object (grape cluster) within the RoI and the bounding box surrounding the object.
- **Mask Head:** Similarly to the bbox head, the mask head consists of multiple convolutional layers that predict a fixed-size segmentation mask for each RoI. The masks are then resized to overlay the RoI, matching the segmented object in the original image.
- **Loss:** The loss $L = L_{cls} + L_{bbox} + L_{mask}$ combines classification (L_{cls}), the bbox (L_{bbox}) and mask (L_{mask}) losses. L evaluates the model's performance in classification, localization, and instance segmentation. Through optimization techniques such as the Adam optimizer [44] or stochastic gradient descent [5], the model progressively reduces this loss, thereby learning to detect and segment objects more effectively.

By adjusting these components, Mask R-CNN [38] can be finely tuned for computer vision applications, including detection and segmentation of grape clusters.

3.4. Tracking

This study examines two-step multi object tracking approaches, separating the process into distinct detection and tracking stages. As a result, detection functions independently of tracking. This modular framework allows easy substitution of either the detector or the tracker. This section introduces three algorithms for the tracking step.

3.4.1. Simple Online and Realtime Tracking (SORT)

The multi object tracking algorithm SORT, as introduced by Bewley et al. (2016) [12], is designed for speed and efficiency, making it well-suited for real-time applications. The mathematical tracker does not depend on training data, which allows the application of SORT [12] on [datasets](#) lacking instance IDs in the training data.

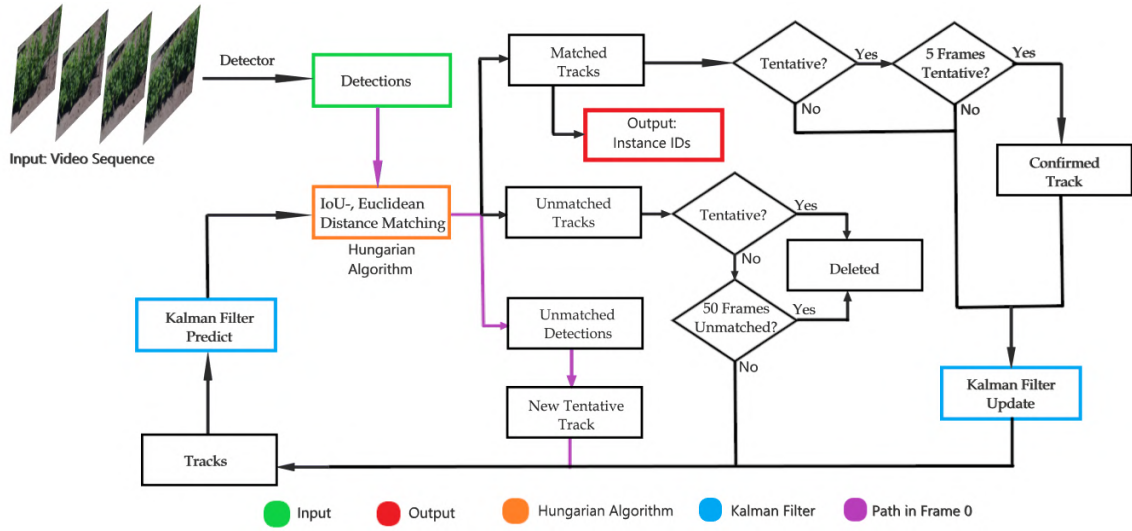


Figure 3.5.: Simple Online and Realtime Tracking Overview [Own Figure]

Upon detecting all instances (grape clusters) in frame n , SORT [12] employs a [Kalman Filter](#) [86] to predict the position of each grape cluster in the subsequent frame $n + 1$. The [Kalman Filter](#) [86], through a set of mathematical equations, iteratively estimates the position and velocity of an object (grape cluster) with noisy data. This allows the [Kalman Filter](#) [86] to predict future positions.

The [Hungarian matching algorithm](#) [48] is then used to optimally assign each predicted position to at most one detected instance in frame $n + 1$, within certain regional constraints. The [Kalman Filter](#) [86] continues to predict the positions of unmatched tracks for up to 50 frames, until a detection match is found, or the track is terminated. Detections not matched with any track initiate new tentative tracks, which are promptly removed if they remain unmatched. After a duration of five frames, a tentative track is confirmed. This helps to exclude false positives (*FPs*) from the count, as they are less likely to occur in five consecutive frames.

3.4.1.1. Kalman Filter

The Kalman Filter [86] is designed to quickly converge onto the true position and velocity (state estimate) of an object. Although based on the assumption of linear motion, it can also handle non-linear motion, given that a sufficient number of data points from non-linear motion approximate a linear trajectory. In the context of multi object tracking, SORT [12] initializes a Kalman Filter [86] for each individual track.

The Kalman Filter [86] iteratively predicts the state estimate (position and velocity) X_t for future frames and calculates an error covariance matrix P_t . This matrix quantifies the filter's uncertainty regarding its predictions, offering a dynamic measure of confidence into the state estimate. Upon detecting a new measurement (matching a detection with the track) the Kalman Filter [86] updates both the state estimate and the error covariance matrix (uncertainty) of the track. The state estimate $X_t \in \mathbb{R}^8$ as mean vector and the error covariance matrix $P_t \in \mathbb{R}^{8 \times 8}$ together characterize a multivariate normal distribution

$N(X_t, P_t)$. Over time, the elements of the error covariance matrix P_t decrease, indicating growing confidence into the state estimate's accuracy. Thus, the Kalman Filter [86] converges onto the true position and velocity of the tracked object. The Kalman Filter [86] algorithm is explained in the following.

1. Initialization

- Initiate the **State Estimate** X_0 . Based on the camera's movement, the velocity in x -direction is initialized at 5 pixels per frame. The state estimate X_t represents the Kalman Filter's estimation of a grape cluster's position and velocity in frame t .

$$X_0 = \begin{pmatrix} c_x \\ c_y \\ a \\ h \\ v_{c_x} \\ v_{c_y} \\ v_a \\ v_h \end{pmatrix} = \begin{pmatrix} c_x \\ c_y \\ a \\ h \\ \pm 5 \\ 0 \\ 0 \\ 0 \end{pmatrix} \quad (3.16)$$

(c_x, c_y) = center of bbox; a = width/height ratio; h = height; v = velocity or change in ratio and height.

- Initiate the **(Process) Error Covariance Matrix** P_0 , indicating the Kalman Filter's uncertainty regarding the current state estimate.

$$P_0 = \text{diag}((2w_ph)^2, (2w_ph)^2, 10^{-4}, (2w_ph)^2, (10w_vh)^2, (10w_vh)^2, 10^{-10}, (10w_vh)^2) \quad (3.17)$$

$w_p = \frac{1}{10}$ = position weight; $w_v = \frac{1}{80}$ = velocity weight; h = height.

2. Predict

Predicts the position and velocity of a grape cluster in the next frame, and how certain the Kalman Filter [86] is about its prediction.

$$X_{t-1} \rightarrow X_{t_p} = AX_{t-1} \quad (3.18)$$

$$P_{t-1} \rightarrow P_{t_p} = AP_{t-1}A^T + Q_t \quad (3.19)$$

The Process Noise Covariance Matrix Q_t represents the uncertainty within the model, such as deviations from the linear model presumed by the Kalman Filter [86].

$$Q_t = \text{diag}((w_pX_{t-1}[4])^2, (w_pX_{t-1}[4])^2, 10^{-4}, (w_pX_{t-1}[4])^2, (w_vX_{t-1}[4])^2, (w_vX_{t-1}[4])^2, 10^{-10}, (w_vX_{t-1}[4])^2) \quad (3.20)$$

$w_p = \frac{1}{10}$ = position weight; $w_v = \frac{1}{80}$ = velocity weight; $X_{t-1}[4]$ = height of the bbox estimation in the last frame.

The Motion Matrix \mathbf{A} is the linear motion of state X_t in a single time step ($t \rightarrow t + 1$).

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (3.21)$$

The [Hungarian Algorithm](#) [48] utilizes the prediction X_{t_p} to identify a matching detection in frame t . If a track does not match any detection, [SORT](#) [12] skips the update step, leading to an increase in the elements of the error covariance matrix \mathbf{P}_{t_p} , signifying greater uncertainty. If the Kalman update step is skipped:

$$X_{t_p} \rightarrow X_t \quad (3.22)$$

$$\mathbf{P}_{t_p} \rightarrow \mathbf{P}_t \quad (3.23)$$

3. Update (Optional)

If the [Hungarian Algorithm](#) [48] finds a matching detection, the Kalman Filter [86] uses this detection to refine its estimates. The larger the elements of the [Error Covariance Matrix](#), the more the detection influences the update of the state estimate.

$$\mathbf{K} = \mathbf{P}_{t_p} \mathbf{H}^\top [\mathbf{H} \mathbf{P}_{t_p} \mathbf{H}^\top + \mathbf{R}_t]^{-1} \quad (3.24)$$

$$\iff \mathbf{K} [\mathbf{H} \mathbf{P}_{t_p} \mathbf{H}^\top + \mathbf{R}_t] = \mathbf{P}_{t_p} \mathbf{H}^\top \quad (3.25)$$

$$\iff [\mathbf{H} \mathbf{P}_{t_p} \mathbf{H}^\top + \mathbf{R}_t]^\top \mathbf{K}^\top = \mathbf{H} \mathbf{P}_{t_p}^\top \quad (3.26)$$

$$\xrightarrow{\text{Cholesky}} \mathbf{K}^\top$$

$$X_t = X_{t_p} + \mathbf{K} [Y_t - \mathbf{H} X_{t_p}] \quad (3.27)$$

$$\mathbf{P}_t = [\mathbf{I} - \mathbf{K} \mathbf{H}] \mathbf{P}_{t_p} \quad (3.28)$$

\mathbf{K} = Kalman Gain; X_{t_p} , \mathbf{P}_{t_p} : predicted state estimate and error covariance matrix from (3.18); $Y_t = ((c_x)_t, (c_y)_t, a_t, h_t) \in \mathbb{R}^4$: matched detection in frame t ; \mathbf{H} changes dimensions:

$$\mathbf{H} = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (3.29)$$

The Measurement Noise Covariance Matrix \mathbf{R}_t represents the uncertainty of the measurement Y_t , such as cases where the detector might only identify half of the grape cluster area.

$$\mathbf{R}_t = \text{diag}((w_p Y_t[4])^2, (w_p Y_t[4])^2, 10^{-2}, (w_p Y_t[4])^2) \quad (3.30)$$

The Kalman gain \mathbf{K} is efficiently calculated using a Cholesky Decomposition \mathbf{L} of $[\mathbf{H}\mathbf{P}_t\mathbf{H}^\top + \mathbf{R}_t]^\top$ and solving a linear system of equations for each column of \mathbf{K} . Through complete induction, it can be shown that $[\mathbf{H}\mathbf{P}_t\mathbf{H}^\top + \mathbf{R}_t]^\top$ is symmetric and positive definite for every t . This is proven in Section A.1.1. For an in-depth understanding of the Cholesky Decomposition, the reader is referred to Dörfler and Rieder (2022) [27]. The Kalman gain \mathbf{K} dictates the extent to which new measurements influence the prediction. Over time, the elements of \mathbf{K} diminish, signifying increasing confidence into the estimation.

For simplification, we assume that the variables in the state estimation are independent of each other. Therefor the covariance matrices \mathbf{P}_t , \mathbf{Q}_t and \mathbf{R}_t are initialized as diagonal matrices. \mathbf{P}_t becomes non-diagonal after one iteration.

3.4.1.2. Hungarian Matching Algorithm

The Hungarian Algorithm [48], depicted by an orange box in Figure 3.5, determines the minimum total cost for achieving a one-to-one assignment between track predictions and detections. The cost matrix $\mathbf{C} \in \mathbb{R}^{n \times m}$ could consist of the Euclidean distances between the centers of bounding boxes, their *IoU-overlaps* or the *Mahalanobis distance* [58]. For instance, the cost $c_{i,j} = 1 - \text{IoU}(\text{Pred}_i, \text{Det}_j)$ represents the complement of the *IoU-overlap* between the i th track prediction and the j th detection. The Hungarian Algorithm [48] makes the best one-to-one assignment by minimizing the total cost $c = \sum_M c_{i,j}$, where M denotes the set of all matched pairs:

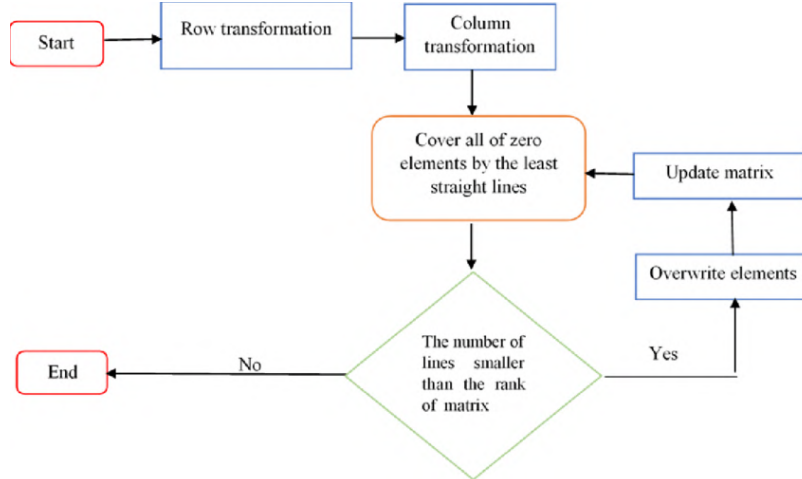


Figure 3.6.: Flowchart of the Hungarian Algorithm. [31]

1. Delete rows and columns, where every entry equals infinity.

- Certain matches can be excluded from consideration by setting their cost $c_{i,j} = \inf$
- Deleted rows are unmatched tracks, while deleted columns are unmatched detections.

2. **For each row, subtract its minimum element from every element in that row.**
 - All elements become non-negative.
 - With only non-negative values, an assignment with total cost $c = 0$ is minimum.
3. **For each column, subtract its minimum element from every element in that column.**
 - This often allows for an optimal assignment, where the total cost is $c = 0$.
4. **Cover all zeros with the fewest possible number of lines across rows and columns.**
 - For a detailed explanation of this step, refer to Kuhn (1955) [48].
 - The number of lines is less than or equal to the smaller of the two dimensions, n or m .
5. **If the number of lines equals n or m , an optimal solution is found. → Stop**
 - If not, continue to the next step.
 - If an optimal solution is found, one zero can be chosen in every row or every column, depending on which dimension is smaller. For example, choosing a 0 in row i and column j means matching track prediction i with detection j .
 - Pairs where the cost exceeds a certain threshold are unmatched. For instance, without any *IoU-overlap* between detections and predictions, the cost matrix will consist solely of ones, leading to random matches.
 - Remaining rows and columns represent unmatched tracks or predictions, respectively.
6. **Subtract the smallest non-covered entry from each uncovered row and add it to each crossed-out column.**
 - → Return to Step 4.

3.4.2. DeepSORT

Simple Online and Realtime Tracking with a Deep Association Metric (DeepSORT), developed by Wojke, Bewley, and Paulus (2017) [88], extends the SORT [12] algorithm by integrating a classifier to compare appearances of objects (grape clusters) across frames. This feature makes DeepSORT [88] potentially well-suited for tracking grapes, because of the inanimate grape clusters and its capability to re-identify grapes that have been occluded or undetected for several frames.

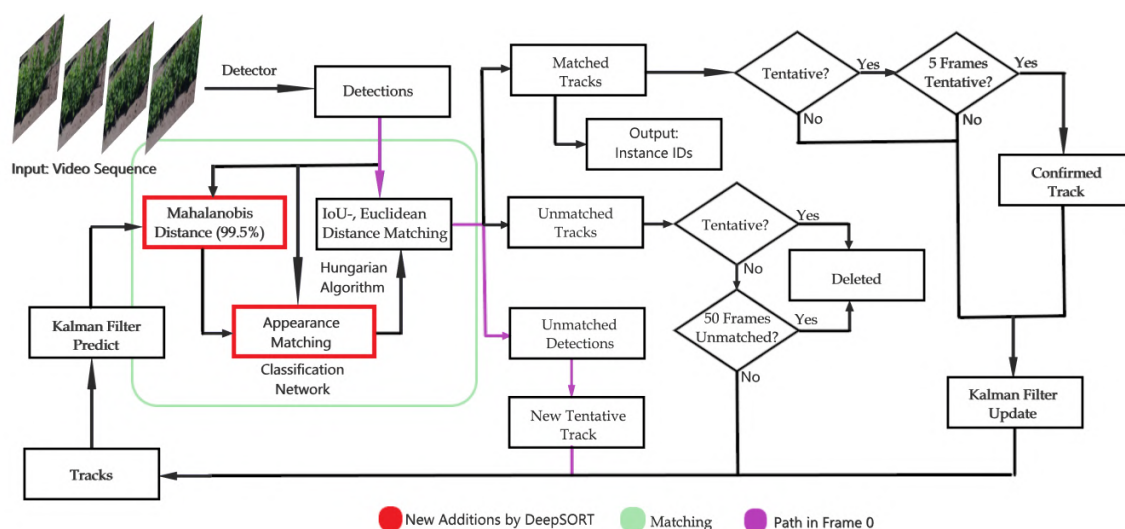


Figure 3.7.: DeepSORT as an extension of SORT. [Own Figure]

As can be seen in figure 3.7, DeepSORT [88] builds upon the matching mechanism of SORT [12] by implementing the Mahalanobis distance [58] to filter out improbable matches (below 0.5% probability) and employing a classification network that extracts appearance features of objects.

Because the classification network does not consider the physical proximity between detections and track predictions, Wojke, Bewley, and Paulus (2017) [88] incorporate the [Mahalanobis distance](#) [58]. The [Mahalanobis distance](#) [58] measures how far a detection is from a probability distribution, characterized by the state estimate X_{t_p} and the covariance matrices (uncertainty) of both the prediction \mathbf{P}_{t_p} and the measurement (detection) \mathbf{R}_t . Detections are matched to the track prediction using the [Hungarian Algorithm](#) [48] only if their [Mahalanobis distance](#) [58] is below the 99.5% threshold. This threshold, derived from the chi-squared distribution [16] is the [Mahalanobis distance](#) [58] beyond which matches between a detection and a track are considered extremely improbable (less than 0.5% probability).

3.4.2.1. Mahalanobis Distance

The Mahalanobis distance [58] is a measure of distance between a point and a distribution. In the context of the [Kalman Filter](#) [86] the predicted state estimate X_{t_b} and the covariance

matrix $\mathbf{S}_t = [\mathbf{H}\mathbf{P}_{t_p}\mathbf{H}^\top + \mathbf{R}_t]$ together form a multivariate normal distribution $N(X_{t_p}, \mathbf{S}_t)$. Matches where the Mahalanobis distance [58] between detection Y_t and the track prediction $N(X_{t_p}, \mathbf{S}_t)$ exceed a certain threshold are excluded. To not consider these matches in the [Hungarian Algorithm](#) [48], their cost $c_{i,j}$ is set to infinity. The threshold for the Mahalanobis distance [58] can be derived from the chi-squared distribution [16], where the squared Mahalanobis distance [58], $d(\vec{x}, \vec{y}, N)^2$ is a chi-squared distribution [16] with n degrees of freedom. Here, n refers to the number of variables in the track prediction \vec{x} (simplification of X_{t_p}) and detection \vec{y} (simplification of Y_t). This relationship is proven in Thill (2017) [82]. The variables in \vec{x} and \vec{y} can include just the bounding box center (c_x, c_y) or additional attributes like the width/height ratio a and height h . The chi-0.995-quantile $x_{0.995;n}$ with either 2 or 4 degrees of freedom can be applied, resulting in thresholds of $x_{0.995;2} = 10.597$ or $x_{0.995;4} = 14.860$ respectively, to filter highly unlikely matches (0.5% probability) out.

The squared **Mahalanobis distance** [58] between detection \vec{y} and the track prediction \vec{x} with respect to the multivariate normal distribution $N(0, \mathbf{S})$ is defined as:

$$D = d(\vec{x}, \vec{y}, N)^2 = (\vec{x} - \vec{y})^\top \mathbf{S}_t^{-1} (\vec{x} - \vec{y}) \quad (3.31)$$

$$= (\vec{x} - \vec{y})^\top \mathbf{S}_t^{-\frac{1}{2}} \mathbf{S}_t^{-\frac{1}{2}} (\vec{x} - \vec{y}) \quad (3.32)$$

$$= (\mathbf{S}_t^{-\frac{1}{2}} (\vec{x} - \vec{y}))^\top (\mathbf{S}_t^{-\frac{1}{2}} (\vec{x} - \vec{y})) \quad (3.33)$$

$$= \|\mathbf{S}_t^{-\frac{1}{2}} (\vec{x} - \vec{y})\|^2 \quad (3.34)$$

$$\iff \|\mathbf{S}_t^{\frac{1}{2}}\|^2 D = \|\mathbf{S}_t^{\frac{1}{2}} \mathbf{S}_t^{-\frac{1}{2}} (\vec{x} - \vec{y})\|^2 \quad (3.35)$$

$$\iff \|\mathbf{S}_t^{\frac{1}{2}} \cdot \vec{z}\|^2 = \|\vec{x} - \vec{y}\|^2 \quad (3.36)$$

$$\iff \|\mathbf{L}_t^\top \vec{z}\|^2 = \|\vec{x} - \vec{y}\|^2 \quad (3.37)$$

The track prediction \vec{x} is the mean vector and \mathbf{S}_t the covariance matrix of $N(\vec{x}, \mathbf{S}_t)$. It follows that $(\vec{x} - \vec{y}) \sim N(0, \mathbf{S}_t)$. When computing the Mahalanobis distance [58] using only the center of the bounding boxes, the covariance matrix is $\mathbf{S}_t = \mathbf{T}[\mathbf{H}\mathbf{P}_{t_p}\mathbf{H}^\top + \mathbf{R}_t]\mathbf{T}^\top$, where \mathbf{T} adjusts the dimensions:

$$\mathbf{T} = \begin{pmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{pmatrix} \quad (3.38)$$

Otherwise, the covariance matrix is $\mathbf{S}_t = [\mathbf{H}\mathbf{P}_{t_p}\mathbf{H}^\top + \mathbf{R}_t]$. The covariance matrix $[\mathbf{H}\mathbf{P}_{t_p}\mathbf{H}^\top + \mathbf{R}_t]$ incorporates the [error covariance matrix](#) \mathbf{P}_{t_p} from equation (3.18) and the [measurement noise covariance matrix](#) \mathbf{R}_t , which represent the uncertainty of state estimate X_{t_p} and the measurement Y_t . $\mathbf{S}_t = [\mathbf{H}\mathbf{P}_{t_p}\mathbf{H}^\top + \mathbf{R}_t]$ is symmetric and positive definite. For proof, the reader is referred to Section A.1.1. Therefore, the Cholesky decomposition \mathbf{L}_t of $[\mathbf{H}\mathbf{P}_{t_p}\mathbf{H}^\top + \mathbf{R}_t]$ exists and \mathbf{S}_t^{-1} is symmetric and positive definite as well. It follows that $\mathbf{S}_t^{\frac{1}{2}}$ and $\mathbf{S}_t^{-\frac{1}{2}}$ exist, with $\mathbf{S}_t^{\frac{1}{2}} \mathbf{S}_t^{\frac{1}{2}} = \mathbf{S}_t$ and $\mathbf{S}_t^{-\frac{1}{2}} \mathbf{S}_t^{-\frac{1}{2}} = \mathbf{S}_t^{-1}$. For a proof of $\|\mathbf{S}_t^{\frac{1}{2}} \vec{z}\|^2 = \|\mathbf{L}_t^\top \vec{z}\|^2$, refer to Section A.1.2. Using the Cholesky decomposition \mathbf{L}_t of $[\mathbf{H}\mathbf{P}_{t_p}\mathbf{H}^\top + \mathbf{R}_t]$ one can calculate \vec{z} , where $\|\vec{z}\|^2 = D$.

For each track prediction, the Mahalanobis distance [58] to every detection is calculated. Detections, where the distance is above the chi-0.995-quantile $x_{0.995;n}$ are not considered

for matching with that prediction.

The Mahalanobis distance [58] alone, without incorporating appearance features, can be used within the cost matrix of the [Hungarian Algorithm](#) [48] to improve [SORT](#) [12].

3.4.2.2. Classification Network

The classification network is a convolutional neural network (CNN), such as ResNet [47] or EfficientNet [46]. Unlike the CNN-backbone in [Mask R-CNN](#) [38], which processes the entire image, the classification CNN extracts a feature vector $v_j \in \mathbb{R}^n$ from the cut-out bbox of every detection in the current frame. For each track, the CNN generates a feature vector $(v_i)_l \in \mathbb{R}^n$ from the cut-out bbox of the last 10 detections matched with that track. The track's feature vector $v_i \in \mathbb{R}^n$, obtained as the mean of these vectors $(v_i)_l$, is then compared to the feature vectors v_j of all detections.

$$v_i = \frac{1}{10} \begin{pmatrix} \sum_{l=1}^{10} (v_i)_l[1] \\ \dots \\ \sum_{l=1}^{10} (v_i)_l[n] \end{pmatrix} \quad (3.39)$$

The distance $d(i, j)$ in the appearance space between track i and detection j is calculated as the Euclidean distance between the respective normalized feature vectors \tilde{v}_i and \tilde{v}_j .

$$\tilde{v}_i = \frac{v_i}{||v_i||} \quad (3.40)$$

$$\tilde{v}_j = \frac{v_j}{||v_j||} \quad (3.41)$$

$$d(i, j) = \sqrt{\sum_{k=1}^n (\tilde{v}_i[k] - \tilde{v}_j[k])^2} \quad (3.42)$$

The cost matrix C for matching track predictions and detections is a weighted sum of the squared [Mahalanobis distance](#) [58] and the Euclidean distance in the appearance space:

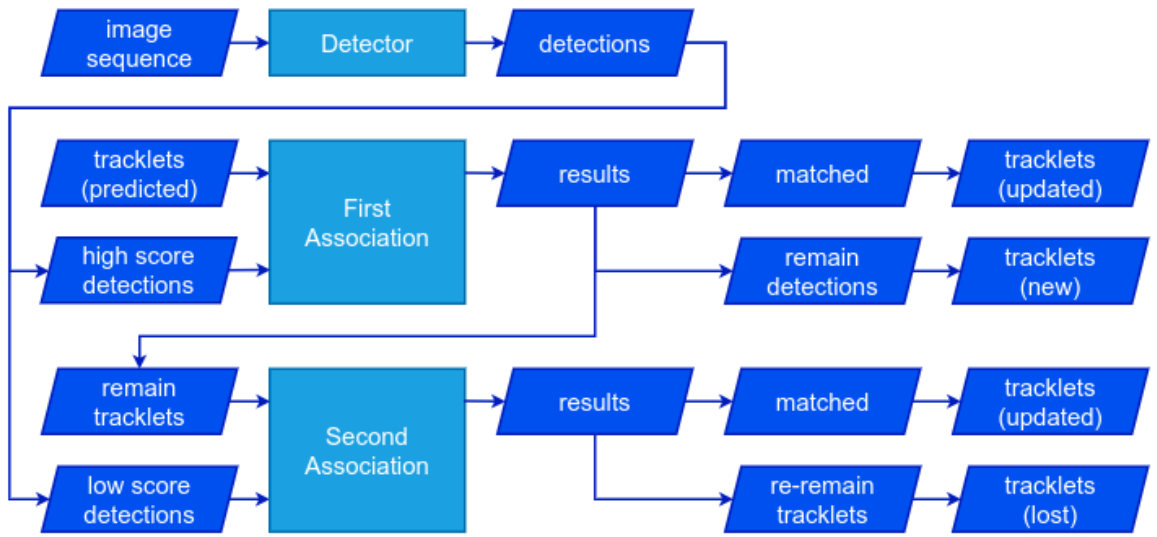
$$c_{i,j} = \lambda D + (1 - \lambda)d(i, j) \quad (3.43)$$

where $\lambda \in [0, 1]$. For rapid camera movement $\lambda = 0$ is a popular choice [88], since the [Mahalanobis distance](#) [58] might be inaccurate.

3.4.3. ByteTrack

In traditional vineyards, the occlusion of grape clusters significantly hinders detection efforts. Partly occluded objects receive lower confidence scores by the detection algorithm and are discarded if these scores fall below a specific threshold. This process results in the loss of true objects and leads to fragmented tracks.

ByteTrack, developed by Zhang et al. (2022) [97], addresses this issue by recovering low confidence detections that fit existing tracks. ByteTrack [97] uses many of [SORT's](#) [12] building blocks.



© 2023 Luffca Inc.

Figure 3.8.: ByteTrack Overview [18]

- A [Kalman Filter](#) [86] estimates the position and velocity of an object (grape cluster) and predicts its future position in subsequent frames.
- Tracks begin as “tentative” and require matching with high-confidence detections across five consecutive frames to be confirmed. If no high confidence detection matches with a tentative track, the track gets removed immediately.
- Confirmed tracks are retained for up to 50 frames without detection matches before being deleted.
- In Figure 3.8 the “Associations” represent the usage of the [Hungarian Algorithm](#) [48], which matches track predictions with detections. The matching algorithm utilizes a cost matrix C calculated based on the *IoU-overlap*: $c_{i,j} = 1 - IoU(Pred_i, Det_j)$. Alternatives to the *IoU-overlap* can also be used, such as Euclidean or [Mahalanobis distances](#) [58], or even appearance features from a re-identification network (similar to [DeepSORT](#) [88]).

To handle scenarios where [Mask R-CNN](#) [38] detects two bounding boxes for the same track, ByteTrack scales the *IoU-overlap* between track predictions and detections based on the confidence scores of the detections. This ensures that detections with higher confidence scores are given greater weight during the matching process, potentially increasing matching accuracy. The following outlines the matching order:

1. During the first association, a [Hungarian Algorithm](#) [48] matches high-confidence detections with confirmed tracks first, applying a relatively low threshold for *IoU-overlap*.

2. Another [Hungarian Algorithm](#) [48] matches any remaining high confidence detections with tentative tracks. Here a slightly higher *IoU*-threshold is applied, due to the questionable status of these tracks.
 - Remaining tentative tracks are removed.
 3. The last [Hungarian Algorithm](#) [48] matches the remaining confirmed tracks with low confidence detections, which have to pass a relatively high *IoU*-threshold.
 - Remaining low confidence detection are discarded.
- Similar to SORT [12], the [Kalman Filter](#) [86] in ByteTrack [97] predicts the future position of unmatched confirmed tracks for up to 50 frames, until they are either matched or removed. These tracks bypass the update step.
 - The [Kalman Filter](#) [86] updates the state estimate of tracks using matched detections.

ByteTrack [97] repeats these steps for each frame, thereby tracking grape clusters with the help of recovered low-confidence detections.

4. Concept

This thesis evaluates the tracking performance of SORT [12], DeepSORT [88] and ByteTrack [97] using the Mask R-CNN detector [38] on the UAV RGB Video dataset [7]. This chapter explains the implementation and specific configuration of the dataset, the detector, and the trackers as introduced in the previous chapter "Materials and Methods". The Code for this comparative evaluation is going to be publicly accessible on GitHub (<https://github.com/Jax377/Grape-Detection-and-Tracking.git>).

4.1. Code Environment

The algorithms were implemented, trained and evaluated using Open-MMLab's MMTTracking toolbox (<https://github.com/open-mmlab/mtracking>) on a Windows 10 platform. MMTTracking is a comprehensive video perception toolkit facilitating object detection, object segmentation, single object tracking, multi object tracking and video instance segmentation, using python. Its modular architecture allows for easy swapping of components, like SORT [12] or Mask R-CNN [38]. However, the original toolbox requires annotations for every frame to evaluate tracking performance. To evaluate videos with only a few annotated frames and fully implement all tracking algorithms, MMTTracking was modified and new code added.

4.2. Dataset Configuration and Data Transformation

The Challenges with the UAV RGB Video dataset [7], identified in section 3.1, such as the presence of very small or missed grape clusters, complicate the training of a good detector, critical for achieving accurate tracking performance. This research primarily explores the tracking step in a two-step multi-object tracking approach, by employing the same Mask R-CNN detector [38] in all tracking algorithms. To ensure the best performance of Mask R-CNN [38], which in itself is of secondary interest in this thesis, the UAV RGB Video dataset [7] was split into a training and a test subsets, excluding a validation set typically used for epoch selection. The dataset was partitioned into an 80-20 split, allocating 528 images for training and 136 for testing. The training subset is used to iteratively train the Mask R-CNN network [38] to detect and segment grape clusters from the training set, while the test set contains videos for following grape tracking evaluations. This breaks the convention, where an additional validation subset is used to choose the best epoch (weights) for the Mask R-CNN network [38]. Using the test set directly for epoch selection

instead, guarantees the utilization of the most effective [Mask R-CNN](#) model [38] for both grape detection and tracking in the videos of the test set. The following Table 4.1 shows the adopted 80-20 split, where video sequences from all four vine rows are included in the test set.

Subset	Rows
Training set	4.3.2, 4.4.2, 4.4.4, 6.1.3, 6.1.4, 6.2.1, 6.2.2, 6.3, 7.1.3, 7.1.4, 7.2.1, 7.2.2, 7.2.3, 7.2.4, 7.3.1, 7.3.2, 7.3.3, 7.3.4, 7.4.1, 7.4.2, 8.2, 8.3, 8.4
Test set	4.2.1, 6.1.1, 6.1.2, 7.1.1, 7.1.2, 8.1

Table 4.1.: For the training/test split, four vine rows were segmented into subsections and further into sub-subsections, each ranging from 250 to 2000 frames. On average, each video section includes approximately 22 annotated frames.

Data Transformation

To improve the [Mask R-CNN](#) network’s [38] performance as a detector, the training subset images undergo various transformations. These modifications bolster the model’s robustness and its capacity to generalize onto unseen data. Transformations such as rotation, flipping and changes in illumination mimic varying real-world conditions, including different weather scenarios or camera positions. Implementing such transformations is crucial for preventing overfitting, thus ensuring the detection and segmentation of objects, regardless of their orientation, size, location, or lighting conditions in the image.

Training the [Mask R-CNN](#) detector [38] involves a series of transformations outlined in Table 4.2, aimed at improving object detection accuracy of unseen data:

Photo Metric Distortion	
Brightness Adjustment	Random scalar from [0.5, 1.5]
Contrast Adjustment	Random scalar from [0.5, 1.5]
Saturation Adjustment	Random scalar from [0.5, 1.5]
Hue Adjustment	Random degree from $[-18, 18]$ by which pixels hues are shifted using the circular HSV color model.
Horizontal Flip	50% of the time
Pixel Value Normalization	Mean = [123.675, 116.28, 103.53], Standard deviation = [58.395, 57.12, 57.375]
Padding	Image dimensions dividable by 32 by adding black pixels.

Table 4.2.: Transformations applied to the training data of the [Mask R-CNN](#) network [38]

For brightness, contrast and saturation adjustment, different scalars within the range [0.5, 1.5] are randomly chosen to scale their respective characteristic. Hue adjustment utilizes the circular HSV color model to rotate the color of each pixel by a degree randomly chosen from the range $[-18, 18]$. Half of the images are mirrored along the vertical axis. For

image normalization, the model employs a mean of [123.675, 116.28, 103.53] and a standard deviation of [58.395, 57.12, 57.375]. To normalize the images, the mean is subtracted from each pixel's RGB values, centering the pixel values around zero. Subsequently, the pixel's RGB values are divided by the respective standard deviation to standardize their variance to one. The images are then padded with black pixels to ensure that both height and width are divisible by $32 = 2^5$. This allows the ResNet-50 backbone [47] and the Feature Pyramid Network [51] in Mask R-CNN [38] to half the image's spatial dimensions five times.

After training the Mask R-CNN [38] on the training dataset, the resulting model is applied to unseen images, which are normalized and padded. The model uses the same mean = [123.675, 116.28, 103.53] and standard deviation = [58.395, 57.12, 57.375] as during the training phase. This standardization ensures that the tested data matches the RGB-range of the training data. Similar to the training set, images are padded with black pixels so that their height and width are divisible by 32, a requirement for ResNet-50 [47] and the FPN [51] in Mask R-CNN [38].

During the training and application of the classification network in DeepSORT [88], similar transformations are applied, which are specified in Table 4.3. The CNN (classification network) within DeepSORT [88] is trained on cut-out ground truth bonding boxes of the grape clusters in the training data, aiming to extract feature vectors and matching similar looking grape clusters.

Resizing to fixed size	(128, 256)
Horizontal Flip	50% of the time
Pixel Value Normalization	mean = [123.675, 116.28, 103.53], standard deviation = [58.395, 57.12, 57.375]

Table 4.3.: Transformations applied to the training data of the CNN (classification network) within DeepSORT [88]

During application in DeepSORT [88] on unseen data, the classification network processes the cut-out bounding boxes by resizing and normalizing them with the mean = [123.675, 116.28, 103.53] and standard deviation = [58.395, 57.12, 57.375] established during training. This ensures that the images fed into the network during the application are in the same format and have the same RGB range as the images it was trained on, allowing for consistent and accurate feature extraction.

4.3. Detector Configuration

This study focuses on the evaluation of grape tracking, since grape detection has been extensively explored in previous research. Olenskyj et al. (2022) [65], Victorino et al. (2019) [83], Lopes and Cadima (2021) [54] identified the area of grape clusters as the single most significant predictor for grape yield estimation. Given that yield estimation, besides robotic applications, serves as a primary motivation for grape tracking, this research

has chosen a segmentation network capable of generating pixel-level masks. In 2020 Santos et al. (2020) [75] demonstrated that [Mask R-CNN](#) [38] outperforms early versions of YOLO (You-Only-Look-Once) networks. However, recent studies in 2023, such as Sapkota, Ahmed, and Karkee (2023) [76] indicate that newer versions of YOLO outperform [Mask R-CNN](#) [38] in complex orchard settings, making YOLO the preferred choice for grape detection and segmentation [77, 8, 22]. Despite this, this thesis has chosen to utilize [Mask R-CNN](#) [38], as [Mask R-CNN](#) frameworks [38] are widely accessible to many researchers and generally require less fine-tuning to achieve satisfactory detection and segmentation outcomes. Table A.1, which can be found in the Appendix, outlines the configuration of all components within the [Mask R-CNN](#) [38] framework utilized in this study.

The following Table 4.4: outlines the settings for the training of the [Mask R-CNN](#) network [38]:

Anchor Assignment	
Positive: Anchors $> IoU$ -threshold	0.7
Negative: Anchors $< IoU$ -threshold	0.3
Positive (low quality): Anchors $> IoU$ -threshold	0.3
Random Samples for Training	256 (50% positive)
Mask/Bbox Assignment	
Positive: Anchors $> IoU$ -threshold	0.5
Negative: Anchors $< IoU$ -threshold	0.5
Random Samples for Training	512 (25% positive)
	<i>GT</i> are included for training
Mask Binary Threshold	0.5
Maximum Epochs	25

Table 4.4.: Training Configurations of the [Mask R-CNN](#) Network [38]

During testing, which includes tracking, [Mask R-CNN](#) [38] discards all bounding boxes (bboxes) with a confidence score below 60%, except for [Mask R-CNN](#) [38] in [ByteTrack](#) [97]. In [ByteTrack](#) [97] the model recovers low-confidence bboxes by matching them with unmatched track predictions.

During Non-Maximum-Suppression [64], only the bbox with the highest confidence score is kept among all bboxes that have an *IoU*-overlap above 5% with each other. This unusually low *IoU*-threshold was chosen, because grape clusters rarely overlap or are very close to one another. By adopting this threshold, it becomes less common to have more than one bbox for the same grape cluster. Following Non-Maximum-Suppression [64], a maximum of 300 bboxes are kept, based on the highest confidence scores. This approach ensures that the system prioritizes the most probable grape cluster detections, minimizing the chance of duplicate detections or false positives, thus improving object detection and segmentation.

Optimizer Configuration

To train the [Mask R-CNN](#) network [38] an optimizer adjusts the network's weights to minimize a loss function, thereby improving the network's accuracy in object detection and instance segmentation. It controls the learning process through mechanisms, such as gradient calculation and weight updates, using a strategy determined by the specific type of optimizer chosen. Both stochastic gradient descent (SGD) [5] and the Adam optimizer [44] are evaluated to determine the most effective optimizer for this task. Both optimizers use a weight decay of 0.0001 and SGD [5] a momentum of 0.9, while being tested across various learning rates and assessed based on the resulting networks' *mAP*. The findings are detailed in the [“Results”](#) chapter.

A learning rate scheduler was employed to dynamically adjust the learning rate (the optimizer's step size), potentially improving the model's performance and the speed of convergence. Each epoch in the training consists of 264 iterations, which is half the number of images in the training subset. The optimizer starts with 1% of the set learning rate, linearly increasing its step size to reach the full learning rate by the 100th iteration. After three epochs, the learning rate is decreased to 10% of its original value, because the model approaches its optimum state. The configuration of the learning rate scheduler is depicted in Figure 4.1, providing a visual representation of this adaptive adjustment strategy.

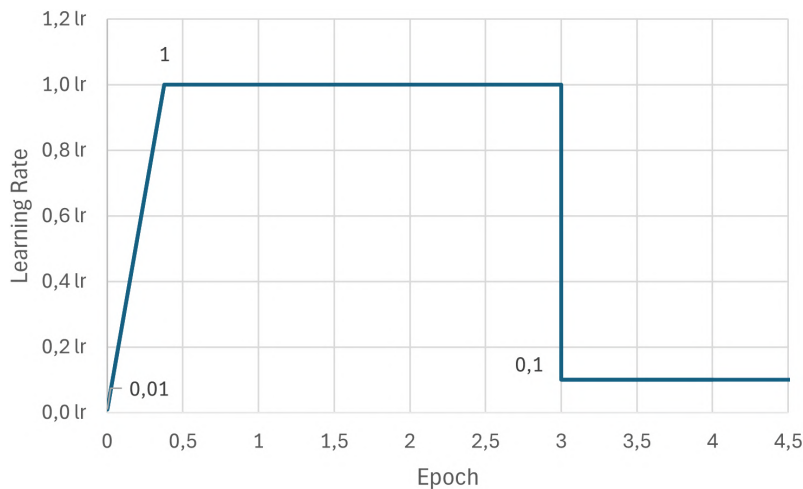


Figure 4.1.:
Learning Rate Scheduler
[Own Figure]

4.4. Tracker Configurations

As mentioned in Chapter 3 on “Materials and Methods”, the tracking algorithms [SORT](#) [12], [DeepSORT](#) [88], and [ByteTrack](#) [97] offer numerous configuration possibilities. This is particularly true for the matching of track predictions with their corresponding detections, which offers multiple variations in the choice of similarities utilized by the [Hungarian Algorithm](#) [48], as well as the selection of thresholds to discard unlikely matches. An advanced approach involves employing multiple [Hungarian Algorithms](#) [48] in sequence, using different similarities to pair remaining detections with remaining tracks. Possible

similarities for this purpose include the *IoU*-overlap, the [Mahalanobis distance](#) [58], the Euclidean distance and appearance features extracted by a [classification network](#).

The original implementations of [SORT](#) [12] and [ByteTrack](#) [97] rely solely on *IoU*-overlap for matching tracks with detections. [DeepSORT](#) [88] additionally incorporates appearance features in combination with the [Mahalanobis distance](#) [58] in its initial matching phase, before using the *IoU*-overlap to match remaining tracks and detections. Besides these original implementations, this thesis expands the matching processes of [SORT](#) [12] and [DeepSORT](#) [88] by employing multiple similarities in sequence, referring to the enhanced versions as SORT+ and DeepSORT+. These alterations aim to leverage the complementary strengths of different similarity measures to potentially increase matching accuracy, leading to improved tracking performance, and counting accuracy.

4.4.1. SORT Configuration

This section describes the implementation of the original SORT algorithm [12], using fine-tuned thresholds that yield the best tracking results. The specific configuration and values used for SORT [12] are listed in Table 4.5.

Detector	Mask R-CNN [38]
Confidence score threshold	0.6
Frames until removing unmatched tracks	50
Frames until confirming tentative tracks	5
Kalman Filter [86]	
position weight	$w_p = \frac{1}{10}$
velocity weight	$w_v = \frac{1}{80}$
Matching	Hungarian Algorithm [48]
1 – <i>IoU</i> overlap	Threshold: 0.965

Table 4.5.: Configuration of the SORT Algorithm [12]

To be comparable, all trackers employ a [Mask R-CNN](#) network [38] as the detector. Confirmed tracks that fail to be matched with detections are removed after 50 frames. A tentative track transitions to the confirmed state five frames after initialization. The position weight w_p and velocity weight w_v fine-tune the initialization of the covariance matrices \mathbf{P}_0 , \mathbf{Q}_t and \mathbf{R}_t in the [Kalman Filter](#) [86]. However, altering these weights does not yield any significant changes in the results. The cost matrix utilized by the [Hungarian Algorithm](#) [48] consists of the *IoU*-overlap between detections and track predictions. Only matches with an *IoU*-overlap greater than 3.5% are retained.

4.4.2. SORT+ Configuration

SORT+ extends [SORT](#) [12] by incorporating additional similarities to the *IoU*-overlap in order to match detections and tracks. The specific configuration and values used for SORT+ that yield the best tracking results are listed in Table 4.6.

Detector	Mask R-CNN [38]
Confidence score threshold	0.6
Frames until removing unmatched tracks	50
Frames until confirming tentative tracks	5
Kalman Filter [86]	
position weight	$w_p = \frac{1}{10}$
velocity weight	$w_v = \frac{1}{80}$
Matching	Hungarian Algorithm [48]
1. $1 - IoU_{overlap}$	Threshold: 0.8
2. Mahalanobis distance [58]	Threshold: $x_{0.9;2} = 4.605$
Variables used for the Mahalanobis distance [58]	(c_x, c_y)
3. $1 - IoU_{overlap}$	Threshold: 0.965
4. Euclidean distance	Threshold: $w_{Pred} + h_{Pred}/2$

Table 4.6.: Configurations of the SORT+ Algorithm

For comparability, all trackers utilize a [Mask R-CNN](#) network [38] as the detector. Confirmed tracks that fail to be matched with detections are removed after 50 frames. A tentative track transitions to the confirmed state five frames after initialization. The position weight w_p and velocity weight w_v fine tune the [Kalman Filter](#) [86].

To potentially improve matching accuracy, SORT+ incorporates four [Hungarian Algorithms](#) [48]. First, all detections and track predictions are matched where the *IoU*-overlap exceeds 20%, indicating highly probable matches.

Subsequently, using the [Mahalanobis distance](#) [58], detections are matched to the multivariate normal distribution that represents the closest track. Only matches with a [Mahalanobis distance](#) [58] below the chi-squared-0.9-quantil $x_{0.9;2} = 4.605$ are kept, as matches with larger distances are correct only 10% of the time.

Following this, the *IoU*-overlap is again utilized to match remaining detections with remaining track predictions, with a minimum *IoU*-overlap of 3.5%.

The last [Hungarian Algorithm](#) [48] employs the Euclidean distance between the centers of the bounding boxes for matching. Given that the Euclidean distance depends significantly on the size of the bounding boxes and the [Kalman Filter](#) [86] causes the track estimate to converge onto the average size of the grape cluster's bbox, the track bbox's width and half of the height are used as a threshold ($w_{Pred} + h_{Pred}/2$) for matches. The Euclidean distance aims to recover matches where [Mask R-CNN](#) [38] detected only part of a grape cluster, resulting in no *IoU*-overlap with the track prediction.

4.4.3. DeepSORT Configuration

This section explains the implementation of the original DeepSORT algorithm [88], using fine-tuned values that give the best tracking performance. The specific configuration and values utilized by [DeepSORT](#) [88] are listed in Table 4.7.

Detector	Mask R-CNN [38]
Confidence score threshold	0.6
Frames until removing unmatched tracks	50
Frames until confirming tentative tracks	5
Kalman Filter [86]	
position weight	$w_p = \frac{1}{10}$
velocity weight	$w_v = \frac{1}{80}$
Classification Network	ResNet-50 [47] Global Average Pooling [40] 1 Fully Connected Layer
Types of Matching	Hungarian Algorithm [48]
1. Appearance Features	Mahalanobis distance [58] < 10.597 & Appearance Features < 1.5
Variables for the Mahalanobis distance [58]	(c_x, c_y)
2. $1 - IoU_{overlap}$	Threshold: 0.965

Table 4.7.: Configurations of the DeepSORT Algorithm [88]

For consistency, all trackers employ a [Mask R-CNN](#) network [38] as the detector. Confirmed tracks that fail to be matched with detections are removed after 50 frames. A tentative track transition to the confirmed state five frames after initialization. The position weight w_p and velocity weight w_v fine tune the [Kalman Filter](#) [86].

[DeepSORT](#) [88] employs appearance feature vectors, the [Mahalanobis distance](#) [58] and the IoU -overlap to match detections with track predictions. The convolutional neural network (CNN) that extracts the feature vectors incorporates a [ResNet-50](#) model [47], Global Average Pooling [40] and a fully connected layer. Figure 4.2 shows the framework of the employed [classification network](#).

The implemented Pairwise Loss is a Triplet Loss, which utilizes two matching grape clusters and one non-matching grape cluster to train feature vectors that are spatially close if the appearances match, and distant otherwise. Both Stochastic Gradient Descent [5] and the Adam optimizer [44] are tested for training the classification network. The results are shown in section 5.1.3. The margin scales how much the weights are changed by the optimizer. Batch normalization [62] normalizes the RGB-values of each batch to a mean = [0, 0, 0] and a standard deviation = [1, 1, 1]. The convolutional neural network utilizes the ReLU (Rectified Linear Unit) [1] activation function:

$$ReLU(x) = \max(0, x) \quad (4.1)$$

Table 4.8 lists the configuration and training parameters of the classification network.

The classification network learns to generate spatially close feature vectors for identical grape clusters. The [Hungarian Algorithm](#) [48] then utilizes the Euclidean distance between feature vectors to match tracks with their corresponding detection. Matches are disregarded if the [Mahalanobis distance](#) [58] surpasses the chi-squared threshold at the 0.995 quantile ($x_{0.995;2} = 10.597$) or if the Euclidean distance between feature vectors

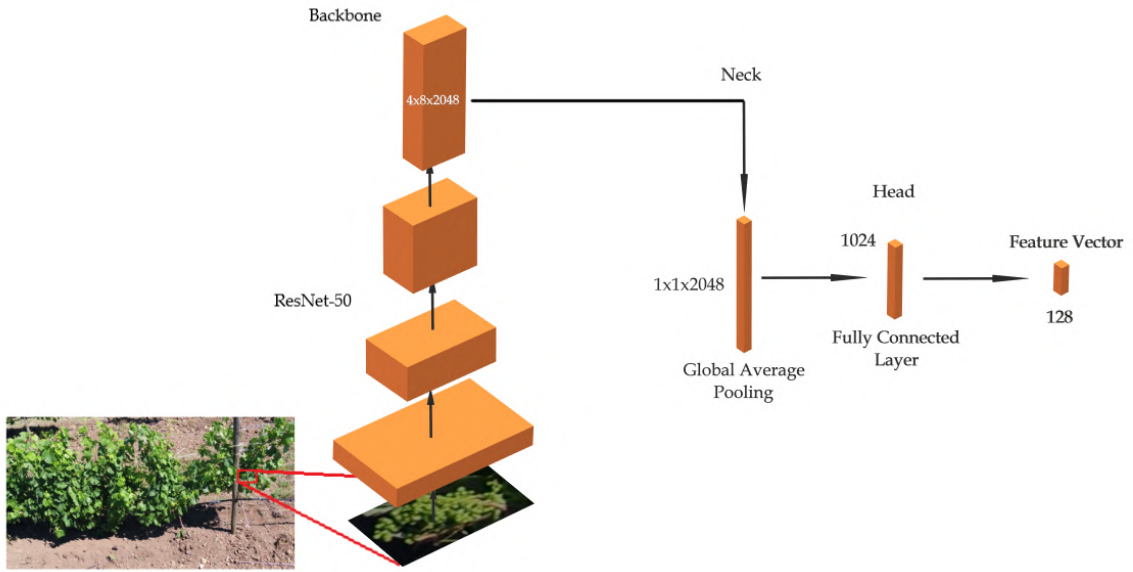


Figure 4.2.: The Classification Network in DeepSORT. [Own Figure]

Backbone	ResNet-50 [47]
Output Stage	4 stages
	4
	Pretrained on the ImageNet dataset [25]
Neck	Global Average Pooling [40]
In channels	$(4 \times 8 \times 2048)$
Out channels	$(1 \times 1 \times 2048)$
Head	One Fully Connected Layer
In channels	2048
Neurons	1024
Out Channels	128
Classes	586
Classification Loss	Cross-entropy loss
Pairwise Loss ($\times 2$)	Triplet Loss (Margin: 0.5)
	Batch Normalization [62]
	Rectified Linear Unit Activation Function [1]

Table 4.8.: Configuration of the Classification Network in DeepSORT [88]

exceeds 1.5.

This threshold was established by analyzing the Euclidean distances of positive and negative matches. The feature vectors of positive matches typically exhibit a Euclidean distance of approximately 0.8 with a variance of 0.6, whereas negative matches show a distance of around 2.1 with a variance of 0.4. A threshold of 1.5 effectively distinguishes positive from negative matches, with the [Mahalanobis distance](#) [58] serving as an additional filter to avoid negative matches.

Similar to [SORT](#) [12], DeepSORT matches remaining detections and track predictions

with *IoU*-overlaps greater than 3.5%. DeepSORT [88] aims to re-identify grape clusters when there is no overlap between detection and track prediction following a period of occlusion, potentially reducing instance ID switches (*IDsw*) and increasing tracking performance.

4.4.4. DeepSORT+ Configuration

DeepSORT+ builds upon DeepSORT [88] by adding three additional matching steps. The specific configuration and values that produce the best tracking results for DeepSORT+ are listed in Table 4.9.

Detector	Mask R-CNN [38]
Confidence score threshold	0.6
Frames until removing unmatched tracks	50
Frames until confirming tentative tracks	5
Kalman Filter [86]	
position weight	$w_p = \frac{1}{10}$
velocity weight	$w_v = \frac{1}{80}$
Classification Network	ResNet-50 [47] Global Average Pooling [40] 1 Fully Connected Layer
Types of Matching	Hungarian Algorithm [48]
1. Appearance Features	Mahalanobis distance [58] < 10.597 & Appearance Features < 1.5
Variables for the Mahalanobis distance [58]	(c_x, c_y)
2. $1 - IoU_{overlap}$	Threshold: 0.8
3. Mahalanobis distance [58]	Threshold: $x_{0.995;2} = 4.605$
4. $1 - IoU_{overlap}$	Threshold: 0.965
5. Euclidean distance	Threshold: $w_{Pred} + h_{Pred}/2$

Table 4.9.: Configurations of the DeepSORT+ Algorithm

To ensure comparability, all trackers utilize a Mask R-CNN network [38] for detection. If a confirmed track is not matched with any detection within 50 frames, it is removed. A tentative track is confirmed after matching detections for five frames. Fine-tuning of the Kalman Filter [86] is achieved through the position weight w_p and velocity weight w_v , similar to section 4.4.1.

DeepSORT+ leverages its classification network to match visually similar grape clusters, identified by their spatially close feature vectors. Only pairs with a Mahalanobis distance [58] smaller than the chi-squared-0.995-quantil $x_{0.995;2} = 10.597$ and a Euclidean distance between the feature vectors below 1.5 are retained.

Like SORT+, the algorithm matches detections and track predictions when the Intersection over Union (*IoU*) overlap exceeds 20%.

Next, using the Mahalanobis distance [58], detections are matched to the multivariate

normal distribution that represents the closest track. Only matches with a [Mahalanobis distance](#) [58] below the chi-squared-0.9-quantil $x_{0.9;2} = 4.605$ are kept, as matches with larger distances are correct only 10% of the time.

The *IoU*-overlap is again utilized to match remaining detections with remaining track predictions, with a minimum *IoU*-overlap of 3.5%.

The last [Hungarian Algorithm](#) [48] matches detections with track predictions, depending on their Euclidean distance. Considering that the Euclidean distance between track prediction and detection relies on the size of the bboxes and the [Kalman Filter](#) [86] makes the track estimate converge onto a stable average size of the grape cluster’s bbox, the track bbox’s width and half of the height are used as a threshold ($w_{pred} + h_{pred}/2$) for matches. This approach aids in recovering matches where [Mask R-CNN](#) [38] detects part of a grape cluster, resulting in no *IoU*-overlap with the track prediction.

4.4.5. ByteTrack Configuration

As described in section 3.4.3, [ByteTrack](#) [97] recovers low-confidence detections by matching them with confirmed tracks. Table 4.10 lists the configurations and specific thresholds utilized by [ByteTrack](#) [97]. To handle cases where [Mask R-CNN](#) [38] detects two bboxes for the same track, [ByteTrack](#) [97] scales the *IoU*-overlap between track predictions and detections based on the confidence scores of the detections. This adjustment ensures that more confident detections are prioritized for matching, potentially increasing matching accuracy.

Detector	Mask R-CNN [38]
Frames keeping unmatched tracks	50
Frames until tentative tracks confirmed	5
Kalman Filter [86]	
position weight	$w_p = \frac{1}{10}$
velocity weight	$w_v = \frac{1}{80}$
High-Confidence Detections	> 0.6
Low-Confidence Detections	> 0.05
Confidence to Initialize Track	> 0.6
Matching (Hungarian Algorithm [48])	$1 - IoU_{overlap}$
1. Confirmed Tracks with High-Confidence Detections	Threshold: 0.965
2. Tentative Tracks with High-Confidence Detections	Threshold: 0.915
3. Confirmed Tracks with Low-Confidence Detections	Threshold: 0.8

Table 4.10.: Configurations of the [ByteTrack](#) Algorithm [97]

To be comparable, all trackers employ a [Mask R-CNN](#) network [38] as the detector. Tracks that fail to be matched with detections are removed after 50 frames. The position weight w_p and velocity weight w_v fine-tune the initialization of the covariance matrices \mathbf{P}_0 , \mathbf{Q}_t and \mathbf{R}_t in the [Kalman Filter](#) [86]. However, adjustments to these weights do not yield significant changes in results.

ByteTrack [97] categorizes detections into high-confidence detections with confidence scores above 60% and low-confidence detections with confidence scores above 5%. Only high-confidence detections can start an initially tentative track, which transitions to the confirmed state after matching high-confidence detections for five subsequent frames. The matching strategy consists of three stages. Firstly, high-confidence detections are matched with predictions of confirmed tracks with at least 3.5% *IoU*-overlap. Next, remaining high-confidence detections are matched with the tentative tracks if the *IoU*-overlap exceeds 8.5%. The last Hungarian Algorithm [48] matches low-confidence detections with predictions of confirmed tracks, requiring a minimum of 20% *IoU*-overlap. Thus, ByteTrack [97] recovers detections, such as occluded grape clusters, potentially decreasing track fragmentation and increasing tracking accuracy.

5. Results

5.1. Detection Results

5.1.1. Optimizer Results

To train the [Mask R-CNN](#) detector [38] both the Adam Optimizer [44] and Stochastic Gradient Descent (SGD) [5] were evaluated using multiple learning rates. Each optimizer is configured with a weight decay of 0.0001 and SGD [5] with a momentum of 0.9 for the entire training. To determine the optimizer's performance, the evaluation utilizes the highest bounding box *mAP* across all epochs during validation on the test subset. The bbox *mAP* as the primary metric was chosen, because the accuracy of the [Mask R-CNN](#) network [38] in detecting bounding boxes is more important than the segmentation accuracy for tracking applications, such as [SORT](#) [12], [DeepSORT](#) [88] and [ByteTrack](#) [97]. In the field *mAP* is traditionally the most important metric to assess detection performance. The Table 5.1, as well as Figure 5.1 and Figure 5.2 show the comparative results of employing Stochastic Gradient Descent (SGD) [5] and the Adam Optimizer [44] with various learning rates.

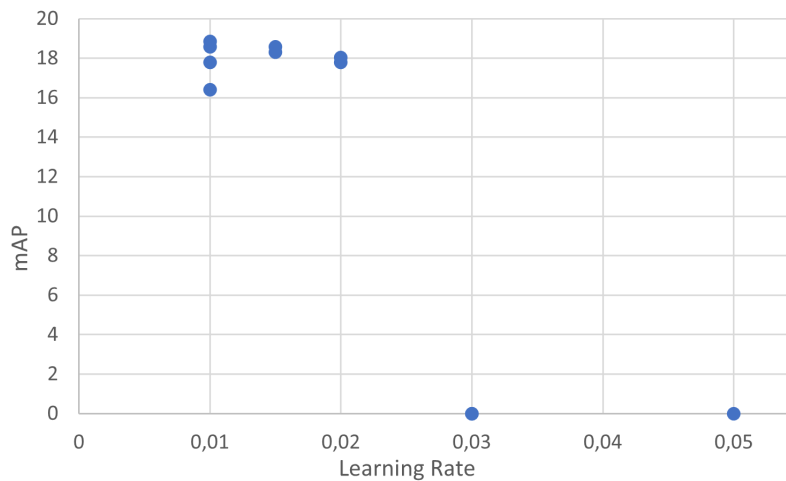


Figure 5.1.:
Testing of Stochastic Gradient Descent (SGD) [5]
[Own Figure]

Both Stochastic Gradient Descent (SGD) [5] and the Adam Optimizer [44] yield similar training outcomes for [Mask R-CNN](#) [38], with bbox *mAP* values ranging between 17% and 19%. The Stochastic Gradient Descent (SGD) Optimizer [5] requires roughly one hundred times the learning rate (step size) compared to the Adam Optimizer [44] to achieve comparable results. The [Mask R-CNN](#) model [38] achieves a peak *mAP* of 19.4%, using the Adam Optimizer [44] with a learning rate of 0.000105 and a weight decay of $= 0.0001$.

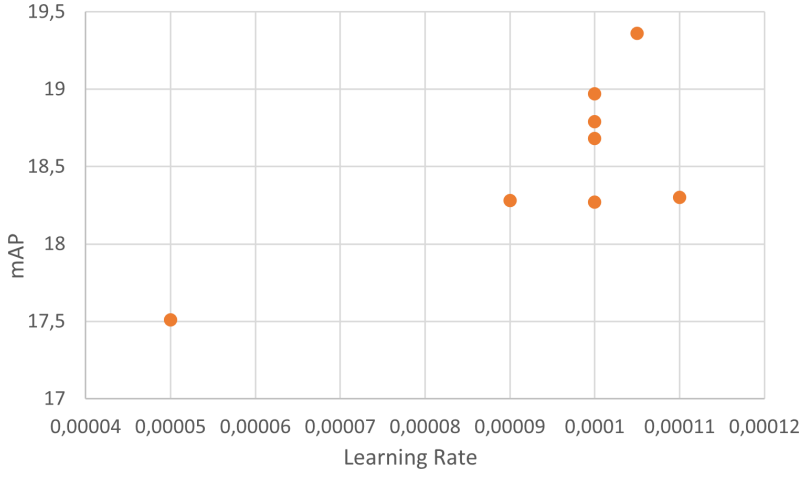


Figure 5.2.:
Testing of Adam [44] [Own
Figure]

Optimizer	Learning Rate	↑ mAP
SGD [5]	0.01	16.4%
SGD [5]	0.01	17.8%
SGD [5]	0.01	18.6%
SGD [5]	0.01	18.9%
SGD [5]	0.015	18.3%
SGD [5]	0.015	18.6%
SGD [5]	0.02	17.8
SGD [5]	0.02	18.0%
SGD [5]	0.03	0.0%
SGD [5]	0.03	0.0%
SGD [5]	0.05	0.0%
Adam [44]	0.00005	17.5%
Adam [44]	0.00009	18.3%
Adam [44]	0.0001	18.3%
Adam [44]	0.0001	18.7%
Adam [44]	0.0001	18.8%
Adam [44]	0.0001	19.0%
Adam [44]	0.000105	19.4%
Adam [44]	0.00011	18.3%

Table 5.1.: Mask R-CNN Optimizer Testing

5.1.2. Mask R-CNN Results

The Mask R-CNN network [38], which is optimized using the Adam Optimizer [44] ($lr = 0.000105$) and thereby achieving a bbox mAP of 19.4%, is utilized in all following detection and tracking evaluations.

Table 5.2 shows the conventional bounding box mAP alongside the mAP values using the IoU -thresholds of 0.5 and 0.75. Additionally, Table 5.2 lists mAP_s , mAP_m and mAP_l by

categorizing the detected bounding boxes into small (*s*), medium (*m*) and large (*l*), like in section 3.2.2 and calculating the mean Average Precision (*mAP*).

Detector	\uparrow mAP	\uparrow mAP ₅₀	\uparrow mAP ₇₅	\uparrow mAP _s	\uparrow mAP _m	\uparrow mAP _l
Mask R-CNN [38]	19.4%	44.8%	14.1%	5.5%	19.2%	28.0%

Table 5.2.: Mask R-CNN Bounding Box Results

Similarly, Table 5.3 evaluates the segmentation performance of Mask R-CNN [38] by presenting comparable *mAP* metrics using the pixel-level segmentation masks instead of the bounding box:

Detector	\uparrow mAP	\uparrow mAP ₅₀	\uparrow mAP ₇₅	\uparrow mAP _s	\uparrow mAP _m	\uparrow mAP _l
Mask R-CNN [38]	14.5%	44.3%	4.5%	4.4%	13.8%	23.0%

Table 5.3.: Mask R-CNN Segmentation Results

5.1.3. Classification Network Results

The [classification network](#) used in DeepSORT [88] is trained to generate appearance feature vectors from detected grape clusters. The feature vectors are spatially close for similar appearances. This model incorporates a ResNet-50 [47] architecture, Global Average Pooling [40] and a fully connected layer. Both Stochastic Gradient Descent [5] and the Adam Optimizer [44] were tested for training the convolutional neural network and evaluated using the *mAP* (See Section 3.2.1). The learning rate scheduler shown in Figure 4.1 in Section 4.3 was employed to dynamically adjust the learning rate. Table 5.4 summarizes the results.

Optimizer	Learning Rate	\uparrow mAP
SGD [5]	0.01	75.7%
SGD [5]	0.0105	76.1%
SGD [5]	0.011	75.9%
SGD [5]	0.0115	75.7%
SGD [5]	0.0125	75.7%
SGD [5]	0.015	75.3%
SGD [5]	0.02	71.1%
SGD [5]	0.1	64.8%
Adam [44]	0.000105	80.6%

Table 5.4.: Classification Network Optimizer Testing

As with the Mask R-CNN [38] optimizer testing in section 5.1.1, the Adam optimizer [44] - with a learning rate of 0.000105 and weight decay of 0.0001 - outperforms SGD [5]. Using this configuration, the convolutional neural network achieves a *mAP* of 80.6%, and is subsequently utilized in all following DeepSORT [88] implementations for feature extraction.

5.2. Tracking Results

The Tables 5.5 and 5.6 fulfill one of the main goals of this study by comparing the tracking results of SORT [12], DeepSORT [88], ByteTrack [97], SORT+ and DeepSORT+. The best value for each metric is highlighted in bold. Since the *MOTP*, *MT*, *PT* and *ML* metrics depend solely on the detector, the variance in results stems from bootstrapping [60]. Therefore, no values were highlighted for these metrics.

Tracker	↑ MOTA	↑ IDF1	↓ IDsw	↑ MOTP
SORT [12]	37.69% ± 0.33%	60.42% ± 0.09%	73.50 ± 1.20	65.41% ± 0.08%
DeepSORT [88]	42.25% ± 0.37%	66.93% ± 0.21%	17.05 ± 0.29	65.46% ± 0.07%
ByteTrack [97]	41.53% ± 0.37%	66.01% ± 0.19%	26.09 ± 0.33	65.44% ± 0.08%
SORT+	41.50% ± 0.40%	66.48% ± 0.18%	28.24 ± 0.49	65.40% ± 0.07%
DeepSORT+	42.71% ± 0.37%	67.10% ± 0.19%	14.90 ± 0.23	65.39% ± 0.08%
PointTrack [8]	(-8.2%)	-	(19)	(66.6%)

Tracker	↑ MT	PT	↓ ML	↓ FM
SORT [12]	57.01 ± 0.34	54.12 ± 0.66	36.04 ± 0.52	43.68 ± 0.76
DeepSORT [88]	57.03 ± 0.36	54.08 ± 0.64	35.84 ± 0.50	45.33 ± 0.81
ByteTrack [97]	56.90 ± 0.37	54.04 ± 0.65	35.65 ± 0.50	44.71 ± 0.80
SORT+	57.14 ± 0.36	54.55 ± 0.65	35.97 ± 0.52	44.70 ± 0.83
DeepSORT+	57.19 ± 0.35	54.23 ± 0.64	36.34 ± 0.50	44.63 ± 0.78

Table 5.5.: Bootstrapped Tracking Results [60]

Table 5.5 shows the bootstrapped tracking results of all five tracking algorithms. Bootstrapping [60] estimates the variability of the results by creating and evaluating 1000 random samples with replacement from the test set. The values in Table 5.5 indicate the average results across all 1000 bootstrap samples, as well as the 95%-confidence interval around this mean. This implies that with 95% confidence, the true mean performance of the tracking algorithms, when applied to similar datasets, would fall within this interval.

The included results of PointTrack from Ariza-Sentís et al. (2023) [8] are in brackets to emphasize the limited comparability of their findings with this research. It is unknown what data was used for testing by Ariza-Sentís et al. (2023) [8]. Depending on the size of the testing subset, the number of instance ID switches (*IDsw*) can be interpreted either positively or negatively. Furthermore, while Ariza-Sentís et al. (2023) [8] employ the Multi-Object Tracking and Segmentation Accuracy (*MOTSA*) and Precision (*MOTSP*) in their evaluation, this study uses *MOTA* and *MOTP*. However, it is important to note that *MOTSA* and *MOTSP* are derived using the same foundational formulas as *MOTA* and *MOTP*, as described in Section 3.2. This similarity in calculation methods provides a basis for comparison, albeit with the noted limitations.

Figure 5.3 shows the *MOTA* scores of all five tracking algorithms, Figure 5.4 the *IDF1* scores, and Figure 5.5 the number of ID switches made by the trackers. If the 95%-confidence

intervals of two individual trackers overlap, it is inconclusive which tracking algorithms performs better.

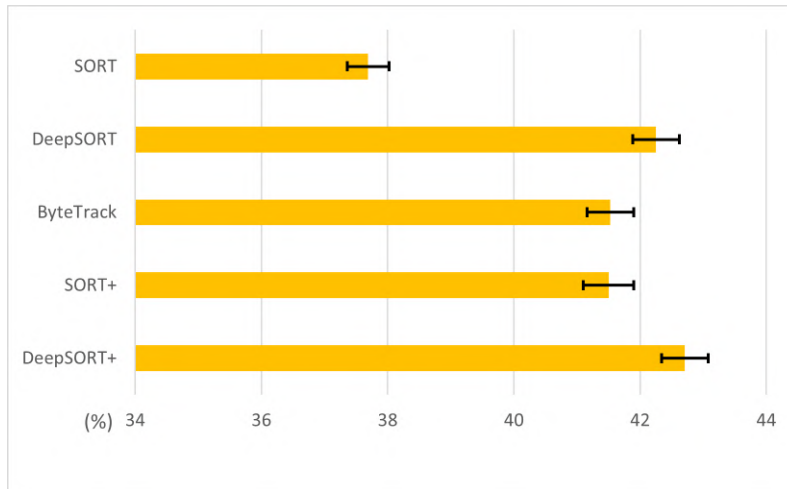


Figure 5.3.:
MOTA scores with their corresponding 95% confidence interval. [Own Figure]

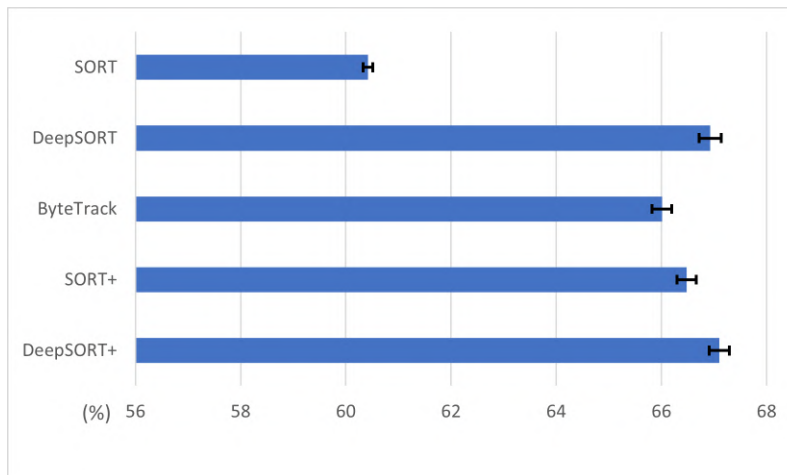


Figure 5.4.:
IDF1 scores with their corresponding 95% confidence interval. [Own Figure]

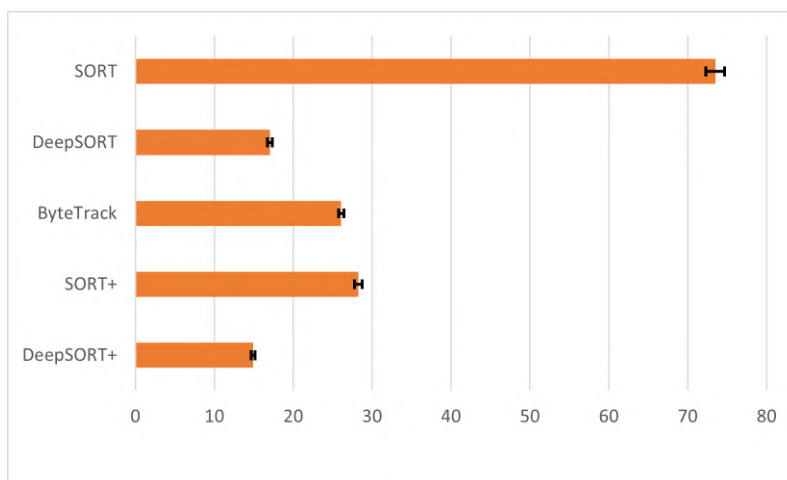


Figure 5.5.:
Number of ID switches with their corresponding 95% confidence interval. [Own Figure]

Victorino et al. (2019) [83] and Olenskyj et al. (2022) [65] show that the count of grape clusters together with the grape cluster area is a strong indicator of grape yield. As

grape yield estimation is an important motivation of this research, the following Table 5.6 presents the number of grape clusters counted by the tracking algorithms.

	AG	HC (f)	HC (b)	SORT	DSORT	ByteTrack	SORT+	DSORT+
Count	147	150	36	313	195	238	192	186
Counting Accuracy	-	-	-	32.7%	95.2%	72.0%	96.8%	100%

Table 5.6.: Grape Cluster (Track) Count and Counting Accuracy

A tracking algorithm counts the grape clusters within the test subset by tallying the number of confirmed tracks. The test subset contains 147 annotated grapes (AG). However, due to the problem of missing annotations within the dataset from Ariza-Sentís, Vélez, and Valente (2023) [7] (refer to Figure 3.1), there are additional visible grape clusters. The manual hand counting (HC) of grapes in the video confirms at least 150 grape clusters in the foreground (f) and 36 grape clusters in the background, resulting in a total of at least 186 visible grape clusters. Instead of the number of annotated grapes, the hand counted grape clusters serve here as a reference for evaluating the counting accuracy of the tracking algorithms.

6. Discussion

6.1. Evaluation of Tracking Algorithms

This study evaluates the performance of SORT [12], DeepSORT [88] and ByteTrack [97] in tracking and counting grape clusters, employing a uniform Mask R-CNN detector [38] for all trackers. Enhanced versions, SORT+ and DeepSORT+, are also tested to assess the impact of improving the matching processes.

Summarizing the results, the selected Mask R-CNN network [38] achieves a bounding box *mAP* of 19.4% and a segmentation *mAP* of 14.5%. Despite the low detection accuracy, all trackers demonstrate robust tracking performance, with *MOTA* scores ranging from 37% to 43% and *IDF1* scores from 60% to 67%.

The tracking algorithms tend to overcount grape clusters by up to 68%. However, DeepSORT [88], SORT+ and DeepSORT+ maintain a more accurate count, with only 5% overcount. This improved counting accuracy is likely due to the use of the Mahalanobis distance [58] as their only unique similarity measure.

DeepSORT+ consistently outperforms the other tracking algorithms across almost every metric, achieving the highest *MOTA* score of 42.7% ($\pm 0.4\%$) and an *IDF1* score of 67.1% ($\pm 0.2\%$), with only 15 ID switches over 186 counted tracks. Meanwhile, the original implementation of DeepSORT [88] performs comparably, with overlapping 95%-confidence intervals in *MOTA* and *IDF1* and 17 ID switches in 195 counted tracks. Due to the overlapping confidence intervals, it remains inconclusive which tracking algorithm consistently achieves better tracking results on similar datasets. As only the classification network distinguishes the top-performing trackers DeepSORT [88] and DeepSORT+, from the other algorithms, it proves highly effective in generating accurate feature vectors to match detections with their corresponding tracks. During evaluation, the classification network achieves a *mAP* of 80.6%.

The original implementation of SORT [12], being the precursor to the other implementations, exhibits the most identity switches — more than twice those of SORT+ and ByteTrack [97], and over four times those of DeepSORT [88] and DeepSORT+. Nevertheless, SORT+ achieves results comparable to ByteTrack [97] and within the reach of DeepSORT(+), showing that integrating additional similarities in the matching process can make SORT [12] a viable tracker.

Choosing a tracking algorithm for grape tracking involves considering not only the tracking performance and counting accuracy, but also the implementation complexity. Although DeepSORT [88] and DeepSORT+ yield the best results in tracking and counting, ByteTrack [97] and SORT+ present viable alternatives. These mathematical algorithms, which only

require training of the detector, increase the number of ID switches by approximately ten, while only slightly underperforming DeepSORT(+) by around 1% in *MOTA* and *IDF1*. SORT+, in particular, achieves good counting accuracy, identifying 192 grape clusters of at least 186 visible clusters, suggesting that lost IDs are recovered later by another ID switch. Because ByteTrack [97] and SORT+ do not rely on a *classification network*, which requires training and labeled instance IDs in the dataset, both algorithms are easier to implement and feasible for broader applications. However, for achieving the best tracking performance, the advanced capabilities of DeepSORT [88] and DeepSORT+ are most promising.

6.2. Contextualizing the Results

This thesis ties in with current research on grape tracking [77, 8, 22], demonstrating that accurate grape tracking and counting is feasible on challenging datasets that resemble real-world conditions.

Ariza-Sentís et al. (2023) [8] evaluate grape tracking using the same dataset [7] as this study, achieving a *MOTSP* of 66.6%, which is comparable to the *MOTP* of 65.4% ($\pm 0.1\%$) reported here. Anyhow, their *MOTSA* score of -8.2% using PointTrack [91] indicates a high number of false positives (*FPs*) and false negatives (*FNs*). This is likely due to the small size of the grape clusters, which are not distinct enough from the background for effective pattern matching. In contrast, SORT [12], DeepSORT [88], ByteTrack [97], SORT+ and DeepSORT+ reach *MOTA* scores between 37% and 43%. This suggests that Kalman Filter [86] based tracking, which estimates the location and velocity of grape clusters, tends to outperform purely geometric based re-identification trackers on datasets where objects blend into the background.

Figure 6.1 shows example images from the three datasets that have been used to evaluate grape tracking in previous studies [77, 8, 22].

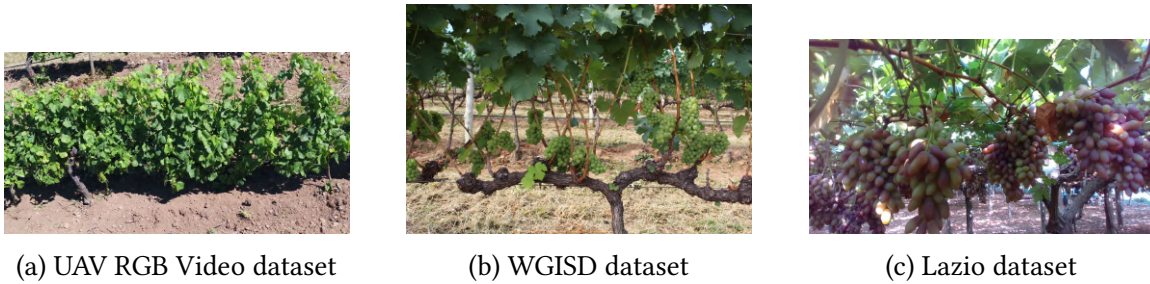


Figure 6.1.: Three datasets presenting unique challenges, impacting tracking performance: a) Ninth annotated image in Row 4.2.1 in the UAV RGB Video dataset [7], b) Annotated image CDY_2037 showing Chardonnay from the WGISD dataset [74], c) Sixth annotated image from “CloseUp 2” in the Lazio dataset [77].

Compared to this study, Ciarfuglia et al. (2023) [22] achieve slightly higher *MOTA* scores up to 46.7% and *MOTP* up to 72.9% on the WGISD dataset [74] using a Mask R-CNN model [38] with a *mAP* of 53.4% and DeepSORT [88] as the tracking algorithm. The better performance of Mask R-CNN [38], with significantly higher *mAP* (53.4% vs. 19.4%), can

be attributed to the dataset's larger grape clusters (0.2% to 7% of the frame) and minimal foliage, which simplify detection. However, the close proximity of clusters in the WGISD dataset [74] complicates the matching process during tracking and evaluation, explaining the modest improvement in tracking performance.

Saraceni et al. (2023) [77] achieve *MOTA* scores up to 66.1% and *IDF1* scores up to 73.7% on the Lazio dataset [77] using a YOLOv5 detector [20] and their AgriSORT [77] tracker. Although AgriSORT [77] performs exceptionally well on some subsets, it does not consistently outperform SORT [12], which achieves *MOTA* scores up to 62.7%. In comparison, the higher *MOTA* scores (62.7% vs. 37.7%) are likely attributable to the dataset's [77] larger, more distinguishable red grape clusters which range from 0.3% to 20% of the frame.

The UAV RGB Video dataset [7] used in this thesis contains at least 186 visible grape clusters, of which 147 are annotated, as well as small grape clusters (0.01% to 0.17% of the frame) and significant occlusion. These challenges hinder the training of Mask R-CNN [38], which achieves 19.4% bbox *mAP* and 14.5% segmentation *mAP*. Notably, Mask R-CNN [38] shows better performances at lower *IoU*-thresholds, which are used for matching detections with *GTs* during evaluation. This can be seen in Table 5.2 and Table 5.3 in Section 5.1.2, where the model detects bboxes and masks with a mAP_{50} of 44.8% and 44.3% respectively, but only 14.1% and 4.5% at a mAP_{75} , using the 0.75 *IoU*-threshold. This discrepancy suggests, that Mask R-CNN [38] regularly detects only parts of a grape cluster. Nonetheless, these partial detections prove sufficient for tracking the corresponding grape cluster, thereby explaining the comparably good tracking results of all algorithms, despite the low detection accuracy.

6.3. Limitations and Further Research

This section discusses the limitations of this study and proposes methods for future research to address these challenges.

DeepSORT [88], SORT+ and DeepSORT+ achieve at least 95% counting accuracy, with DeepSORT+ counting the exact number of hand-counted visible grapes. However, these results should be interpreted with caution due to the unknown margin of error. For instance, it is improbable that DeepSORT+ maintains 100% counting accuracy for every other dataset. To confirm the results, bootstrapping could be utilized, as used for the tracking results (See Table 5.5 in Section 5.2, [60]).

Previous studies by Victorino et al. (2019) [83] and Olenskyj et al. (2022) [65] suggest that the counted grape clusters, together with the grape cluster mask area, are a reliable indicator for grape yield estimation. Using linear regression, various studies [42, 65, 94, 54, 41, 84, 35, 6] demonstrate a linear correlation between yield contributors - such as cluster count, cluster area, or berry count - and the actual grape yield. By applying a linear regression function from one of these studies or an average of multiple correlations, the grape yield from the UAV RGB Video dataset [7] could be estimated. However, evaluating yield estimation is problematic, as the UAV RGB Video dataset [7] provides no data on grape weight or wine harvested from the vine rows.

As noted by Hacking et al. (2019) [36], Jaramillo, Vanden Heuvel, and Petersen (2021) [41], grape bunch area is the most reliable individual yield indicator. Future research could reduce occlusion error by utilizing every track detection to potentially reconstruct the entire grape cluster mask. Furthermore, following Kierdorf et al. (2022) [43], a generative adversarial network trained on defoliated vines can be used to estimate the number of occluded grapes.

The [UAV RGB Video dataset](#) [7] features four rows of “Vitis Vinifera” in its early ripening stage. The characteristics of these vine rows, such as color, size, camera perspective, and cluster proximity, vary depending on the maturity stage, grape variety, and specific environmental influences. Consequently, the detector and tracking algorithms fine-tuned for the [UAV RGB Video dataset](#) [7] are expected to perform worse on datasets with different characteristics.



Figure 6.2.: The second annotated image in Row 7.4.2 showing “Vitis Vinifera” of the [UAV RGB Video dataset](#) [7].

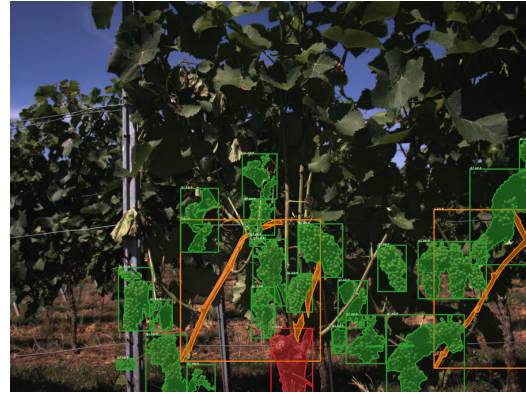


Figure 6.3.: An annotated image showing a dataset with grape clusters very close to each other. [Internal Fraunhofer IOSB dataset]

Comparing Figure 6.2 with Figure 6.3, the proximity of many grape clusters complicates the matching of track detections with ground truths (GTs). Similar to Liu et al. (2020) [53], who utilize circular bounding boxes for tomatoes, adopting trapezoid bboxes might prevent detection losses during Non-Maximum-Suppression (NMS) [64] and improve the matching of track predictions with GTs, thereby reducing false ID switches during evaluation.

To develop a grape tracking algorithm that achieves reliable results across all grape varieties, it is necessary to collect a comprehensive dataset that includes a large variety of grape species, colors, shapes, and ripening stages. Alternatively, future research could extend the work of Bellocchio et al. (2020) [11], who successfully translate fruits in datasets into other fruits utilizing a C-GAN [98]. This approach could enable the training of a detector and tracker for multiple grape species using a dataset that contains only a few varieties.

For the training of a network, the quality of the dataset is a deciding factor for the results. The [UAV RGB Video dataset](#) [7] presents three problems. First, it contains visible grape clusters in the background, which should not be counted. Adjusting the camera angle

to exclude background grapes, using a depth sensor to distinguish between foreground and background [90], or capturing images at night with strong illumination [65, 41, 6, 45, 85] are potential solutions. Each of these approaches is either inconvenient or expensive. Hence, further research could explore training a network to differentiate foreground from background autonomously.

Second, the dataset contains visible grape clusters that were missed during annotation, complicating the training of reliable object detectors and thus limiting tracking performance.

Third, the grape clusters in the high-resolution images (2160×4096 pixels) represent only 0.01% to 0.17% of the frame. Given the increasing use of UAVs in agriculture with an annual market growth of 20% [2], and their ability to capture high-resolution videos, future object detectors should be able to capitalize on high resolutions. Since Mask R-CNN [38] is designed for resolutions up to 1280×720 pixels, future research, like Wu et al. (2021) [89] should consider adapting Mask R-CNN [38] to handle higher resolutions or develop other suited detectors that can exploit high-resolution imagery [34].

Despite these limitations, the low detection accuracies of Mask R-CNN [38] (bbox mAP of 19.4% and segmentation mAP of 14.5%) are partially mitigated in tracking applications, because SORT [12], DeepSORT [88], ByteTrack [97], SORT+ and DeepSORT+ maintain identification over intermittent detections.

This thesis introduces advanced versions of SORT [12] and DeepSORT [88], calling them SORT+ and DeepSORT+, respectively. Table 5.5 in Section 5.2 shows that SORT+ achieves significantly better results compared to the original SORT [12] with $MOTA$ increasing from 37.7% to 41.5% and ID switches decreasing from 73.5 to 28.2. DeepSORT+ exhibits minor improvements over its predecessor, reducing ID switches from 17.1 to 14.9. Using only the IoU -overlap for matching, ByteTrack [97] achieves results comparable to SORT+ and approaches the performance of DeepSORT [88] and DeepSORT+. Therefore, implementing and evaluating ByteTrack+ employing the same enhanced matching strategy as SORT+ and DeepSORT+ is worth investigating in future research.

7. Conclusion

This thesis aims to identify the most effective tracking algorithm for tracking and counting grape clusters in challenging real-world vineyard settings. Accurate tracking and counting of grape clusters allows grape yield estimation and enables precise robotic harvesting that does not damage plants. To this end, five tracking algorithms - SORT [12], DeepSORT [88], ByteTrack [97], SORT+ and DeepSORT+ - are implemented and compared. Enhanced versions, SORT+ and DeepSORT+, incorporate a novel matching algorithm, that leverages the complementary strengths of the *IoU*-overlap, the Mahalanobis distance [58] and the Euclidean distance to improve tracking performance and counting accuracy.

The results demonstrate the feasibility of accurate grape tracking and counting in videos captured by UAVs, even at comparably large distances, despite a low detection accuracy. The uniform Mask R-CNN [38] detector achieves only a *mAP* of 19.4% due to the challenging nature of the UAV RGB Video dataset [7]. All trackers could mostly maintain track identification over partial and missing detections. Furthermore, the trackers SORT+, DeepSORT [88] and DeepSORT+ achieve over 95% counting accuracy, suggesting the viability of these algorithms for estimating grape yield.

DeepSORT+, closely followed by DeepSORT [88], achieves the highest *MOTA* score of 42.7% ($\pm 0.4\%$) and an *IDF1* score of 67.1% ($\pm 0.2\%$), with only 15 ID switches over 186 tracks. However, it remains inconclusive which tracker consistently performs better on unseen similar data, due to the overlapping bootstrapped 95%-confidence intervals of DeepSORT [88] and DeepSORT+.

The enhanced matching process significantly reduces ID switches in SORT [12] from 74 to 28 in SORT+ and improves the counting accuracy from 32.7% to 95.2%. This demonstrates the potential of the advanced matching process and suggests implementing this process in ByteTrack [97] for future studies. Considering the complexity and additional requirements to the dataset for implementing the classification network in DeepSORT(+), SORT+ may present a more viable option in some scenarios.

These findings not only align with, but also expand upon the current research on grape tracking [77, 8, 22]. As Ciarfuglia et al. (2023) [22] use DeepSORT [88] and Saraceni et al. (2023) [77] compare SORT [12], ByteTrack [97] and StrongSORT [29] (improved DeepSORT [88]), differences in tracking performance can largely be attributed to dataset characteristics. Given that all grape tracking research up-to-date focuses solely on single grape species, further research could aim to develop robust grape detection, tracking and counting algorithms applicable to various grape species.

The accurate tracking and counting of small grape clusters that blend into the background are feasible using either [DeepSORT+](#), [DeepSORT](#) [88] or [SORT+](#). [DeepSORT+](#) seems most promising, but [SORT+](#) might be the better choice, where implementing the classification network proves difficult. Using a UAV to count the number of visible grape clusters, and determine their mask area, offers a fast and inexpensive method for grape yield estimation. Therefore, the implications of this research extend beyond academic interest, offering practical insight for vineyard managers. By potentially improving yield estimation and nondestructive robotic farming, these findings support sustainable farming practices and economic growth.

Bibliography

- [1] Abien Fred Agarap. *Deep Learning using Rectified Linear Units (ReLU)*. Feb. 7, 2019. DOI: [10.48550/arXiv.1803.08375](https://doi.org/10.48550/arXiv.1803.08375). arXiv: [1803.08375\[cs, stat\]](https://arxiv.org/abs/1803.08375). URL: <http://arxiv.org/abs/1803.08375> (visited on 04/18/2024).
- [2] *Agriculture Drones Market - Size, Companies & Industry Share*. URL: <https://www.mordorintelligence.com/industry-reports/agriculture-drones-market> (visited on 04/16/2024).
- [3] André Silva Aguiar et al. "Grape Bunch Detection at Different Growth Stages Using Deep Learning Quantized Models". In: *Agronomy* 11.9 (Sept. 2021). Number: 9 Publisher: Multidisciplinary Digital Publishing Institute, p. 1890. ISSN: 2073-4395. DOI: [10.3390/agronomy11091890](https://doi.org/10.3390/agronomy11091890). URL: <https://www.mdpi.com/2073-4395/11/9/1890> (visited on 02/27/2024).
- [4] Nir Aharon, Roy Orfaig, and Ben-Zion Bobrovsky. *BoT-SORT: Robust Associations Multi-Pedestrian Tracking*. July 7, 2022. DOI: [10.48550/arXiv.2206.14651](https://doi.org/10.48550/arXiv.2206.14651). arXiv: [2206.14651\[cs\]](https://arxiv.org/abs/2206.14651). URL: <http://arxiv.org/abs/2206.14651> (visited on 03/05/2024).
- [5] Shun-ichi Amari. "Backpropagation and stochastic gradient descent method". In: *Neurocomputing* 5.4 (June 1, 1993), pp. 185–196. ISSN: 0925-2312. DOI: [10.1016/0925-2312\(93\)90006-0](https://doi.org/10.1016/0925-2312(93)90006-0). URL: <https://www.sciencedirect.com/science/article/pii/S0925231293900060> (visited on 03/28/2024).
- [6] Arturo Aquino et al. "Automated early yield prediction in vineyards from on-the-go image acquisition". In: *Computers and Electronics in Agriculture* 144 (Jan. 1, 2018), pp. 26–36. ISSN: 0168-1699. DOI: [10.1016/j.compag.2017.11.026](https://doi.org/10.1016/j.compag.2017.11.026). URL: <https://www.sciencedirect.com/science/article/pii/S0168169917303964> (visited on 09/25/2023).
- [7] Mar Ariza-Sentís, Sergio Vélez, and João Valente. "Dataset on UAV RGB videos acquired over a vineyard including bunch labels for object detection and tracking". In: *Data in Brief* 46 (Feb. 1, 2023), p. 108848. ISSN: 2352-3409. DOI: [10.1016/j.dib.2022.108848](https://doi.org/10.1016/j.dib.2022.108848). URL: <https://www.sciencedirect.com/science/article/pii/S2352340922010514> (visited on 10/08/2023).
- [8] Mar Ariza-Sentís et al. "Object detection and tracking on UAV RGB videos for early extraction of grape phenotypic traits". In: *Computers and Electronics in Agriculture* 211 (Aug. 1, 2023), p. 108051. ISSN: 0168-1699. DOI: [10.1016/j.compag.2023.108051](https://doi.org/10.1016/j.compag.2023.108051). URL: <https://www.sciencedirect.com/science/article/pii/S0168169923004398> (visited on 09/26/2023).

- [9] Andrea Arlotta, Martina Lippi, and Andrea Gasparri. “An EKF-Based Multi-Object Tracking Framework for a Mobile Robot in a Precision Agriculture Scenario”. In: *2023 European Conference on Mobile Robots (ECMR)*. 2023 European Conference on Mobile Robots (ECMR). ISSN: 2767-8733. Sept. 2023, pp. 1–6. DOI: [10.1109/ECMR59166.2023.10256338](https://doi.org/10.1109/ECMR59166.2023.10256338). URL: <https://ieeexplore.ieee.org/abstract/document/10256338> (visited on 03/01/2024).
- [10] Ms Dhanashree Barbole and Dr Parul Jadhav. “COMPARATIVE ANALYSIS OF DEEP LEARNING ARCHITECTURES FOR GRAPE CLUSTER INSTANCE SEGMENTATION”. In: *INFORMATION TECHNOLOGY IN INDUSTRY* 9.1 (Mar. 1, 2021). Number: 1, pp. 344–352. ISSN: 2203-1731. DOI: [10.17762/itii.v9i1.138](https://doi.org/10.17762/itii.v9i1.138). URL: <http://it-in-industry.org/index.php/itii/article/view/138> (visited on 02/27/2024).
- [11] Enrico Bellocchio et al. “Combining Domain Adaptation and Spatial Consistency for Unseen Fruits Counting: A Quasi-Unsupervised Approach”. In: *IEEE Robotics and Automation Letters* 5.2 (Apr. 2020). Conference Name: IEEE Robotics and Automation Letters, pp. 1079–1086. ISSN: 2377-3766. DOI: [10.1109/LRA.2020.2966398](https://doi.org/10.1109/LRA.2020.2966398). URL: <https://ieeexplore.ieee.org/document/8957569> (visited on 02/27/2024).
- [12] Alex Bewley et al. “Simple Online and Realtime Tracking”. In: *2016 IEEE International Conference on Image Processing (ICIP)*. Sept. 2016, pp. 3464–3468. DOI: [10.1109/ICIP.2016.7533003](https://doi.org/10.1109/ICIP.2016.7533003). arXiv: [1602.00763\[cs\]](https://arxiv.org/abs/1602.00763). URL: <http://arxiv.org/abs/1602.00763> (visited on 10/28/2023).
- [13] Achilleas Blekos et al. “A Grape Dataset for Instance Segmentation and Maturity Estimation”. In: *Agronomy* 13.8 (Aug. 2023). Number: 8 Publisher: Multidisciplinary Digital Publishing Institute, p. 1995. ISSN: 2073-4395. DOI: [10.3390/agronomy13081995](https://doi.org/10.3390/agronomy13081995). URL: <https://www.mdpi.com/2073-4395/13/8/1995> (visited on 11/05/2023).
- [14] Davide Botturi et al. “STEWIE: eSTimating grapE berries number and radius from images using a Weakly supervised nEural network”. In: *2023 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor)*. 2023 IEEE International Workshop on Metrology for Agriculture and Forestry (MetroAgriFor). Nov. 2023, pp. 277–282. DOI: [10.1109/MetroAgriFor58484.2023.10424094](https://doi.org/10.1109/MetroAgriFor58484.2023.10424094). URL: <https://ieeexplore.ieee.org/abstract/document/10424094> (visited on 03/01/2024).
- [15] Sara Bouraya and Abdessamad Belangour. “Deep Learning based Neck Models for Object Detection: A Review and a Benchmarking Study”. In: *International Journal of Advanced Computer Science and Applications* 12.11 (2021). ISSN: 21565570, 2158107X. DOI: [10.14569/IJACSA.2021.0121119](https://doi.org/10.14569/IJACSA.2021.0121119). URL: <http://thesai.org/Publications/ViewPaper?Volume=12&Issue=11&Code=IJACSA&SerialNo=19> (visited on 03/20/2024).
- [16] Richard G. Brereton. “The chi squared and multinormal distributions”. In: *Journal of Chemometrics* 29.1 (2015). _eprint: <https://onlinelibrary.wiley.com/doi/pdf/10.1002/cem.2680>, pp. 9–12. ISSN: 1099-128X. DOI: [10.1002/cem.2680](https://doi.org/10.1002/cem.2680). URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/cem.2680> (visited on 03/25/2024).

-
- [17] Matthew Brown and David G. Lowe. "Automatic Panoramic Image Stitching using Invariant Features". In: *International Journal of Computer Vision* 74.1 (Aug. 1, 2007), pp. 59–73. ISSN: 1573-1405. DOI: [10.1007/s11263-006-0002-3](https://doi.org/10.1007/s11263-006-0002-3). URL: <https://doi.org/10.1007/s11263-006-0002-3> (visited on 03/05/2024).
- [18] *ByteTrack: Multi-Object Tracking by Associating Every Detection Box*. Luffca. URL: <https://www.luffca.com/2023/06/multiple-object-tracking-bytetrack/> (visited on 03/27/2024).
- [19] Jinkun Cao et al. *Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking*. Mar. 15, 2023. arXiv: [2203.14360\[cs\]](https://arxiv.org/abs/2203.14360). URL: <http://arxiv.org/abs/2203.14360> (visited on 12/02/2023).
- [20] Y. Chen et al. "An Improved YOLOv5 Real-time Detection Method for Aircraft Target Detection". In: 2022 27th International Conference on Automation and Computing: Smart Systems and Manufacturing, ICAC 2022. 2022. ISBN: 978-1-66549-807-4. DOI: [10.1109/ICAC55051.2022.9911114](https://doi.org/10.1109/ICAC55051.2022.9911114).
- [21] Yanmin Chen et al. "Instance Segmentation and Number Counting of Grape Berry Images Based on Deep Learning". In: *Applied Sciences* 13.11 (Jan. 2023). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 6751. ISSN: 2076-3417. DOI: [10.3390/app13116751](https://doi.org/10.3390/app13116751). URL: <https://www.mdpi.com/2076-3417/13/11/6751> (visited on 03/01/2024).
- [22] Thomas A. Ciarfuglia et al. "Weakly and semi-supervised detection, segmentation and tracking of table grapes with limited and noisy data". In: *Computers and Electronics in Agriculture* 205 (Feb. 1, 2023), p. 107624. ISSN: 0168-1699. DOI: [10.1016/j.compag.2023.107624](https://doi.org/10.1016/j.compag.2023.107624). URL: <https://www.sciencedirect.com/science/article/pii/S0168169923000121> (visited on 09/26/2023).
- [23] Luca Coviello et al. "GBCNet: In-Field Grape Berries Counting for Yield Estimation by Dilated CNNs". In: *Applied Sciences* 10.14 (Jan. 2020). Number: 14 Publisher: Multidisciplinary Digital Publishing Institute, p. 4870. ISSN: 2076-3417. DOI: [10.3390/app10144870](https://doi.org/10.3390/app10144870). URL: <https://www.mdpi.com/2076-3417/10/14/4870> (visited on 02/28/2024).
- [24] Geng Deng et al. "TSGYE: Two-Stage Grape Yield Estimation". In: *Neural Information Processing*. Ed. by Haiqin Yang et al. Communications in Computer and Information Science. Cham: Springer International Publishing, 2020, pp. 580–588. ISBN: 978-3-030-63820-7. DOI: [10.1007/978-3-030-63820-7_66](https://doi.org/10.1007/978-3-030-63820-7_66).
- [25] Jia Deng et al. "ImageNet: A large-scale hierarchical image database". In: *2009 IEEE Conference on Computer Vision and Pattern Recognition*. 2009 IEEE Conference on Computer Vision and Pattern Recognition. ISSN: 1063-6919. June 2009, pp. 248–255. DOI: [10.1109/CVPR.2009.5206848](https://doi.org/10.1109/CVPR.2009.5206848). URL: <https://ieeexplore.ieee.org/abstract/document/5206848> (visited on 04/18/2024).
- [26] Aishit Dharwal. *Complete Guide to Understanding Precision and Recall Curves*. Analytics India Magazine. Nov. 21, 2021. URL: <https://analyticsindiamag.com/>

- [complete-guide-to-understanding-precision-and-recall-curves/](#) (visited on 03/18/2024).
- [27] Willy Dörfler and Andreas Rieder. *Numerische Mathematik 1/2*. Mar. 2, 2022.
 - [28] Wensheng Du and Ping Liu. “Instance Segmentation and Berry Counting of Table Grape before Thinning Based on AS-SwinT”. In: *Plant Phenomics* 5 (Aug. 29, 2023). Publisher: American Association for the Advancement of Science, p. 0085. DOI: [10.34133/plantphenomics.0085](#). URL: <https://spj.science.org/doi/full/10.34133/plantphenomics.0085> (visited on 03/01/2024).
 - [29] Yunhao Du et al. “StrongSORT: Make DeepSORT Great Again”. In: *IEEE Transactions on Multimedia* 25 (2023). Conference Name: IEEE Transactions on Multimedia, pp. 8725–8737. ISSN: 1941-0077. DOI: [10.1109/TMM.2023.3240881](#). URL: <https://ieeexplore.ieee.org/abstract/document/10032656> (visited on 03/05/2024).
 - [30] Wentai Fang et al. “A novel apple fruit detection and counting methodology based on deep learning and trunk tracking in modern orchard”. In: *Computers and Electronics in Agriculture* 197 (June 1, 2022), p. 107000. ISSN: 0168-1699. DOI: [10.1016/j.compag.2022.107000](#). URL: <https://www.sciencedirect.com/science/article/pii/S0168169922003179> (visited on 09/25/2023).
 - [31] *Figure 1. Flowchart of the Hungarian algorithm. To better understand...* ResearchGate. URL: https://www.researchgate.net/figure/Flowchart-of-the-Hungarian-algorithm-To-better-understand-the-traditional-Hungarian_fig1_376479617 (visited on 03/26/2024).
 - [32] Luca Ghiani et al. “In-Field Automatic Detection of Grape Bunches under a Totally Uncontrolled Environment”. In: *Sensors* 21.11 (Jan. 2021). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 3908. ISSN: 1424-8220. DOI: [10.3390/s21113908](#). URL: <https://www.mdpi.com/1424-8220/21/11/3908> (visited on 02/27/2024).
 - [33] Jonatan Grimm et al. “An adaptable approach to automated visual detection of plant organs with applications in grapevine breeding”. In: *Biosystems Engineering* 183 (July 1, 2019), pp. 170–183. ISSN: 1537-5110. DOI: [10.1016/j.biosystemseng.2019.04.018](#). URL: <https://www.sciencedirect.com/science/article/pii/S1537511018311838> (visited on 02/28/2024).
 - [34] Wei Guo et al. “Geospatial Object Detection in High Resolution Satellite Images Based on Multi-Scale Convolutional Neural Network”. In: *Remote Sensing* 10.1 (Jan. 2018). Number: 1 Publisher: Multidisciplinary Digital Publishing Institute, p. 131. ISSN: 2072-4292. DOI: [10.3390/rs10010131](#). URL: <https://www.mdpi.com/2072-4292/10/1/131> (visited on 04/16/2024).
 - [35] Chris Hacking, Nitesh Poona, and Carlos Poblete-Echeverria. “Vineyard yield estimation using 2-D proximal sensing: a multitemporal approach”. In: *OENO One* 54.4 (Oct. 23, 2020). Number: 4, pp. 793–812. ISSN: 2494-1271. DOI: [10.20870/oeno-](#)

-
- one.2020.54.4.3361. URL: <https://oeno-one.eu/article/view/3361> (visited on 02/28/2024).
- [36] Chris Hacking et al. “Investigating 2-D and 3-D Proximal Remote Sensing Techniques for Vineyard Yield Estimation”. In: *Sensors* 19.17 (Jan. 2019). Number: 17 Publisher: Multidisciplinary Digital Publishing Institute, p. 3652. ISSN: 1424-8220. DOI: [10.3390/s19173652](https://doi.org/10.3390/s19173652). URL: <https://www.mdpi.com/1424-8220/19/17/3652> (visited on 02/28/2024).
- [37] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Google-Books-ID: si3R3Pfa98QC. Cambridge University Press, 2003. 676 pp. ISBN: 978-0-521-54051-3.
- [38] Kaiming He et al. *Mask R-CNN*. Jan. 24, 2018. DOI: [10.48550/arXiv.1703.06870](https://doi.org/10.48550/arXiv.1703.06870). arXiv: [1703.06870\[cs\]](https://arxiv.org/abs/1703.06870). URL: <http://arxiv.org/abs/1703.06870> (visited on 03/02/2024).
- [39] João F. Henriques et al. “High-Speed Tracking with Kernelized Correlation Filters”. In: *IEEE Transactions on Pattern Analysis and Machine Intelligence* 37.3 (Mar. 2015). Conference Name: IEEE Transactions on Pattern Analysis and Machine Intelligence, pp. 583–596. ISSN: 1939-3539. DOI: [10.1109/TPAMI.2014.2345390](https://doi.org/10.1109/TPAMI.2014.2345390). URL: <https://ieeexplore.ieee.org/abstract/document/6870486> (visited on 03/05/2024).
- [40] Ting-Yun Hsiao et al. “Filter-based deep-compression with global average pooling for convolutional networks”. In: *Journal of Systems Architecture* 95 (May 1, 2019), pp. 9–18. ISSN: 1383-7621. DOI: [10.1016/j.sysarc.2019.02.008](https://doi.org/10.1016/j.sysarc.2019.02.008). URL: <https://www.sciencedirect.com/science/article/pii/S1383762118302340> (visited on 04/18/2024).
- [41] Jonathan Jaramillo, Justine Vanden Heuvel, and Kirstin H. Petersen. “Low-Cost, Computer Vision-Based, Prebloom Cluster Count Prediction in Vineyards”. In: *Frontiers in Agronomy* 3 (2021). ISSN: 2673-3218. URL: <https://www.frontiersin.org/articles/10.3389/fagro.2021.648080> (visited on 02/28/2024).
- [42] Muhammad Rizwan Khokher et al. “Early Yield Estimation in Viticulture Based on Grapevine Inflorescence Detection and Counting in Videos”. In: *IEEE Access* 11 (2023), pp. 37790–37808. ISSN: 2169-3536. DOI: [10.1109/ACCESS.2023.3263238](https://doi.org/10.1109/ACCESS.2023.3263238). URL: <https://ieeexplore.ieee.org/document/10087292/> (visited on 12/12/2023).
- [43] Jana Kierdorf et al. “Behind the Leaves: Estimation of Occluded Grapevine Berries With Conditional Generative Adversarial Networks”. In: *Frontiers in Artificial Intelligence* 5 (2022). ISSN: 2624-8212. URL: <https://www.frontiersin.org/articles/10.3389/frai.2022.830026> (visited on 02/27/2024).
- [44] Diederik P. Kingma and Jimmy Ba. *Adam: A Method for Stochastic Optimization*. Jan. 29, 2017. DOI: [10.48550/arXiv.1412.6980](https://doi.org/10.48550/arXiv.1412.6980). arXiv: [1412.6980\[cs\]](https://arxiv.org/abs/1412.6980). URL: <http://arxiv.org/abs/1412.6980> (visited on 03/28/2024).
- [45] A. Koirala et al. “Deep learning for real-time fruit detection and orchard fruit load estimation: benchmarking of ‘MangoYOLO’”. In: *Precision Agriculture* 20.6 (Dec. 1,

- 2019), pp. 1107–1135. ISSN: 1573-1618. DOI: [10.1007/s11119-019-09642-0](https://doi.org/10.1007/s11119-019-09642-0). URL: <https://doi.org/10.1007/s11119-019-09642-0> (visited on 02/27/2024).
- [46] Brett Koonce. “EfficientNet”. In: *Convolutional Neural Networks with Swift for TensorFlow: Image Recognition and Dataset Categorization*. Ed. by Brett Koonce. Berkeley, CA: Apress, 2021, pp. 109–123. ISBN: 978-1-4842-6168-2. DOI: [10.1007/978-1-4842-6168-2_10](https://doi.org/10.1007/978-1-4842-6168-2_10). URL: https://doi.org/10.1007/978-1-4842-6168-2_10 (visited on 03/31/2024).
- [47] Brett Koonce. “ResNet 50”. In: *Convolutional Neural Networks with Swift for TensorFlow: Image Recognition and Dataset Categorization*. Ed. by Brett Koonce. Berkeley, CA: Apress, 2021, pp. 63–72. ISBN: 978-1-4842-6168-2. DOI: [10.1007/978-1-4842-6168-2_6](https://doi.org/10.1007/978-1-4842-6168-2_6). URL: https://doi.org/10.1007/978-1-4842-6168-2_6 (visited on 03/31/2024).
- [48] H. W. Kuhn. “The Hungarian method for the assignment problem”. In: *Naval Research Logistics Quarterly* 2.1 (1955), pp. 83–97. ISSN: 1931-9193. DOI: [10.1002/nav.3800020109](https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109). URL: <https://onlinelibrary.wiley.com/doi/abs/10.1002/nav.3800020109> (visited on 03/21/2024).
- [49] Huipeng Li et al. “A real-time table grape detection method based on improved YOLOv4-tiny network in complex background”. In: *Biosystems Engineering* 212 (Dec. 1, 2021), pp. 347–359. ISSN: 1537-5110. DOI: [10.1016/j.biosystemseng.2021.11.011](https://www.sciencedirect.com/science/article/pii/S1537511021002786). URL: <https://www.sciencedirect.com/science/article/pii/S1537511021002786> (visited on 09/26/2023).
- [50] Yang Li, Zhiyuan Bao, and Jiangtao Qi. “Seedling maize counting method in complex backgrounds based on YOLOV5 and Kalman filter tracking algorithm”. In: *Frontiers in Plant Science* 13 (2022). ISSN: 1664-462X. URL: <https://www.frontiersin.org/journals/plant-science/articles/10.3389/fpls.2022.1030962> (visited on 02/28/2024).
- [51] Tsung-Yi Lin et al. “Feature Pyramid Networks for Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2017, pp. 2117–2125. URL: https://openaccess.thecvf.com/content_cvpr_2017/html/Lin_Feature_Pyramid_Networks_CVPR_2017_paper.html (visited on 03/20/2024).
- [52] Tsung-Yi Lin et al. *Microsoft COCO: Common Objects in Context*. Feb. 20, 2015. DOI: [10.48550/arXiv.1405.0312](https://arxiv.org/abs/1405.0312). arXiv: [1405.0312\[cs\]](https://arxiv.org/abs/1405.0312). URL: <http://arxiv.org/abs/1405.0312> (visited on 03/31/2024).
- [53] Guoxu Liu et al. “YOLO-Tomato: A Robust Algorithm for Tomato Detection Based on YOLOv3”. In: *Sensors* 20.7 (Jan. 2020). Number: 7 Publisher: Multidisciplinary Digital Publishing Institute, p. 2145. ISSN: 1424-8220. DOI: [10.3390/s20072145](https://www.mdpi.com/1424-8220/20/7/2145). URL: <https://www.mdpi.com/1424-8220/20/7/2145> (visited on 02/27/2024).
- [54] Carlos Lopes and Jorge Cadima. “Grapevine bunch weight estimation using image-based features: comparing the predictive performance of number of visible berries and bunch area”. In: *OENO One* 55.4 (Dec. 8, 2021). Number: 4, pp. 209–226. ISSN:

-
- 2494-1271. DOI: [10.20870/oeno-one.2021.55.4.4741](https://doi.org/10.20870/oeno-one.2021.55.4.4741). URL: <https://oeno-one.eu/article/view/4741> (visited on 02/28/2024).
- [55] Shenglian Lu et al. "Swin-Transformer-YOLOv5 for Real-Time Wine Grape Bunch Detection". In: *Remote Sensing* 14.22 (Jan. 2022). Number: 22 Publisher: Multidisciplinary Digital Publishing Institute, p. 5853. ISSN: 2072-4292. DOI: [10.3390/rs14225853](https://doi.org/10.3390/rs14225853). URL: <https://www.mdpi.com/2072-4292/14/22/5853> (visited on 09/26/2023).
- [56] Bruce D Lucas and Takeo Kanade. "An Iterative Image Registration Technique with an Application to Stereo Vision". In: *IJCAI'81: 7th international joint conference on Artificial intelligence*. Vol. 2. Vancouver, Canada, Aug. 1981, pp. 674–679. URL: <https://hal.science/hal-03697340> (visited on 03/05/2024).
- [57] Wei Lyu et al. "A survey on image and video stitching". In: *Virtual Reality & Intelligent Hardware* 1.1 (Feb. 1, 2019), pp. 55–83. ISSN: 2096-5796. DOI: [10.3724/SP.J.2096-5796.2018.0008](https://doi.org/10.3724/SP.J.2096-5796.2018.0008). URL: <https://www.sciencedirect.com/science/article/pii/S2096579619300063> (visited on 03/04/2024).
- [58] Geoffrey John McLachlan. "Mahalanobis Distance". In: *Indian Academy of Sciences* (June 1999).
- [59] Lucas Mohimont et al. "Comparison of Machine Learning and Deep Learning Methods for Grape Cluster Segmentation". In: *Smart and Sustainable Agriculture*. Ed. by Selma Boumerdassi, Mounir Ghogho, and Éric Renault. Communications in Computer and Information Science. Cham: Springer International Publishing, 2021, pp. 84–102. ISBN: 978-3-030-88259-4. DOI: [10.1007/978-3-030-88259-4_7](https://doi.org/10.1007/978-3-030-88259-4_7).
- [60] Christopher Z. Mooney, Robert D. Duval, and Robert Duvall. *Bootstrapping: A Nonparametric Approach to Statistical Inference*. Google-Books-ID: ZxaRC4I2z6sC. SAGE, Aug. 9, 1993. 84 pp. ISBN: 978-0-8039-5381-9.
- [61] Josep Ramon Morros et al. *AI4Agriculture Grape Dataset*. Version 1.0.0. Nov. 9, 2021. DOI: [10.5281/zenodo.5660081](https://doi.org/10.5281/zenodo.5660081). URL: <https://zenodo.org/records/5660081> (visited on 03/08/2024).
- [62] Jishnu Mukhoti et al. *On Batch Normalisation for Approximate Bayesian Inference*. Dec. 24, 2020. DOI: [10.48550/arXiv.2012.13220](https://doi.org/10.48550/arXiv.2012.13220). arXiv: [2012.13220\[cs, stat\]](https://arxiv.org/abs/2012.13220). URL: <http://arxiv.org/abs/2012.13220> (visited on 04/18/2024).
- [63] Anjana K. Nellithimaru and George A. Kantor. "ROLS : Robust Object-Level SLAM for Grape Counting". In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). ISSN: 2160-7516. June 2019, pp. 2648–2656. DOI: [10.1109/CVPRW.2019.00321](https://doi.org/10.1109/CVPRW.2019.00321). URL: <https://ieeexplore.ieee.org/document/9025552> (visited on 03/01/2024).
- [64] A. Neubeck and L. Van Gool. "Efficient Non-Maximum Suppression". In: *18th International Conference on Pattern Recognition (ICPR'06)*. 18th International Conference on Pattern Recognition (ICPR'06). Vol. 3. ISSN: 1051-4651. Aug. 2006, pp. 850–855.

- DOI: [10.1109/ICPR.2006.479](https://doi.org/10.1109/ICPR.2006.479). URL: <https://ieeexplore.ieee.org/abstract/document/1699659> (visited on 04/02/2024).
- [65] Alexander G. Olenskyj et al. “End-to-end deep learning for directly estimating grape yield from ground-based imagery”. In: *Computers and Electronics in Agriculture* 198 (July 1, 2022), p. 107081. ISSN: 0168-1699. DOI: [10.1016/j.compag.2022.107081](https://doi.org/10.1016/j.compag.2022.107081). URL: <https://www.sciencedirect.com/science/article/pii/S0168169922003982> (visited on 02/28/2024).
- [66] Baden Parr, Mathew Legg, and Fakhrul Alam. “Analysis of Depth Cameras for Proximal Sensing of Grapes”. In: *Sensors* 22.11 (Jan. 2022). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 4179. ISSN: 1424-8220. DOI: [10.3390/s22114179](https://doi.org/10.3390/s22114179). URL: <https://www.mdpi.com/1424-8220/22/11/4179> (visited on 02/28/2024).
- [67] Baden Parr, Mathew Legg, and Fakhrul Alam. “Grape yield estimation with a smartphone’s colour and depth cameras using machine learning and computer vision techniques”. In: *Computers and Electronics in Agriculture* 213 (Oct. 1, 2023), p. 108174. ISSN: 0168-1699. DOI: [10.1016/j.compag.2023.108174](https://doi.org/10.1016/j.compag.2023.108174). URL: <https://www.sciencedirect.com/science/article/pii/S0168169923005628> (visited on 03/01/2024).
- [68] Yun Peng, Shengyi Zhao, and Jizhan Liu. “Segmentation of Overlapping Grape Clusters Based on the Depth Region Growing Method”. In: *Electronics* 10.22 (Jan. 2021). Number: 22 Publisher: Multidisciplinary Digital Publishing Institute, p. 2813. ISSN: 2079-9292. DOI: [10.3390/electronics10222813](https://doi.org/10.3390/electronics10222813). URL: <https://www.mdpi.com/2079-9292/10/22/2813> (visited on 02/27/2024).
- [69] Isabel Pinheiro et al. “Deep Learning YOLO-Based Solution for Grape Bunch Detection and Assessment of Biophysical Lesions”. In: *Agronomy* 13.4 (Apr. 2023). Number: 4 Publisher: Multidisciplinary Digital Publishing Institute, p. 1120. ISSN: 2073-4395. DOI: [10.3390/agronomy13041120](https://doi.org/10.3390/agronomy13041120). URL: <https://www.mdpi.com/2073-4395/13/4/1120> (visited on 11/05/2023).
- [70] Joseph Redmon and Ali Farhadi. *YOLOv3: An Incremental Improvement*. Apr. 8, 2018. DOI: [10.48550/arXiv.1804.02767](https://doi.org/10.48550/arXiv.1804.02767). arXiv: [1804.02767\[cs\]](https://arxiv.org/abs/1804.02767). URL: <http://arxiv.org/abs/1804.02767> (visited on 03/05/2024).
- [71] Joseph Redmon et al. “You Only Look Once: Unified, Real-Time Object Detection”. In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016, pp. 779–788. URL: https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html (visited on 03/05/2024).
- [72] Shaoqing Ren et al. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Jan. 6, 2016. DOI: [10.48550/arXiv.1506.01497](https://doi.org/10.48550/arXiv.1506.01497). arXiv: [1506.01497\[cs\]](https://arxiv.org/abs/1506.01497). URL: <http://arxiv.org/abs/1506.01497> (visited on 03/19/2024).

-
- [73] Robert Rudolph et al. "Efficient identification, localization and quantification of grapevine inflorescences in unprepared field images using Fully Convolutional Networks". In: *ArXiv* (July 10, 2018). URL: <https://www.semanticscholar.org/paper/f1b32960bfc772353146a43a4c88c87d85cd7bfa> (visited on 02/28/2024).
- [74] Thiago Santos et al. *Embrapa Wine Grape Instance Segmentation Dataset – Embrapa WGISD*. Version 1.0.0. July 26, 2019. DOI: [10.5281/zenodo.3361736](https://zenodo.org/records/3361736). URL: <https://zenodo.org/records/3361736> (visited on 03/08/2024).
- [75] Thiago T. Santos et al. "Grape detection, segmentation, and tracking using deep neural networks and three-dimensional association". In: *Computers and Electronics in Agriculture* 170 (Mar. 1, 2020), p. 105247. ISSN: 0168-1699. DOI: [10.1016/j.compag.2020.105247](https://www.sciencedirect.com/science/article/pii/S0168169919315765). URL: <https://www.sciencedirect.com/science/article/pii/S0168169919315765> (visited on 09/26/2023).
- [76] Ranjan Sapkota, Dawood Ahmed, and Manoj Karkee. "Comparing YOLOv8 and Mask RCNN for object segmentation in complex orchard environments". In: *Qeios* (Dec. 11, 2023). ISSN: 2632-3834. DOI: [10.32388/ZB9SB0](https://www.qeios.com/read/ZB9SB0). URL: <https://www.qeios.com/read/ZB9SB0> (visited on 04/02/2024).
- [77] Leonardo Saraceni et al. *AgriSORT: A Simple Online Real-time Tracking-by-Detection framework for robotics in precision agriculture*. Sept. 28, 2023. DOI: [10.48550/arXiv.2309.13393](https://arxiv.org/abs/2309.13393). arXiv: [2309.13393\[cs\]](https://arxiv.org/abs/2309.13393). URL: <http://arxiv.org/abs/2309.13393> (visited on 03/01/2024).
- [78] Kah Phooi Seng et al. "Computer Vision and Machine Learning for Viticulture Technology". In: *IEEE Access* 6 (2018). Conference Name: IEEE Access, pp. 67494–67510. ISSN: 2169-3536. DOI: [10.1109/ACCESS.2018.2875862](https://ieeexplore.ieee.org/document/8502206). URL: <https://ieeexplore.ieee.org/document/8502206> (visited on 03/08/2024).
- [79] Lei Shen et al. "Real-time tracking and counting of grape clusters in the field based on channel pruning with YOLOv5s". In: *Computers and Electronics in Agriculture* 206 (Mar. 1, 2023), p. 107662. ISSN: 0168-1699. DOI: [10.1016/j.compag.2023.107662](https://www.sciencedirect.com/science/article/pii/S0168169923000509). URL: <https://www.sciencedirect.com/science/article/pii/S0168169923000509> (visited on 09/25/2023).
- [80] Marco Sozzi et al. "wGrapeUNIPD-DL: An open dataset for white grape bunch detection". In: *Data in Brief* 43 (Aug. 1, 2022), p. 108466. ISSN: 2352-3409. DOI: [10.1016/j.dib.2022.108466](https://www.sciencedirect.com/science/article/pii/S2352340922006606). URL: <https://www.sciencedirect.com/science/article/pii/S2352340922006606> (visited on 11/05/2023).
- [81] Madeleine Stein, Suchet Bargouti, and James Underwood. "Image Based Mango Fruit Detection, Localisation and Yield Estimation Using Multiple View Geometry". In: *Sensors* 16.11 (Nov. 2016). Number: 11 Publisher: Multidisciplinary Digital Publishing Institute, p. 1915. ISSN: 1424-8220. DOI: [10.3390/s16111915](https://www.mdpi.com/1424-8220/16/11/1915). URL: <https://www.mdpi.com/1424-8220/16/11/1915> (visited on 02/27/2024).

- [82] Markus Thill. *The Relationship between the Mahalanobis Distance and the Chi-Squared Distribution*. ML & Stats. Sept. 5, 2017. URL: <https://MarkusThill.github.io/mahalanbis-chi-squared/> (visited on 03/25/2024).
- [83] G. Victorino et al. "Grapevine yield prediction using image analysis - improving the estimation of non-visible bunches". In: *Proceedings 12th EFITA International Conference, 27-29 June 2019* (2019). Accepted: 2022-01-14T13:34:24Z Publisher: EFITA. URL: <https://www.repository.utl.pt/handle/10400.5/23066> (visited on 02/27/2024).
- [84] Gonalo Filipe Victorino et al. "Yield components detection and image-based indicators for non-invasive grapevine yield prediction at different phenological phases". In: *OENO One* 54.4 (Oct. 29, 2020). Number: 4, pp. 833–848. ISSN: 2494-1271. DOI: [10.20870/oeno-one.2020.54.4.3616](https://doi.org/10.20870/oeno-one.2020.54.4.3616). URL: <https://oeno-one.eu/article/view/3616> (visited on 03/01/2024).
- [85] Zhenglin Wang, Kerry Walsh, and Anand Koirala. "Mango Fruit Load Estimation Using a Video Based MangoYOLO–Kalman Filter–Hungarian Algorithm Method". In: *Sensors* 19.12 (Jan. 2019). Number: 12 Publisher: Multidisciplinary Digital Publishing Institute, p. 2742. ISSN: 1424-8220. DOI: [10.3390/s19122742](https://doi.org/10.3390/s19122742). URL: <https://www.mdpi.com/1424-8220/19/12/2742> (visited on 09/26/2023).
- [86] Greg Welch. "An Introduction to the Kalman Filter". In: *UNC Computer Science* (Dec. 25, 1997).
- [87] *Wine - Worldwide | Statista Market Forecast*. Statista. URL: <https://www.statista.com/outlook/cmo/alcoholic-drinks/wine/worldwide> (visited on 04/19/2024).
- [88] 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)*. 2017 IEEE International Conference on Image Processing (ICIP). ISSN: 2381-8549. Sept. 2017, pp. 3645–3649. DOI: [10.1109/ICIP.2017.8296962](https://doi.org/10.1109/ICIP.2017.8296962). URL: <https://ieeexplore.ieee.org/abstract/document/8296962> (visited on 03/05/2024).
- [89] Qifan Wu et al. "Improved Mask R-CNN for Aircraft Detection in Remote Sensing Images". In: *Sensors* 21.8 (Jan. 2021). Number: 8 Publisher: Multidisciplinary Digital Publishing Institute, p. 2618. ISSN: 1424-8220. DOI: [10.3390/s21082618](https://doi.org/10.3390/s21082618). URL: <https://www.mdpi.com/1424-8220/21/8/2618> (visited on 04/16/2024).
- [90] Zhenchao Wu et al. "NDMFCS: An automatic fruit counting system in modern apple orchard using abatement of abnormal fruit detection". In: *Computers and Electronics in Agriculture* 211 (Aug. 1, 2023), p. 108036. ISSN: 0168-1699. DOI: [10.1016/j.compag.2023.108036](https://doi.org/10.1016/j.compag.2023.108036). URL: <https://www.sciencedirect.com/science/article/pii/S0168169923004246> (visited on 09/26/2023).
- [91] Zhenbo Xu et al. *Segment as Points for Efficient Online Multi-Object Tracking and Segmentation*. July 3, 2020. DOI: [10.48550/arXiv.2007.01550](https://doi.org/10.48550/arXiv.2007.01550). arXiv: [2007.01550\[cs\]](https://arxiv.org/abs/2007.01550). URL: <http://arxiv.org/abs/2007.01550> (visited on 10/20/2023).

-
- [92] Shuo-Han Yeh and Shang-Hong Lai. “Real-time video stitching”. In: *2017 IEEE International Conference on Image Processing (ICIP)*. 2017 IEEE International Conference on Image Processing (ICIP). ISSN: 2381-8549. Sept. 2017, pp. 1482–1486. DOI: [10.1109/ICIP.2017.8296528](https://doi.org/10.1109/ICIP.2017.8296528). URL: <https://ieeexplore.ieee.org/document/8296528> (visited on 03/05/2024).
- [93] Laura Zabawa et al. “Detection of Single Grapevine Berries in Images Using Fully Convolutional Neural Networks”. In: *2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW)*. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). ISSN: 2160-7516. June 2019, pp. 2571–2579. DOI: [10.1109/CVPRW.2019.00313](https://doi.org/10.1109/CVPRW.2019.00313). URL: <https://ieeexplore.ieee.org/document/9025531> (visited on 02/27/2024).
- [94] Laura Zabawa et al. “Image-based analysis of yield parameters in viticulture”. In: *Biosystems Engineering* 218 (June 1, 2022), pp. 94–109. ISSN: 1537-5110. DOI: [10.1016/j.biosystemseng.2022.04.009](https://doi.org/10.1016/j.biosystemseng.2022.04.009). URL: <https://www.sciencedirect.com/science/article/pii/S1537511022000861> (visited on 02/27/2024).
- [95] Chuandong Zhang et al. “Grape Cluster Real-Time Detection in Complex Natural Scenes Based on YOLOv5s Deep Learning Network”. In: *Agriculture* 12.8 (Aug. 2022). Number: 8 Publisher: Multidisciplinary Digital Publishing Institute, p. 1242. ISSN: 2077-0472. DOI: [10.3390/agriculture12081242](https://doi.org/10.3390/agriculture12081242). URL: <https://www.mdpi.com/2077-0472/12/8/1242> (visited on 02/27/2024).
- [96] Xin Zhang et al. “Full Stages of Wine Grape Canopy and Clusters”. In: (Mar. 30, 2020). Publisher: Washington State University. DOI: [10.7273/000001846](https://doi.org/10.7273/000001846). URL: <https://rex.libraries.wsu.edu/esploro/outputs/dataset/Full-Stages-of-Wine-Grape-Canopy/99900502168101842> (visited on 03/08/2024).
- [97] Yifu Zhang et al. “ByteTrack: Multi-object Tracking by Associating Every Detection Box”. In: *Computer Vision – ECCV 2022*. Ed. by Shai Avidan et al. Lecture Notes in Computer Science. Cham: Springer Nature Switzerland, 2022, pp. 1–21. ISBN: 978-3-031-20047-2. DOI: [10.1007/978-3-031-20047-2_1](https://doi.org/10.1007/978-3-031-20047-2_1).
- [98] Jun-Yan Zhu et al. “Unpaired Image-To-Image Translation Using Cycle-Consistent Adversarial Networks”. In: *Proceedings of the IEEE International Conference on Computer Vision*. 2017, pp. 2223–2232. URL: https://openaccess.thecvf.com/content_iccv_2017/html/Zhu_Unpaired_Image-To-Image_Translation_ICCV_2017_paper.html (visited on 03/05/2024).

A. Appendix

A.1. Proofs

A.1.1. S_t is symmetric and positive definite

A.1.1.1. Summary of used Matrices and Formulas:

$H \in \mathbb{R}^{4 \times 8}$ changes the dimensions:

$$H = \begin{pmatrix} 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 0 \end{pmatrix} \quad (A.1)$$

$R_t \in \mathbb{R}^{4 \times 4}$ is the [Measurement Noise Covariance Matrix](#):

$$R_t = \text{diag}((w_p Y_t[4])^2, (w_p Y_t[4])^2, 10^{-2}, (w_p Y_t[4])^2) \quad (A.2)$$

$w_p = \frac{1}{10}$ = position weight; $w_v = \frac{1}{80}$ = velocity weight; $Y_t[4]$ = height of the detection bbox matched with the track.

$P_{t_p} \in \mathbb{R}^{8 \times 8}$ is the iteratively updated [Error Covariance Matrix](#) from the [Kalman Filter](#) [86]. It is initialized as:

$$P_{1_p} = A P_0 A^\top + Q_t \quad (A.3)$$

Where:

$$P_0 = \text{diag}((2w_p h)^2, (2w_p h)^2, 10^{-4}, (2w_p h)^2, (10w_v h)^2, (10w_v h)^2, 10^{-10}, (10w_v h)^2) \quad (A.4)$$

$w_p = \frac{1}{10}$ = position weight; $w_v = \frac{1}{80}$ = velocity weight; h = height of the bbox initializing the track.

P_{t_p} changes iteratively, based on whether a detection was matched with the track.

$$P_{t_p} = \begin{cases} A P_{(t-1)_p} A^\top + Q_t & \text{,if no detection was matched with the track} \\ A [I - K_t H] P_{(t-1)_p} A^\top + Q_t & \text{,if a detection was matched with the track} \end{cases} \quad (A.5)$$

$K_t \in \mathbb{R}^{8 \times 4}$ is the [Kalman Gain](#) in frame t :

$$K_t = P_{(t-1)_p} H^\top [H P_{(t-1)_p} H^\top + R_{t-1}]^{-1} \quad (A.6)$$

$\mathbf{A} \in \mathbb{R}^{8 \times 8}$ is the **Motion Matrix**:

$$\mathbf{A} = \begin{pmatrix} 1 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 & 0 & 0 & 0 & 1 \\ 0 & 0 & 0 & 0 & 1 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 1 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \end{pmatrix} \quad (\text{A.7})$$

$\mathbf{Q}_t \in \mathbb{R}^{8 \times 8}$ is the **Process Noise Covariance Matrix**:

$$\mathbf{Q}_t = \text{diag}((w_p X_{t-1}[4])^2, (w_p X_{t-1}[4])^2, 10^{-4}, (w_p X_{t-1}[4])^2, \\ (w_v X_{t-1}[4])^2, (w_v X_{t-1}[4])^2, 10^{-10}, (w_v X_{t-1}[4])^2) \quad (\text{A.8})$$

$w_p = \frac{1}{10}$ = position weight; $w_v = \frac{1}{80}$ = velocity weight; $X_{t-1}[4]$ = height of the bbox estimation in the last frame.

A.1.1.2. Summary of Lemmas used:

Lemma 1:

Let Matrices \mathbf{A} and \mathbf{B} be symmetric, then $\mathbf{A} + \mathbf{B}$ and $\mathbf{A} - \mathbf{B}$ are also symmetric.

Lemma 2:

Let Matrices \mathbf{A} and \mathbf{B} be positive definite, then $\mathbf{A} + \mathbf{B}$ is positive definite.

Lemma 3:

For any symmetric and positive definite matrix \mathbf{A} , the inverse \mathbf{A}^{-1} exists and is also symmetric and positive definite.

Lemma 4:

Let matrix \mathbf{A} be symmetric, then \mathbf{TAT}^\top is symmetric for any matrix \mathbf{T} .

Lemma 5:

Let matrix \mathbf{A} be positive definite and matrix \mathbf{U} has full rank, then \mathbf{UAU}^* is positive definite. For a real matrix \mathbf{U} , it holds that $\mathbf{U}^* = \mathbf{U}^\top$.

Lemma 6:

Let \mathbf{A}, \mathbf{B} be positive definite. If $\mathbf{A} - \mathbf{B}$ is positive definite, this can be expressed as $\mathbf{A} > \mathbf{B}$. It holds that $\mathbf{A} > \mathbf{B}$ if and only if $\mathbf{B}^{-1} > \mathbf{A}^{-1}$.

Claim:

The matrix $\mathbf{S}_t = [\mathbf{HP}_t \mathbf{H}^\top + \mathbf{R}_t] \in \mathbb{R}^{4 \times 4}$ is symmetric and positive definite for every $t \in \mathbb{N}$.

Proof:

To prove this claim for every $t \in \mathbb{N}$, complete induction is used.

Induction Start ($t = 1$):

$$\mathbf{P}_{1_p} = \mathbf{A}\mathbf{P}_0\mathbf{A}^\top + \mathbf{Q}_t \quad (\text{A.9})$$

\mathbf{P}_0 is a diagonal matrix with positive (eigen-)values, hence positive definite. Using Lemma 4 and Lemma 5, $\mathbf{A}\mathbf{P}_0\mathbf{A}^\top$ is symmetric and positive definite, because \mathbf{A} has full rank. Given that \mathbf{Q}_t is also a diagonal matrix with positive (eigen-)values, \mathbf{P}_{1_p} is symmetric and positive definite using Lemma 1 and Lemma 2. Therefore, $\mathbf{S}_t = [\mathbf{H}\mathbf{P}_{1_p}\mathbf{H}^\top + \mathbf{R}_t]$ is symmetric and positive definite, utilizing Lemma 4, Lemma 5, Lemma 1 and Lemma 2, given \mathbf{H} has full rank and \mathbf{R}_t is a diagonal matrix with only positive (eigen-)values.

Induction Requirement (IR): Let \mathbf{P}_{t_p} and $\mathbf{S}_t = [\mathbf{H}\mathbf{P}_{t_p}\mathbf{H}^\top + \mathbf{R}_t]$ be symmetric and positive definite for a particular $t \in \mathbb{N}$.

Induction Conclusion ($t - 1 \rightarrow t$):

Case 1: No detection was matched with the track in the last step.

$$\mathbf{P}_{t_p} = \mathbf{A}\mathbf{P}_{(t-1)_p}\mathbf{A}^\top + \mathbf{Q}_t \quad (\text{A.10})$$

Following the same rationale as the induction start, if $\mathbf{P}_{(t-1)_p}$ is symmetric and positive definite (IR), then \mathbf{P}_{t_p} and subsequently $\mathbf{S}_t = [\mathbf{H}\mathbf{P}_{t_p}\mathbf{H}^\top + \mathbf{R}_t]$ are also symmetric and positive definite. This derives from the full rank of the Matrices \mathbf{A} , \mathbf{H} and \mathbf{Q}_t , \mathbf{R}_t being symmetric and positive definite, using Lemma 4, Lemma 5, Lemma 1 and Lemma 2.

Case 2: A detection was matched with the track during the last step.

$$\mathbf{P}_{t_p} = \mathbf{A}[\mathbf{I} - \mathbf{K}_t\mathbf{H}]\mathbf{P}_{(t-1)_p}\mathbf{A}^\top + \mathbf{Q}_t \quad (\text{A.11})$$

$$\iff \mathbf{P}_{t_p} = \mathbf{A}\mathbf{P}_{(t-1)_p}\mathbf{A}^\top - \mathbf{A}\mathbf{K}_t\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{A}^\top + \mathbf{Q}_t \quad (\text{A.12})$$

$$\iff \mathbf{P}_{t_p} = \mathbf{A}\mathbf{P}_{(t-1)_p}\mathbf{A}^\top - \mathbf{A}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top [\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top + \mathbf{R}_{t-1}]^{-1}\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{A}^\top + \mathbf{Q}_t \quad (\text{A.13})$$

Here, \mathbf{K}_t is well-defined, as $\mathbf{S}_{t-1} = [\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top + \mathbf{R}_{t-1}]$ is symmetric and positive definite (IR), ensuring the inverse $[\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top + \mathbf{R}_{t-1}]^{-1}$ exists and is symmetric, positive definite (Lemma 3). Given \mathbf{Q}_t being positive definite and based on Lemma 2, \mathbf{P}_{t_p} is positive definite, if $\mathbf{A}\mathbf{P}_{(t-1)_p}\mathbf{A}^\top - \mathbf{A}\mathbf{K}_t\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{A}^\top$ is positive definite. Applying the strict order for positive definite matrices, this relationship can be expressed as:

$$\mathbf{P}_{t_p} \text{ pos. dif.} \Leftarrow \mathbf{A}\mathbf{P}_{(t-1)_p}\mathbf{A}^\top > \mathbf{A}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top [\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top + \mathbf{R}_{t-1}]^{-1}\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{A}^\top \quad (\text{A.14})$$

Note that matrix multiplication cannot be used for this order, as the product of two positive definite matrices, is not necessarily positive definite. Only addition, subtraction, and similarity transformations ($\mathbf{A} \rightarrow \mathbf{T}\mathbf{A}\mathbf{T}^\top$) can be used. Given that the motion matrix \mathbf{A} is an upper triangular matrix with 1s on its diagonal, the spectrum $\sigma(\mathbf{A})$ equals $\{1\}$ making \mathbf{A} invertible. By applying a similarity transformation, Equation (A.14) can be changed to:

$$\iff \mathbf{P}_{(t-1)_p} > \mathbf{P}_{(t-1)_p}\mathbf{H}^\top [\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top + \mathbf{R}_{t-1}]^{-1}\mathbf{H}\mathbf{P}_{(t-1)_p} \quad (\text{A.15})$$

By multiplying both sides of Equation (A.15) by \mathbf{H} and \mathbf{H}^\top respectively, it yields:

$$\iff \mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top > \mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top [\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top + \mathbf{R}_{t-1}]^{-1}\mathbf{H}\mathbf{P}_{(t-1)_p}\mathbf{H}^\top \quad (\text{A.16})$$

$$\Longleftrightarrow$$

$$\mathbf{HP}_{(t-1)p} \mathbf{H}^\top (\mathbf{HP}_{(t-1)p} \mathbf{H}^\top)^{-1} \mathbf{HP}_{(t-1)p} \mathbf{H}^\top > \mathbf{HP}_{(t-1)p} \mathbf{H}^\top [\mathbf{HP}_{(t-1)p} \mathbf{H}^\top + \mathbf{R}_{t-1}]^{-1} \mathbf{HP}_{(t-1)p} \mathbf{H}^\top$$

Given that matrix \mathbf{H} has full rank and $\mathbf{P}_{(t-1)p}$ is symmetric and positive definite (IR), $\mathbf{HP}_{(t-1)p} \mathbf{H}^\top$ is also symmetric and positive definite (Lemma 4, Lemma 5) and thus has an inverse (Lemma 3). Using a similarity transformation, gives:

$$\Longleftrightarrow (\mathbf{HP}_{(t-1)p} \mathbf{H}^\top)^{-1} > [\mathbf{HP}_{(t-1)p} \mathbf{H}^\top + \mathbf{R}_{t-1}]^{-1} \quad (\text{A.17})$$

Considering that $[\mathbf{HP}_{(t-1)p} \mathbf{H}^\top + \mathbf{R}_{t-1}]^{-1}$ and $(\mathbf{HP}_{(t-1)p} \mathbf{H}^\top)^{-1}$ are well-defined, Lemma 6 is used onto Equation (A.17):

$$\Longleftrightarrow \mathbf{HP}_{(t-1)p} \mathbf{H}^\top + \mathbf{R}_{t-1} > \mathbf{HP}_{(t-1)p} \mathbf{H}^\top \quad (\text{A.18})$$

$$\Longleftrightarrow \mathbf{R}_{t-1} > 0 \Longleftrightarrow \mathbf{R}_{t-1} \text{ pos. dif.} \quad (\text{A.19})$$

This is a true statement, because \mathbf{R}_{t-1} is positive definite for all $t \in \mathbb{N}$. Thus, \mathbf{P}_{t_p} is positive definite.

Applying Lemma 1, Lemma 3 and Lemma 4 onto Equation (A.13) and using the symmetry of $\mathbf{P}_{(t-1)p}$, shows that \mathbf{P}_{t_p} is symmetric with both \mathbf{Q}_t and \mathbf{R}_t being symmetric for all $t \in \mathbb{N}$.

Utilizing Lemma 1, Lemma 2, Lemma 4 and Lemma 5, proves that $\mathbf{S}_t = [\mathbf{HP}_{t_p} \mathbf{H}^\top + \mathbf{R}_t]$ is symmetric and positive definite in both cases. Since \mathbf{S}_1 is symmetric and positive definite, and this condition holds true for every subsequent \mathbf{S}_{t+1} , \mathbf{S}_t is symmetric and positive definite for all $t \in \mathbb{N}$. \square

A.1.2. $||\mathbf{S}_t^{\frac{1}{2}} \vec{z}|| = ||\mathbf{L}_t^\top \vec{z}||$

In the following, the equality $||\mathbf{S}_t^{\frac{1}{2}} \vec{z}|| = ||\mathbf{L}_t^\top \vec{z}||$ is proven for $\mathbf{S}_t = [\mathbf{HP}_{t_p} \mathbf{H}^\top + \mathbf{R}_t] \in \mathbb{R}^{4 \times 4}$. This equality can similarly be demonstrated for $\mathbf{S}_t = \mathbf{T}[\mathbf{HP}_{t_p} \mathbf{H}^\top + \mathbf{R}_t]\mathbf{T}^\top \in \mathbb{R}^{2 \times 2}$, which is also symmetric and positive definite (Lemma 4, Lemma 5).

As shown in section A.1.1, $\mathbf{S}_t = [\mathbf{HP}_{t_p} \mathbf{H}^\top + \mathbf{R}_t]$ is symmetric and positive definite for every $t \in \mathbb{N}$. Consequently, it's Cholesky decomposition \mathbf{L}_t and the symmetric, positive definite square root $\mathbf{S}_t^{\frac{1}{2}}$ exist, with $\mathbf{S}_t^{\frac{1}{2}} \mathbf{S}_t^{\frac{1}{2}} = \mathbf{S}_t$ and $\mathbf{L}_t \mathbf{L}_t^\top = \mathbf{S}_t$.

Claim:

For all $\vec{z} \in \mathbb{R}^4$ it holds that $||\mathbf{S}_t^{\frac{1}{2}} \vec{z}|| = ||\mathbf{L}_t^\top \vec{z}||$.

Proof:

Let \vec{z} be arbitrarily chosen from \mathbb{R}^4 . Then:

$$\begin{aligned} ||\mathbf{S}_t^{\frac{1}{2}} \vec{z}|| &= (\mathbf{S}_t^{\frac{1}{2}} \vec{z})^\top (\mathbf{S}_t^{\frac{1}{2}} \vec{z}) = \vec{z}^\top (\mathbf{S}_t^{\frac{1}{2}})^\top \mathbf{S}_t^{\frac{1}{2}} \vec{z} \\ &= \vec{z}^\top \mathbf{S}_t \vec{z} \\ &= \vec{z}^\top \mathbf{L}_t \mathbf{L}_t^\top \vec{z} = (\mathbf{L}_t^\top \vec{z})^\top (\mathbf{L}_t^\top \vec{z}) = ||\mathbf{L}_t^\top \vec{z}|| \end{aligned} \quad (\text{A.20})$$

The symmetry of $\mathbf{S}_t^{\frac{1}{2}}$ is used to prove the equality. \square

A.2. List of Abbreviations

Abbreviation	Definition
AG	Annotated Grapes
b	background
bbox	bounding box
COCO	Common Objects in Context
CVAT	Computer Vision Annotation Tool
DeepSORT	Simple Online and Realtime Tracking with a Deep Association Metric
f	foreground
FN	False Negative
FP	False Positive
FPN	Feature Pyramid Network
GNSS	Global navigation satellite system
HC	Hand Counted
IDF1	Identification F1 Score
IDFN	False Negative Identities
IDFP	False Positive Identities
IDP	Identification Precision
IDR	Identification Recall
IDTP	True Positive Identities
IDsw	ID-switches
IoU	Intersection over Union
IR	Induction Requirement
MAE	Mean Average Error
mAP	mean Average Precision
MOTA	Multi Object Tracking Accuracy
MOTP	Multi Object Tracking Precision
MOTS	Multi Object Tracking and Segmentation
MOTSA	Multi Object Tracking and Segmentation Accuracy
MOTP	Multi Object Tracking and Segmentation Precision
pos. dif.	positive definite
R-CNN	Region Based Convolutional Neural Network
ReLU	Rectified Linear Unit
ResNet	Residual Network
RoI	Region of Interest
RPN	Region Proposal Network
SfM	Structure from Motion
SORT	Simple Online and Realtime Tracking
TP	True Positive
UAV	Unmanned Aerial Vehicle

A.3. Mask R-CNN Configuration Table

Table [A.1](#) on the next page outlines the configuration of all components within the [Mask R-CNN](#) [38] framework utilized in this study.

Backbone	ResNet50 [47] 4 stages. Stage 1 is frozen. Pretrained on the COCO dataset [52]
Neck	Feature Pyramid Network (FPN) [51]
In channels	$(1088 \times 2048 \times 256)$, $(544 \times 1024 \times 512)$, $(272 \times 512 \times 1024)$, $(136 \times 256 \times 2048)$
Out channels	$(1088 \times 2048 \times 256)$, $(544 \times 1024 \times 256)$, $(272 \times 512 \times 256)$, $(136 \times 256 \times 256)$, $(68 \times 128 \times 256)$
Region Proposal Network Head	
In channels	$(1088 \times 2048 \times 256)$, $(544 \times 1024 \times 256)$, $(272 \times 512 \times 256)$, $(136 \times 256 \times 256)$, $(68 \times 128 \times 256)$
Anchor Generator	(8×8) , (8×16) , (16×8) with strides 4, 8, 16, 32, 64 for each layer respectively
Bbox $[x, y, h, w]$ coder	mean = $[0, 0, 0, 0]$,
normalization	standard deviation = $[1, 1, 1, 1]$
Bbox loss	L^1 loss
Classification loss	Cross-entropy loss
Non-Maximum-Suppression (NMS) [64]:	
Proposals kept before NMS	2000
Maximum proposal kept after NMS	1000
IoU-threshold	0.6
Region of Interest (RoI) Head	RoI Align
Bbox branch output	$7 \times 7 \times 256$
Mask branch output	$14 \times 14 \times 256$
Strides	4, 8, 16, 32
Bbox Head	2 Fully Connected Layers
In channels	$7 \times 7 \times 256$
Out channels of fully connected layers	1024
Classes	1
Bbox $[x, y, h, w]$ coder	mean = $[0, 0, 0, 0]$,
normalization	standard deviation = $[0.1, 0.1, 0.2, 0.2]$
Bbox loss	L^1 loss
Classification loss	Cross-entropy loss
Mask Head	4 Convolutions
In channels	$14 \times 14 \times 256$
Out channels of convolutional layers	$4 \times (14 \times 14 \times 256)$, $(28 \times 28 \times 256)$, (28×28)
Classes	1
Mask loss	Cross-entropy loss

Table A.1.: Configuration of the Mask R-CNN Network [38]