ELTE
EÖTVÖS LORÁND
UNIVERSITY

MSc, Computer Science
Department of Artificial Intelligence

AI Laboratory Report

Bahruz Huseynov
Supervisor: Dr. Gulyás László

6th December, 2024

Contents

1 Introduction

The advancements of the Computer Vision helps people with different opportunities. Particularly object detection and tracking have also become significant part of the studies to develop modern algorithms to overcome various challenges. These components enable machines to understand dynamic environments. Object detection aims to identify and localize objects in the image based on the predefined labels. On the other side, object tracking extends the detection by associating objects through sequential frames so that it checks the identities over frames and tries to receive the trajectories of objects. There are various recent applications, such as healthcare monitoring, autonomous driving, and anomaly detection, accomplished by analyzing benchmarks and existing methods [1]. However, various segmentation models can also be used to enhance the detection and tracking pipelines. They can handle occlusions, remove the incorrect predictions and filter the background out to differ the object from the spatial environment.

In the carried experiments, the main goal was to build the object counter by analyzing and searching about performance of the algorithms on the videos. Different types of LEGO bricks have been used as objects. Briefly, the performance of the detection, segmentation and tracking pipeline on the LEGO bricks has been explored together with various pre-processing techniques. Another essential aspect is the techniques that is related to the creation of a custom dataset based on the real-world recordings of the videos. In general, the research covers distinct strategies applied with several pipelines in which detection algorithms of YOLOv8 and RT-DETR, DeepLabv3 as a semantic segmentation model and DeepSORT tracking algorithm has been utilized.

There are many recent applications of various sequence of models to track objects. One of them has been mainly done Luka Farkaš so that the combination of EL-YOLOv8 and StrongSORT algorithms have been used to track objects [2]. During the experiment, the challenging VisDrone dataset has been utilized and it is focused on pedestrians and vehicles divided into 10 classes including people and vehicles. As a result, improved detection accuracy in the drone footage has been accessed during experiments. In addition, an automated checkout system or retail counter in another name has been developed by Arpita Vats and David C. Anastasiu to handle real-world challenges like occlusion, motion blur, and item similarity [3]. In their method, detection-tracking pipeline has been done by YOLOv8 and DeepSORT algorithms by using video inpainting for removing unwanted objects from the region of interest. the pipeline showed its high performance by achieving an F1 score of 0.8177 on tracking scanned products.

Aichen Wang et al. developed NVW-YOLOv8s with improved feature extraction, class balancing, and loss functions for tomato crop estimation by the usage of detection and segmentation pipeline. The method achieved detection mAP50 of 91.4% and real-time processing at 60.2 FPS, addressing occlusion and small target challenges [4]. Another detection-segmentation pipeline has been executed by Tingting Yang et al. They worked on an approach for precise segmentation of plant leaf images using a fusion of YOLOv8 and improved DeepLabv3+ [5]. The technique outperformed classical segmentation methods by obtaining a mean Intersection over Union (mIoU) of 90.8% for leaf segmentation on the public dataset.

2 Methodology

2.1 Tools and datasets

In this research, Python was used as the main programming language for executing the code, as it provides essential libraries for computer vision and data manipulation. To fine-tune models, PyTorch was utilized, a Python-based framework designed for building machine learning applications. Furthermore, technologies such as Kaggle, for storing datasets, and Google Colab, due to its computational capabilities, played a key role in simplifying the workflow. Google Colab's free GPU resources were particularly beneficial, enabling efficient model training and saving significant time during the experiment.

It is another important point that the dataset has been manually created. Thus, it has been recorded using a smartphone and consisted of 12 videos. Out of these, 10 videos were allocated for training, and 2 for testing. Furthermore, RoboFlow web application has been used to annotate LEGO bricks frame-by-frame which was mostly time consuming part of the research. Table 1 and Table 2 widely shows information about the dataset. The videos capture conveyor belts carrying LEGO bricks, and recordings have been taken from multiple angles (top, front, and diagonal) as in Figure 1. The dataset includes videos of various complexities categorized as "Overlapping," "Normal," or "Simple". Entire dataset has been saved in Kaggle together with explanation of the usage and other details in the description for the further improvements or applications. Here is the link: https://www.kaggle.com/datasets/hbahruz/multiple-lego-tracking-dataset/

Training

	View	**Complexity**	**Only Lego**	**Frames per second**	**Approx. duration (seconds)**	**Resolution**
Video 1	Top	Overlapping	+	16	19	1280x720
Video 2	Front	Overlapping	+	13	16	1280x720
Video 3	Diagonal	Normal	+	20	56	720x1280
Video 4	Top	Overlapping	+	20	42	1280x720
Video 5	Diagonal	Overlapping	+	21	14	720x1280
Video 6	Top	Normal	-	20	50	720x1280
Video 7	Top	Simple	+	15	20	1280x720
Video 8	Diagonal	Normal	+	13	13	720x1280
Video 9	Top	Normal	+	19	28	1280x720
Video 10	Front	Normal	-	20	58	720x1280

Test

	View	**Complexity**	**Only Lego**	**Frames per second**	**Approx. duration (seconds)**	**Resolution**
Video 1	Top	Normal	+	20	68	1280x720
Video 2	Diagonal	Normal	-	25	57	720x1280

Table 1: General information about LEGO dataset

Training

	Number of frames	Number of objects	Average number of objects per frame
Video 1	300	2220	7.40
Video 2	196	1648	8.41
Video 3	1136	10549	9.29
Video 4	839	10546	12.57
Video 5	294	3563	12.12
Video 6	1000	8299	8.30
Video 7	303	1388	4.58
Video 8	277	2056	7.42
Video 9	537	2662	4.96
Video 10	1162	20142	17.33

Test

	Number of frames	Number of objects	Average number of objects per frame
Video 1	1401	10628	7.59
Video 2	1444	12601	8.73

Table 2: Information about frames and number of objects

Analyzing Table 1, the 'View' column indicates the angles from which the videos were recorded as discussed earlier. This diversity in angles is crucial to ensure the dataset captures various perspectives for better model training. The 'Complexity' column, on the other hand, describes the level of difficulty in predicting the positions of LEGO bricks based on their arrangement, with categories like "Overlapping," "Normal," or "Simple" indicating how challenging the layouts might be. Additionally, some videos include objects other than LEGO bricks into the dataset, compelling models to distinguish LEGO bricks from unrelated objects. This variation is reflected in the "Only Lego" column, which specifies whether the video contains only LEGO bricks or additional objects.

Furthermore, discrepancies can arise between the number of frames, frames per second (fps), and video duration. Ideally, the total frame count should equal the product of fps and the duration in seconds. However, due to the approximate nature of the duration measurement, which omits milliseconds, a small discrepancy may result, causing the actual frame count to slightly exceed manual calculations. Regarding the resolution of the videos, it is primarily 720x1280, a common standard for most smartphones. However, since some videos were captured in both vertical and horizontal orientations, the resolution sometimes appears to be 1280x720.

Moreover, Table 2 mainly concentrates on the frame and object count. The number of frames and objects highly change for training videos; however, the numbers are much closer to each other for test videos. Looking at average number of objects per frame, the values can be misleading because in some cases, especially at the beginning of the videos, there are no any LEGO bricks or fewer of them occur in the video. However, in the middle of the videos, there are situations in which a frame contains more than 15-16 objects.

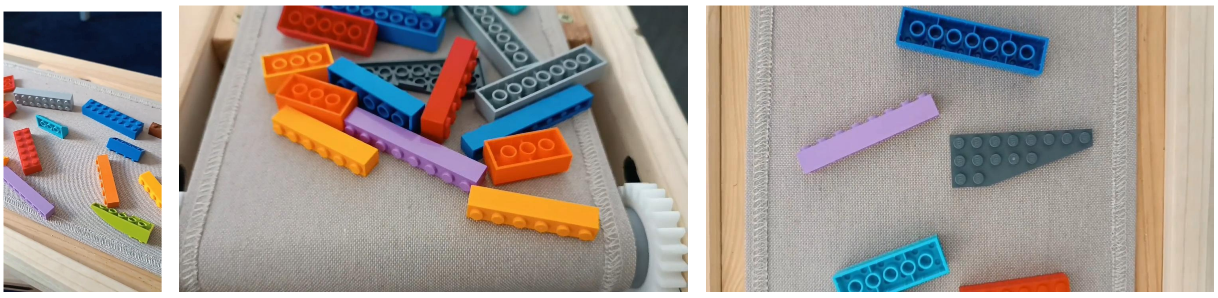


Figure 1: Different frames and views

2.2 Data Pre-processing

2.2.1 Data augmentation

Considering the low number of samples, some data augmentation methods have been used during the application. The samples are artificially increased or converted into different forms for a better generalization of the model. These included adjustments in hue, saturation, and brightness, geometric transformations such as rotation, scaling, and shear to introduce spatial variations. Moreover, vertical and horizontal flipping was applied to account for the symmetry in the data. In addition, advanced augmentation methods such as mosaic-combination of multiple images into one and mixing of images have been utilized [6]–[9]. These augmentations contributed to a more comprehensive dataset and all parameters of the augmentations done during the training is represented in Table 3.

2.2.2 Dataset partitioning

The dataset preparation to train the detection models involved two methods for splitting the training, validation, and testing data. For the initial technique represented in Figure 2a for the training videos, every fifth frame was uniformly selected to form the validation set, while the remaining frames were used for the training set. As a result 80% of the frames included into training set and 20% for validation set where samples were chosen uniformly across the training recordings. Meanwhile, the test set was constructed by including entire frames from the test videos to evaluate the the performance of the model on entirely unseen data. However, this method resulted in overfitting as the validation frames closely resembled those in the training set because of the continuous frame sequence in the videos.

Secondly used splitting strategy, which can be thought as a systematic sampling, shown in Figure 2b, overcame the issue. In the method, a uniform sampling approach was applied to both the training and test videos. In this case, instead of using all frames from the recordings, every fifth frame of each video was selected to create a reduced dataset like systematic sampling [10]. For example, from a video containing 200 frames, 40 frames were created by uniformly selecting each fifth frame. These sampled frames were then used to split the training and validation sets, again through uniform sampling [11]. This

Parameter	Value	Description
hsv_h	0.015	Adjusts the image hue randomly within ±1.5% of the total hue range to enhance color variability.
hsv_s	0.3	Modifies the saturation by ±30% to simulate different levels of color intensity.
hsv_v	0.3	Alters the brightness by ±30% to account for varying lighting conditions.
degrees	90.0	Rotates the image randomly within ±90 degrees to improve robustness to orientation changes.
translate	0.1	Shifts the image up to 10% of its dimensions horizontally or vertically to handle positional variation.
scale	0.5-1.5	Resizes the image randomly by scaling factors between 0.5 and 1.5 to simulate size variations.
shear	1.3	Applies a shearing transformation up to 1.3 degrees to introduce geometric distortions.
flipud	0.7	Vertically flips the image with a 70% probability to mimic different viewpoints.
fliplr	0.7	Horizontally flips the image with a 70% probability to address symmetry variations.
mosaic	1.0	Combines four images into one with a 100% probability to enrich the training data with varied contexts.
mixup	0.7	Blends two images and their labels with a 70% probability to create diverse training examples.

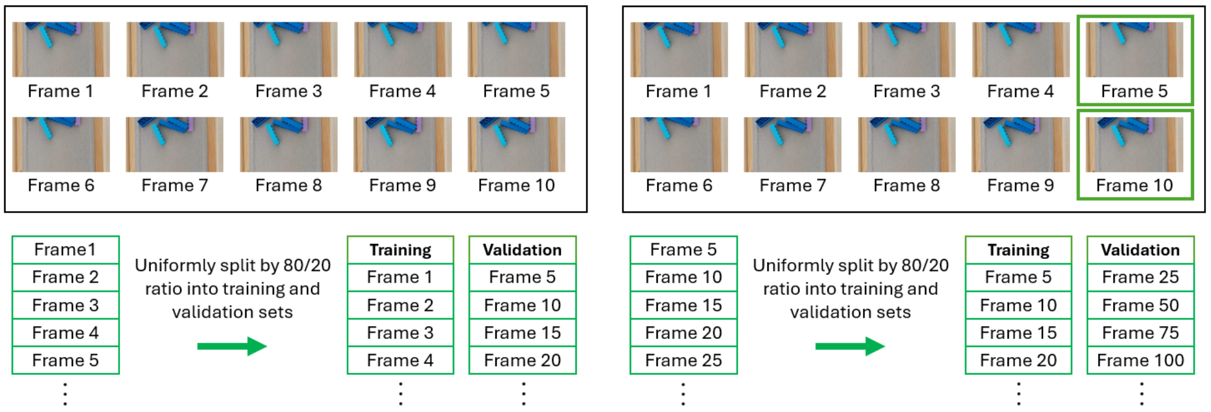
Table 3: Augmentation parameters and their descriptions.

ensured that the training and validation sets were both diverse and distinct. Similarly, the test dataset was created by uniformly sampling every fifth frame from the test videos.

2.3 Detection models

Detection of boundary box coordinates of the LEGO pieces is an initial stage to track them. For this purpose, YOLOv8 and RT-DETR from ultralytics were utilized [12]. YOLOv8 is one of the modern iterations of the YOLO (You Only Look Once) series and is designed for high-speed real-time object detection [13]. Focused on the architecture given in Figure 3a, it consists of three main components which are a Backbone, a Feature Pyramid Network , and a Detection Head. The Backbone aims for the extraction of hierarchical features from the input image. The Feature Pyramid Network fuses features across different scales, enabling the detection of objects of varying sizes. The Detection Head predicts bounding boxes, class probabilities, and confidence scores using an anchor-free approach. In general, it divides the image into a grid and predicting bounding boxes and class probabilities for each grid cell. YOLOv8 uses anchor-free predictions and applies a non-maximum suppression algorithm to reduce overlapping detections.

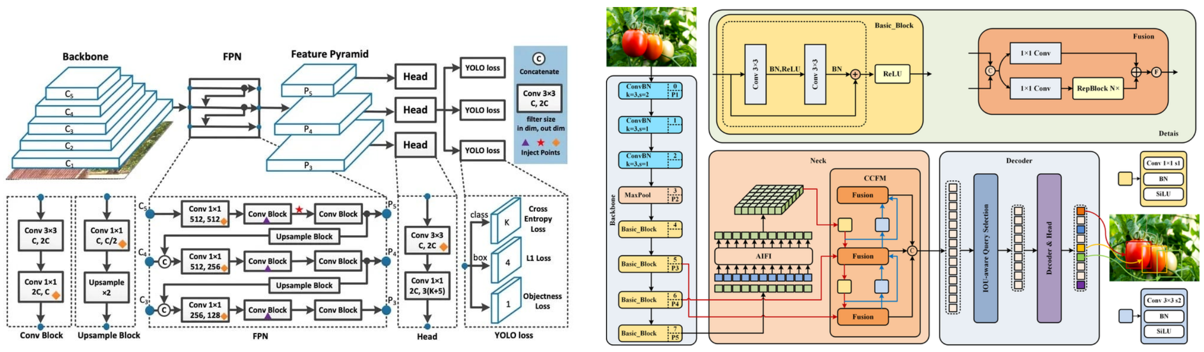
On the other hand, RT-DETR (Real-Time Detection Transformer) is a transformer-based object detection model optimized for real-time applications. Unlike YOLOv8's CNN-



(a) Normal data split (b) Systematic splitting

Figure 2: Data splitting of training video frames

based approach, RT-DETR utilizes an encoder-decoder mechanism, where the encoder transforms image features into embeddings, and the decoder uses learnable object queries to predict bounding boxes and class probabilities [14]. Figure 3b shows the working mechanism of RT-DETR, where the Efficient Hybrid Encoder extracts multi-scale features using a backbone network, combining low-level details and high-level semantics through the Adaptive Instance Feature Integration (AIFI) module for enhanced object detection. The IoU-aware Query Selection module dynamically matches object queries to high-quality candidate regions, improving precision and reducing computational demands. The architecture uses a transformer-based self-attention mechanism for global spatial relationships, bypassing anchor boxes to directly regress object locations.



(a) YOLOv8 architecture [15] (b) RT-DETR architecture [16]

Figure 3: Detection models

2.4 Semantic segmentation

In terms of distinguishing the objects from the environment and to overcome overlapping or other complex scenarios, segmentation can be applied as a part of the pipeline. During the research, DeepLabV3 was used as the semantic segmentation framework to achieve pixel-wise classification of images. DeepLabV3 is a semantic segmentation framework designed with an encoder-decoder architecture to effectively capture and process multi-scale features. Its encoder consists of Atrous Spatial Pyramid Pooling (ASPP) to integrate features at multiple dilation rates, combining low-level details with high-level context [17]. The decoder refines these features to produce precise segmentation maps, preserving fine details such as object boundaries. By leveraging image-level pooling and efficient up-sampling techniques, DeepLabV3 achieves accurate segmentation even in complex visual scenarios. This architecture generalizes across varying object sizes made it well-suited for this study, delivering efficient and high-quality results after fine-tuning.

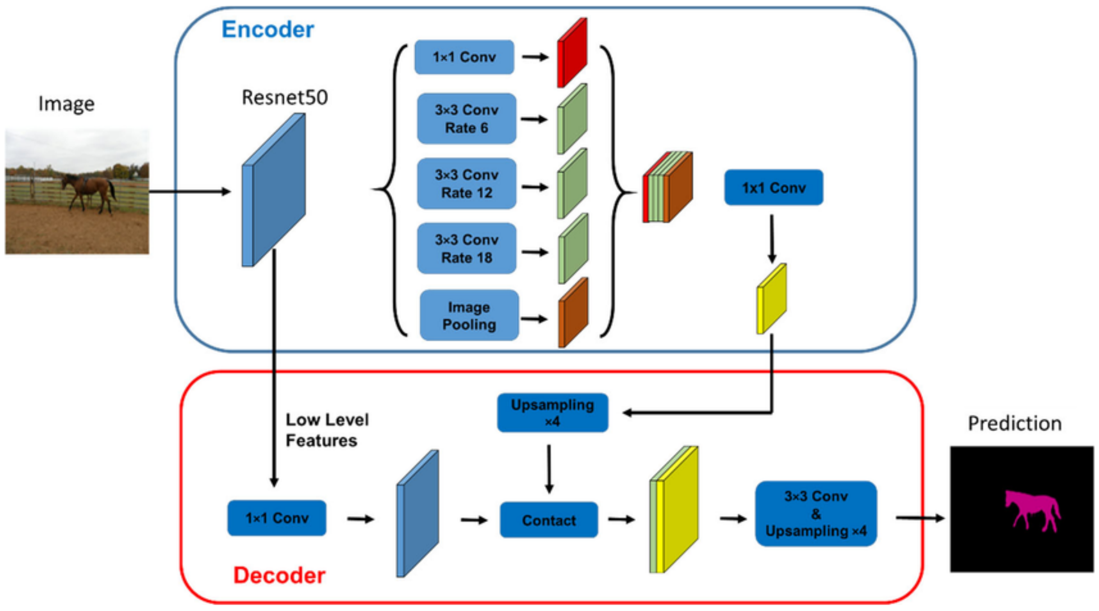


Figure 4: DeepLabV3 architecture [18]

2.5 Overlap Based Filtering

Overlap-based filtering, given in Figure 5, is a technique to refine model predictions by checking how well the segmentation mask aligns with the detection bounding box. The detection model outputs a bounding box, while the segmentation model provides a mask with white pixels representing the object. To decide if a detection is valid, the number of white pixels in the bounding box is divided by the total pixels in the box. If this ratio is above a set threshold, the detection is kept and otherwise, it is filtered out. This helps ensure only accurate detections are retained, improving the model's overall reliability. In the execution of the codes, threshold has been set to 0.3.

2.6 Tracking algorithm

For the last step of the pipeline and object counting, DeepSORT tracking algorithm was utilized to get consistent object identities across video frames. Rather than the detection algorithm applied at the beginning, DeepSORT builds upon the SORT algorithm by incorporating a deep network detection previously. This enhancement allows DeepSORT to address occlusions and re-identify objects that temporarily disappear from the frame. It adapts well to dynamic scenes with complex interactions, making it ideal for real-world applications. The methodology combines motion prediction using a Kalman filter

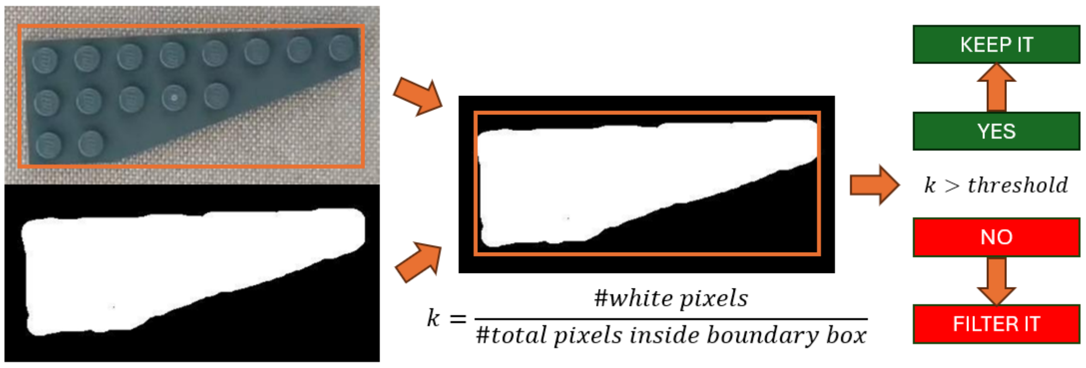

KEEP IT
YES
$k > threshold$
NO
FILTER IT
$k = \frac{\#white\ pixels}{\#total\ pixels\ inside\ boundary\ box}$

Figure 5: Overlap based filtering. (k is the overlap ratio)

with association metrics. The deep appearance descriptor further ensures reliable re-identification by associating objects based on their visual features. Sequentially, the Hungarian algorithm is then applied to efficiently solve the data association problem, minimizing the cost of linking detections and tracks. This integration of motion and appearance modeling makes DeepSORT a robust solution for tracking objects [19].

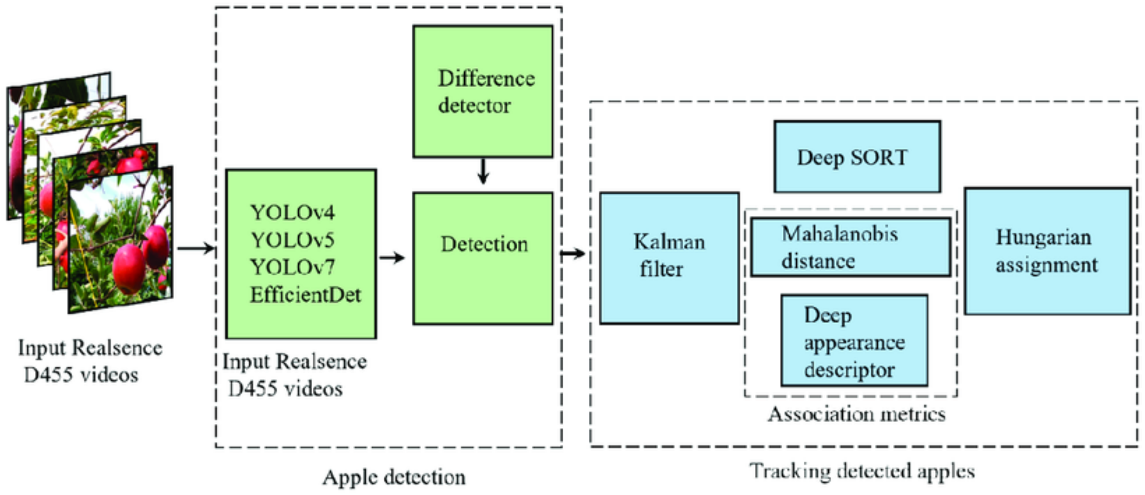
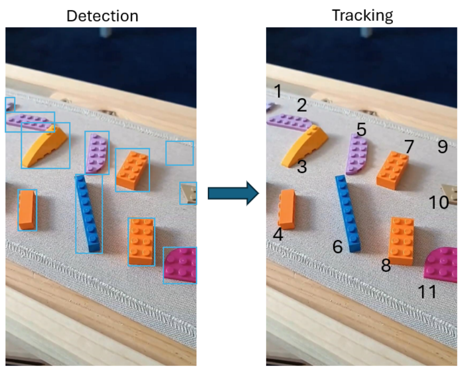


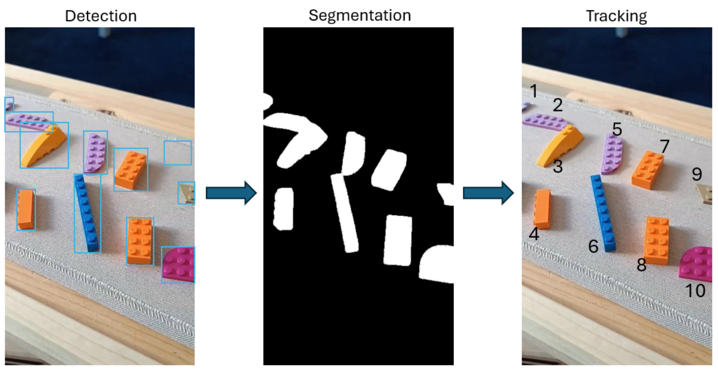
Figure 6: DeepSORT architecture [20].

2.7 Pipeline

Two distinct pipelines were implemented to evaluate the performance of object detection and tracking systems. The first pipeline as represented in Figure 7a involved a pure detection and tracking approach, where detected objects were directly fed into the DeepSORT algorithm. This pipeline relied solely on bounding box-level information for object localization and tracking, offering a simpler and faster solution but with limitations in getting more details about the detected objects. Figure 7b shows the diagram of the second pipeline which combines detection, semantic segmentation, and tracking. After detecting objects, semantic segmentation was applied to refine the object masks, providing pixel-level information about their shapes and boundaries. These refined outputs enhance accuracy in scenarios involving overlapping objects or cluttered scenes. Unlike the first pipeline, this approach offered more detailed object representation and improved tracking robustness. The main difference between the pipelines lies in their computational complexity. Although the inclusion of semantic segmentation added computational overhead but significantly improved precision in challenging scenarios. This comparison highlights the trade-off between efficiency and accuracy in object detection and tracking systems.



(a) Detection and tracking



(b) Detection, segmentation and tracking

Figure 7: Overview of the working mechanism of the pipelines

The pipeline used in the experiment is shown in Figure 8 represents the semantic segmentation, object detection, and tracking sequence in detail. Each frame is processed in parallel, where semantic segmentation identifies regions of interest, and object detection predicts bounding boxes for potential objects. An overlap-based filtering mechanism is then applied to ensure alignment between the segmented regions and detected bounding boxes, retaining only those that meet a specified threshold. The validated detections are passed to the DeepSORT tracking algorithm, which assigns unique IDs to objects and maintains their associations across frames.

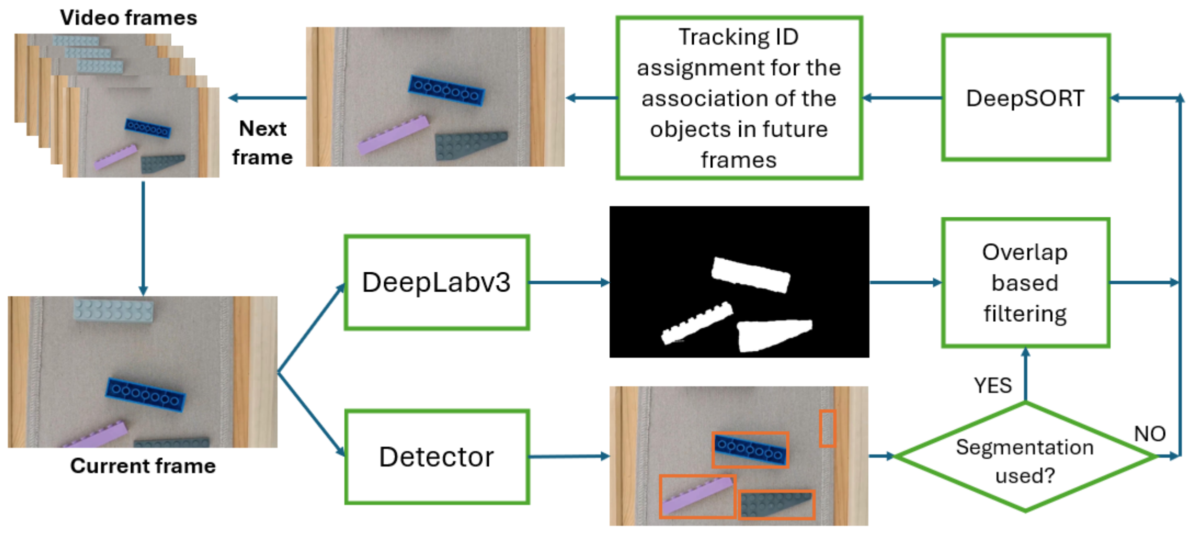


Figure 8: Pipeline. If segmentation is used it is like in Figure 7a, on the contrary, if segmentation is not used it turns to be Figure 7b.

3 Results and Discussions

3.1 Evaluation metrics

The evaluation metrics are essential to assess the performance of the models and to decide readiness of the model for the real world scenarios. As we have a pipeline of the models, we need to analyze their performance based on the type of the model.

3.1.1 Detection metrics

These metrics aim to compute how well the model detects the correct class and right boundary box coordinates of the objects. The main metrics for the detection models are precision, recall and mAP50, however before analyzing them the essence of terms like True Positive (TP), True Negative (TN), False Positive (FP) and False Negative (FN) must be taken into account to describe the results of the classification. Basically, they are used to compute other metrics such as accuracy, precision, recall to conclude an idea on the model's prediction.

$$\text{Precision} = \frac{\text{TP}}{\text{TP} + \text{FP}}$$

$$\text{Recall} = \frac{\text{TP}}{\text{TP} + \text{FN}}$$

Turning to Mean Average Precision (mAP), it is commonly used metric to quantify both the localization and classification performance of the model. Basically, it is the mean of the Average Precision (AP) values across all classes. AP is calculated as the area under the Precision-Recall curve for a single class. The AP is defined as:

$$\text{AP} = \int_0^1 \text{Precision}(\text{Recall})\, d\text{Recall}$$

In practice, the integral is approximated by summing the precision values at discrete recall points. To compute mAP, the AP values for all object classes are averaged. If there are N object classes, the mAP is given by:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i$$

A key component of mAP calculation is the Intersection over Union (IoU), which determines whether a predicted bounding box is a True Positive. An IoU threshold, commonly 0.5, is used to classify a detection as a True Positive. mAP at IoU threshold 0.5 is denoted as mAP50 [21]. They are defined as:

$$\text{IoU} = \frac{\text{Area of Overlap}}{\text{Area of Union}}$$

$$\text{mAP50} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i \, (\text{where IoU} \geq 0.5).$$

High mAP indicates that the model performs well in both detecting and correctly classifying objects across all classes.

3.1.2 Tracking metrics

To evaluate the performance of the multi-object tracking algorithms, the following metrics were used: MOTA (Multi-Object Tracking Accuracy), MOTP (Multi-Object Tracking Precision), IDF1 (Identity F1 Score), and other auxiliary metrics like Recall, Precision, and associated error metrics such as Number of Matches, Number of False Positives, Number of Misses, and Number of Identity Switches. Among these metrics, MOTA, MOTP and IDF1 are commonly evaluated. MOTA measures the overall tracking accuracy by penalizing missed detections, false positives, and identity switches. On the other side, MOTP evaluates the localization precision by calculating the average positional error between the predicted and ground truth objects [22]. Furthermore, IDF1 measures the ability of the tracker to maintain consistent identities throughout the sequence [23]. It is the harmonic mean of identity precision and identity recall. They are defined as :

$$\text{MOTA} = 1 - \frac{\sum_t (\text{FN}_t + \text{FP}_t + \text{IDSW}_t)}{\sum_t \text{GT}_t}$$

$$\text{MOTP} = \frac{\sum_{t,i} d_{i,t}}{\sum_t c_t}$$

$$\text{IDF1} = \frac{2 \cdot \text{ID Precision} \cdot \text{ID Recall}}{\text{ID Precision} + \text{ID Recall}}$$

FN_t, FP_t, IDSW_t and GT_t are number of false negatives (misses), false positives, identity switches and ground truth objects at time t in the formula of MOTA. For the equation of MOTP, $d_{i,t}$ is the distance between the predicted and ground truth position of object i at time t, while c_t is the number of matches at time t. Considering IDF1 score, ID Precision and ID Recall are the fraction of correctly identified tracks among all predicted tasks and ground truth tasks, responsively.

Additional error metrics used to analyze performance include:

- Precision is the proportion of correctly tracked objects among detected objects.
- Recall measures the ability of the tracker to detect ground truth objects.
- Number of Matches (c_t): The total number of correctly matched predicted and ground truth objects.
- False Positives (FP): Predicted objects that do not correspond to any ground truth.
- Misses (FN): Ground truth objects not detected by the tracker.
- Identity Switches (IDSW): Instances where the identity of a tracked object is incorrectly reassigned to another object.

3.2 Discussion of results

For the better performance of the pipeline each component or model must work accurately. Initial point is about the performance of the detection models and analysis of hyperparameters or pre-processing methods on the outputs. Table 4, Table 5 and Table 6 represent used training and post-processing hyperparameters together with the metrics obtained in the application of the detection models. Initially, based on the normal data split, Yolov8 model was fine-tuned with various hyperparameters and it is seen from Table 4 that the results are not satisfactory, however for Table 5, results significantly increased due to the uniform data split. To deeply investigate the tables, constant post-processing hyperparameters are seen in Table 4, however the performance of the model was so poor that other metrics based on the change of 'iou' and 'conf' values have not been saved as the results were also not satisfactory.

Num_epochs	**Batch size**	**Optimizer**	**lr0**	**lrf**	**iou**	**conf**	**Precision**	**Recall**	**mAP50**
30	**8**	**AdamW**	**0.01**	**0.01**	**0.7**	**0.25**	**0.925**	**0.779**	**0.911**
30	8	AdamW	0.001	0.01	0.7	0.25	0.912	0.672	0.643
30	8	AdamW	0.01	0.1	0.7	0.25	0.933	0.419	0.500
30	16	AdamW	0.01	0.01	0.7	0.25	0.889	0.724	0.841
40	8	SGD (Momentum)	0.01	0.01	0.7	0.25	0.917	0.738	0.890
40	8	RMSProp	0.01	0.01	0.7	0.25	0.906	0.220	0.430

Table 4: Yolov8 hyperparameters and metrics with normal data split on Lego dataset

Num_epochs	**Batch size**	**Optimizer**	**lr0**	**lrf**	**iou**	**conf**	**Precision**	**Recall**	**mAP50**
20	8	AdamW	0.01	0.01	0.5	0.5	0.995	0.938	0.969
20	8	RMSProp	0.01	0.01	0.7	0.25	0.982	0.92	0.945
20	**16**	**AdamW**	**0.001**	**0.1**	**0.7**	**0.25**	**0.989**	**0.948**	**0.975**
20	16	RMSProp	0.001	0.1	0.7	0.5	0.967	0.917	0.965
30	8	AdamW	0.01	0.01	0.7	0.25	0.984	0.932	0.97
30	8	RMSProp	0.01	0.01	0.5	0.5	0.978	0.961	0.958

Table 5: Yolov8 hyperparameters and metrics with uniform data split on Lego dataset

Because of the complexity of the tables, analyzing the effect of the pre-processing on the performance of the model can be done based on Figure 9. In the figure, results of the experiments represents the superiority of the uniformly sampled splitting approach over the normal data split. The results are taken from Table 4 and Table 5. This difference underscores the importance of thoughtful data preparation strategies in machine learning workflows. Uniform sampling ensures that both training and validation sets are representative of the entire dataset, leading to more robust evaluations. Considering that all hyperparameters are same except number of epochs and optimizer, the former one significantly improved the performance of the model in terms of precision, recall and mAP50. In normal splitting, all consecutive frames of the training videos were selected and split into training and validation sets. Due to this approach, a high similarity between training and validation datasets occurred. As a consequence, the lack of diversity between distribution of the sets made model struggle to generalize to unseen data. Low recall value indicates that the model failed to identify a large amount of positive cases in the test data. Although the precision is not as low as recall, its relatively small value compared to another method shows the presence of more false positives in the inference mode. In contrast, the uniform sampling strategy introduced diversity in the dataset and this resulted in dramatically higher metrics by effectively reducing the dataset diversity.

After determining the main pre-processing technique, merely that method is applied while RT-DETR model was fine-tuned. Moreover, as shown in Table 5, the metric results are

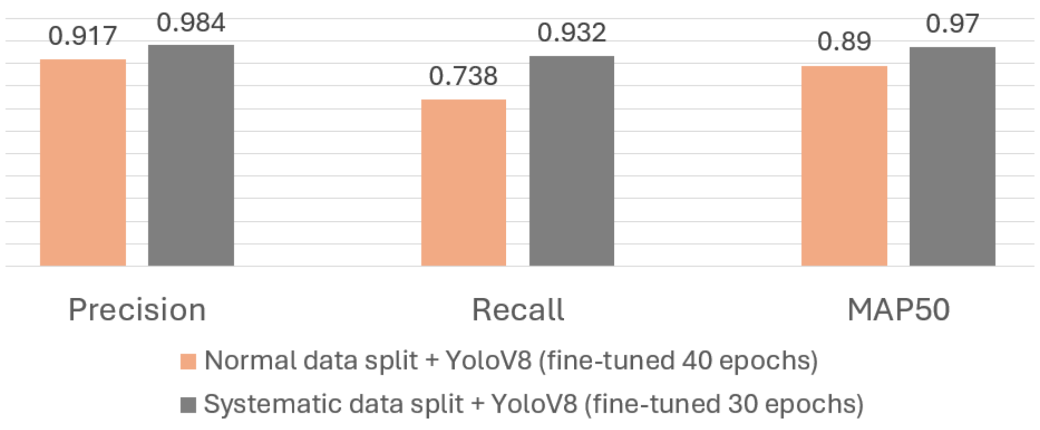


Figure 9: Effects of pre-processing methods on metrics.

better with 'AdamW' optimizer and taking that into account, same optimizer has been used during experiments with RT-DETR. This consistency in methodology ensures a fair comparison and highlights the importance of hyperparameter optimization in model performance. It also reflects a pragmatic approach to adapting models to specific use cases, such as detecting LEGO bricks. One of the main points about the application of model on LEGO bricks is about confidence threshold. The model detects the objects with a high level of the confidence and as it has more than 30M parameters, small number of epochs with larger batch size has been used in order to efficiently share the GPU resources. In addition, how initial learning rate of 0.01 deficiently affects the metrics are clearly seen from the first row of the table.

Num_epochs	Batch size	Optimizer	lr0	lrf	iou	conf	Precision	Recall	mAP50
10	8	AdamW	0.01	0.01	0.7	0.5	0.652	0.734	0.45
10	8	AdamW	0.001	0.1	0.7	0.9	0.97	0.92	0.94
10	16	AdamW	0.001	0.1	0.7	0.9	0.986	0.934	0.954
15	16	AdamW	0.001	0.01	0.7	0.7	0.96	0.921	0.932
15	**16**	**AdamW**	**0.001**	**0.1**	**0.7**	**0.9**	**0.97**	**0.944**	**0.956**
20	16	AdamW	0.001	0.01	0.5	0.7	0.94	0.923	0.94
20	16	AdamW	0.001	0.1	0.5	0.9	0.982	0.931	0.952

Table 6: RT-DETR hyperparameters and metrics with uniform data split on Lego dataset

For better analysis of the confidence level, Figure 10 shows the change of recall while increasing the post-processing confidence threshold for the models. One of the models is the selected Yolov8 with best results indicated in Table 5, another one is the selected RT-DETR model that is indicated in Table 6. This comparison highlights the varying sensitivity of the models to confidence thresholds, demonstrating Yolov8's reliance on precise tuning for optimal performance. Conversely, RT-DETR's stability under varying thresholds showcases its robustness for diverse scenarios. It can be seen that recall value decreases for Yolov8 as confidence level increases while RT-DETR keeps it at constant around 0.92-0.93 for all confidence levels.

Turning to the analysis of tracking metrics, Figure 11 and Figure 12 represents MOTA and MOTP percentages of pipelines according to DeepSORT hyperparameters. Use Yolov8 and RT-DETR models are ones that have been indicated in Table 5 and Table 6, respon-

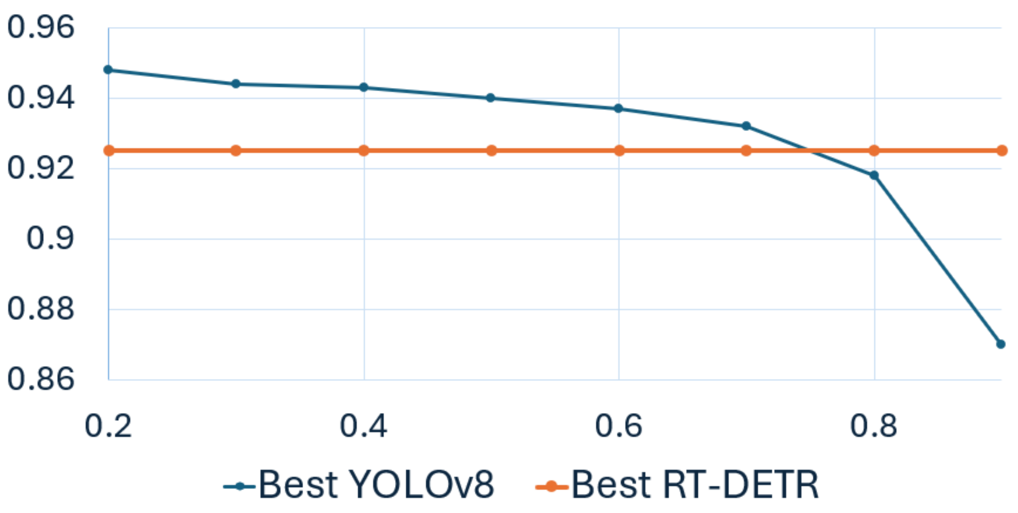


Figure 10: How confidence threshold influences recall

sively. Firstly, MOTA values are quite satisfactory and mostly more than 90%. So, overall tracking performance is accurate in terms of identifying and maintaining the correct objects over time. Considerint MOTP values, while objects are being successfully tracked, the exact positioning or overlap of the bounding boxes with the ground truth is not as refined. In addition, sometimes gaps occurring between detections can be result of not so high score.

Explaining hyperparameters, max_age defines the maximum number of frames a track can remain inactive before it is deleted, n_init is the minimum number of consecutive frames in which a detection is considered a valid track and nms_max_overlap determines the maximum allowable overlap between bounding boxes in non-maximum suppression during the detection process. In general, while max_age is equal to 10 and n_init is taken as 3, MOTA values are slightly better compared to others. On contrast, MOTP scores are significantly higher for black bars where "max age" is 20. The reason why the number of frames to detect the objects influences the pipeline is related to capturing more information. Thus the model gets more frames which increases the detection of the correct trajectory for the objects.

Focusing on the MOTA and MOTP results, it is seen how DeepLabV3 semantic segmentation enhances the performance of the model for the black bars. For the pipeline of RT-DETR and DeepSORT, MOTA is 84.8%, however after applying semantic segmentation, it increases by approximately 4.3% and reaches 89.1%. Another notable pattern is the rise of MOTP for Yolov8 and DeepSORT pipeline after the application of the segmentation from 76.8% to 78.6%. For some cases, the scores can decrease after the segmentation, which mainly occurs as the detection models have the maximum performance on the frames and segmentation model removes or tries to highly rectify the boundary boxes. Commonly, these declines do not affect the general working mechanism of the pipeline in a remarkable manner. Furthermore, grow in nms_max_overlap benefits MOTA but can slightly reduce MOTP, as it may allow overlapping detections to persist.

As can be seen, the hyperparameters used for green bars are relatively smaller than those for blue and orange ones. That is why their usage in detection-segmentation-tracking pipeline was redundant from getting higher result of point of the view and they were not applied with DeepLabV3 again. For investigating differences between the effect of the detection model on the general performance, although Yolov8 results in more MOTA, MOTP scores for RT-DETR model are higher or equal than that for Yolov8. Even after application of the segmentation, almost similar trend happens in the segmentation included pipeline with a few partial declines.

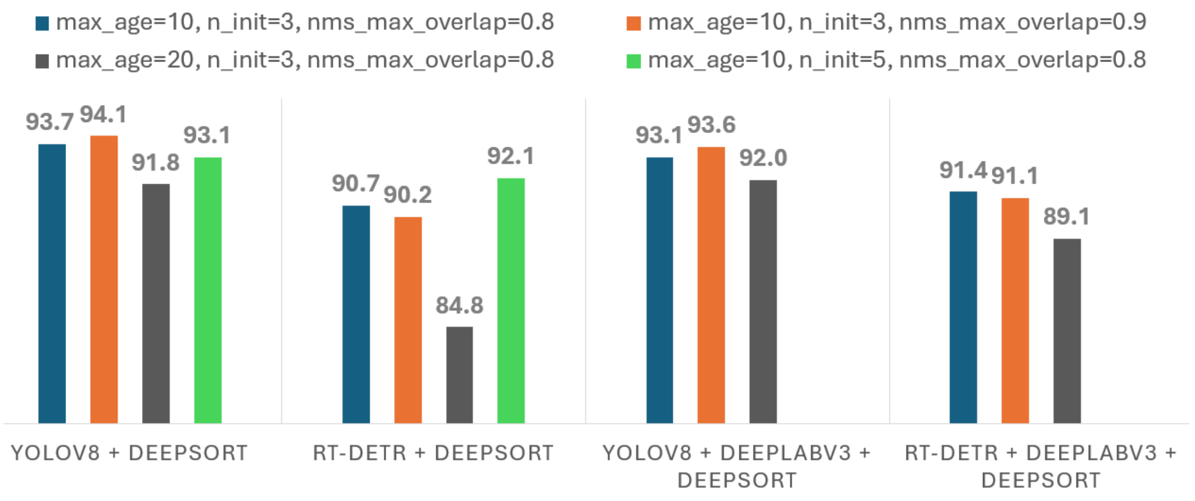


Figure 11: MOTA of pipelines based on DeepSORT hyperparameters

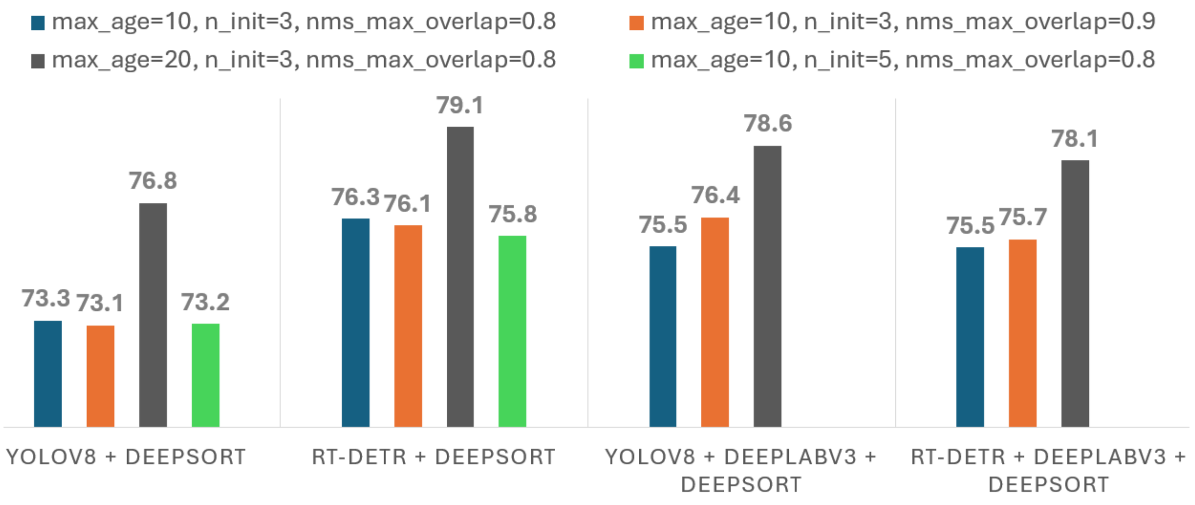


Figure 12: MOTP of pipelines based on DeepSORT hyperparameters

Considering max_age parameter as 10 and n_init parameter equal to 3, the results are better than other combinations of the hyperparameters whether nms_max_overlap is 0.8 or 0.9. That is why 0.8 is regarded as a nms_max_overlap to compare other metrics of the pipelines based on the best performance. Similarly, Figure 13 indicates the number of misses and false positives formed on the various pipelines considering static hyperparameter of the DeepSORT tracking algorithm. High number of false positives mean that the pipeline detects the wrong object type in which DeepLabv3 segmentation is used to handle the situation. On the contrary, misses occur when the system fails to detect or identify something that is actually present. In object detection or tracking, misses can be thought as false negatives which must be decreased to improve the performance.

Stand on Figure 13, it can be understood that after the application of segmentation between detection and tracking algorithms, there is a noteworthy fall off in the number of false positives which has gone from 254.5 to 211.5 for Yolov8 and decreased from 735 to 153 for RT-DETR detection pipeline. Segmentation helps reduce false positives and ensures that the model accurately distinguishes objects from their background. By focusing only on valid regions of interest, segmentation eliminates spurious detections, especially in complex environments. When combined with detection, segmentation enhances spatial precision and removes redundant or overlapping detections. This makes the system more reliable and reduces the chance of incorrectly identifying non-objects.

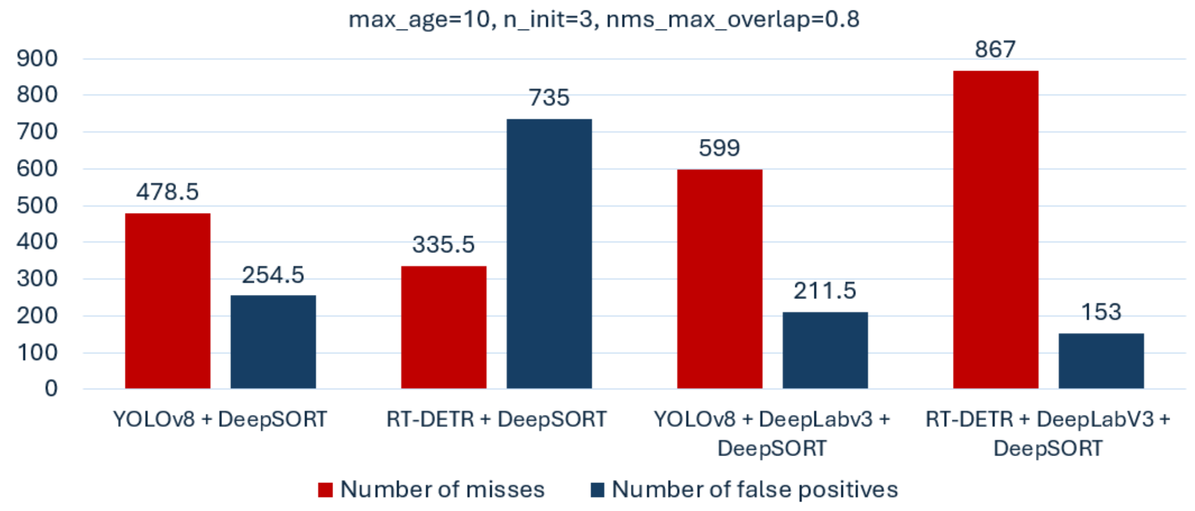


Figure 13: Comparison of pipelines based on number of misses and false positives

The problem is that merely reduction of false positives does not mean the pipeline works better. That is the reason why the number of misses must also be taken into account to check the performance. If segmentation is not applied correctly, the number of misses drastically boosts up as it is shown in the Figure 13. It increased from 478.5 to 599 for Yolov8 based pipeline, which was a slight change and balance was maintained. However, it climbed to 867 for RT-DETR pipeline after including the DeepLabV3 segmentation. It is obvious that somehow segmentation affects the capabilities of the model but lowering the number of false positives by keeping the number of misses same is the result of better pipeline.

Briefly explaining, incorporating segmentation can significantly reduce false positives by refining the regions of interest, enabling more accurate detection and tracking. But segmentation model must work with a high accuracy to enhance the pipeline effectively. Otherwise, it can strike a balance by increasing false positives with significant number of increasing misses.

4 Conclusion

The experiment evaluated detection, segmentation, and tracking pipelines based on the LEGO videos in which the aim was to build the object counter. For the pipeline, YOLOv8 and RT-DETR for detection, DeepLabV3 for segmentation, and DeepSORT for tracking were utilized. Initial challenge was overfitting of the YOLOv8 model during dataset preparation that was coped with systematic sampling to improve generalization. YOLOv8 demonstrated high detection precision and recall, while RT-DETR excelled in real-time detection. DeepSORT maintained consistent object identities in dynamic scenes, further strengthening the pipeline.

The results showed that integrating segmentation significantly enhanced tracking accuracy, particularly in handling occlusion and overlapping objects. The second pipeline, combining detection, segmentation, and tracking, outperformed the pure detection-and-tracking approach by improving MOTA and MOTP scores while refining object boundaries. In general, this research highlights combination of detection, segmentation, and tracking for robust performance in several scenarios.

All code, datasets, and additional materials related to this research are available in the GitHub repository - Object Tracking AI Lab

References

[1] S. K. Pal, A. Pramanik, J. Maiti, and P. Mitra, "Deep learning in multi-object detection and tracking: State of the art", *Springer Nature*, Apr. 2021. DOI: 10.1007/s10916-021-01756-5.

[2] L. Farkaš, "Object tracking and detection with yolov8 and strongsort algorithms captured by drone", Master's thesis, University of Split, Faculty of Science / Sveučilište u Splitu, Prirodoslovno-matematički fakultet, 2023. [Online]. Available: https://urn.nsk.hr/urn:nbn:hr:166:238937.

[3] A. Vats and D. C. Anastasiu, "Enhancing retail checkout through video inpainting, yolov8 detection, and deepsort tracking", in *AI City Challenge 2023 Proceedings*, Track 4: Multi-Class Product Counting and Recognition, Santa Clara, CA, USA, 2023. [Online]. Available: https://www.aicitychallenge.org.

[4] A. Wang, W. Qian, A. Li, *et al.*, "Nvw-yolov8s: An improved yolov8s network for real-time detection and segmentation of tomato fruits at different ripeness stages", *Computers and Electronics in Agriculture*, vol. 219, 2024.

[5] T. Yang, S. Zhou, A. Xu, J. Ye, and J. Yin, "An approach for plant leaf image segmentation based on yolov8 and the improved deeplabv3+", *Plants*, vol. 12, no. 19, 2023.

[6] H. Zhang, M. Cisse, Y. N. Dauphin, and D. Lopez-Paz, "Mixup: Beyond empirical risk minimization", in *International Conference on Learning Representations*, 2018. [Online]. Available: https://arxiv.org/abs/1710.09412.

[7] S. Yun, D. Han, S. J. Oh, S. Chun, J. Choe, and Y. Yoo, "Cutmix: Regularization strategy to train strong classifiers with localizable features", in *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, 2019, pp. 6023–6032. [Online]. Available: https://arxiv.org/abs/1905.04899.

[8] A. Bochkovskiy, C.-Y. Wang, and H.-Y. M. Liao, "Yolov4: Optimal speed and accuracy of object detection", *arXiv preprint arXiv:2004.10934*, 2020. [Online]. Available: https://arxiv.org/abs/2004.10934.

[9] E. Harris, I. Rubachev, V. Zhmoginov, and M. Sandler, "Fmix: Enhancing mixed sample data augmentation", in *NeurIPS Workshop on Machine Learning for Creativity and Design*, 2020. [Online]. Available: https://arxiv.org/abs/2002.12047.

[10] Y. Zhi, Z. Tong, L. Wang, and G. Wu, "MGSampler: An Explainable Sampling Strategy for Video Action Recognition", *arXiv preprint arXiv:2104.09952*, Aug. 2021, Version 2.

[11] Y. Xu and R. Goodacre, "On splitting training and validation set: A comparative study of cross-validation, bootstrap and systematic sampling for estimating the generalization performance of supervised learning", *Journal of Chemometrics*, vol. 32, no. 10, e3045, 2018. DOI: 10.1002/cem.3045.

[12] Ultralytics, *Ultralytics github repository*, https://github.com/ultralytics/ultralytics, Visited on 2023-09-01.

[13] J. R. Terven and D. M. Córdova-Esparza, "A comprehensive review of yolo architectures in computer vision: From yolov1 to yolov8 and yolo-nas", *Machine Learning and Knowledge Extraction*, vol. 5, no. 1, pp. 1–20, 2023. DOI: 10.3390/make5010001.

[14] Y. Zhao, W. Lv, S. Xu, *et al.*, “Detrs beat yolos on real-time object detection”, *arXiv preprint arXiv:2304.08069*, Apr. 2023. [Online]. Available: `https://arxiv.org/abs/2304.08069`.

[15] A. Sarhan, R. Abdel-Rahem, B. Darwish, *et al.*, “Egyptian car plate recognition based on yolov8, easy-ocr, and cnn”, *Springer Nature*, 2024.

[16] S. Wang, H. Jiang, J. Yang, *et al.*, “Lightweight tomato ripeness detection algorithm based on the improved rt-detr”, *Frontiers in Plant Science*, vol. 15, p. 1 415 297, Jul. 2024. DOI: `10.3389/fpls.2024.1415297`. [Online]. Available: `https://www.frontiersin.org/articles/10.3389/fpls.2024.1415297`.

[17] L.-C. Chen, G. Papandreou, F. Schroff, and H. Adam, “Rethinking atrous convolution for semantic image segmentation”, *arXiv preprint arXiv:1706.05587*, Dec. 2017. [Online]. Available: `https://arxiv.org/abs/1706.05587`.

[18] L. Kou, M. Sysyn, S. Fischer, J. Liu, *et al.*, “Optical rail surface crack detection method based on semantic segmentation: Replacement for magnetic particle inspection”, *Sensors*, vol. 22, no. 21, p. 8214, Oct. 2022. DOI: `10.3390/s22218214`. [Online]. Available: `https://www.researchgate.net/publication/364658172`.

[19] N. Wojke, A. Bewley, and D. Paulus, “Simple online and realtime tracking with a deep association metric”, in *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2017, pp. 3645–3649. DOI: `10.1109/ICIP.2017.8296962`.

[20] R. M. R. D. Abeyrathna, V. M. Nakaguchi, *et al.*, “Recognition and counting of apples in a dynamic state using a 3d camera and deep learning algorithms for robotic harvesting systems”, *Sensors*, vol. 23, no. 8, p. 3810, Apr. 2023. DOI: `10.3390/s23083810`. [Online]. Available: `https://www.researchgate.net/publication/369884480`.

[21] P. Henderson and V. Ferrari, “End-to-end training of object class detectors for mean average precision”, *arXiv preprint arXiv:1607.03476*, 2016. [Online]. Available: `https://arxiv.org/abs/1607.03476`.

[22] K. Bernardin and R. Stiefelhagen, “Evaluating multiple object tracking performance: The clear mot metrics”, *EURASIP Journal on Image and Video Processing*, vol. 2008, pp. 1–10, 2008.

[23] E. Ristani, F. Solera, R. S. Zou, R. Cucchiara, and C. Tomasi, “Performance measures and a data set for multi-target, multi-camera tracking”, in *Proceedings of the European Conference on Computer Vision (ECCV) Workshop*, Oct. 2016, pp. 17–35.