UNIVERSITAT POLITÈCNICA DE CATALUNYA
BARCELONATECH
Escola Tècnica Superior d'Enginyeria
de Telecomunicació de Barcelona

Ants as liquid brains
Multi-agent computer vision on a discrete lattice

Master Thesis
submitted to the Faculty of the
Escola Tècnica d'Enginyeria de Telecomunicació de Barcelona
Universitat Politècnica de Catalunya
by

Ignasi Nogueiras Marco

In partial fulfillment
of the requirements for the master in
Master's degree in Telecommunications **ENGINEERING** *(MET)*
multimedia specialty

Advisor: Josep Ramon Morros Rubió
Barcelona, Date 10-02-2023

Contents

1 Introduction **6**
1.1 Work plan . . . 7

2 State of the art **8**
2.1 Tracking algorithms . . . 8
2.1.1 SORT . . . 9
2.1.2 Deep SORT . . . 11
2.1.3 ByteTrack . . . 12
2.1.4 OC-SORT . . . 12
2.1.5 Location metrics from OC-SORT . . . 14
2.1.6 Strong SORT . . . 16
2.1.7 Deep OC-SORT . . . 17
2.2 YOLOv8n . . . 18
2.3 Bag of Tricks . . . 20
2.4 Tracking software . . . 20
2.4.1 AnimalTA . . . 20
2.4.2 AnTracks . . . 21
2.4.3 AnTraX . . . 21
2.4.4 idTracker . . . 21
2.4.5 Ctrax . . . 21
2.4.6 DA-Tracker . . . 22

3 Methodology **23**
3.1 Data and Datasets . . . 24
3.1.1 CVAT . . . 30
3.1.2 Lonely Ant Frame Segmentation . . . 31
3.1.3 Purging by Deletion . . . 31
3.1.4 Cleaning by Crop Deletion . . . 32
3.1.5 Labeling strategies summary . . . 32
3.2 Metrics . . . 33
3.2.1 Object Detection Metrics . . . 33
3.2.2 Identity Re-identification Metrics . . . 35
3.2.3 Multiple Object Tracking Metrics . . . 36
3.3 Detection Models . . . 39
3.3.1 YOLOv8 Training . . . 42
3.4 Appearance Model . . . 44
3.4.1 Bag of Tricks Training . . . 44
3.4.2 Appearance Tests . . . 45
3.5 Tracking Models . . . 46

4 Results **47**
4.1 Detection Training . . . 47
4.2 Detection Results . . . 50
4.3 Appearance Training . . . 50

4.4 Appearance Tests Results 53
4.5 Tracking Results 55
4.6 Tracking Results analysis 56

5 Conclusions **60**

6 Future Work **60**

References **61**

List of Figures **64**

List of Tables **64**

List of Acronyms **66**

Revision history and approval record

Revision	Date	Purpose
0	13/03/2023	Document creation
1	17/04/2023	State of the Art revision
2	28/08/2023	Document revision
3	31/08/2023	Document approval
4	01/09/2023	Document submission

DOCUMENT DISTRIBUTION LIST

Name	e-mail
Ignasi Nogueiras Marco	ignasi.nogueiras@estudiantat.upc.edu
Ramon Morros Rubió	ramon.morros@upc.edu

Written by:		Reviewed and approved by:	
Date	28/08/2023	Date	31/08/2023
Name	Ignasi Nogueiras Marco	Name	Ramon Morros Rubio
Position	Project Author	Position	Project Supervisor

Abstract

Multiple object tracking is fundamental for collective behavior research, and ants offer a unique perspective on social structures, making ant tracking a valuable tool for biologists. This master's thesis explores ant tracking using advanced computer vision techniques.

The research begins by assessing the current state of multiple object tracking and its applications in ant tracking. It becomes evident that existing tools require enhancements through state-of-the-art computer vision models.

One primary challenge is the scarcity of suitable data with accurate annotations. This work addresses this issue by collecting new raw data and developing annotation tools.

A YOLOv8n detector, with a validation mAP@50-95 of 85.5%, is trained and integrated into the tracking models. The detection performance decreases during testing but proves crucial for overall tracking improvements.

A BoT appearance descriptor for re-identification, achieving a validation Rank-1 accuracy of 74%, is trained and integrated into the tracking models. Subsequent testing identifies model flaws, leading to its exclusion from the final tracker. However, the analysis highlights the appearance model's potential for future investigation.

The results, obtained using an OC-SORT tracker, establish a baseline for future research, achieving a 49% improvement in HOTA for the testing set.

In conclusion, this master's thesis lays the foundation for future research by preparing data, identifying key components, and establishing an initial baseline.

The codebase of this project is accessible through the following GitHub repository link[1]

[1] `https://github.com/imatge-upc/AntTracking/tree/Ignasi`

1 Introduction

Ant colonies offer an interesting view of collective behavior. The ability to track multiple objects within these colonies is essential for understanding their behavior and has beneficial implications for biology research.

This project explores various approaches within the **tracking by detection** methodology, evaluating **deep learning-based detection** models for precise and efficient **multiple object tracking**. Ants, with their complex movements and unique social structures within colonies, provide an ideal subject for this research.

Recognizing the profound impact that tracking technology can have on biological investigations, this project collaborates closely with the Consejo Superior de Investigaciones Científicas (CSIC). The CSIC, an institution in the field of scientific research, has provided essential unlabeled data that forms the foundation of our research.

Beyond the development of a "multiple object tracking software" tailored to the CSIC's environment, this project also provides the essential "software for training" for each element of the tracker and a user-friendly "tutorial for annotating" new training videos, leveraging the power of automatic tracking results.

As we delve into this thesis, we will meticulously examine the intricacies of the chosen tracking methodology. The accompanying block diagram (Figure 5 within the methodology section) vividly illustrates our approach. Through this dedicated work, we aim to deepen our understanding of ant behavior, offering valuable contributions to the realm of animal behavior studies.

1.1 Work plan

The tracking by detection problem is divided into four critical components: **detection**, **estimation**, **association**, and **track management**, offering a comprehensive solution.

The research plan contains a set of five core objectives, each contributing to the goal of this project, guiding our exploration of multiple object tracking using the tracking by detection methodology:

- **Location Model (Association Component)**: This objective focuses on the application of a location model based on estimations and matchings. The research emphasis lies on the development of position estimators and position matching metrics and algorithms. The performance can be enhanced by the implementation of expert knowledge and heuristics specifically tailored to the ant tracking problem.
- **Appearance Model (Association Component)**: This objective focuses on the training of an appearance description model for capturing distinctive visual features to distinguish individual ants.
- **Detection Model**: The successful localization of ants with accuracy and efficiency is reliant on a robust detection model. This objective encompasses the training of an advanced object detection model, a critical component of our tracking system.
- **Postprocessing**: This step explores the refinement of the tracking results. It involves exploring various techniques, including path matching models, the application of appearance descriptors on sequences, and track interpolation.
- **Datasets Annotation**: We aim to significantly reduce the time required for annotating datasets. This step is fundamental for enhancing dataset readiness and expediting the tracking process.

Data Availability Challenges Throughout the project, we encountered various phases of data availability; each one with a different setting. These phases of data availability were pivotal in shaping our research:

1. The first phase mainly facilitated the analysis of location models.
2. The second phase allowed the training for both the appearance and detection models.
3. The third phase was mainly used for testing and evaluation.

Deviations from Initial Plan These data challenges necessitated adaptations to our research plan. The scarcity of labeled data postponed the path matching model training; at the end, it was discarded as a potential future task. Furthermore, the implementation of expert knowledge was impossible because the experts need this project to obtain that knowledge; at the end of their research, it could be implemented to validate their results.

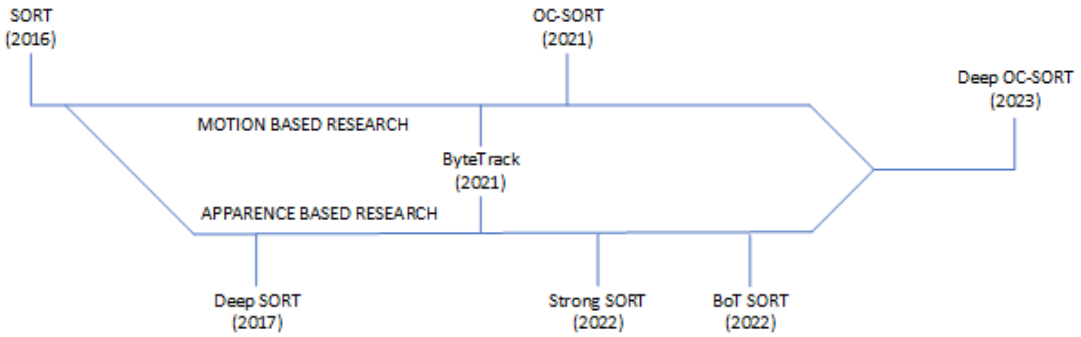


Figure 1: Timeline of some Simple Online and Realtime Tracking (SORT) based algorithms.

2 State of the art

The state of the art section has been divided in four parts:

1. The first part is about tracking models from the literature, the main foundation of this project.
2. The second part roughly describes the YOLOv8n model, the detection model.
3. The third part roughly describes the Bag of Tricks, the appearance model.
4. The fourth part is about existing software for ant tracking, different implementations of the first part models.

2.1 Tracking algorithms

Multiple Object Tracking (MOT) is a branch of computer vision which deals with sequences of images. The goal of MOT is the location and consistent identification of all the target objects within the sequence.

Currently, there are three strategies to solve this problem:

- *Detection-Free Tracking*: This strategy involves manually initializing targets, and the models search for these targets in the subsequent frames. However, it is not contemplated for this project due to the drawbacks of manual initialization and the inability of increasing the number of identities.
- *Tracking by Detection*: In this approach, each frame undergoes an object detection step, followed by an identity association step.
- *End-to-End Tracking*: This strategy aims to detect identities from previous frames or generate new identities within a single step. For example, the MOTR[1] model uses transformers to achieve this. However, it requires a substantial amount of data and is not within the scope of this project.

This subsection provides a review of a family of **tracking by detection** algorithms that follow the structure introduced by **Simple Online and Realtime Tracking (SORT)** (see Figure 1).

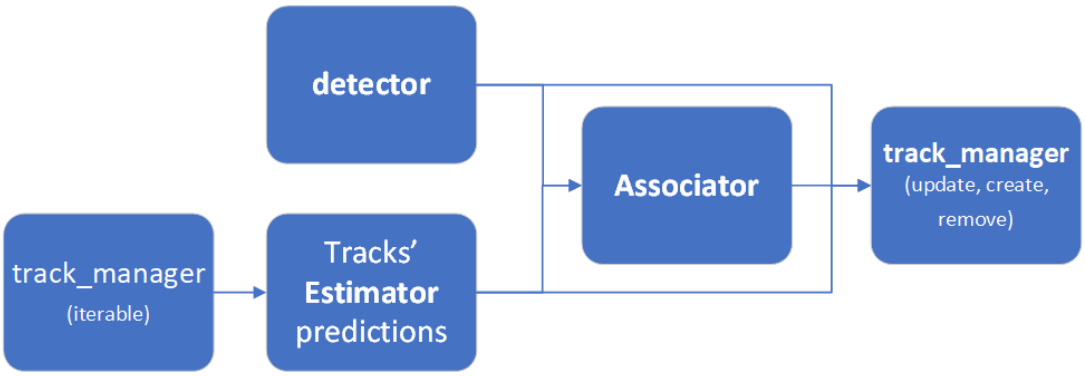


Figure 2: Block diagram of the SORT algorithm.

2.1.1 SORT

SORT[2] is a model published in 2016, it reached competitive accuracies and speed simultaneously. As it can be seen from the block diagram of the figure 2, the authors propose to use a system with the following four components:

1. **Detector**: The detector solves the problem of finding an undefined number of regions that fit with a set of target objects within an image (frame). The authors of SORT use a Convolutional Neural Network (CNN) based model that yields better detections than previous models, this reduce the estimations noise.
2. **Estimator**: The estimator employs a Kalman filter, a recursive filter, to estimate the movement of each individual object based on noisy observations.
3. **Associator**: The associator resolves the matching of the current frame detections with the active tracks estimations. The usage of a Intersection over Union (IoU) distance matrix and the Hungarian algorithm improved the speed with respect other systems.
4. **Track manager**: The track manager creates and deletes the tracks given the other components results.

Detector Mathematically, the detector is defined as an operation that, given an input image I, returns a list of output vectors b_i containing the confidence c, the object position x and y, the object dimensions h and w and the class probability p_c (this project only has one class):

$$Detector(I) = \{b_0, b_1, \ldots, b_n\} \; with \; b_n = \begin{bmatrix} c \\ x \\ y \\ h \\ w \\ p_c \end{bmatrix} \tag{1}$$

Estimator The **Kalman filter**[3] operates in two main steps: prediction and updating. The prediction step estimates the **state mean** x and covariance matrix $\mathbf{P}$ with the state transition matrix $\mathbf{F}$ and the process noise matrix $\mathbf{Q}$:

$$\hat{x}_k = \mathbf{F} \cdot x_{k-1} \quad with\ x = \begin{bmatrix} u \\ v \\ s \\ r \\ \dot{u} \\ \dot{v} \\ \dot{s} \end{bmatrix} \tag{2}$$

$$\hat{\mathbf{P}}_k = \mathbf{F} \cdot \mathbf{P}_{k-1} \cdot \mathbf{F}^T + \mathbf{Q} \tag{3}$$

Where u *and* v is the object center, s is the object scale, r is the object aspect ratio and $\dot{u}, \dot{v}$ *and* $\dot{s}$ are their velocities.

And, the updating step requires an observation z (modified detection matched by the associator) to correct the estimated state mean x and covariance matrix $\mathbf{P}$ which improves the following iterations. This is done with the Kalman gain $\mathbf{K}$, the measurements mask $\mathbf{H}$ and the measurements uncertainty matrix $\mathbf{R}$:

$$\mathbf{K} = \hat{\mathbf{P}}_k \cdot \mathbf{H}^T \cdot (\mathbf{H} \cdot \mathbf{P}_k \cdot \mathbf{H}^T + \mathbf{R})^{-1} \tag{4}$$

$$x_k = \hat{x}_k + \mathbf{K} \cdot (z - \mathbf{H} \cdot \hat{x}_k) \quad with\ z = \begin{bmatrix} u \\ v \\ s \\ r \end{bmatrix} \tag{5}$$

$$\mathbf{P}_k = (\mathbf{K} \cdot \mathbf{H})^{-1} \cdot \hat{\mathbf{P}}_k \cdot (\mathbf{K} \cdot \mathbf{H})^{-T} + \mathbf{K} \cdot \mathbf{R} \cdot \mathbf{K}^T \tag{6}$$

Associator The associator defines a **cost matrix** using negative IoU between detections and estimations, with IoU values below a certain threshold being rejected.

$$IoU(A, B) = \frac{|A \cap B|}{|A \cup B|}$$
$$|A \cap B| = max(min_{x2}(A, B) - max_{x1}(A, B), 0) \cdot max(min_{y2}(A, B) - max_{y1}(A, B), 0)$$
$$|A \cup B| = Area(A) + Area(B) - |A \cap B| \tag{7}$$

The **Hungarian algorithm** associate the rows and columns assigning at most one row to each column and at most one column to each row with the minimum cost. At the end, 3 sets are defined:

$$A = \{(row, col) \mid (row, col) \text{ is in the output of } Hungarian(-\mathbf{IoU})\} \tag{8a}$$

$$B = \{row \mid \forall\, col \nexists\, (row, col) \in A\} \tag{8b}$$

$$C = \{col \mid \forall\, row \nexists\, (row, col) \in A\} \tag{8c}$$

Track manager The Track Manager component uses the set of matched tracks and detections (A in equation 8a) to update the Kalman filter for each assigned detection and track. It also creates new potential tracks from unmatched detections (B in equation 8b) and potentially terminates tracks with missing associations (C in equation 8c).

This model is the basis for subsequent tracking models discussed in this section.

2.1.2 Deep SORT

Deep SORT[4] is a model published in 2017, it upgrades the SORT model with a new associator, containing appearance model.

Appearance model The key innovation in Deep SORT is the introduction of an appearance model. The appearance model defines an operation that transforms the image crop that represent the detected object in the image into a new unitary norm feature space, with the resulting feature vectors represented as r_n.

$$Appearance(crop(\mathbf{I}, b_n)) = r_n \;\mid\; ||\, r_n\, || = 1 \tag{9}$$

Associator The resulting feature space enables online identity definition using classical models such as K-Nearest Neighbours (KNN), distance to the class mean or distance to the last instance. In the paper, the authors use the minimum cosine distance with the last 100 instances of each track as appearance distance as shown in the following equation 10.

$$d_c(i, j) = min\{1 - r_j^T \cdot r_k^i \;\mid\; r_k^i \in \text{"Last 100 appearance of identity i"}\} \tag{10}$$

The Deep SORT associator combines two distances: the location distance, represented by the Mahalanobis distance (equation 11), and the appearance distance. The Mahalanobis distance considers the expected covariance between detections and track estimates.

$$d_m(i, j) = (z_j - z_i)^T \cdot P_i^{-1} \cdot (z_j - z_i) \tag{11}$$

Here, z_j is the j-th bounding box detection, and z_i is the i-th track's bounding box estimation, both having the same shape (z in equation 5). P_i represents the covariance matrix estimation of the i-th track ($\hat{\mathbf{P}}_k$ in equation 3).

The associator employs these distances to make inadmissible match discards. The Mahalanobis distance can be thresholded using the inverse chi-squared (χ^2) distribution, while the appearance distance threshold can be data-driven.

Subsequentially, both distances are combined using a weighted sum to make a cost matrix. The authors argue in favor of using only the appearance distance. This cost matrix is **split by track age** before applying linear assignments within a **matching cascade**, where recently tracked tracks are prioritized.

Finally, a last assignment is done using a IoU cost matrix computed from the unassigned detections and unassigned tracks of age 1 (recently tracked tracks and potential new tracks from the previous frame).

2.1.3 ByteTrack

ByteTrack (BYTE)[5], published in 2021, builds upon the SORT model with a two-staged associator and a deeper detector (YOLOX-X[6]).

Associator The two-staged associator divides **high confidence detections** from **low confidence detections**, allowing for **different assignment criteria**. The authors propose either a combination of appearance for the high confidence and IoU for the low confidence or IoU for both stages.

2.1.4 OC-SORT

Observation-Centric SORT (OC-SORT)[7] is a 2021 model that builds upon the BYTE model. It enhances the motion model, introduces an improved location score, and adds a third association stage, excluding the appearance model for research purposes.

Motion model In OC-SORT, the Kalman filter is improved by freezing its state until a new observation is assigned. The model updates each skipped frame using a linear model when new observations arrive (observation centric), other models update their state with their own estimations (estimation centric). SORT is able to roughly follow a non-linear motion if the time between observations is short enough, OC-SORT reduce the estimation noise originated on missed frames.

Location score The location score from OC-SORT is a global direction metric. To compute this metric, OC-SORT performs a search of the most relevant older observation: the oldest observation from the last $delta_t = 3$ frames, or the newest observation available. The relevant observation and the newest observation available are used to obtain the track global direction vector:

$$direction = \frac{(z_{new} - z_{old})}{||(z_{new} - z_{old})||} \tag{12}$$

A potential global direction is computed using the set of high score detections as potential new observations. At the end, each track has its own global direction and the hypothetical global direction in case of using a certain new detection. For computation, the previous data is stored as four matrix of number of tracks (M) rows and number of high score detections (N) columns:

$$\mathbf{\Delta x_{tracks}} = \begin{bmatrix} direction_1[x] & direction_1[x] & \dots & direction_1[x] \\ direction_2[x] & direction_2[x] & \dots & direction_2[x] \\ \vdots & \vdots & \vdots & \vdots \\ direction_M[x] & direction_M[x] & \dots & direction_M[x] \end{bmatrix} \tag{13a}$$

$$\mathbf{\Delta y_{tracks}} = \begin{bmatrix} direction_1[y] & direction_1[y] & \dots & direction_1[y] \\ direction_2[y] & direction_2[y] & \dots & direction_2[y] \\ \vdots & \vdots & \vdots & \vdots \\ direction_M[y] & direction_M[y] & \dots & direction_M[y] \end{bmatrix} \tag{13b}$$

$$\mathbf{\Delta x_{detections}} = \begin{bmatrix} direction_{1,1}[x] & direction_{1,2}[x] & \dots & direction_{1,N}[x] \\ direction_{2,1}[x] & direction_{2,2}[x] & \dots & direction_{2,N}[x] \\ \vdots & \vdots & \vdots & \vdots \\ direction_{M,1}[x] & direction_{M,2}[x] & \dots & direction_{M,N}[x] \end{bmatrix} \tag{13c}$$

$$\mathbf{\Delta y_{detections}} = \begin{bmatrix} direction_{1,1}[y] & direction_{1,2}[y] & \dots & direction_{1,N}[y] \\ direction_{2,1}[y] & direction_{2,2}[y] & \dots & direction_{2,N}[y] \\ \vdots & \vdots & \vdots & \vdots \\ direction_{M,1}[y] & direction_{M,2}[y] & \dots & direction_{M,N}[y] \end{bmatrix} \tag{13d}$$

This matrixes are operated with the Hadamard product[2] and matrix addition to obtain a component to components dot product, afterwards, the arccosine is applied to obtain the angular distance (from 0 to π) of the track and detections directions, finally it is transformed into a normalized angular score (from -0.5 to 0.5):

$$\begin{aligned} \mathbf{\Delta}\phi &= arccos(\Delta x_{tracks} \odot \Delta x_{detections} + \Delta y_{tracks} \odot \Delta y_{detections}) \\ \phi_{\mathbf{score}} &= 0.5 - \left(\frac{|\mathbf{\Delta}\phi|}{\pi}\right) \end{aligned} \tag{14}$$

[2]The notation for element-wise product will be $\odot$ because $\oslash$ could be used for element-wise division.

Associator During the association step, OC-SORT categorizes the detections in three groups:

1. Initially high score detections.
2. Initially low score detections.
3. High score associations that were not successfully associated

The first step performs a linear assignment with the Hungarian algorithm. Using a weighted sum of the IoU and the angular score from equation 14 pondered by the detections confidence (c_n from the detector's output b_n) as the cost:

$$\mathbf{confidence_{detections}} = \begin{bmatrix} c_1 & c_2 & \dots & c_N \\ c_1 & c_2 & \dots & c_N \\ \vdots & \vdots & \vdots & \vdots \\ c_1 & c_2 & \dots & c_N \end{bmatrix} \tag{15}$$

$$cost_1 = -\mathbf{IoU} - \lambda \cdot (\phi_{\mathbf{score}} \odot \mathbf{confidence_{detections}}) \tag{16}$$

At the end of the first stage, a threshold on the IoU is applied.

The second association stage is the same as the BYTE model (section 2.1.3): the low confidence detections are assigned with a second location metric and threshold.

The third stage of OC-SORT is the same as the second stage; however, it uses the last known observation of the remaining unmatched tracks and the remaining high confidence detections.

2.1.5 Location metrics from OC-SORT

While not originally part of OC-SORT, the publicly available code from the OC-SORT authors includes a range of location metrics. These metrics were initially developed for object detection but have found utility in object tracking scenarios, particularly in improving the accuracy of associations compared to traditional IoU. This section presents an exploration of three location metrics utilized in OC-SORT's codebase.

GIoU The Generalized Intersection over Union[8] is a metric that extends the IoU into the negatives and smoothen it by accounting for both positive and negative scenarios. However, it scores positively bigger boxes. The upgrade consist in subtracting a term computed from the enclosure area and the complementary of the union with the enclosure area:

$$GIoU(A,B) = IoU(A,B) - \frac{|enclosure(A,B)| - |A \cup B|}{|enclosure(A,B)|}$$
$$with\ |enclosure(A,B)| = (max_{x2}(A,B) - min_{x1}(A,B)) \cdot (max_{y2}(A,B) - min_{y1}(A,B)) \tag{17}$$

DIoU The Distance Intersection over Union[9] is a metric similar to the GIoU, but with a different approach. It replace the previous enclosure distance term with a the novel metric called Inner over Outer (IoO). IoO is computed as the center distance normalized by the enclosure diagonal lenght:

$$
\begin{aligned}
&DIoU(A,B) = IoU(A,B) - \frac{InnerDistance(A,B)}{OuterDistance(A,B)} \\
&with\ InnerDistance(A,B) = d(center(A), center(B)) \\
&with\ OuterDistance(A,B) = d(BottomRight(AB), UpperLeft(AB)) \\
&with\ UpperLeft(AB) = [min_{x1}(A,B), min_{y1}(A,B)] \\
&with\ BottomRight(AB) = [max_{x2}(A,B), max_{y2}(A,B)] \\
&with\ d(P_1,P_2) = \sqrt{(P_{2_x} - P_{1_x})^2 + (P_{2_x} - P_{1_x})^2} \\
&\quad \text{using AB as a shortened notation for enclosure(A, B)}
\end{aligned}
\tag{18}
$$

CIoU The Complete Intersection over Union[10], also from the same authors as DIoU, further refines object association by accounting for variations in aspect ratios. This new feature requires the computation of an aspect ratio distance, the authors uses a normalized quadratic angular distance, and an additional cost value based in the IoU. The CIoU score is defined as:

$$
\begin{aligned}
&CIoU(A,B) = DIoU(A,B) - AspectRatioDistance(A,B) \cdot AspectRatioCost(A,B) \\
&with\ AspectRatioDistance(A,B) = \frac{4}{\pi^2} \cdot \left(arctan\left(\frac{B_w}{B_h}\right) - arctan\left(\frac{A_w}{A_h}\right)\right)^2 \\
&with\ AspectRatioCost(A,B) = \frac{AspectRatioDistance(A,B)}{AspectRatioDistance(A,B) + 1 - IoU}
\end{aligned}
\tag{19}
$$

2.1.6 Strong SORT

Strong SORT[11] is a model published in 2022, it upgrades the Deep SORT model, introducing several notable features and improvements:

- **Advanced Modules**: In contrast to Deep SORT, which employs a Faster R-CNN[12, 13] for detection and a simple CNN for appearance modeling, Strong SORT employs more advanced models for both tasks. Specifically, it utilizes YOLOX-X for detection and a Bag of Tricks (BoT)[14] for appearance.
- **Exponential Moving Average (EMA)**: Instead of maintaining a memory of the last 100 instances for the appearance model of each track, Strong SORT uses an exponential moving average.
- **Enhanced Correlation Coefficient (ECC)**: While Strong SORT includes a camera movement compensation model, this feature is not relevant to the objectives of this project.
- **NSA Kalman**: An improvement of the Kalman filter that estimates the measurements noise covariance using the detection confidence score.
- **Motion Cost**: While Deep SORT assigns a weight of 1 on appearance and 0 on motion for track association, Strong SORT suggests a weight of 0.98 on appearance and 0.02 on motion.
- **Vanilla Matching**: Strong SORT departs from the policy of prioritizing the recently tracked tracks during matching.
- **AFLink**: An appearance-free deep learning model to postprocess results by joining tracklets[3]. Note that it requires a substantial amount of training data.
- **Gaussian-Smoothed Interpolation (GSI)**: A space-temporal interpolator to fill the gaps caused by missing detections. It employs a Gaussian process regression to model nonlinear motion.

Gaussian-Smoothed Interpolation The GSI describes the i-th trajectory position p_t at the time t as a Gaussian process with kernel $k(x, x')$ and Gaussian noise ε:

$$\begin{aligned} p_t &= f^{(i)}(t) + \varepsilon \\ with\ f^{(i)} &\in GP\left(0,\ k(\cdot,\cdot)\right) \\ with\ \varepsilon &\sim N\left(0,\ \sigma^2\right) \\ with\ k(x,x') &= e^{-\frac{||x-x'||^2}{2\cdot\lambda 2}} \end{aligned} \tag{20}$$

[3]Tracklet (neologism): Continuous fragment of a track.

The predicted nonlinear positions P^* (smoothed gap filled track) are obtained by fitting the linear predicted positions P (gap filled track) into the previous Gaussian model trained per each track:

$$\begin{gathered}\mathbf{P}^* = \mathbf{K}(\mathbf{F}^*, \mathbf{F}) \cdot \left(\mathbf{K}(\mathbf{F}^*, \mathbf{F}) + \sigma^2 \cdot \mathbf{I}\right)^{-1} \cdot \mathbf{P} \\ with\ K(\cdot,\cdot)\ \text{as the covariance function based on}\ k(\cdot,\cdot)\end{gathered} \tag{21}$$

Because this project only uses the GSI from Strong SORT, the other features were briefly explained.

2.1.7 Deep OC-SORT

Deep OC-SORT[15] is a model published in 2023, it builds upon the OC-SORT model with the incorporation of an appearance model. Although it also includes a feature for camera motion compensation (CMC), this particular functionality is not within the scope of relevance for this project.

Appearance model Similar to the Strong SORT model, Deep OC-SORT employs the BoT model for extracting appearance embeddings (e_t) for each detection. However, Deep OC-SORT uses a new tracking by detection oriented moving average as a track representation; it is a dynamic moving average influenced by the detection confidence (s_{det}), the detection acceptance threshold (σ), and a constant minimum weight of the previous memory (α_f):

$$\begin{gathered}e_t = \alpha_t \cdot e_{t-1} + (1 - \alpha_t) \cdot e^{new} \\ with\ \alpha_t = \alpha_f + (1 - \alpha_f) \cdot \left(1 - \frac{s_{det} - \sigma}{1 - \sigma}\right) \\ \text{when } s_{det} = \sigma : \alpha_t = 1 \rightarrow e_t = e_{t-1} \\ \text{when } s_{det} = 1 : \alpha_t = \alpha_f \rightarrow e_t = \alpha_f \cdot e_{t-1} + (1 - \alpha_f) \cdot e^{new}\end{gathered} \tag{22}$$

This dynamic appearance memory mitigates the impact from low quality frames (for example, a blurred object) and instances of occlusions during the tracked trajectory.

Associator For the task of associating tracks with new detections, Deep OC-SORT employs a method involving cosine similarity. This method results in the creation of an appearance matrix (A_c).

Like Deep SORT, this similarity is combined with the IoU through a weighted sum to obtain the association cost matrix. What distinguishes Deep OC-SORT is its introduction of an individual appearance weight for each pair of tracks and detections. These weights are computed based on the discriminativeness of the appearance data.

In the context of Deep OC-SORT, the authors define the discriminativeness[4] (z_{diff}) as the upper clipped difference of the highest and second highest score of each row or column (low scoring matches are irrelevant) of the appearance matrix (A_c). The mean of row discriminativeness and column discriminativeness plus a base weight (a_w) is their adaptive appearance weight:

$$
\begin{aligned}
w(m,n) &= a_w + \frac{z_{diff}^{track}(\mathbf{A_c}, m) + z_{diff}^{det}(\mathbf{A_c}, n)}{2} \\
with\ z_{diff}^{track}(A_c, m) &= \min\left(\max_i\left(\mathbf{A_c}[m,i]\right) - \max_{j\neq i}\left(\mathbf{A_c}[m,j]\right), \epsilon\right) \\
with\ z_{diff}^{det}(A_c, n) &= \min\left(\max_i\left(\mathbf{A_c}[i,n]\right) - \max_{j\neq i}\left(\mathbf{A_c}[j,n]\right), \epsilon\right)
\end{aligned}
\tag{23}
$$

2.2 YOLOv8n

The **You Only Look Once version 8 nano (YOLOv8n)** model[5], with approximately 3.2 million parameters, is a key component of this research. It implements a non-linear mathematical expression without any real meaning but enough complexity to solve **detection** problems, its true significance emerges after rigorous training. This model is structured around a backbone module, consisting of five levels. Each level becomes progressively smaller in spatial dimensions while incorporating richer features. The output from the last three levels serves as the input for the following pyramidal model.

YOLOv8n employs a pyramidal model to process the output from each level effectively. This processing stage leverages contextual information from both lower and higher-level features, enhancing the model's understanding of the input data when locating the detections.

Towards the final stage of the model, each level independently generates an output within a detection head. These detections are subject to post-processing techniques, primarily focused on suppressing highly overlapped detection results.

The various components and their interconnection can be seen on Figure 3.

This model was included in this project due to its simplicity for deployment and training through the Ultralytics library[16]. The specific rationale behind its selection and its role will be elaborated in the methodology section.

[4]Discriminativeness: The authors of Deep OC-SORT use this word to refer to the quantification of discriminability.

[5]YOLOv8n still does not have a paper for citation. The proper citation before their publication refers to the Ultralytics GitHub page[16].

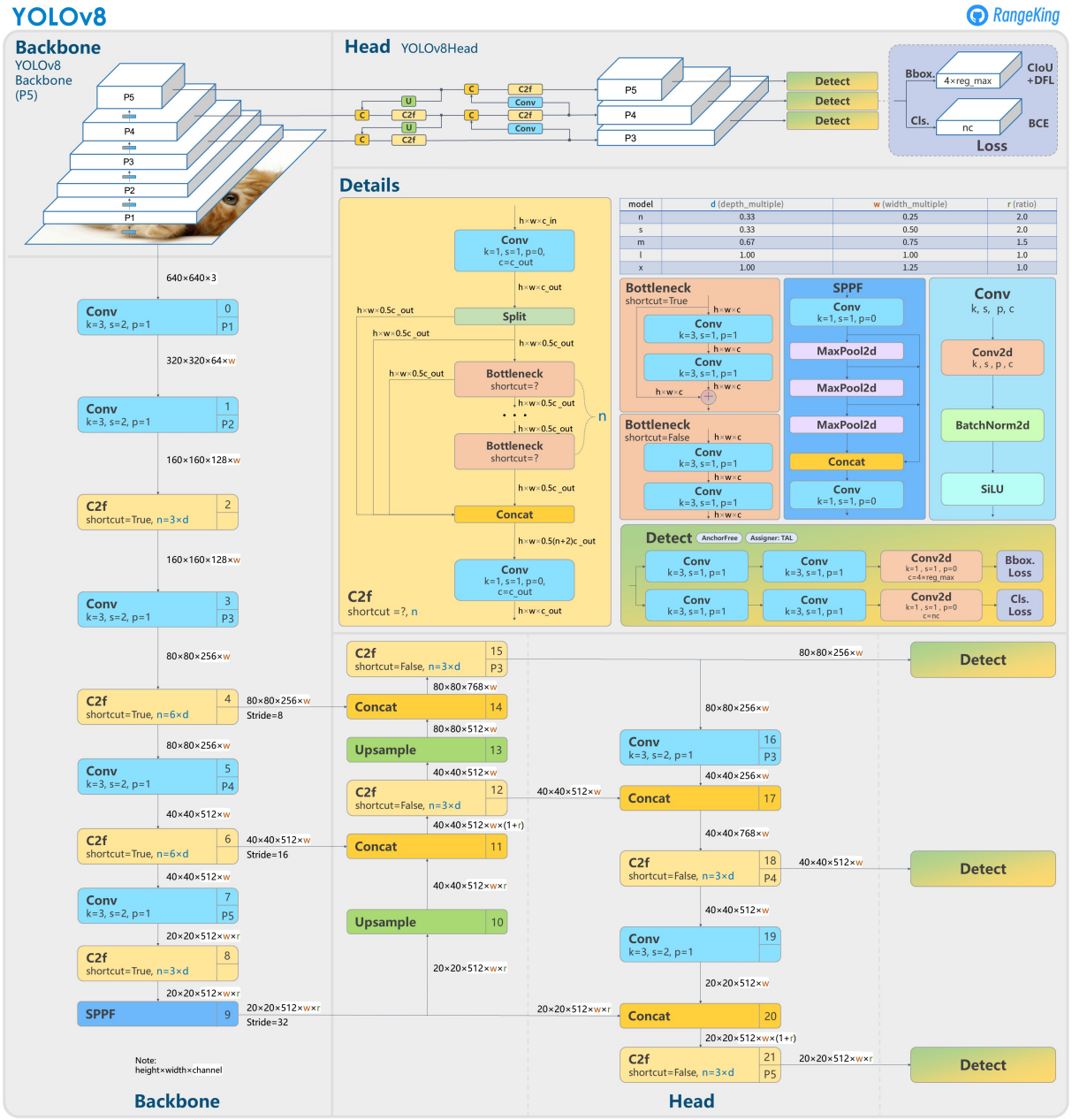


Figure 3: Block diagram of the You Only Look Once version 8 (YOLOv8) architecture extracted from an issue on Ultralytics' GitHub from the user RangeKing[17].

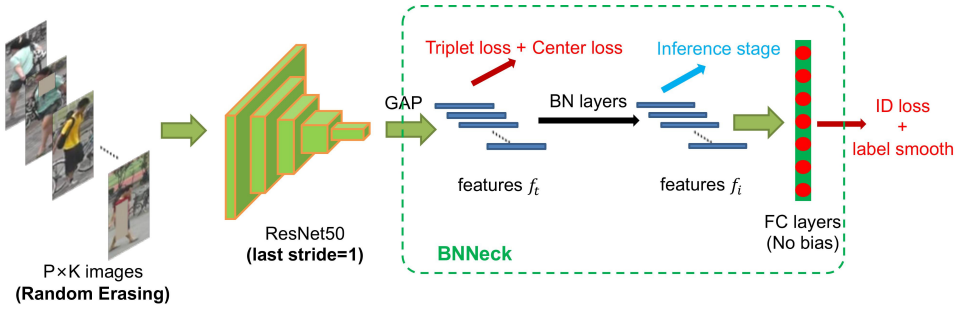


Figure 4: Block diagram of the Bag of Tricks model from the Bag of Tricks paper[14].

2.3 Bag of Tricks

The Bag of Tricks[14] model is a vital component of contemporary appearance-based object tracking. It finds application in Strong SORT and Deep OC-SORT, highlighting its significance in state-of-the-art tracking algorithms; this is also the reason to use BoT within this project.

BoT is a composed by a backbone module and a neck. The last layer from the backbone before the backbone task realization (on a classification head) is extracted as features for the BoT neck. Within the BoT neck there is a sequence of operations consisting in: a batch normalization layer, its output is the appearance metric output, and a generalized mean pooling layer, its output is a classification output. A block diagram of BoT is shown in Figure 4.

The backbone used is the residual network of 50 layers (ResNet50) with non-local blocks. This choice of architecture empowers the model with the ability to capture intricate spatial dependencies and semantic information.

A more comprehensive exploration of the project-specific considerations will be presented in the methodology section.

2.4 Tracking software

Tracking ants and insects poses unique challenges and is not a straightforward task. Moreover, trackers based on appearance models require species-specific tuning. After extensive research on the topic, five applications and one research were found.

2.4.1 AnimalTA

AnimalTA[18], developed in 2023, is an application designed for animal tracking based on minimum distance with respect to the last observation. However, it lacks motion estimation capabilities, making it older in terms of technology compared to SORT. The detection process relies on background extraction, binarization, and connected component analysis. It's worth noting that using connected components for detection can lead to challenges in crowded environments, as objects in close proximity may become fused.

Empirical observations suggest that AnimalTA's tracking performance is inferior to that of the SORT model.

In this context, **background extraction** refers to a technique that models the background from a video to subtract it from each frame, isolating the foreground. This is typically achieved by training a parametric model, such as a Gaussian Mixture Model, with the previous frames as training data. All frames can be used for training since foreground objects are transient and do not affect the model's memory.

2.4.2 AnTracks

AnTracks[19], developed in 2010, is an application that accommodates various object detection tools, including background extraction. Although the specifics of their tracking algorithm are not public, observed performance indicates it falls behind of AnimalTA.

2.4.3 AnTraX

AnTraX[20] is an application from 2020. It uses a background extraction detector. The tracking is performed in two stages:

- The first stage is based on optical flow, it only assign identities when there is a high level of certainty, making tracklets.
- The second stage join the tracklets into tracks using a small CNN-based appearance model (tracklets that are simultaneous are discarded).

2.4.4 idTracker

IdTracker[21, 22], initially developed in 2013 and upgraded in 2018 by a group of researchers from CSIC, relies on background extraction for detection. Similar to Deep SORT, it employs appearance descriptors obtained from a small CNN. However, a notable limitation is its requirement for prior knowledge of the number of tracked objects.

2.4.5 Ctrax

Ctrax[23], introduced in 2009, is noteworthy because its algorithm closely resembles SORT[6] (see Figure 2), although it has certain components downgraded:

1. Ctrax **detection step** uses a background extraction model.
2. The **estimation step** applies a constant velocity model to predict the next position for each track.
3. The **association step** is performed applying the Hungarian algorithm on a Euclidean distance cost matrix.
4. Finally, a **track manager** allows the creation, resuming, destruction and ending of tracks.

[6]SORT is a model from the 2016.

2.4.6 DA-Tracker

DA-Tracker[24], developed as a research project in May 2023, combines detection and tracking using a model extended with domain discriminator modules.

This innovative approach includes an adversarial training strategy for domain adaptation alongside tracking objectives. While this model was discovered during the latter stages of our project, it has been included in the state of the art section, despite not being tested.

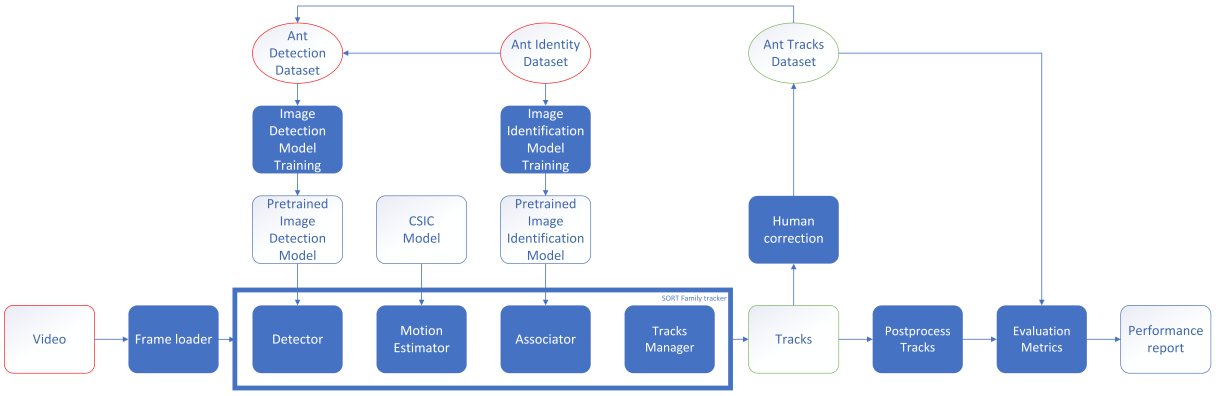


Figure 5: Block diagram of the project

3 Methodology

This section offers an in-depth exploration of the research framework, starting with a high-level overview of the project's key components. The research project encompasses three core components:

- **Detection**: This component focuses on the initial step of locating ants within video frames. It plays a fundamental role in the subsequent stages of re-identification and tracking. It requires a reasonable amount of labeled image data.
- **Re-identification (ReID)**: ReID is the aspect of the research involving the identification of previously detected ants through visual features. It aims to associate ants with their unique visual identities as they move throughout the video, highlighting the case where ants disappear for a long period. It demands a substantial number of labeled identities for a significant volume of cropped detection images.
- **Tracking**: The tracking component is the culmination of the research framework, combining detection and re-identification to trace the ants trajectories across time. The contemplated approches need few labeled video data to test its performance.

This high-level system architecture, as well as the requiered resources, inputs and outputs, is depicted in the Figure 5. Additionally, the three components have their own performance measurments and metrics.

Following the research framework and system overview, the methodology section is structured into five subsections:

1. **Data and Datasets**: This subsection details the gathered data and the methodologies used for labeling the ants for the different tasks.
2. **Metrics**: This subsection comprehensively presents the performance metrics and criteria used to assess the performance of the detection (mAP), re-identification (rank-N), and tracking models (HOTA).
3. **Detection model**: This subsection explores the specifics of both an initial deductive model and the chosen detection model, including their implementation, training processes (for the chosen model), and key considerations.
4. **Appearance model**: This subsection delves into the specificities of the appearance model including the implementation, training process and key considerations.
5. **Tracking**: Finally, we explore the tracking component, were detection, re-identification, motion estimation and location metrics are integrated into a cohesive system for continuous multiple ants tracking.

Each subsection corresponds to a critical aspect of the research and is discussed in detail.

The software development took place within a Python 3.10.12 environment on a Ubuntu (Linux) platform. Access to the codebase of this project is available through the GitHub repository link[7].

3.1 Data and Datasets

The input data for visual tracking consists of a series of videos, each characterized by specific and common attributes (see Table 1 for the common attributes). These videos form the basis for our experiments and the subsequent development of our tracking system.

Table 1: Input video characterization

Feature Name	Value
Frame width	4000 px
Frame height	2992 px
Frame rate	15 fps
Color channels	Grayscale as RGB
Scene type	Static
Camera location	Above
Encoding	MP4

[7] https://github.com/imatge-upc/AntTracking/tree/Ignasi

It is important to notice that the static nature of the scenes allowed the omision of the camera motion models. The frame rate of 15 fps is relatively low and it could affect the ability to roughly follow non-linear motion. Nevertheless, the camera location reduces the occlusions, simplifying the tracking problem. The high resolution (4000x2992 px) is beneficial to experiment with the appearance models.

Video content Our datasets consists of videos with varying content. Initially, we had access to a single one-hour video containing multiple ants.

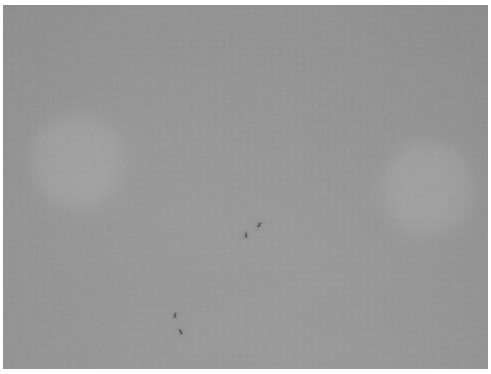
Figure 6a displays a frame from the initial video, offering a view of the scene, including a zoomed-in perspective to observe a net that affected the results. It is important to note that the smooth table background differs from the current one employed by the CSIC researchers, they also stopped using the net. As a result, this video was eventually replaced once additional data became available.

The CSIC researchers recorded 14 additional videos within two sessions.

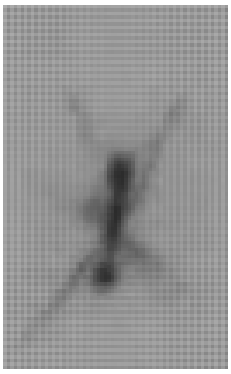
The first session included six 1-hour and six 45-minute videos. The new setting featured a **hexagonal laberinth table** background, as depicted in Figure 7. These recordings primarily served the purpose of training an appearance model and introduced several unique characteristics:

- Each frame featured at most one ant.
- Whenever the CSIC researchers introduced a new ant to the table, it was distinct from all other ants in the 12 videos.
- During ant exchanges, the CSIC researchers hands briefly appears in the frame, resulting in a temporary trace of corrupted pixels.

The second session contains two 1-hour videos. These videos featured a tray contained multiple ants over the hexagonal laberinth table from the previous session, as illustrated in Figure 8. One of these recordings featured 87 ants, while the other contained 46 ants.



(a) A frame from the initial video.



(b) A crop from the initial video.

Figure 6: The Figure 6b exposes the net on the upper layer of the Figure 6a.

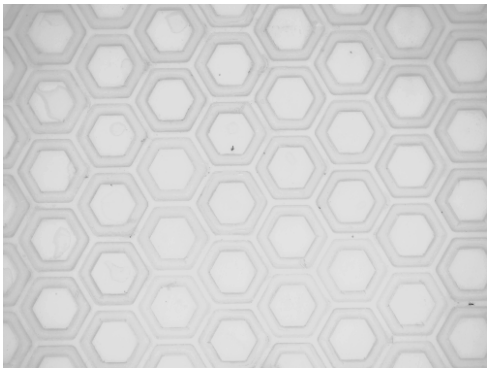


Figure 7: A frame from one of the second set of videos.

(a) A frame from the tray video with 87 ants.

(b) A frame from the tray video with 46 ants.

Figure 8: Frames from the tray videos setup.

To make the collected data useful for the tracking system, it must undergo a labeling process. The main annotation format that was used is the **MOT Challenge format**[25], which is well-suited for both object tracking and object detection in videos. This format is implemented as a Comma-Separated Values (CSV) text file per video with each line containing 10 fields as described in Table 2:

Table 2: MOT Challenge format with default values.

Column	Field Name	Detection	Tracking
1	Frame number	<frame>	<frame>
2	Identity number	-1	<id>
3	Detection left	<bb_left>	<bb_left>
4	Detection top	<bb_top>	<bb_top>
5	Detection width	<bb_width>	<bb_width>
6	Detection height	<bb_height>	<bb_height>
7	Detection confidence	<conf>	<conf>
8	3D World x	-1	-1
9	3D World y	-1	-1
10	3D World z	-1	-1

In addition to the main MOT format, this project employs various dataset formats to simplify the training and deployment of different models. These formats include the You Only Look Once (YOLO) dataset format[26] for image object detection, the Market-1501 dataset format[27] for appearance modeling, and a custom format tailored for segmenting valid frames and IDs from the appearance videos.

YOLO Format The YOLO format is utilized for labeling single images extracted from videos. Each image is associated with an equally named CSV text-file that contains the following column fields for each detection:

"class id, x-axis center, y-axis center, width, height"

It's important to note that the image fields are normalized based on the image shape. Additionally, the YOLO format defines a specific folder structure that separates images and labels into subfolders stemming from the same root. A YAML file is used to describe the data paths and class indexes for each dataset.

A tool was developed to convert the videos with a MOT Challenge annotations into images with a valid YOLO format dataset. This tool offers the flexibility to crop images (and the corresponding labels) into different segments, ensuring that each segment contains at least one whole ant in a random position. The cropping feature maximizing the number of whole ants within each crop while guaranteeing that each ant is fully visible in at least one crop. Figure 9 shows one example of the results of this tool on a video.

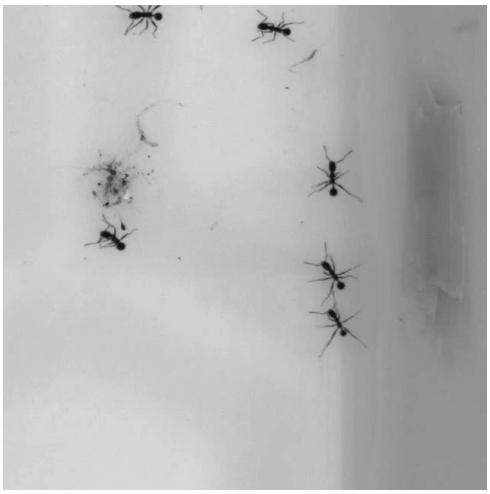


Figure 9: A crop of 640x640 px generated from a video with a MOT Challenge annotations file.

Market-1501 Format Market-1501 is a widely-used reidentification dataset organized based on folder and filename structures. In this project, we mimic this format to simplify the deployment of the appearance model training, we call it Market-1501 Format.

In Market-1501 Format, each "ground truth detection" is cropped from the video and saved as an image within one of three folders: train, test, or test query. and contains its identity on the filename. The identity of each detection is encoded in the filename.

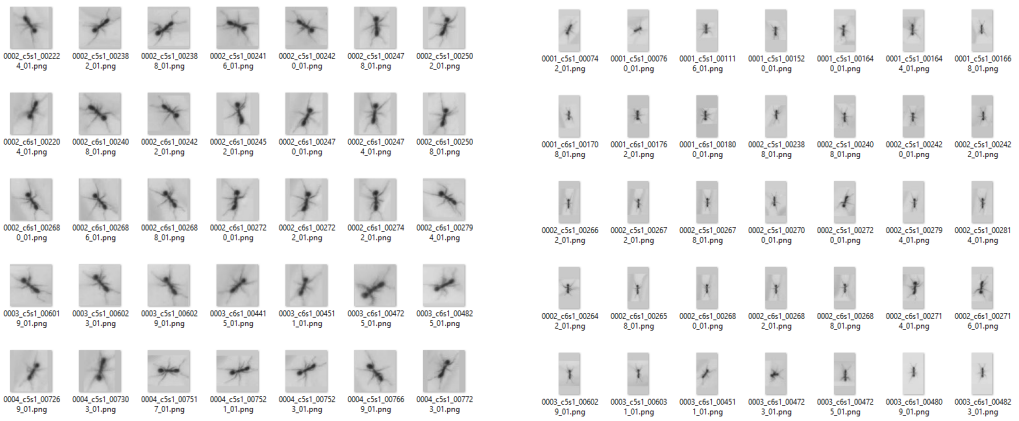
To convert the videos with MOT Challenge annotations into a Market-1501 dataset, a pair of tools were developed. The first tool generates square crops using one of three strategies: "cropping and padding", "cropping, padding the small size, and reshaping", or "cropping and reshaping". The second tool generates rectangular crops with the head of the ant pointing downwards; the shape of the crop is adjusted using similar methods to those employed by the first tool, with the "padding the small size" option in the second tool being replaced by "padding to obtain the correct aspect ratio."

The process of orienting the head of the ant downwards involves four steps:

1. The segmentation of the cropped ant by thresholding[8] the luminance level of the image.
2. The computation of the direction of the body of the ant through principal component analysis.
3. Determining the movement direction of the ant by comparing the current frame position with the nearest frame were the ants appears (if it is a previous frame, the origin point is the previous frame).
4. Selecting the body direction of the ant that minimize the angular difference with the movement direction.

[8]Thresholding (neologism): Act of splitting a set of values using a threshold value.

Figure 10 provides an example of the output from each tool. In both cases, padding was applied to one side before reshaping the crop. While there is an improvement, it's worth noting that the orientation solution in Figure 10b has some limitations, with a total of 35 instances, including 17 correct cases, 15 upside-down cases, and 3 instances with incorrect orientation.



(a) 35 samples of the unoriented appearance dataset.

(b) 35 samples of the oriented appearance dataset.

Figure 10: Comparison between the output of the oriented and unoriented versions of the appearance dataset generation tools.

Custom Format The custom format for segmenting video fragments is structured as a Tab-Separated Values (TSV) file with three columns. The first column indicates the initial frame number, the second column indicates the final frame number, and the last column comprises a Python-styled list of lists that represents the invalid intervals. These intervals are encoded by specifying their initial and final frames.

Labeling is a time-consuming task that necessitates human supervision. To reduce the time required, we employed an existing application and developed three additional tools. The research on data annotation was conducted concurrently with the creation of a tutorial designed to guide the annotation of new training videos, which was one of the project's objectives.

Figure 11: A screenshot of the CVAT interface.

3.1.1 CVAT

Computer Vision Annotation Tool (CVAT)[28] is an interactive annotation tool designed for computer vision applications. It provides support for importing and exporting annotations in various popular formats. In this project, we primarily relied on tracking results for labeling to save time on the annotation process.

To set up CVAT, it should be installed as a server through their Docker image and accessed through any web browser. Figure 11 shows the annotation interface.

CVAT manage each id as a class. Within a frame, it offers functionalities to create, delete, and modify detections. Additionally, any track can be created, deleted, marked as finished, split, merged, or have its identity changed. For unfinished tracks with gaps in the annotations, CVAT employs linear interpolation, which can be adjusted manually in the desired frames.

One limitation of CVAT is that it supports a maximum resolution of 3000x2000 pixels. To work around this, we utilized ffmpeg[29] to downsample the video file and developed two scripts to adapt the annotations accordingly. The commands in Listing 1 are used to generate the input video and importable annotations.

Listing 1: Commands to obtain the CVAT compatible data

```
    ffmpeg -y -i input.mp4 -vf scale=2000:-2,setsar=1:1 -c:v \
libx264 "input_downsampled.mp4"
    python3 minimum_id.py input_1.txt input_2.txt
    python3 prde_to_cvat.py input_2.txt output.zip
```

A script was also created to convert manually corrected but downsampled annotations into the desired resolution.

Despite the availability of preannotations from tracking models and the utility of this tool, annotating 1375 frames (equivalent to approximately 1 hour and 30 minutes of video) still requires around 2 hours.

3.1.2 Lonely Ant Frame Segmentation

In the process of generating video segment annotations from the collection of videos for appearance training, a text file is filled manually with the previously listed specifications. To speed up the process, a command that write the frame number on each frame using the ffmpeg tool is employed, as illustrated in Listing 2:

Listing 2: Command to write the frame number on each frame

```
ffmpeg -i input.mp4 -vf "drawtext=fontfile=Arial.ttf: \
text=%{n}: fontsize=24: x=(w-tw)/2: y=h-(2*lh): fontcolor=white@0.5: \
box=1:boxcolor=0x00000099@0.3" -y input_frame_num.mp4
```

This approach reduces the time required for annotation to approximately 15 minutes for every hour of video footage. Note that a video can be played at an accelerated speed, and this annotations only require pausing the video.

Moreover, when dealing with sequences involving only one ant per frame and employing a high-performance detector, this method becomes analogous to annotating in the MOT Challenge format. By utilizing **detection by tracking**, the detector can effectively allow the low-confidence detections[9] and utilize a tracker to eliminate false positives. Even if the detector does not eliminate all the false positives, adhering to the strict criterion of one ant per frame enables the initial exclusion of frames featuring multiple ants, which can subsequently be interpolated as required.

3.1.3 Purging by Deletion

A reliable object detector relies on a clean object detection dataset. Given the impracticality on annotating with CVAT, a different approach to create a object detection dataset was designed. This process utilized the script previously explained for generating a cropped (or uncropped) YOLO dataset, as illustrated in Figure 9. To accomplish this, two additional scripts were developed.

The initial step involves generating the object detection dataset using the YOLO dataset script. The image filenames are assigned based on the frame IDs (if no cropping was applied, it can be easily transformed into the MOT format).

Next, the second script comes into play, which annotates the images and saves them in a new folder. In cases where only one detection exists, a crop is retained, as it simplifies the process of assessing its accuracy via a file explorer.

Subsequently, annotators are tasked with deleting the invalid images from this folder.

In the final step, a script is employed to construct a new dataset by copying only the remaining images. This results in an object detection dataset where only a fraction of the frames are preserved, typically less than half of the images prove to be valid.

[9]This improves the detection recall, recall is a metric that will be explained in the following subsection about metrics.

As an example, 50 seconds (equivalent to 750 frames) are transformed into 13,000 crops, equivalent to a duration of approximately 14 minutes and 27 seconds. The annotation process for this sequence typically takes around 4 hours to complete.

3.1.4 Cleaning by Crop Deletion

While discarding images containing errors is an acceptable practice for object detection datasets, the aim is often to maximize available data. To address this, we implemented another set of scripts, similar to the ones described previously.

In this scenario, the data is initially in MOT format both at the beginning and the end of the process. The first script is designed to store cropped images alongside their corresponding frame IDs and line numbers. This script can source data from either one or two models. In cases involving two models, it combines their outputs and attempts to identify true positives (which can be reviewed more swiftly), false positives, and false negatives.

The second script generates a MOT format file containing only the remaining lines, meaning that annotations for frames are corrected rather than deleted.

In this case, cleaning 1 hour of video containing a single ant (equivalent to 54,000 frames) can be accomplished in roughly 6 hours. This process was applied to 9 out of the 12 individual ant videos.

3.1.5 Labeling strategies summary

The following table shows the speed of each strategy:

Table 3: Labeling strategies speed

Strategy	**Speed**
CVAT	0.2 fps
Lonely ant frames	60 fps
Purging by deletion	0.9 fps
Cleaning by deletion	2.5 fps

3.2 Metrics

In the development of any engineering system, a fundamental aspect is the measurement of its performance. In this project, we have one primary system, the MOT tracker, along with two subsystems: the object detector and the appearance model. Each of these components has its own set of evaluation metrics to quantify its performance

3.2.1 Object Detection Metrics

An object detector solves the object localization and classification problems at the same time. This kind of problem is evaluated by the precision, recall, F1-Score and Mean Average Precision (mAP) metrics.

These metrics require the definition of true positives, false positives and false negatives:

- The **true positives** (TP) are defined by correctly matching a class detections with ground truth objects of the same class. Usually a IoU threshold is used to match the detections, similar to the association step of a SORT tracker.
- The **false positives** (FP) are defined by incorrectly assigning a class on the detected object, it applies to matched objects and detected background.
- The **false negatives** (FN) are defined by the absence of matching detections for ground truth objects.

Precision Precision is a metric that measures the correctness of the detected set, a high precision means a high certainty on the detected objects, however it does not contemplate undetected objects.

Recall Recall is a metric that measures the ability to detect targets, a high recall means a high certainty on detecting the relevant objects, however it does not contemplate undesired detections.

Figure 12 describes graphically and mathematically precision and recall.

F1-Score F1-Score is the harmonic mean of precision and recall which symmetrically represents both metrics, it allows a good global evaluation of a model. Its formula is the following:

$$F_1 = 2 \cdot \frac{precision \cdot recall}{precision + recall} \tag{24}$$

Usually, deep learning based detection models return their confidence score. F1-Score, precision and recall computed on a validation dataset can be used to determine that parameter. There is a compromise between precision and recall that makes the choice of a good threshold a non-trivial task.

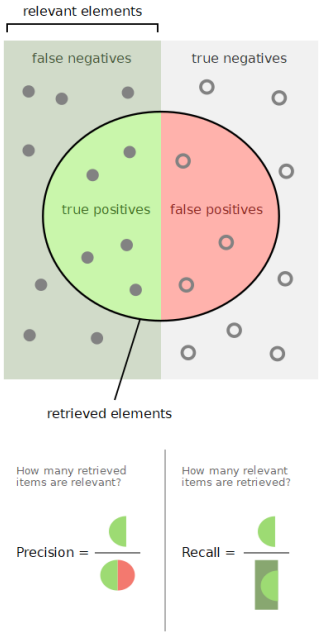


Figure 12: Visual representation of precision and recall extracted from WikipediaCommons[30].

Mean Average Precision Average Precision (AP) and Mean Average Precision (mAP) are other global performance metrics like F1-Score. AP join precision and recall regardless of the confidence threshold for a given class. MAP averages the AP metric across multiple outputs generated by various user-defined settings. These sets of outputs are referred to as queries.

The most popular conventions for the mAP set of settings are the isolation of each class and the definition of multiple IoU threshold used to match the detections with the ground truth. This project only detects one class, which means that only the IoU threshold is modified.

The AP metric is obtained by averaging the precision curve ($P(R)$) in function of the recall (R) from 0 to 1:

$$AP = \int_0^1 P(R)\, dR \tag{25}$$

In practice, the precision and recall curve is estimated by a given dataset becoming a set of not equidistributed points. In order to average the precision and recall curve, the data output is sorted by decreasing confidence (the $i-th$ detection is ranked k_i), the precision is computed within a subset of the k more confident detections ($P(k)$) and, at the end, each ranked precision is weighted averaged by the width of the recall interval from the previous rank to the current rank.

The mathematical expression of the defined weighted average can be simplified into the simple mean of the ranked precision of the ground truth elements, with the false negatives redefined to have a ranked precision of 0:

$$AP = \frac{\sum_{i \in TP} P(k_i)}{|TP| + |FN|} \tag{26}$$

This project uses the **mAP@50** (or AP@50 because there is only one class) and the **mAP@50-95**. The @ notation refers to the IoU threshold queries, being **mAP@50** the AP when the IoU is set to 0.5 and **mAP@50-95** the mean of the AP computed with IoU thresholds of 0.5, 0.55, 0.6, 0.65, 0.7, 0.75, 0.8, 0.85, 0.9 and 0.95.

3.2.2 Identity Re-identification Metrics

An appearance model solves the identity re-identification problem. This kind of problem is evaluated by the rank metrics and the already seen **mAP** using the identities on the desired validation dataset classes as queries.

In re-identification problems true negatives can also be defined, in addition to true positives, false positives and false negatives:

- The **true positives** are defined by correctly matching pairs of ants images with the same identity.
- The **true negatives** (TN) are defined by correctly not matching pairs of ants images with different identities.
- The **false positives** are defined by incorrectly matching pairs of ants images with different identities.
- The **false negatives** are defined by incorrectly not matching pairs of ants images with the same identities.

Rank metrics The k-rank metrics redefine true positives (TP_k) by the existence on any correct match for a query ant within the k nearest appearances. The final metric is the accuracy of the given set of queries:

$$Rank_k = \frac{|TP_k|}{|queries|} \tag{27}$$

This project contemplates the rank-1, rank-5 and rank-10 metrics.

3.2.3 Multiple Object Tracking Metrics

Evaluating multiple object tracking is a complex problem comparable to the tracking task itself. The reason of this complexity is the fact that a tracking system solves 3 problems:

- **Detection**: the target objects are found.
- **Localization**: the found objects are spatially aligned with their target.
- **Association**: the found objects are temporally aligned with their target identity.

For this reason, there exist a lot of metrics that prioritize one problem over the others or evaluate each problem with different criteria. In this project, it was decided to use the Higher Order Tracking Accuracy (HOTA)[31] group of metrics because the authors of these metrics states that the HOTA metric, that jointly evaluates the three problems, is balanced with respect the three problems (see Figure 13).

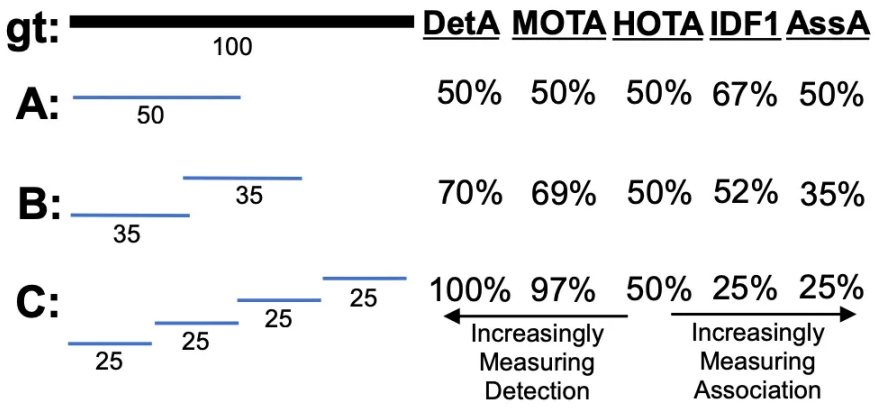


Figure 13: Image extracted from the HOTA paper[31].
HOTA, DetA and AssA compared with other popular metrics (MOTA[32] and IDF1[33]).

Similar to the other metrics, the first step is the definition of true positives, false negatives and false positives. However, to obtain these values, a matching between ground truth and tracks is required. This matching should aim to maximize the final metric while maintaining plausibility. The rationale behind this criterion is that all consistent interpretations of the system's output that align with the ground truth are valid, and the maximized interpretation is likely the most accurate.

The authors of HOTA solve this optimization problem with the Hungarian algorithm using a composed score. The score uses a **current frame** spatial similarity component and a **global** estimated "optimal association" score. The estimated association score is a prematching proxy for an association score that approaches the estimated one for the optimal assignment.

The spatial similarity can be the previously explained IoU, the previous "plausibility" refers to a threshold value applied on this score after the ground truth matching.

The estimated association score is a multiple frames Jaccard index[10], it is computed as the available spatial associations (with a higher spatial similarity than the threshold) of a pair "ground truth identity"-"tracked object identity" for all the frames over the number of frames where at least one of the identities was present (see Figure 14).

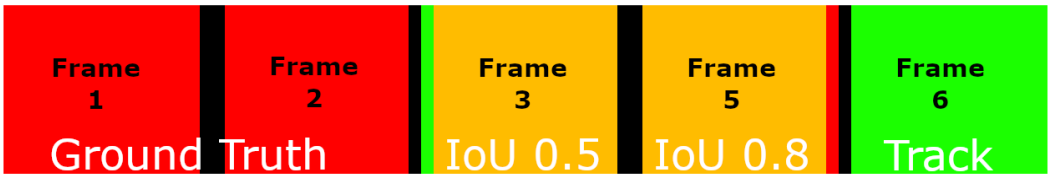


Figure 14: This image illustrates a case where a track of 3 frames overlaps during 2 frames with a ground truth identity of 4 frames. The denominator of the estimated optimal association will be 3-2+4=5. The numerator can be 2 for a threshold smaller than 0.5, 1 for a threshold smaller than 0.8 and 0 for a threshold higher than 0.8. With a threshold of 1 the association score cannot be different than the estimated association score.

Once the spatial similarity ($\mathcal{S}$), the estimated association score ($\mathcal{A}_{max}$) and the similarity threshold (α) are defined, the potential matching score used for the linear assignment joins them as in the equation 28:

$$MS(i,j) = \begin{cases} \frac{1}{\varepsilon} + \mathcal{A}_{max}(i,j) + \varepsilon \cdot \mathcal{S}(i,j) & \text{if } \mathcal{S}(i,j) \geq \alpha \\ 0 & \text{otherwise} \end{cases} \tag{28}$$

The matching algorithm was implemented in order to use the ground truth to analyses our results and test the viability of a model.

With a mapping between ground truth and output tracks, HOTA defines true positives, false negatives, false positives:

- The **true positives** are detections that are matched with a ground truth object, independently of the identity, it measures the performance of the detection and localization problems.
- The **false negatives** are ground truth objects that are not matched.
- The **false positives** are detections that are not matched.

[10]The Jaccard index is the intersection of two sets over their union (IoU).

Additionally, HOTA defines new values called true positives associations (TPA), false negatives associations (FNA) and false positives associations (FPA). The TPA, FNA and FPA are defined independently over all the pairs of "ground truth identity"-"tracked object identity" within the true positives set; a single evaluation has got one set of true positives, false negatives and false positives and $C = |TP|$ sets of true positives associations, false negatives associations and false positives associations. These values measure the performance of the association problem:

- The **true positives associations**, given a true positive of interest "c", are all the true positives which have the same "ground truth identity"-"tracked object identity" pairs than the interest true positive "c"; all TPAs have the same TPA, FNA and FPA which reduce the number of computation although the concept is defined for all the TPs.
- The **false negatives associations**, given a true positive of interest "c", are the ground truth objects with the same identity as the ground truth from "c" that were not assigned the tracked object identity from "c" (different or unassigned identities).
- The **false positives associations**, given a true positive of interest "c", are the detections with the same identity as the detection from "c" that were not assigned the ground truth identity from "c" (different or unassigned identities).

With the new association measures, the previously mentioned association score can be properly defined:

$$\mathcal{A}(C) = \frac{|TPA(c)|}{|TPA(c)| + |FNA(c)| + |FPA(c)|} \tag{29}$$

HOTA$_\alpha$ HOTA$_\alpha$ is a metric defined by a double Jaccard formulation, where the association score ($\mathcal{A}$) is a Jaccard metric over the associations that weights the true positives of a Jaccard metric over the detection concept. And α is the threshold on location similarity.

$$HOTA_\alpha = \sqrt{\frac{\sum_{c \in \{TP\}} \mathcal{A}(c)}{|TP| + |FN| + |FP|}} \tag{30}$$

The square root is needed because the dimensionality of the summation of association scores (multiplied by true positive of value 1) is squared (jointly detection domain and association domain) over unsquared (association domain), and the detection Jaccard makes the denominator squared (jointly detection domain and association domain).

HOTA Theoretically, HOTA is the average of $HOTA_\alpha$ on all the location similarity domain (from 0 to 1). In practice, it can be approximated by a mean:

$$\text{HOTA} = \int_0^1 \text{HOTA}_\alpha \, d\alpha \approx \frac{1}{19} \cdot \sum_{\alpha \in \{0.05 \cdot i \,|\, 1 \leq i \leq 19\}} \text{HOTA}_\alpha \tag{31}$$

LocA Localization Accuracy (Localization Accuracy (LocA)) is a measure of localization that average the different localization thresholds:

$$\text{LocA} = \int_0^1 \frac{1}{|\text{TP}_\alpha|} \cdot \sum_{c \in \{\text{TP}_\alpha\}} \mathcal{S}(c)\, d\alpha \tag{32}$$

DetA Detection Accuracy (Detection Accuracy (DetA)) is the detection Jaccard index with a given localization threshold α. It can be further decomposed into detection precision and detection recall but it is not used in this project.

$$\text{DetA}_\alpha = \frac{|\text{TP}_\alpha|}{|\text{TP}_\alpha| + |\text{FN}_\alpha| + |\text{FP}_\alpha|} \tag{33}$$

AssA Association Accuracy (Association Accuracy (AssA)) is a measure of association that average the association score with a given localization threshold α.

$$\text{AssA}_\alpha = \frac{1}{|\text{TP}_\alpha|} \cdot \sum_{c \in \{\text{TP}_\alpha\}} \mathcal{A}(c)\, d\alpha \tag{34}$$

Another interpretation of HOTA_α is the geometric mean of DetA_α and AssA_α.

3.3 Detection Models

To facilitate the development of multiple detection models and their subsequent integration into the tracker, we designed a versatile program. This program processes input videos and saves the output detections in a MOT format file (as shown in Figure 15). We used the OpenCV library[34] for Python3 to handle the input video data.

Initially, there was no labeled data to train any data-driven model, for this reason, the first implemented model was a deductive model.

This deductive model, as the examples from the tracking software subsection of the state of the art, is based on background extraction. It is implemented using a set of tools from the OpenCV library (refer to Figure 16).

To ensure accurate background estimation, this model requires a set of training frames. The estimation is performed using a Gaussian Mixture-based model, which becomes reliable after proper training.

Once the background is estimated, it is subtracted from the frame to isolate the foreground. A morphological filter is then applied to the foreground to reduce noise, and the remaining connected components with a minimum pixel count are identified as the output detections.

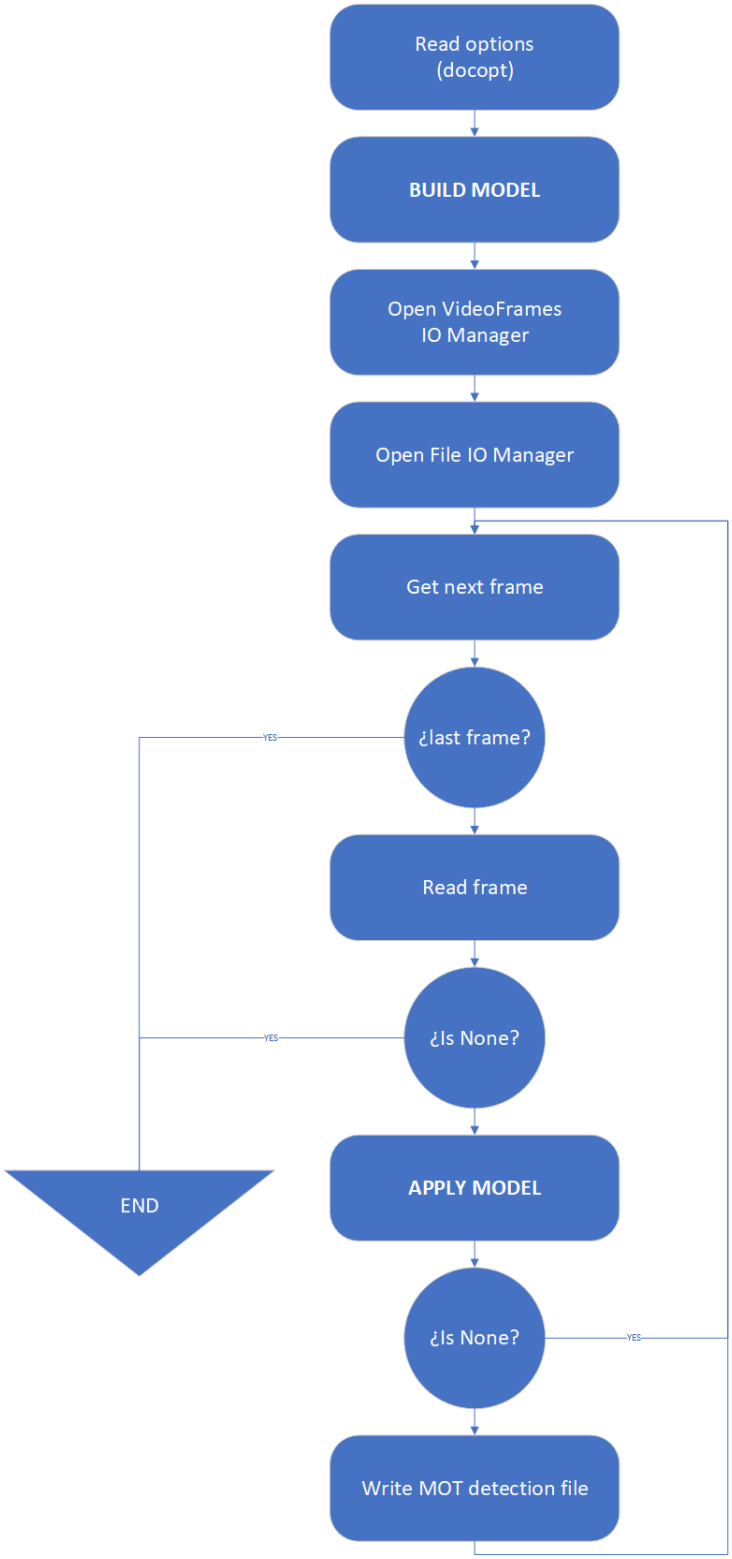


Figure 15: Block diagram of the detection script.

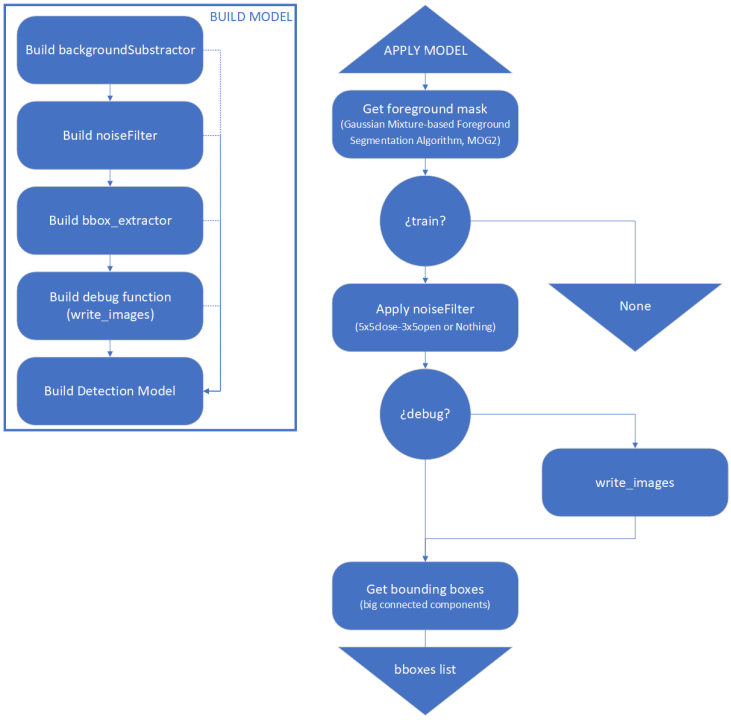


Figure 16: Block diagram of the background extraction based model.

With the first detection model results, filtered by a tracker and corrected with the developed annotation tools, a initial dataset was developed; it was used to train a deep learning model. After some iterations of training, detection of new data and cleaning, the final dataset used for detection training was developed; it has got 8374 images for training and 3662 images for validation. It does not include the first video.

The deep learning model selected for this project is the Ultralytics[16] library You Only Look Once version 8 (YOLOv8) architecture YOLOv8n model (detailed in the state of the art section). The decision to opt for this architecture was based on state of the art performance in popular challenges as well as simplicity of training and inference deploying (Ultralytics implements most of the standard training features).

The choice of model was based on the current problem, the ants on the frames are small objects, and deep neural networks tend to reduce the spatial dimension while expanding the feature dimension; this means that the ant location becomes uncertain. YOLOv8n is the smallest YOLOv8 model, consequentially, the model that will locate better the ants.

Another modification that was deployed because the size of the ants is a slicing windows analysis; for the training 640x640 px crops were used and for the inference a slicing windows was applied using the Slicing Aided Hyper Inference (SAHI) library[35].

3.3.1 YOLOv8 Training

Ultralytics is fully integrated with Weights and Bias (WandB)[36], a machine learning development platform that helps in the real-time tracking and visualization of experiments, it also have an automatized launcher for hyperparameter sweeping. Using WandB resources to visualize enough trainings, the lower and upper thresholds of each available hyperparameter were adjusted until the setting in Table 4.

Table 4: Detection Model hyperparameters.

Hyperparameter Name	**Values**
Hue (maximum deviation)	min=0, max=0.3
Saturation (maximum deviation)	min=0, max=0.5
Value (maximum deviation)	min=0, max=0.5
Rotation (maximum degrees)	min=60º, max=180º
Translation (fraction of input)	min=0, max=0.5
Scale (maximum gain and loss)	min=0, max=1
Flip upside down (probability)	min=0, max=0.5
Flip left-right (probability)	min=0, max=0.5
Mosaic (probability)	min=0, max=1
Shear (maximum degrees)	min=0, max=15
Perspective (fraction)	min=0, max=0.001
Mixup (probability)	min=0, max=0.12
Copy paste (probability)	min=0, max=1
Dropout	{0, 0.25, 0.5, 0.75}
Batch size	min=50, max=100
Optimizer	{SGD, AdaM}
Initial learning rate	min=0.001, max=0.01
Final learning rate	min=0.0001, max=0.001
Warmup epochs	10
Cosine learning rate	{True, False}
Momentum	min=0.9, max=0.974
Maximum epochs	500
Early stopping patience	50 epochs
NMS	{Agnostic, False}
NMS IoU	min=0.5, max=1

The first group of hyperparameters refer to the data augmentation used to help the model generalize better with the same amount of data, as each time a imaged is loaded, it will be randomly modified within a certain limits. It includes modification in the images coloration, the appearance of the objects within the image, small modifications of the objects (a flip will create a symmetrical but inexistent ant) and some combination of various images. Figure 17 shows a set of 16 images after data augmentation.

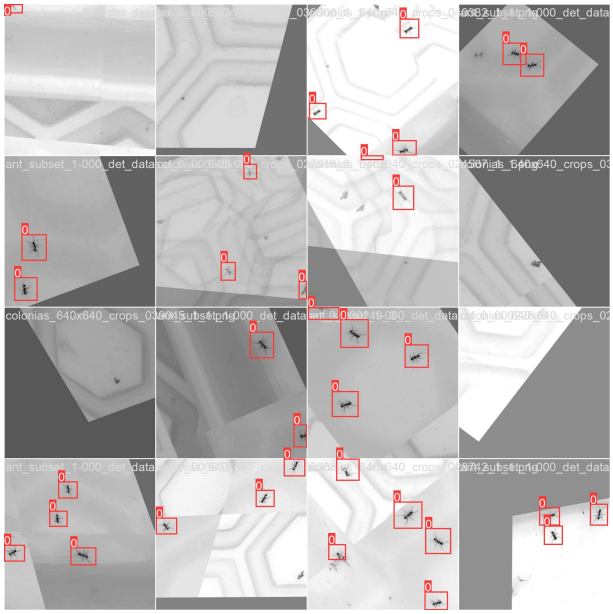


Figure 17: Samples of the augmentated training data for the best detector trained.

The second group of hyperparameters refer to the model, in this case, there is only the dropout; it randomly set to zero certain values in the mathematical expression of a forward pass to reduce the number of useless nodes as well as generalize by randomizing.

The third group of hyperparameters refer to the weight upgrading to reach an optimum result on the loss function. The loss function for these trainings is a combination of binary cross entropy and focal loss for classification (with only one class, ant, both of them are equivalent) and a mean squared error (Mean Squared Error (MSE)) for the object location. The hyperparameters include the number of images used as a representative sample of the world that the model should learn, how the model will find the best weights to fit the current batch samples and the speed (with not a well-defined meaning) at which it will try to reach that weights.

The fourth group set a limit on how long it should take to reach the limit, and the maximum steps admissible without validation improvement.

The last group is about post-processing the outputs, only the suppression of highly overlapped detections is contemplated.

At the end, the best models were selected, added to the detection script with SAHI and their outputs were used in the tracker.

The training of the YOLOv8n model was performed on CALCULA, the computing server from the Teoria del Senyal i Comunicacions (TSC) department of the Universitat Politècnica de Catalunya (UPC), using 1 GPU NVIDIA GeForce RTX 3090 with 24 GB of memory available, 16 cores of CPU and 32 GB of RAM. With this setting and the previous explained hyperparameters, the longest training duration was smaller than 7 hours.

3.4 Appearance Model

Similar to the detection model, the first step was the development of a script that applies the appearance model. In this case, the script takes a MOT format file and its video. It applies the appearance model on each line of the MOT file. And the output is the input MOT file with each line extended with as many columns as appearance features returns the model.

For the appearance model, the choice of model was directly obtained from Deep OC-SORT (a model from 2023): the Bag of Tricks model[14]. The rationale behind this choice was twofold: first, given the limited availability of appearance data, there were modest expectations for its performance in the initial video. Second, Deep OC-SORT represents a state-of-the-art model, further justifying its selection.

The implementation of this model is the same from Deep OC-SORT, it is the model from the FastReID[37] library which includes training scripts. In this case, a tuning of the FastReID source code was needed. The inference function was extracted from the Deep OC-SORT code and applied on the developed inference scripts.

3.4.1 Bag of Tricks Training

Being a relatively simple model architecture (detailed in the state of the art section), the more relevant part of this model is the training, more specifically, the training tricks defined on their paper and code:

- The triplet loss, a derivable function for maximizing interclass euclidean distance and minimizing intraclass euclidean distance, is applied before the batch normalization layer.
- The classification loss is still derivated from the output of the pooling layer.
- The first 1000 iterations from a total of 18000 has the backbone freezed.
- The first 2000 iterations are warm-up iterations and the learning rate increases.
- From the iteration 2000 until the iteration 9000 the learning rate is constant.
- From the iteration 9000 until the end, the learning rate decrease with a cosine decay.

The implementation used for training is the same of the paper, with the only difference in the data augmentation: the code was modified in order to allow the definition of rotation degrees and the use of rotation without any additional affine transformations. Most of the training options were reduced based on the paper.

The biggest ant appearance dataset that was used contained 186 identities, divided into 93 for training and 93 for validating. The validation set will leave a 20% of the identities as unknown identities (these identities are true negatives). At the end, the train set has got 6730 images, the validation query set has got 489 images to assign in 5795 images.

Additionally, a small script to allow the use of WandB to visualize the experiments as a whole and lunch automatic sweeps was written. The set of hyperparameter values that were allowed is defined in Table 5.

Table 5: Appearance Model hyperparameters.

Hyperparameter Name	**Values**
Augmentation mix (probability)	min=0, max=1
Brightness (maximum deviation)	min=0, max=2
Contrast (maximum deviation)	min=0, max=2
Hue (maximum deviation)	min=0, max=0.5
Saturation (maximum deviation)	min=0, max=2
Flips 50% enabled	{True, False}
Rotation (maximum degrees)	min=60, max=100
Batch size	8192
Optimizer	{SGD, AdaM}
Initial learning rate	min=0.00001, max=0.005
Momentum	min=0.9, max=0.95
Maximum epochs	1440

To adjust the color mean and standard deviation, a script to analyze the pixels of a whole video was made.

Finally, the unoriented and the oriented versions of the same dataset trained were trained. The best models independently of the dataset were used to perform a test of viability.

The training of the BoT model was performed on CALCULA using 4 GPUs NVIDIA GeForce GTX 1080 Ti, 8 cores of CPU and 128 GB of RAM (because the code had some memory leakage due to multiprocessing and data stored on non-consecutive memory). With this setting and the previous hyperparameters, the longest training duration was smaller than 1 hour.

3.4.2 Appearance Tests

The training of the appearance features was unconvincing, in order to analyze the discriminability of appearance, a set of tests were designed.

The first test consists in the comparison of ants with themselves rotated and plotting the distance with respect the angular rotation, the goal of this test is the study of the invariance over rotation that a well trained appearance model should have.

The second test is a comparison of ants with other ants rotated, this also checks the invariance over rotation but from the error point of view.

The third test is the splitting of all tracks within a ground truth into tracklets every time two or more ants are near; afterwards, the mean appearance of the tracklets, the limit appearance near a split point and the re-joining capabilities were observed through a set of correct versus incorrect appearance distance histograms.

3.5 Tracking Models

For the tracking model development, a proxy detector that reads the output of the previous detection models or appearance model was used. It removes the time and memory needed for video loading and reduce the time used on redundant computations.

The first model tested was a SORT model with a downgraded detector, the background extraction model (see Figure 16). The next model tested was the OC-SORT model with the same the background extraction model as the model.
The code from these trackers was reimplemented to gain understanding and flexibility to apply modifications.

Afterwards, a manual annotation was performed to define the project baseline. This annotated data allowed the analysis of the ants behavior, by instance, the displacement per frame, the time of potential occlusions, the location metrics distribution, etc. It also allowed the study on the error distribution: the velocity and angular errors from the Kalman estimations.

Before training the deep learning based detector, a test to improve the Kalman filter from the OC-SORT model with an intuition was developed. The intuition consisted on "the ants should move towards the direction their body points". It was implemented by instantaneously modifying the direction of the estimation (less than 180^{o}) using Principal Component Analysis (PCA) on the ants body. To reduce the input loading time, the angle obtained through PCA was included on the MOT file.

With a better detector, the SORT and OC-SORT model were tested again.

Finally, the Deep SORT model was reimplemented and the Deep OC-SORT github code was downloaded. These models were tested with the trained appearance model.

4 Results

This section will present and discuss the training curves of the best models found and the results of the experiments explained at the previous section.

The results are based on 150 frames (10 seconds) from the video with a tray and 87 ants. This source video was not used neither in the training or validation of any subsystem of the tracking model. The annotations where made as accurately as possible using CVAT.

4.1 Detection Training

The YOLOv8 training curves, are a visual representation that serves as a snapshot of the model development.

From the curves on Figure 18, it can be seen the learning process starts fluctuating only on validation because the model is not optimized and the validation contains some randomness. As the training advances, the validation curve stabilizes and slowly improves until the epoch 243; at the epoch 293 (of a maximum of 500) the trainer program early stopped the training. Although the training ended without a clear sign of overfitting, the improvement on the training while the maintained and even declining on the validation, concurrently with the decreasing learning rate (see Figure 19), proves a successful training.

Additionally, it was seen when increasing the dataset size with more difficult data that the model reached a better solution[11]. The validation mAP@50-95 curves on Figure 20 illustrates this statement where both curves have similar shapes but different endpoints.

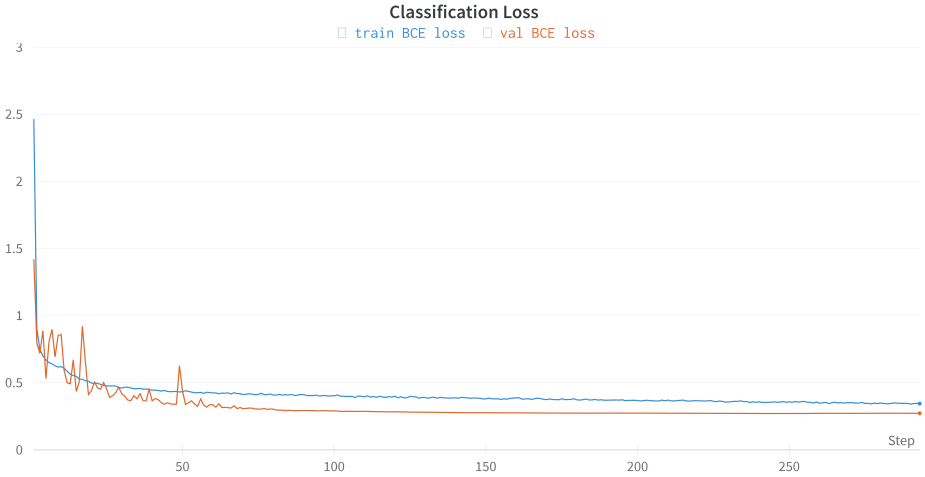
The previous results provided a comprehensive understanding of the training process and the model as a versatile tool. However, it is imperative to shift our focus towards the practical application of the detection model, utilizing the best-performing weights garnered during training.

Figures 21a, 21b and 21c present the precision, recall and F1-Score metrics, assessed at various confidence thresholds. When transitioning from detection to tracking, the primary objective is to attain a high recall rate to minimize the loss of ants during the tracking phase. This approach is rooted in the rationale that spurious false positives can be subsequently filtered out by the tracking model.

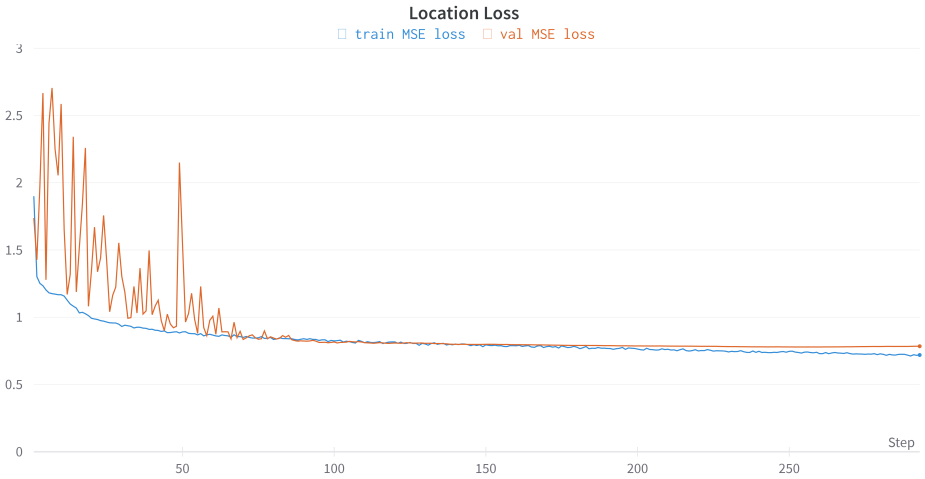
For this purpose, the basic threshold when assamled with the tracking model was initially set to 0.4. This choice aimed to approach the optimal F1-Score, which occurs at 0.5 in Figure 21c, but with a deliberate shift towards favoring higher recall.

Furthermore, the low-high confidence threshold was set at 0.8. At this threshold, the impact of a reduced recall on the F1-Score becomes noticeable, prompting a thoughtful balance in model performance.

[11] This result is valid because the ampliation of the validation dataset increased the difficulty (from a dataset of only one ant to a dataset with multiple ants interacting in groups), however, it would be better to repeat the curves with a unique version of the validation dataset.



(a) Train and validation curves for the class classification loss.



(b) Train and validation curves for the object location loss.

Figure 18: Train and validation loss curves of YOLOv8.

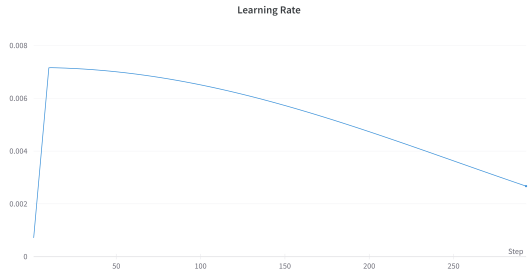
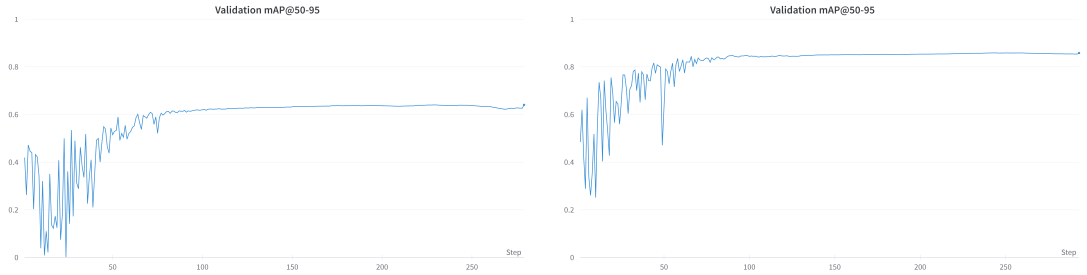


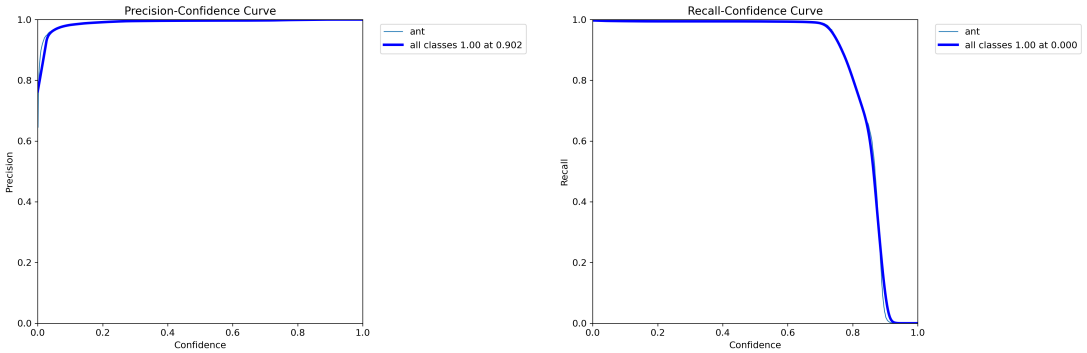
Figure 19: Learning rate curve for the YOLOv8 training.



(a) Validation mAP@50-95 curves of YOLOv8 with the smaller dataset.

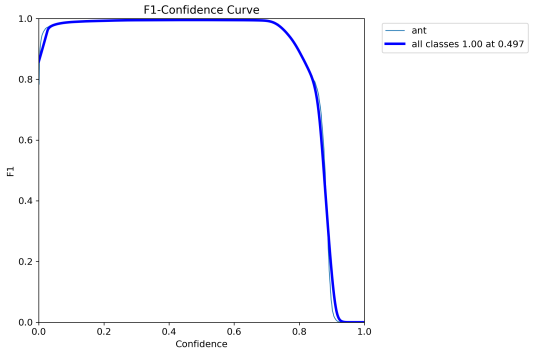
(b) Validation mAP@50-95 curves of YOLOv8 with the bigger and harder dataset.

Figure 20: Validation mAP@50-95 curves of YOLOv8 trained and validated with a dataset and a previous version of the same dataset.



(a) Precision-Confidence curve of the trained YOLOv8.

(b) Recall-Confidence curve of the trained YOLOv8.



(c) F1Score-Confidence curve of the trained YOLOv8.

Figure 21: Validation curves of the trained YOLOv8 for confidence thresholding.

4.2 Detection Results

In object tracking, the selection of a detection model significantly impacts precise location estimation. Table 6 reinforces the rationale behind the chosen detection model by comparing its mean average precision, precision and recall with other trained models and the background extraction model.

Table 6: Comparison of detectors mAP on the testing video.

Model Name	mAP@50	mAP@50-95	Precision	Recall
Background extraction	21%	08%	46%	28%
YOLOv8n	30%	15%	48%	44%

The data unequivocally demonstrates the superiority of YOLOv8n over the background extraction baseline across all evaluated aspects.

Particularly noteworthy is the substantial improvement in recall. In the unlikely event that the false negatives were evenly distributed among frames and tracks, the enhanced recall would signify the loss of slightly more than half the frames, this loss would be mitigated through interframe interpolation.

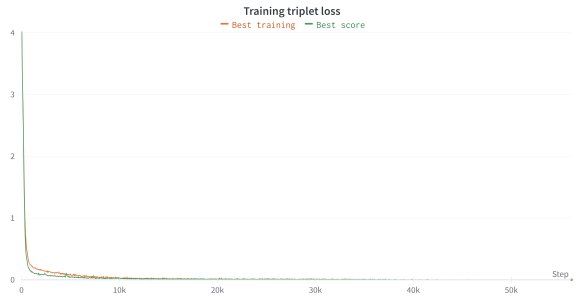
4.3 Appearance Training

The appearance model is a key component for trackers based on re-identification. However, unlike the stable training curves from the detector, the appearance model metrics and loss curves resulted in swift overfitting (see Figures 22, 23 and 24).

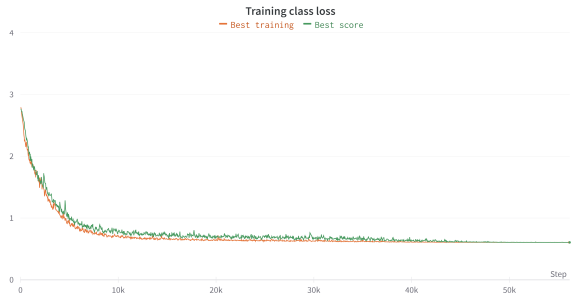
The depicted curves reveal a rapid convergence on the training data, as depicted in Figure 22. However, a notable trend of stagnation or decline becomes evident in the validation results, as illustrated in Figures 23 and 24. A closer examination of the learning rates reveals that the best-performing model, characterized by its absence of decline, operates with a higher learning rate. This observation implies that the training convergence can be attributed to the dataset's relative simplicity in terms of data quantity. The elevated learning rate serves as a safeguard against overfitting.

While the model seems to yields features that contains ant individualities, with a rank-1 accuracy of 74%, it is difficult to measure its certainty due to the training.

The presented training results are only from the unoriented dataset; the oriented dataset (containing the same data) yields results of less than 50% of rank-1 accuracy with the same issues on training.



(a) Training triplet loss curve of BoT for two training settings.



(b) Training classificationloss curve of BoT for two training settings.

Figure 22: Training Loss curves of BoT, it includes 2 training configurations.

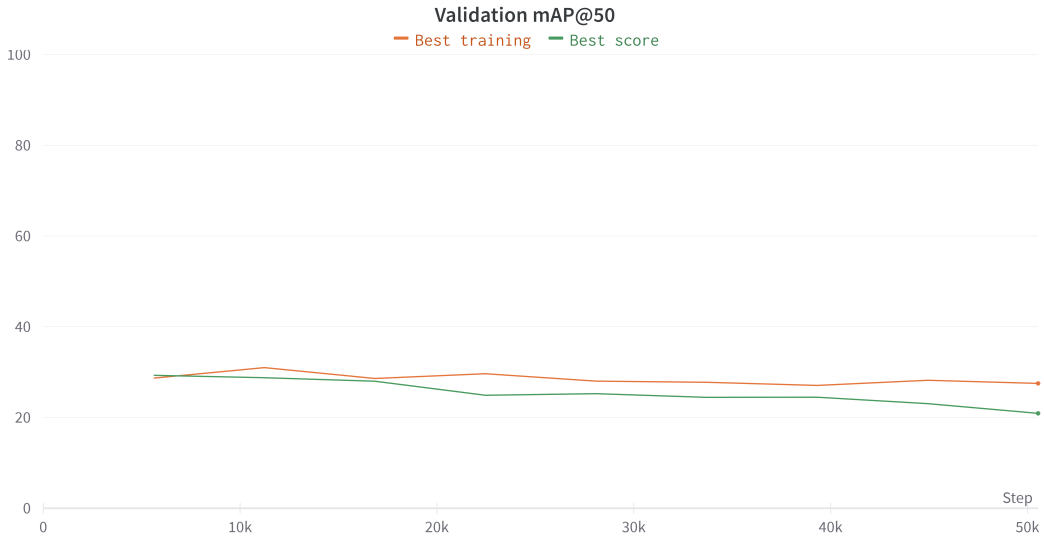
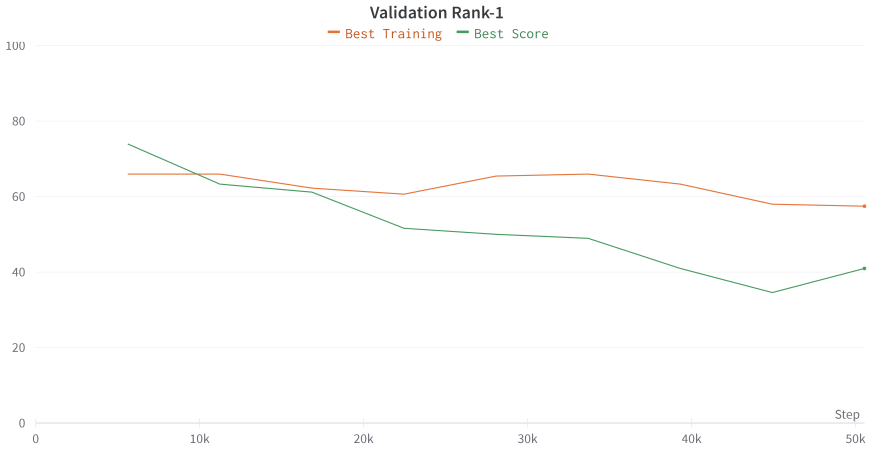
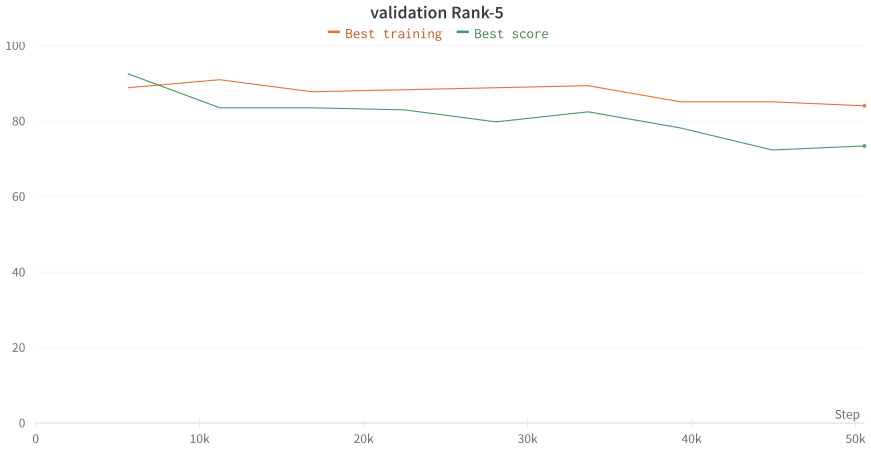


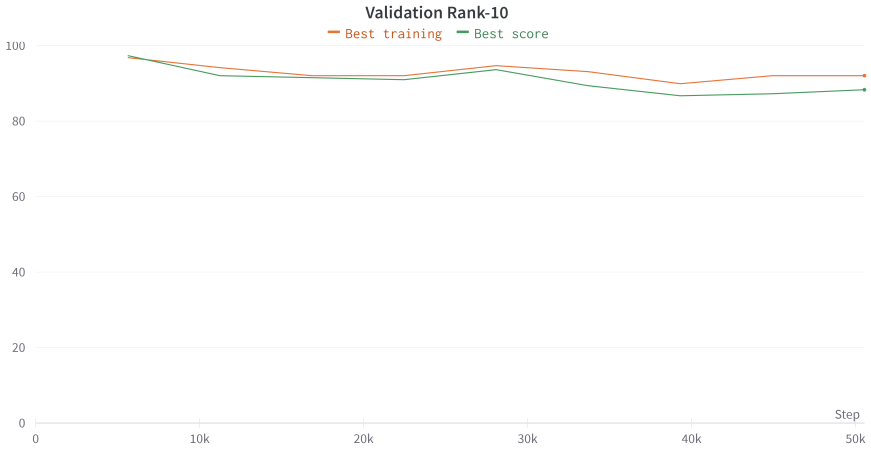
Figure 23: Validation mAP@50 curves of BoT for two training settings.



(a) Validation Rank-1 curve curve of BoT for two training settings.



(b) Validation Rank-5 curve of BoT for two training settings.



(c) Validation Rank-10 curve of BoT for two training settings.

Figure 24: Validation Rank curves of BoT, it includes 2 training configurations.

4.4 Appearance Tests Results

Figure 25 is the outcome of the rotation test. It consist in two polar plots with a characterization of the cosine distance applied on the appearance descriptors.

The left plot, when one crop of one ant is compared with itself, serves to demonstrate the non-invariability over rotation. Without taking into account the 0^{o} rotation with the expected distance 0, the distance is clearly higher at 90^{o} or rotation. However, the fact that a 180^{o} rotation contains another minimum in the similarity opens the direction of using alignment algorithms in future research.

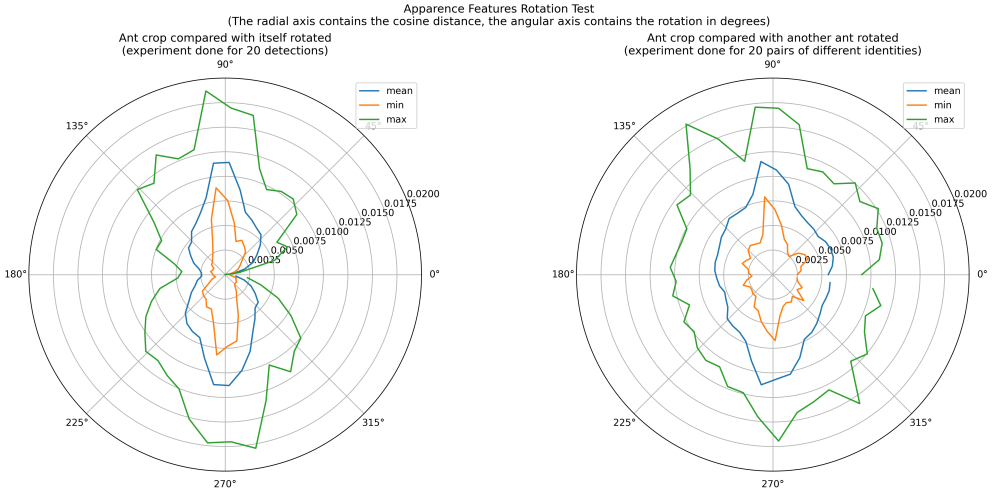
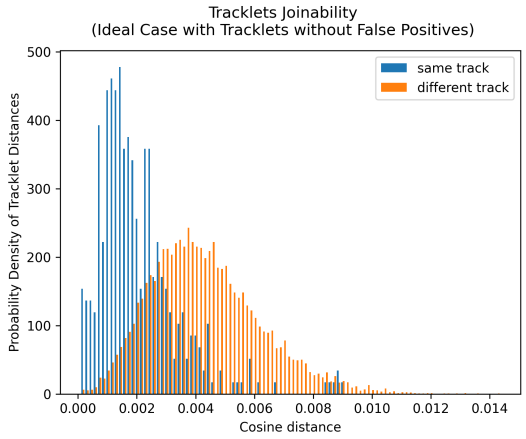


Figure 25: Validation of the rotation invariability property of the trained appearance features.

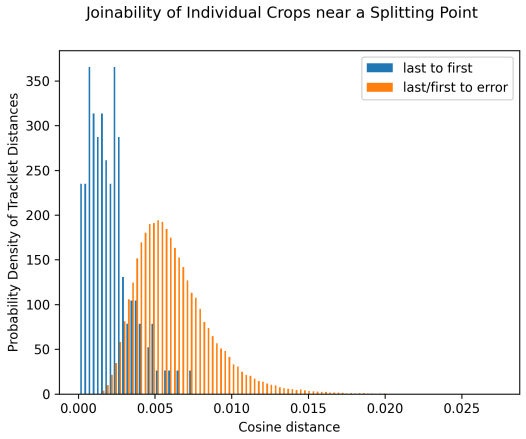
The right plot of Figure 25, compared with the left plot, could be used to have an idea of joinability, however, using the right plot to complement Figures 26 yields a clearer explanation.

Figure 26b depics the likelihood of associating correctly on the relevan points where appearance is the most important feature: the frames where ants are near each other. From Figure 25, the tail of the incorrect matching probability density function mainly contains in distances related to near 90^{o} rotations. From Figure 27, the ant displacement and duration of these cases are characterized, most of them are short meetings within a small scope, this means that the most usual case is a small angular displacement, the better case depicted in Figure 25.

Finally, the Figure 26a exposes the distribution of means of distances of splitted tracklets within one track, and the means of distances of splitted tracklets compared with tracklets from other tracks. It is remarkable the fact that the distributions are normalized in total area, in a real scenario, the problem is unbalanced with more negative cases than positives. From Figure 26a, it can be seen that the appearance features are not usable yet, however, it seems that further research in this topic will be relevant.



(a) Normalized density function to determine the viability of an appearance model that fuse tracklets by their mean descriptor.



(b) Normalized density function to determine the viability of an appearance model that fuse tracklets based on the last known detection.

Figure 26: Appearance features joinability tests obtained by analysising the ground truth with a split version of itself.

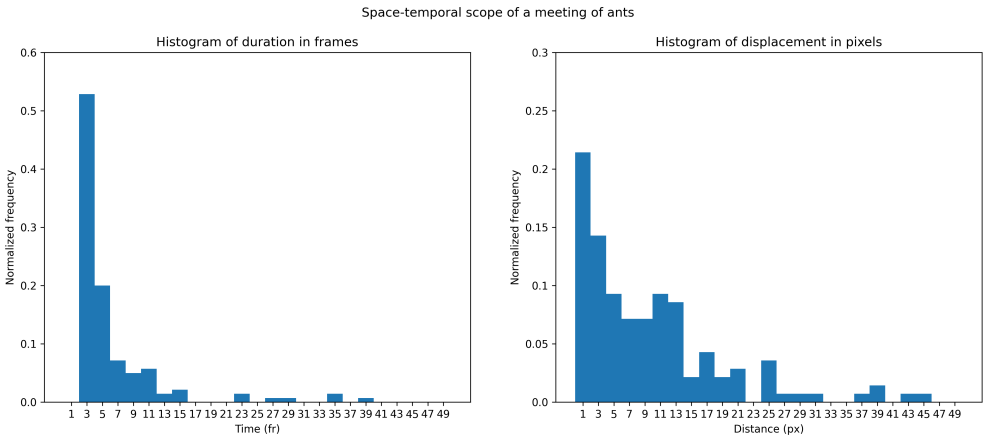


Figure 27: Analysis of the temporal and spatial scope of the occlusions.

4.5 Tracking Results

Finally, after testing the final models, using the different detectors and tracking algorithms, and manually searching for good hyperparameters, the results can be seen on Table 7.

Table 7: Comparison of detectors mAP, "Background extraction" is reduced into fgbg.

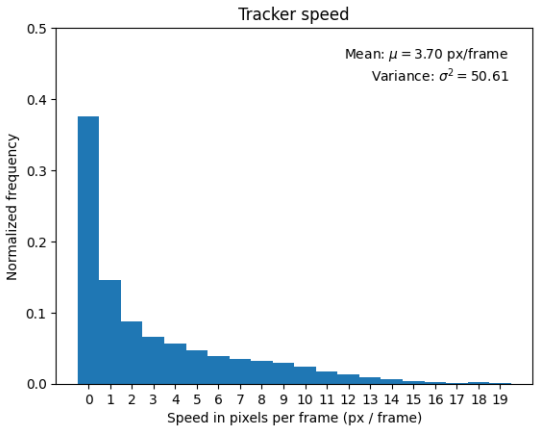
Model Name	**DetA**	**LocA**	**AssA**	**HOTA**
fgbg - SORT	20%	70%	36%	27%
fgbg - OC-SORT	20%	71%	40%	28%
yolo - SORT	**40%**	74%	61%	48%
yolo - Deep SORT	08%	71%	01%	03%
yolo - OC-SORT	**40%**	76%	63%	**49%**
yolo - Deep OC-SORT	**40%**	74%	62%	48%

The usage of the interpolator (GSI) did not affect the results, and the PCA correction on the ant trajectory slightly deteriorate them.

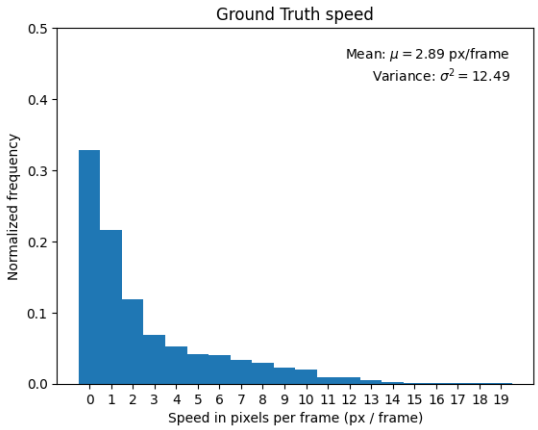
From Table 7 the worse case is the Deep SORT, which only uses appearance to resolve split tracks and fails to associate the observatons, and the best case is the OC-SORT, which discards the appearance model. Additionally, the most relevant component is the detector.

4.6 Tracking Results analysis

From Figure 28 it can be seen that the tracker predicts faster trajectories than the real ones, however, when observing Figure 29, that shows the difference in module in a per case basis, it can be seen the opposite; both ideas can be interpreted together: The tracker correctly tracks most of the fast ants (which may be constantly moving for a certain time), and requires some time to adapt when the ant accelerate or stops (it is the expected behavior of a tracker). Nevertheless, the gross error is small.



(a) Probabilistic representation of the speed of the ants within a sequence processed by the best tracker.



(b) Probabilistic representation of the speed of the ants within a ground truth sequence.

Figure 28: Comparison of the probabilistic representation of the speed of the ants between the best tracker and the ground truth.

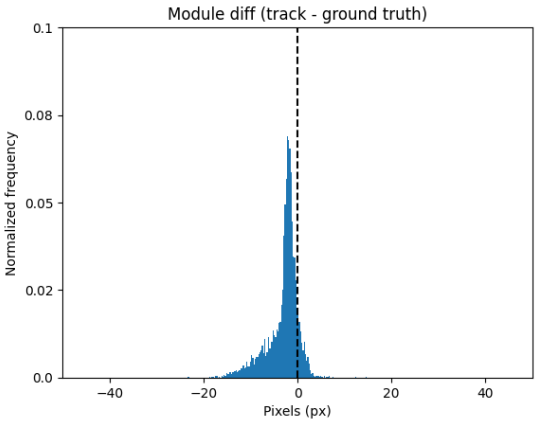
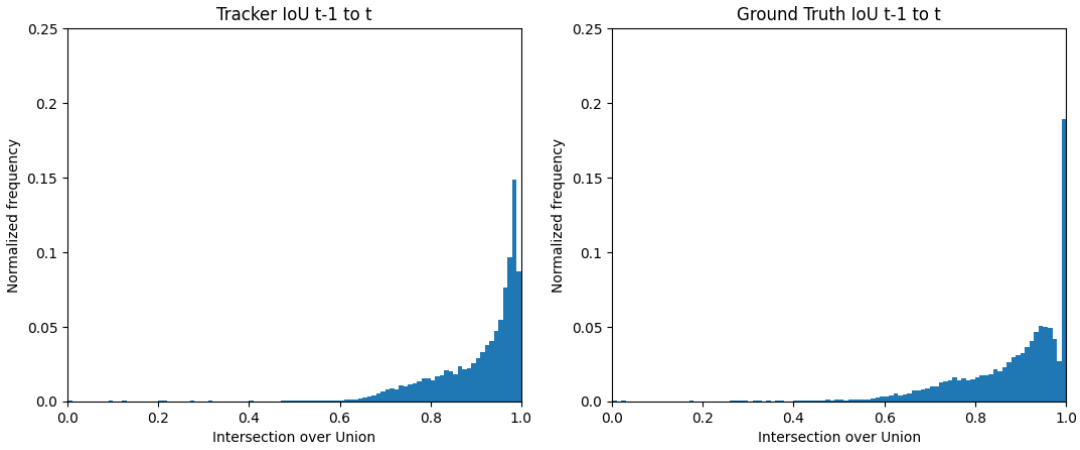


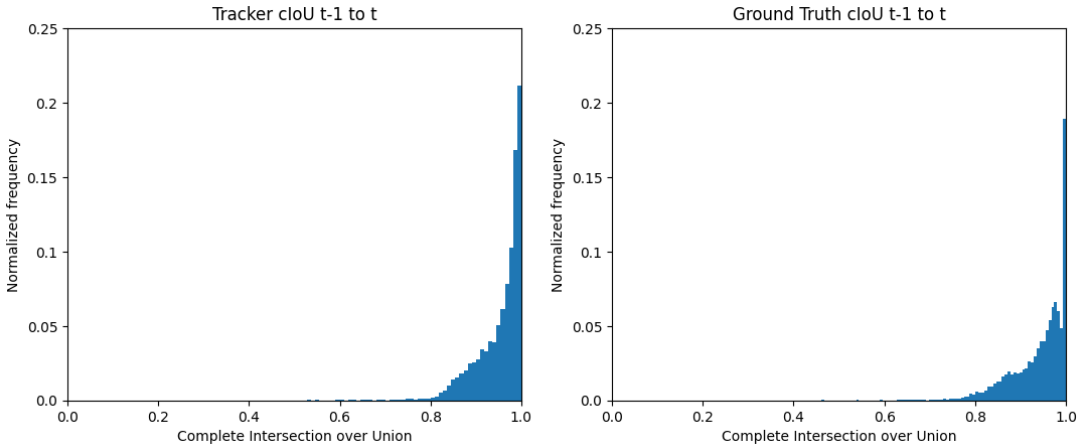
Figure 29: Asymmetric error in tracker displacement.

Another consideration when studying the characterization of a visual tracker, as well as the problem, is the location metrics within a track, if no motion estimation were made. The IoU can be seen in Figure 30a, comparing the ground truth with the tracker, it can be

observed the tracker adapts correctly. The CIoU can be seen in Figure 30a, in these plots, the distribution is displaced towards the high scores, and the ground truth is smoother, which makes it easier for the tracker to reproduce it, by instance, near the score value 1, where the ants are static.



(a) Comparison of the probabilistic representation of the IoU of the ants between the best tracker and the ground truth.



(b) Comparison of the probabilistic representation of the cIoU of the ants between the best tracker and the ground truth.

Figure 30: Comparison of the probabilistic representation of the the locations similarity within a track on the ground truth and the best tracker.

Figure 31 and 32 were done by associating tracks and ground truth as explained in the HOTA subsection from the methodology section.

From Figure 31, the right plot shows that the angular error of the estimated displacement is small, a good explanation for the unsuccessful PCA model: adding complexity when it works will become a noisy source. The left plot shows the module error of the estimated displacement is small, more detailed characterization was done at the beginning of this subsection (using module difference instead of module error).

Figure 32 depicts the IoU of associated tracks and ground truth, it is noticiable the valley at 0.8 IoU, which divides the data of a plot approximately in a 40% at the left and a 60% at the right. Coinciding with the 60% AssA which measures the correct associations of observations and tracks.

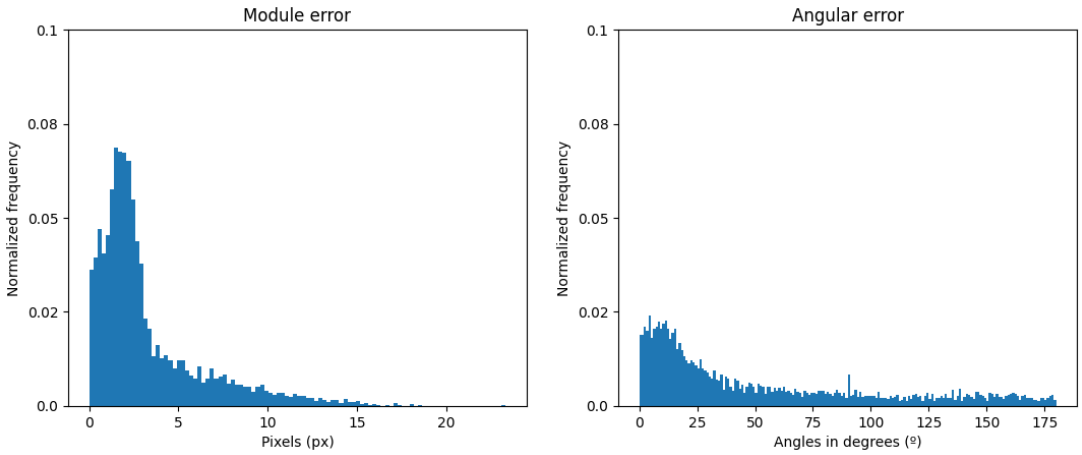


Figure 31: The error in distance and angle of the best tracker.

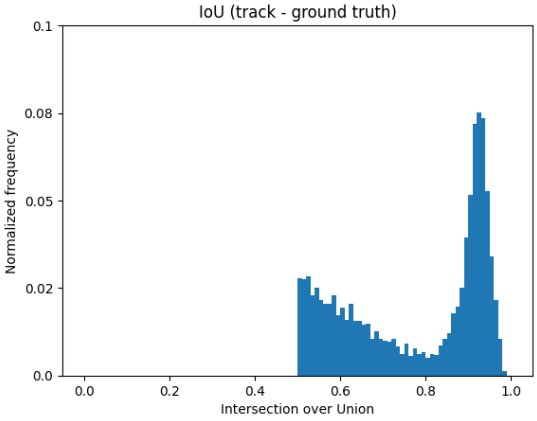


Figure 32: Location similarity between the best tracking and the ground truth

Finally, the HOTA metric is depicted in the Figure 33, and a 2D representation of a tracked sequence compared with its ground truth is shown in Figure 34, where each color line is a track and each row is a different identity.

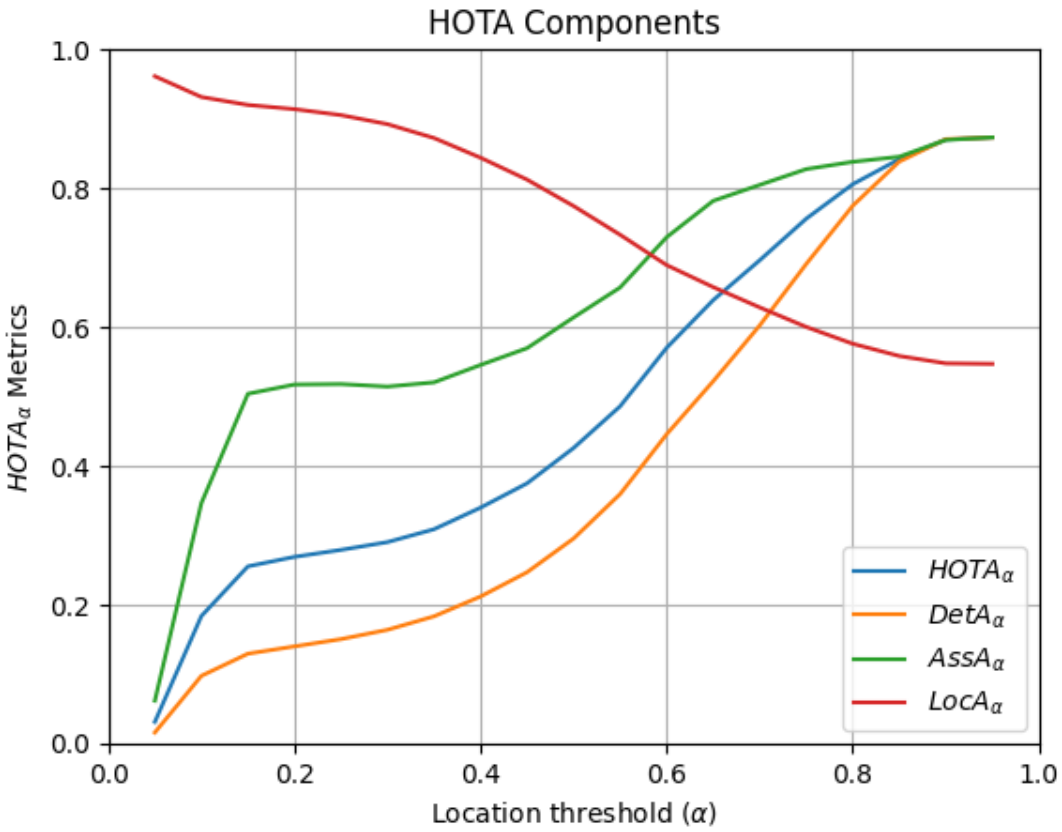


Figure 33: HOTA partial components in function of the threshold for the best tracker.

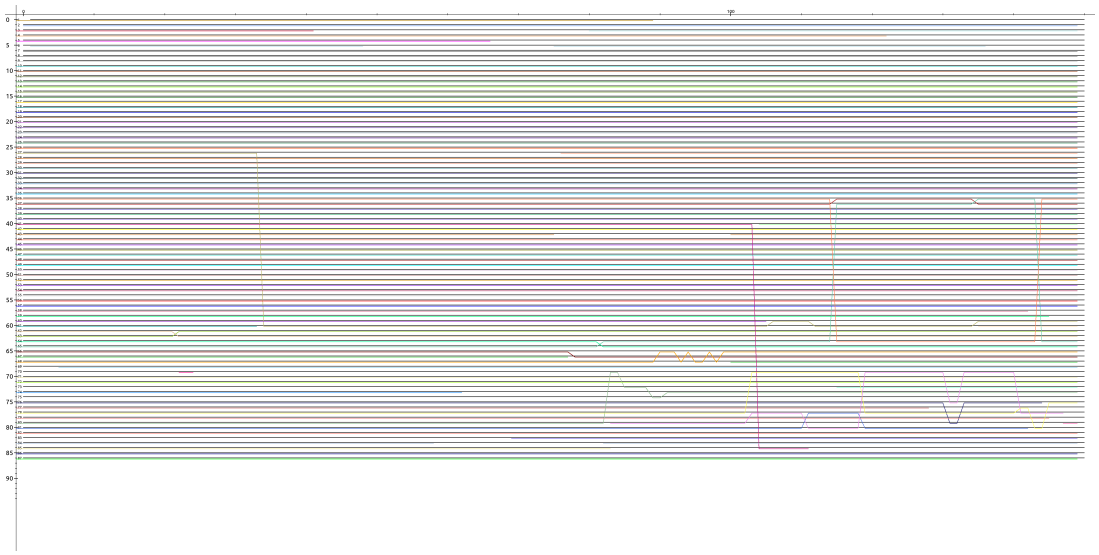


Figure 34: Visual representation of the OCSORT results.

5 Conclusions

This work started an ant tracking project researching and summarizing the theoretical background. It also researched the state of the art and found the need to upgrade the available ant tracking applications with the state of the art models.

A starting project lacks in data, within the realization of this project new raw data was obtained and some of it was labeled, tools to ease the labeling process were researched and developed.

From with data, a YOLOv8n detector with a mAP@50-95 of 85.5% on validation was trained and applied on tracking models using SAHI, however, in the testing sequence it reached a mAP@50-95 of 15%. A BoT appearance model training was started without a successful result.

It was found that the detection component is the most relevant one.

Finally, a set of trackers were tested setting a baseline of 49% of HOTA with the OC-SORT with YOLOv8n model for this ant tracking project next work.

6 Future Work

- The fundamental issue that was found while developing this project was the lack of appropriate data with appropriate annotations. The first future task should be the retrieval and labeling of lots of data.
- The detector is the cornerstone of a tracker, further improvement should be done in this component.
- This work was done oriented towards the SORT model and its successors, all of them are online and causal. Some research should be considered in other tracking architectures (for instance, the models in the papers [38] and [39]). Highlighting bidirectional trackers and fully data driven trackers.
- Our current results are tracks, the CSIC researchers needs some mean to evaluate the certainty of a track; or, in the best case, the certainty improvement for a subset of ants. A future work should try to research and develop the “certainty on tracks” topic.
- The appearance model was unsuccessful; however, it still has the potential to be a powerful tool in ants tracking. A further research on models and trainings should be continued. However, new data should be acquired for it.
- As stated in the introduction, after obtaining enough data, post processing data driven model for traking should be researched. Starting with the AFLink from Strong SORT.
- The development of an ant behavior model is being done in parallel with this work. Once the model is available, it should be a task to test its performance by including it on the tracker.

References

[1] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer, 2022.

[2] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking, 2016.

[3] Rudolph Emil Kalman and Others. A new approach to linear filtering and prediction problems, 1960.

[4] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric, 2017.

[5] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box, 2021.

[6] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021, 2021.

[7] Jinkun Cao, Jiangmiao Pang, Xinshuo Weng, Rawal Khirodkar, and Kris Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking, 2021.

[8] Hamid Rezatofighi, Nathan Tsoi, JunYoung Gwak, Amir Sadeghian, Ian Reid, and Silvio Savarese. Generalized intersection over union, 2019.

[9] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang Ye, and Dongwei Ren. Distance-iou loss: Faster and better learning for bounding box regression, 2020.

[10] Zhaohui Zheng, Ping Wang, Dongwei Ren, Wei Liu, Rongguang Ye, Qinghua Hu, and Wangmeng Zuo. Enhancing geometric factors in model learning and inference for object detection and instance segmentation, 2021.

[11] Yunhao Du, Zhicheng Zhao, Yang Song, Yanyun Zhao, Fei Su, Tao Gong, and Hongying Meng. Strongsort: Make deepsort great again, 2023.

[12] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks, 2017.

[13] Behzad Mirzaei, Hossein Nezamabadi-pour, Amir Raoof, and Reza Derakhshani. Small object detection and tracking: A comprehensive review, 2023.

[14] Hao Luo, Youzhi Gu, Xingyu Liao, Shenqi Lai, and Wei Jiang. Bag of tricks and a strong baseline for deep person re-identification, 2019.

[15] Gerard Maggiolino, Adnan Ahmad, Jinkun Cao, and Kris Kitani. Deep oc-sort: Multi-pedestrian tracking by adaptive re-identification, 2023.

[16] Glenn Jocher, Ayush Chaurasia, and Jing Qiu. YOLO by Ultralytics, 2023.

[17] Range King (GitHub user). Brief summary of yolov8 model structure, 2023. https://github.com/RangeKing.

[18] Violette Chiara and Sin-Yeon Kim. Animalta: A highly flexible and easy-to-use program for tracking and analysing animal movement in different environments, 2023.

[19] Martin Stumpe. Antracks. https://sites.google.com/view/antracks, 2010.

[20] Asaf Gal, Jonathan Saragosti, and Daniel JC Kronauer. antrax, a software package for high-throughput video tracking of color-tagged insects, 2020.

[21] Alfonso Pérez-Escudero, Julián Vicente-Page, Robert Hinz, Sara Arganda, and Gonzalo Polavieja. Idtracker: Tracking individuals in a group by automatic identification of unmarked animals, 2014.

[22] Francisco Romero-Ferrero, Mattia G. Bergomi, Robert Hinz, Francisco J. H. Heras, and Gonzalo G. de Polavieja. idtracker.ai: Tracking all individuals in large collectives of unmarked animals, 2018.

[23] Kristin Branson, Alice Robie, Pietro Perona, and Michael Dickinson. High-throughput ethomics in large groups of drosophila, 2009.

[24] Chamath Abeysinghe, Chris Reid, Hamid Rezatofighi, and Bernd Meyer. Tracking different ant species: An unsupervised domain adaptation framework and a dataset for multi-object tracking, 2023.

[25] L. Leal-Taixé, A. Milan, I. Reid, S. Roth, and K. Schindler. MOTChallenge 2015: Towards a benchmark for multi-target tracking, 2015.

[26] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. You only look once: Unified, real-time object detection, 2016.

[27] Liang Zheng, Liyue Shen, Lu Tian, Shengjin Wang, Jingdong Wang, and Qi Tian. Scalable person re-identification: A benchmark, 2015.

[28] Boris Sekachev, Nikita Manovich, Maxim Zhiltsov, Andrey Zhavoronkov, Dmitry Kalinin, Ben Hoff, TOsmanov, Dmitry Kruchinin, Artyom Zankevich, DmitriySidnev, Maksim Markelov, Johannes222, Mathis Chenuet, a andre, telenachos, Aleksandr Melnikov, Jijoong Kim, Liron Ilouz, Nikita Glazov, Priya4607, Rush Tehrani, Seungwon Jeong, Vladimir Skubriev, Sebastian Yonekura, vugia truong, zliang7, lizhming, and Tritin Truong. opencv/cvat: v1.1.0, 2020.

[29] Suramya Tomar. Converting video formats with ffmpeg, 2006.

[30] Wikimedia Commons (Walber). Precision and recall, 2014.

[31] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking, 2020.

[32] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The clear mot metrics, 2008.

[33] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking, 2016.

[34] G. Bradski. The OpenCV Library, 2000.

[35] Fatih Cagatay Akyon, Sinan Onur Altinuc, and Alptekin Temizel. Slicing Aided Hyper Inference and Fine-tuning for Small Object Detection. *2022 IEEE International Conference on Image Processing (ICIP)*, 2022.

[36] Lukas Biewald. Experiment tracking with weights and biases, 2020. Software available from wandb.com.

[37] Lingxiao He, Xingyu Liao, Wu Liu, Xinchen Liu, Peng Cheng, and Tao Mei. Fastreid: A pytorch toolbox for general instance re-identification, 2020.

[38] Jinyue Zhang, Xiangrong Zhang, Zhongjian Huang, Xina Cheng, Jie Feng, and Licheng Jiao. Bidirectional multiple object tracking based on trajectory criteria in satellite videos, 2023.

[39] Huilan Luo and Zehua Zeng. Real-time multi-object tracking based on bi-directional matching, 2023.

List of Figures

1 SORT algorithms timeline . . . 8
2 SORT algorithm . . . 9
3 YOLOv8 architecture block diagram . . . 19
4 Bag of Tricks model block diagram . . . 20
5 Project's Block diagram . . . 23
6 Frame from the initial video setup . . . 26
7 Frame from the appearance video setup . . . 26
8 Frames from the tray videos setup . . . 26
9 YOLO dataset: crop of 640x640 px . . . 28
10 Samples of the appearance datasets . . . 29
11 CVAT interface . . . 30
12 Precision and recall . . . 34
13 HOTA compared with other metrics . . . 36
14 HOTA: Track overlap with ground truth . . . 37
15 Generic detection script . . . 40
16 Background extraction detection model block diagram . . . 41
17 YOLOv8 data augmentated . . . 43
18 Train and validation loss curves of YOLOv8 . . . 48
19 Learning rate curve of YOLOv8 . . . 48
20 Validation mAP@50-95 curves of YOLOv8 . . . 49
21 YOLOv8: Precision, Recall and F1Score-Confidence curve . . . 49
22 Training Loss curves of BoT . . . 51
23 Validation mAP@50 curves of BoT . . . 51
24 Validation Rank curves of BoT . . . 52
25 Appearance features rotation test . . . 53
26 Appearance features joinability . . . 54
27 Analysis of the occlusions . . . 54
28 Ants speed Normalized histogram . . . 56
29 Asymmetric error in tracker displacement . . . 56
30 Location metrics between frames . . . 57
31 Displacement and angular error of the best tracker . . . 58
32 Location similarity between the best tracking and the ground truth . . . 58
33 Best tracker HOTA . . . 59
34 Visual representation of the OCSORT results . . . 59

List of Tables

1 Input videos characterization . . . 24
2 MOT Challenge format . . . 27
3 Labeling strategies speed . . . 32
4 Detection Model hyperparameters . . . 42
5 Appearance Model hyperparameters . . . 45
6 Detectors mAP Comparison . . . 50

7 Detectors mAP Comparison 55

List of Acronyms

AP Average Precision . . . . . 33 ff.
AssA Association Accuracy 39, 57
BoT Bag of Tricks . . . . . 16 f., 20, 45, 60
BYTE ByteTrack 12, 14
CIoU Complete Intersection over Union 15, 56
CNN Convolutional Neural Network 9, 16, 21
CSIC Consejo Superior de Investigaciones Científicas 6, 21, 25, 60
CSV Comma-Separated Values 27
CVAT Computer Vision Annotation Tool . . . . . 29 ff., 47
DetA Detection Accuracy 39
DIoU Distance Intersection over Union . . . . . 14 f.
ECC Enhanced Correlation Coefficient 16
EMA Exponential Moving Average 16
FN False Negatives 33
FNA False Negatives Associations 37
FP False Positives 33
FPA False Positives Associations 37
GIoU Generalized Intersection over Union 14
GSI Gaussian-Smoothed Interpolation . . . . . 16 f., 55
HOTA Higher Order Tracking Accuracy 36–39, 60
IoO Inner over Outer 14
IoU Intersection over Union . . . . . 9 f., 12, 14, 17, 33–36, 56
KNN K-Nearest Neighbours 11
LocA Localization Accuracy 38
mAP Mean Average Precision . . . . . 33 ff., 47, 60
MOT Multiple Object Tracking 8, 27, 33
MSE Mean Squared Error 43
OC-SORT Observation-Centric SORT . . . . . 12 ff., 17
PCA Principal Component Analysis 46, 55

SAHI Slicing Aided Hyper Inference 41, 43, 60
SORT Simple Online and Realtime Tracking . . . . . 8 f., 12, 21, 60
TN True Negatives 35
TP True Positives 33, 37
TPA True Positives Associations 37
TSC Teoria del Senyal i Comunicacions 43
TSV Tab-Separated Values 29
UPC Universitat Politècnica de Catalunya 43
WandB Weights and Bias 41, 45
YOLO You Only Look Once 27
YOLOv8 You Only Look Once version 8 41
YOLOv8n You Only Look Once version 8 nano 18, 41, 43, 60