Journal of Internet Services and Applications, 2024, 15:1, doi: 10.5753/jisa.2024.3914

OneTrack - An End2End approach to enhance MOT with Transformers

Luiz Araujo [**Universidade Federal do Amazonas** | ***luizcls@icomp.ufam.edu.br***]
Carlos Figueiredo [**Universidade do Estado do Amazonas** | ***cfigueiredo@uea.edu.br***]

Institute of Computing, Universidade Federal do Amazonas, Av. General Rodrigo Octavio Jordão Ramos, 1200 - Coroado I, Manaus - AM, 69067-005, Brazil.

Received: 11 December 2023 • **Accepted:** 04 June 2024 • **Published:** 02 September 2024

Abstract This paper introduces OneTrack, an innovative end-to-end transformer-based model for Multiple Object Tracking (MOT), focusing on enhancing efficiency without significantly compromising accuracy. Addressing the challenges inherent in MOT, such as occlusions, varied object sizes, and motion prediction, OneTrack leverages the power of vision transformers and attention layers, optimizing them for real-time applications. Utilizing a unique Object Sequence Patch Input and a Vision Transformer Encoder, the model simplifies the standard transformer approach by employing only the encoder component, significantly reducing computational costs. This approach is validated using the MOT17 dataset, a benchmark in the field, ensuring a comprehensive evaluation against established metrics like MOTA, HOTA, and IDF1. The experimental results demonstrate OneTrack's capability to outperform other transformer-based models in inference speed, with a marginal trade-off in accuracy metrics. The model's inherent design limitations, such as a maximum of 100 objects per window, are adjustable to suit specific applications, offering flexibility in various scenarios. The conclusion highlights the model's potential as a lightweight solution for MOT tasks, suggesting future work directions that include exploring alternative data representations and encoders, and developing a dedicated loss function to further enhance detection and tracking capabilities. OneTrack presents a promising step towards efficient and effective MOT solutions, catering to the demands of real-time applications.

Keywords: Multiple Object Tracking, transformers, efficient tracking, joint detection and tracking, deep learning.

1 Introduction

Computer vision plays a pivotal role in the broader field of artificial intelligence. The task of enabling machines to interpret visual data presents ongoing challenges. Since the advent of deep learning, innovative approaches have enabled significant advancements across all subfields of computer vision. In facial recognition, the initial approach framed the problem as a classification task, as demonstrated in Schroff *et al*. [2015]. This evolved into models that extract feature vectors to maximize distance measurements for out-of-training data evaluation, as in Deng *et al*. [2022]. In object detection, early models like those in Girshick *et al*. [2014] required spatial cues, such as region proposals, to identify objects within a scene. This approach was revolutionized by Redmon *et al*. [2016], introducing a fast, end-to-end methodology that eliminated the need for an intermediary network. In Multiple Object Tracking (MOT), the field has experienced rapid evolution, transitioning from statistical and heuristic methods to predominantly using various deep learning techniques, reflecting a dynamic and continuously developing landscape.

Current state-of-the-art MOT solutions primarily focus on accuracy in making correct inferences, often sacrificing efficiency, crucial for real-time applications. Many top-performing models, such as Wang *et al*. [2023] and Zhang *et al*. [2023], rely on separate object detection models, trained independently of the tracking component, impacting overall system effectiveness [Reference in Zhang *et al*. [2021]]. Some models employ characteristic differentiation modules, an alternative to traditional motion tracking modules, whose effectiveness in complex scenarios is a subject of ongoing research [Wang *et al*., 2023; Du *et al*., 2023]. A significant challenge arises with MOT solutions using Transformers for temporal contextualization, like Zeng *et al*. [2022]; Sun *et al*. [2021]; Zhang *et al*. [2023]. Despite their innovative approach, these models have struggled with efficiency. Additionally, occlusion, where objects overlap or are temporarily obscured, presents a persistent problem, significantly degrading tracking accuracy.

This work introduces an approach for the real-time use of transformers in MOT systems. We employ a paradigm that simplifies a common step in transformer-based models by eschewing a decoder model for generating object detections and tracks. Thus, the encoder serves as the sole backbone for temporal interpretation of the data, significantly reducing processing time and enhancing speed compared to similar models. While not our primary focus, we address occlusion handling by enlarging the context window for each inference. Our data pre-processing method enables a larger window with minimal impact on inference speed. The contributions of this work are as follows:

- Introduction of **OneTrack**: an end-to-end transformer-based MOT model that surpasses other transformer models in inference speed.
- Development of the **Object Sequence Patch Input** pre-

processing step, making our model nearly invariant to window size changes in inference speed.

The necessity for research on reducing inference times in MOT systems is underscored by applications that depend on real-time processing. One such application is autonomous vehicles, which rely on rapid responses to avoid obstacles and prevent collisions. Similarly, surveillance systems require quick processing to detect anomalies, and suspicious behavior, and to track individuals effectively, necessitating low inference time costs. In robotics and other fields that demand real-time interpretation of the environment in which the intelligent agent operates, it is crucial for the model to provide swift responses.

The article is structured as follows: Section 2 presents a deep and critical analysis of the field, including both transformer and non-transformer methods and forms the basis for some decisions. Section 3 describes all methods and models developed for this project. Section 4 details experiments and results, comparing them with other MOT models. Finally, Section 5 concludes the discussion and suggests directions for future work.

2 Related Works

Multiple Object Tracking is a crucial area in computer vision, with applications ranging from surveillance to autonomous driving. This section focuses on the application of vision transformers in MOT, particularly in the context of the proposed OneTrack model. The goal is to understand how recent advances in vision transformers and data modeling techniques contribute to the efficiency and accuracy of MOT systems.

MOT faces challenges such as occlusions, variations in object size, and motion prediction, making it a complex problem for computer vision. The advent of Machine Learning, and especially Deep Learning, revolutionized the field, moving away from earlier statistical methods and heuristics.

Although there are works that focus on methods to solve offline situations, such as what's presented in Pitie *et al*. [2005], we choose to not focus on such approaches, as they do not require fast inference. As the name suggests, these models also use information from future frames to predict current frame information, meaning it's impossible to run in real time. For such scenarios, online methods are employed, and inference time are important to avoid losing frames due to long processing times. Taking these aspects into account, this work will focus on evaluating the state of current online methods only.

2.1 End-to-End model VS Composite Models

This design decision defines how the model's architecture will process data. In Composite models, tasks are clearly defined, with separate models for detecting objects in a frame and a model responsible for tracking. It also most of the times need post-processing to associate objects to tracklets. Examples include [Bewley *et al*., 2016], [Aharon *et al*., 2022], [Zhang *et al*., 2021], [Du *et al*., 2023], [Zhang *et al*., 2022]. These models, which learn each process step in isolation, are currently the state of the art.

End-to-end models, in contrast, do not have this task separation and typically incorporate a tool for analyzing temporal and spatial frame information. Notable examples are [Zhou *et al*., 2020], [Zeng *et al*., 2022], [Meinhardt *et al*., 2022], [Sun *et al*., 2021]. These models, however, face challenges in both efficiency and efficacy, as they struggle to simultaneously perform both tasks.

2.2 Tracking Method

The tracking method, determined after earlier decisions, defines how the data will be modeled. Models vary in their approach, from calculating object centroid displacement [Mostafa *et al*., 2022] to comparing objects over time [Aharon *et al*., 2022] and predicting future positions [Bewley *et al*., 2016].

2.3 Machine Learning and Deep Learning in MOT

The emergence and diffusion of Deep Learning techniques have broken barriers in MOT, especially in challenging scenarios. Techniques like Convolutional Neural Networks (CNN) are commonly used for extracting local image information, as seen in models like [Du *et al*., 2023], [Zhang *et al*., 2022], [Meinhardt *et al*., 2022]. Additionally, Recursive Neural Networks (RNN) are employed for maintaining relevant temporal information, while transformers, particularly attention layers, have become prominent for tracking in video sequences ([Zeng *et al*., 2022], [Meinhardt *et al*., 2022]). Re-identification modules, used in models like [Aharon *et al*., 2022], [Mostafa *et al*., 2022], focus on object appearance to measure similarity and associate objects.

2.4 Transformers in Computer Vision

Transformers have introduced an innovative approach to computer vision, particularly in MOT. They treat images as sequences of patches, capturing global dependencies effectively. This is exemplified by the Vision Transformer (ViT) [Dosovitskiy *et al*., 2021]. In MOT, transformers integrate spatial and temporal information, aiding in object reidentification and making MOT systems more robust. However, their computational cost and the challenge of data modeling for MOT are significant limitations ([Zeng *et al*., 2022], [Zhou *et al*., 2020]).

2.5 Transformers in MOT

Following the works in correlated fields, transformers were employed in various research to give temporal information and attention to MOT models. Coincidentally, most of these models can operate in end-to-end architectures, meaning a single model is responsible for the entire inference. In [Zeng *et al*., 2022], the model utilizes an encoder-decoder schema to detect and associate IDs to objects at once. Similarly, [Meinhardt *et al*., 2022] also uses this architecture format to do the tracking. While [Zhang *et al*., 2023], noticing the

Table 1. Summary of Related Works in MOT

Works	Methods	Architecture	Attention	Limitations
DeepSORT [Wojke *et al*., 2017]	Appearance association, Kalman filter	Detection + Association	No	Limited in complex scenarios
StrongSORT [Du *et al*., 2023]	Enhanced DeepSORT with improved models	Detection + Embedding + Association	No	Limited in complex scenarios
BotSORT [Aharon *et al*., 2022]	Tracking-by-detection, IoU-ReID fusion	Detector + Associator	No	Camera motion issues in dense scenes
FairMOT [Zhang *et al*., 2021]	Object detection, re-ID features	End-to-end training, requires post-processing	No	Struggles with occlusion and scale variations
ByteTrack [Zhang *et al*., 2022]	Data Association via BYTE, YOLOX	Detector, Associator and ReID	No	Issues with camera motion, occlusion, motion blur
CenterTrack [Zhou *et al*., 2020]	Objects as points, two frames and heatmap	End-to-End	No	Efficiency and efficacy issues
MOTR [Zeng *et al*., 2022]	Set prediction problem, track and detect queries	End-to-End	Yes	High computational cost
MOTRv2 [Zhang *et al*., 2023]	Similar to MOTR, separate YOLOX detector	Separate detection, end-to-end tracking	Yes	High computational cost
TrackFormer [Meinhardt *et al*., 2022]	Object and Tracking Queries	End-to-End	Yes	Needs large datasets, high memory and computational costs
Transtrack [Sun *et al*., 2021]	CNN feature extraction, dual decoders	End-to-End	Yes	Robustness issues with object queries, no long-term info
LMOT [Mostafa *et al*., 2022]	DLA-34 Encoder, linear transformer	Separate training for tracker and detector	Yes	Struggles with varying environment conditions
OneTrack [Mostafa *et al*., 2022]	Vision Transformer Encoder, Object Average Vector, Yolo feature extraction	End-to-End model	Yes	Fixed limited amount of entities per window, lower accuracy metrics compared to slower models, duplicated IDs.

difficulty in performing the object detection by the transformer model itself, decides to use another separate model (YOLOX [Ge *et al*., 2021]) for this task. Another common factor among these models is using both encoder and decoder. Also, the models share a flaw in their inference times, which can be related to this fact.

2.6 Summary of Related Works

In Table 1, we summarize the findings, including a brief description of the method, architecture, whether the model used attention and their respective limitations. We expect this to help have a broader understanding of the current state of Online MOT, also allowing a clear description of the differences between past works and the proposed model.

Notice that our model, OneTrack, has a similar configuration to some of the works brought here. The novelty presented here is how we used the transformer in the architecture, focusing on maintaining practical inference speed, rather than keeping the focus on accuracy-related metrics. We discuss these changes, and how they affect both aspects of the model in further sections of this work, but highlight here the technical differences between current literature and this work.

3 OneTrack - Leveraging Tracking Tokens and Box Encodings for MOT

The proposed method in this work primarily aims to enable the use of transformer models and attention layers in systems addressing Multiple Object Tracking (MOT). Based on this

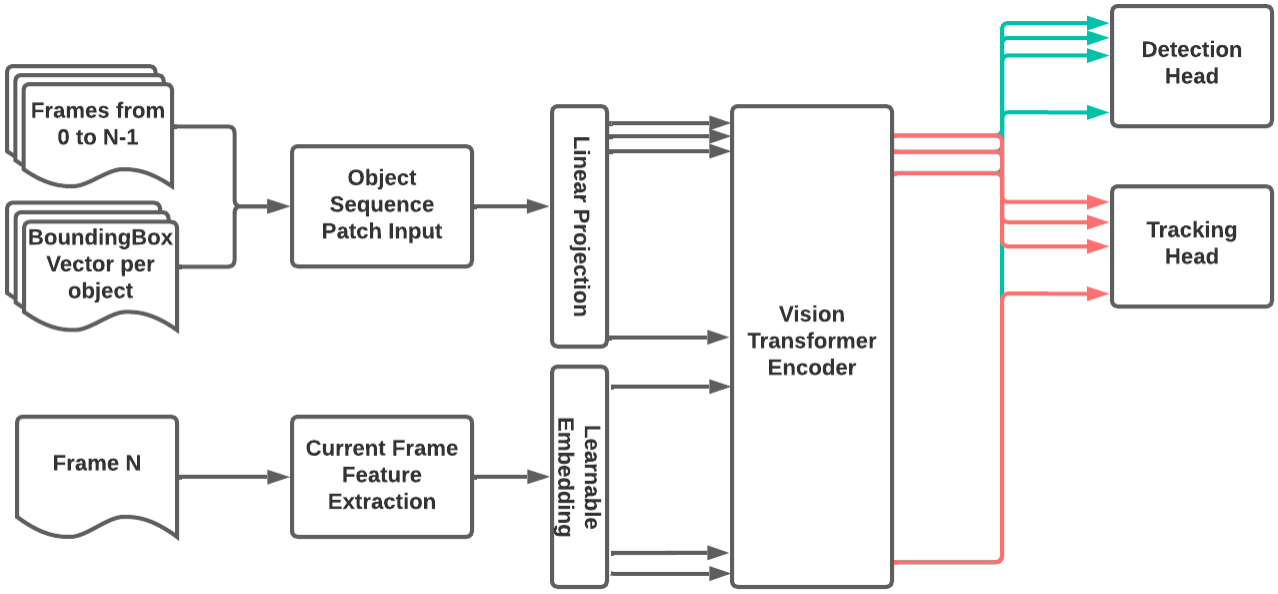

Figure 1. General Architecture of the OneTrack model.

and the points discussed in section 2, this article focuses on an end-to-end model for online operation using transformers for MOT. Efficiency was a key consideration throughout the development process.

3.1 Architecture

The model operates on the hypothesis that by processing a sequence of object cutouts, with an encoding representing their positions over time, it will learn to deduce their current frame position to the sequence. This approach aligns with the concept of set prediction, as proposed by DETR and MOTR in Carion *et al*. [2020] and Zeng *et al*. [2022], respectively.

In the architecture diagram, shown in Figure 1, it's possible to see each component which composes the full model pipeline. There are two types of inputs: a historical input set and a current instant set. The first two, being the frame images themselves, as well as the called Bounding Box Vectors, correspond to the historical data. These are processed and used as context for the model. Then there is just the frame in instant N, which is used to answer what is the current position of each object, as well as relates them to their past IDs.

Each input has its own preprocessing step, as well as a projection step, which turns the inputs into a one-dimensional array representation for each token that will compose the sequence.

Then to process the information built into the input sequence, and correlate both positions and IDs over time, the vision transformer pre-trained encoder is used. Each projected attention map will be then processed through a set of Fully Connected layers to produce the prediction for each detected object, which means their ID and their bounding box coordinates.

Each one of these elements is detailed further in this work.

3.2 Object Sequence Patch Input and Linear Projection

A critical aspect of MOT is data handling. This model aims to simultaneously define *bounding boxes* and IDs for each valid object in a frame. Unlike object detection and classification, a MOT model lacks a deterministic and well-defined reference for classifying each object.

To maintain data abstraction, we structured the data so that each object within a time window is mapped to an ID ranging from zero to the maximum sequence length minus one. Post-processing involves unmapping these IDs for consistency and assigning new IDs for new objects. For instance, Figure 2 shows that two objects were identified first with id values of 13 and 27, respectively. Then the mapping process assigns new values for each object but keeps a reference of this change, which is used after the tracking step, in an unmapping process, giving each object its original, global, ID.

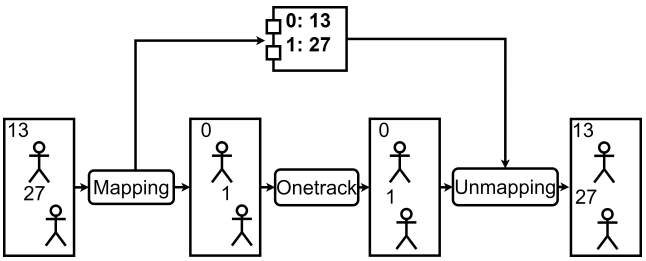

Figure 2. ID re assignation process. To allow the model to consider each object an individual class in its context window.

The architecture hypothesis necessitates a specific data input method. Let K be a value corresponding to the context window size. This means previous frames will be in the range of $0 \leq i \leq K-2$. Each frame will have a subset of tracked objects. Each subset O will have a different amount of objects. But in total, there will be M individual objects, which have to be constrained to N, meaning $M \leq N$. By cropping each object from each frame based on their bound-

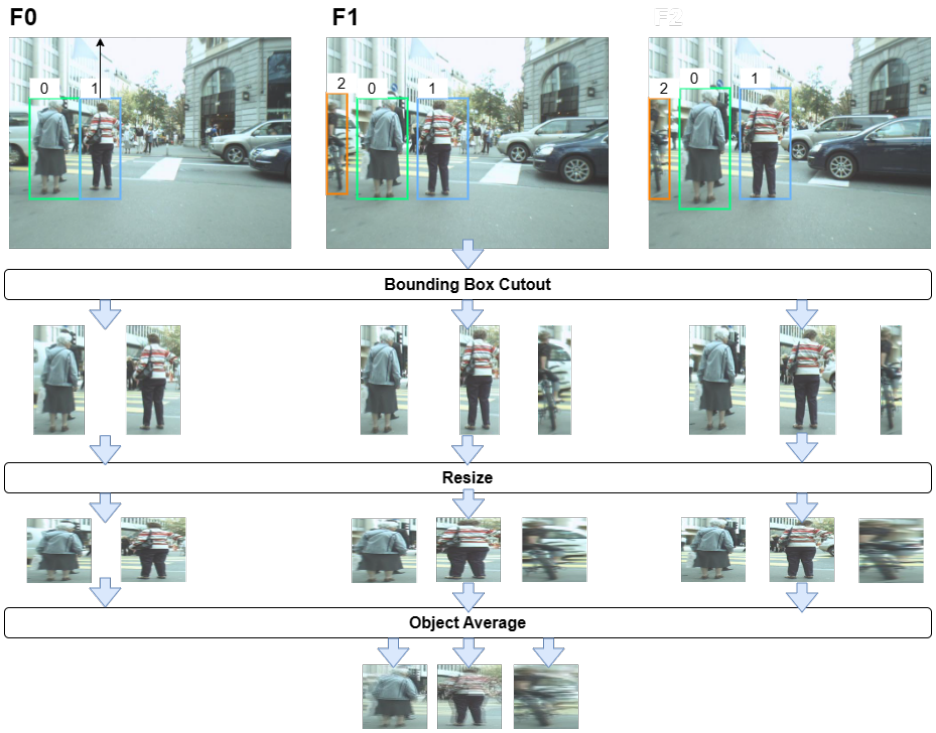

Figure 3. Object Average Vector generation process.

ing box, we can have an array that would look like the following:

$$\begin{bmatrix} f_{00} & f_{01} & \cdots & f_{0o} \\ f_{10} & f_{21} & \cdots & f_{1p} \\ \cdots & \cdots & \cdots & \cdots \\ f_{(K-2)0} & f_{(K-2)1} & \cdots & f_{(K-2)q} \end{bmatrix}$$

where $[o, p, \ldots, q]$ are the number of objects for each frame i. Some elements in this matrix might be empty, meaning the object wasn't found in the corresponding frame. Each column of the matrix has every occurrence of each object along the time window. A resulting matrix would look like this:

$$\begin{bmatrix} f_{m1} \\ f_{m2} \\ \cdots \\ f_{m(K-2)} \end{bmatrix}$$

Each element f_{mi} is created through the following process:

1. Cut out all detections of the object with ID i within the current time window.
2. Resize them to a fixed size of 64x64x3.
3. Calculate a simple pixel-by-pixel average for all images in this set.
4. Apply a dense layer for linear projection of each flattened image average.
5. Fill this stack of linear projections with zero-valued vectors up to N.

This process yields a N 968-position vector corresponding to each object. Steps 1 through 3 can be visualized in Figure 3 yielding Object Averaged Vectors (OAVs), while steps 4 and 5 are illustrated in Figure 4. It is also based on the works of ViT, which, by default, has this shape as the dimension of the each token in the input sequence. We call each position of the final vector a Tracking Token.

To give the model an understanding of each object's movement, we encode the coordinates of each object in past frames, forming a vector of size $(F-1)*4$, where F is the window size and the dimension's size 4 is related to the coordinates of a bounding box: x center, y center, width, and height. Similarly to OAVs, this can be represented by the following matrix:

$$\begin{bmatrix} b_{00} & b_{01} & \cdots & b_{0o} \\ b_{10} & b_{21} & \cdots & b_{1p} \\ \cdots & \cdots & \cdots & \cdots \\ b_{(K-2)0} & b_{(K-2)1} & \cdots & b_{(K-2)q} \end{bmatrix}$$

where $[o, p, \ldots, q]$ are still the number of objects for each frame i, but each element b_{io}, is actually another tuple containing the 4 previously mentioned coordinates.

This vector is placed in a map of learnable *embeddings*, providing local context to the transformer model. These embeddings, as well as the linear projections of each cutout, are 768-position vectors, the same input shape as ViT's encoder transformer.

In the Object Sequence Patch Input module, all data fed contains past information about the previous frames, detections and tracklets. So to add the information from the current frame, we use a convolutional neural network as a feature extractor for this image. This means the image, resized to $[3, 640, 640]$, will be passed through the convolutional layers, and in the end a feature map will be generated.

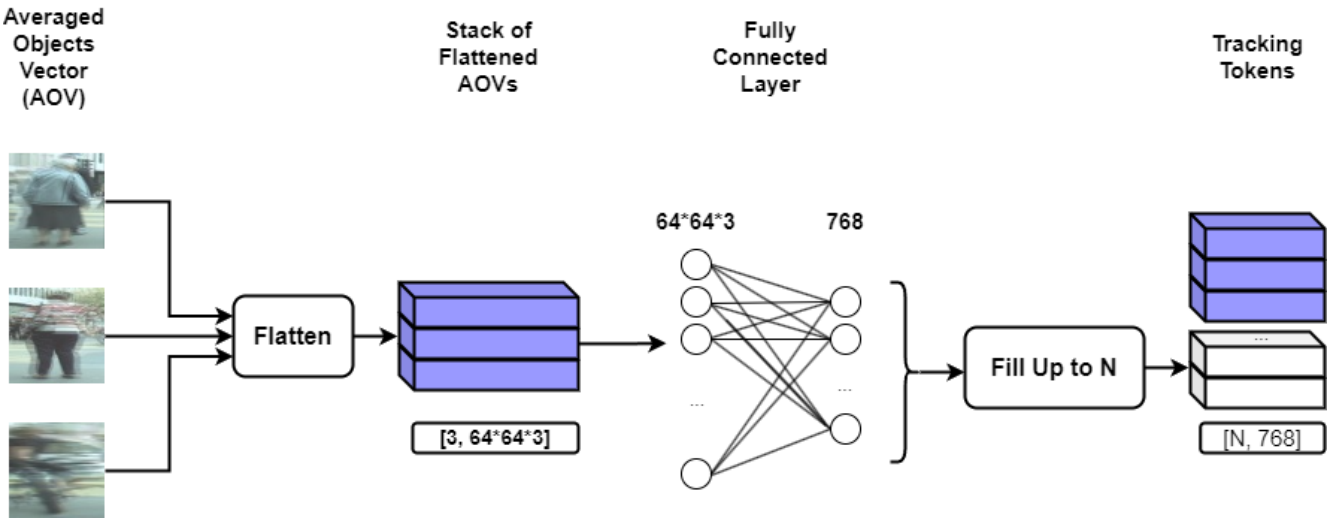

Figure 4. Tracking Tokens generation process.

3.3 Current Frame Feature Extraction and Learnable Embeddings

In the Object Sequence Patch Input module, all data fed contains past information about the previous frames, detections and tracklets. So to add the information from the current frame, we use a convolutional neural network as a feature extractor for this image. This means the image, resized to $[3, 640, 640]$, will be passed through the convolutional layers, and in the end a feature map will be generated. Since a big challenge for these end-to-end models is the object detection aspect, we decided that a good candidate for a feature extraction model would be the backbone of an object detection model. In this case, we used the backbone of YOLOv7 [Wang *et al*., 2022]. In the end of this process, the feature map extracted has shape $[10, 10, 768]$, which is reshaped into $[100, 768]$, becoming a sequence of 100 vectors with 768 positions each, compatible with the Tracking Tokens extracted. On the contrary of some approaches like in Transtrack [Sun *et al*., 2021], which does not use any type of positional encoding for the feature map, we found that this approach wasn't effective in allowing our particular model to learn, but using a set of learnable and randomly initialized embeddings yielded better results than not using them.

3.4 Vision Transformer Encoder

A literature review of MOT models using transformers reveals a classical use of both encoder and decoder. While this is logical, it often results in higher computational costs.

To mitigate this, we employed only the encoder. The Vision Transformer (ViT) model [Dosovitskiy *et al*., 2021] uses this paradigm effectively for image classification tasks. Despite MOT being more complex, ViT's success indicates its potential applicability in MOT. Another model that used this approach and reported state-of-the-art results is in video-swin-transformer, [Liu *et al*., 2021].

Thus, we utilized only the Encoder part of ViT, pre-trained on the ImageNet [Deng *et al*., 2009] dataset. The weights were not frozen due to the different nature of the task and the availability of sufficient data for the model to learn MOT-specific concepts. A more detailed description on the data used for these tests can be found in Section 4.1.

The input and output size of this part of the model was fixed. This was necessary because of the way we track IDs in a scene. Being analogous to classification, it can only contain a pre-determined amount of options. The value was chosen based on tests regarding our target data. If used with another specific domain in mind, this value could be a hyperparameter to be tuned. We illustrate this part of the architecture in Figure 5.

The output of this part of the model represents $N + 100$ attention maps, with the first N relating previous (or empty) objects to potential counterparts in the current frame, either as continuations of their trajectories or as new entries, while the last hundred are not used. This is similar to how ViT uses the encoder by having a classification token. In our case, we have defined 100 tracking tokens to be a sizeable amount (this value was defined by the results obtained in Section 4.2.1), either initialized with values for each tracked object in the frame window, or with empty values, representing positions that could be occupied by new entries.

3.5 Tracking and Detection Heads

These two branches of the network comprise sequences of 3 fully connected layers, with sizes $[768, 768, Output]$. The networks can be seen in detail in Figure 6. Both receive the same input vector, but one head is tasked with detecting bounding boxes via 4-way regression, while the other assigns an ID to each object, from which we can infer based on previous frames information whether the object is new or matched to an existing one.

The output layer size varies for each head. For detection, $Output = 4$, denoting the *bounding box* coordinates. For tracking, $Output = N$, the limit of objects per window, analogous to classification, but aimed at assigning each "class" to a similar or new object from the input. This process is done in parallel. Note that, the only post-processing step needed is doing the process of unmapping defined in Figure 2 and assigning new IDs to unmatched elements.

This approach follow previous works that, when doing

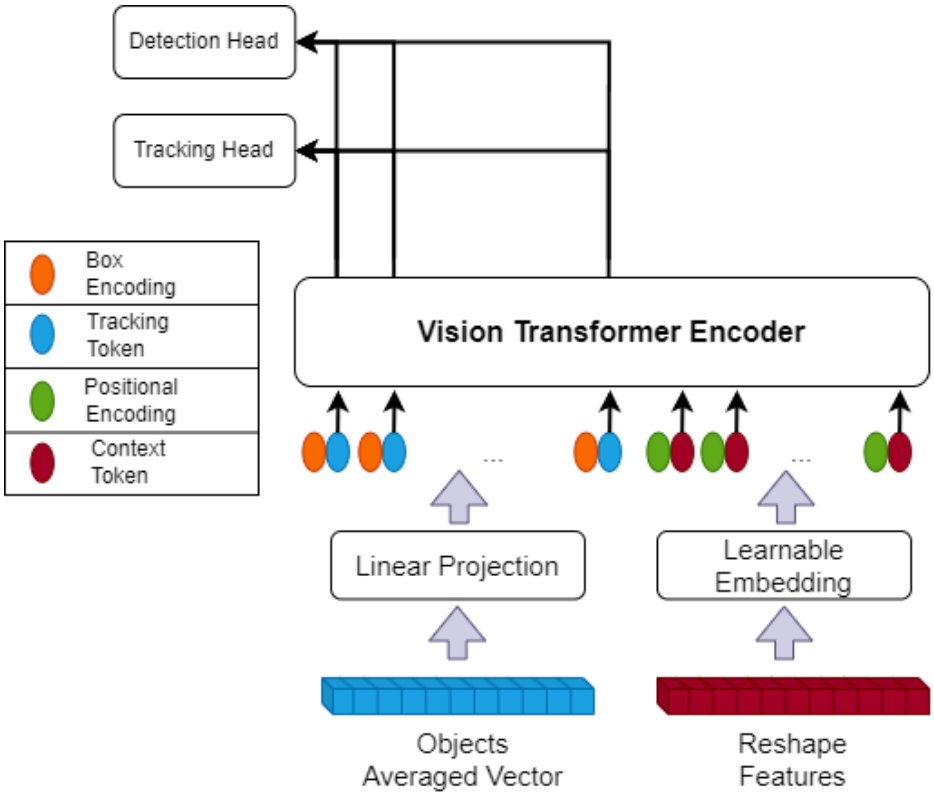

Figure 5. Vision Transformer Encoder schematics.

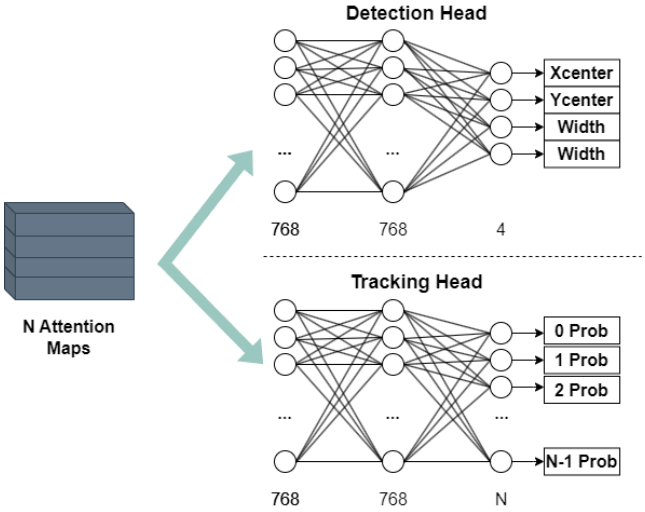

Figure 6. Detection and Tracking Heads. Each receives a N-sized batch of attention maps.

both detection and tracking simultaneously, would actually split this task into two different heads. While some decide against using the same input on both, as in Sun *et al*. [2021], we opted to use the same attention maps extracted from the encoder. As mentioned previously, the positions corresponding to our Tracking Tokens output positions are the ones to learn which bounding box coordinates correlates to which object in current frame, used as the last 100 positions of the input sequence.

3.6 Limitations by Design

This approach aims to portray the MOT problem as a set prediction task, just like DETR [Carion *et al*., 2020] did for object detection and Zeng *et al*. [2022] followed it for MOT. The idea is that instead of the output becoming a sequence where their relative position matter, the network's output represents the situation as a whole. But to use it this way, there needs to be a limit of tracked objects in a given context window. This means two things: firstly some of the outputs might just be throwaway predictions. Secondly we might lose some of the predicted objects if there just happens to be over the object limit. Which is why this value has to be tuned to specific use cases. In more dense scenarios, the model will fail a lot more if the amount of Tracking Tokens is reduced, while in more sparse scenes, we might end up having unnecessary overhead.

4 Experiments

The experiments were conducted to validate the practical capabilities of the solution, bringing common evaluation metrics for the problem and also emphasizing the importance of inference time per image.

4.1 Data Used

For the tests described below, the MOT17 dataset from Milan *et al*. [2016] was used. This dataset is a common benchmark for MOT, making it a good candidate for producing comparisons and evaluating the overall performance of the model.

This dataset has a predefined training and testing subset, where the training data consists of 21 annotated video sequences of people walking in various scenarios, varying cameras, viewing angles, static or mobile recording, number

of people, recording distance, indoor and outdoor environments, and lighting conditions. In total, there are 15948 images, containing 336891 bounding box annotations. For the test set, there are 17757 images with 564228 annotations.

Here we validate the definition of the maximum limits of individual objects in a time window. Below is a visualization of the distribution of objects across frames. This implies that for this specific test case, 100 objects are sufficient to cover almost all objects.

In Figure 7, we can see the boxplot of the distribution of unique IDs in every set of one consecutive frame, meaning $K = 1$. In the plot, we notice how most of the time, there aren't more than 60 objects, and the maximum value is barely over 100.

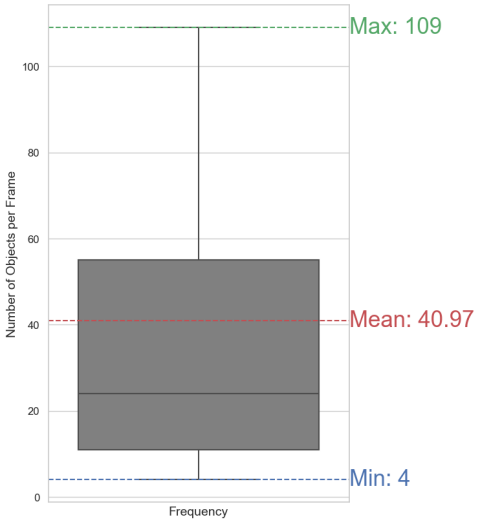

Figure 7. Objects per Window, with K=1 (single frame)

Now in Figure 8, with K=5 (meaning a window 5 times larger than the previous analysis), this boxplot barely changes. So any window size from 1 through 5 can use 100 max elements without losing many detections by default. This analysis meant that this limitation does not necessarily cause lower accuracy metrics for the model with 100 sequence size configuration. Although for other use cases, it could be interesting to make this limit higher, we show how it affects the inference time performance of the model in the next section, also how lowering this value, for more constricted scenarios, could enhance this same metric.

4.2 Model Training Configuration

Following common guidelines in related works Zeng *et al*. [2022]; Zhang *et al*. [2023], the AdamW optimizer [Loshchilov and Hutter, 2019] was employed. The results of the final model configuration, which will be used for comparison with other works, were trained for 150 epochs. Due to memory limitations, the batch size was kept at 16 sequences per sample.

The loss functions directly follow how the data is handled throughout the model. Being a model with tasks that are incompatible in terms of format (one produces a value corresponding to positions in the image, while the other tries to understand the different individual objects), it makes sense that the loss function is composite.

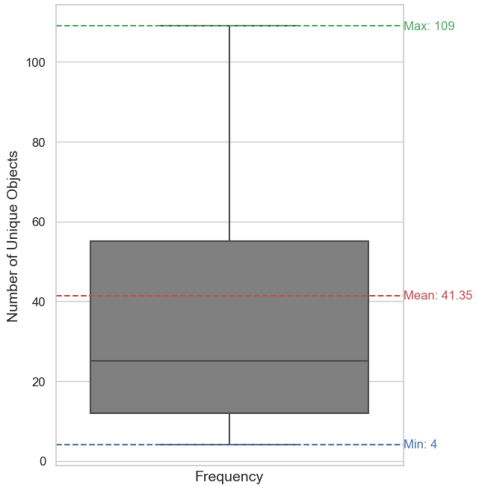

Figure 8. Objects per Window, with K=5

Thus, to adjust the weights based on the ability to find objects in the videos, GIoU was used, first conceptualized in Rezatofighi *et al*. [2019]. This loss function takes into consideration interesting details of bounding boxes. For the tracking head, the *Categorical Cross Entropy* function was used. It works because by reducing the scope to a time window, it becomes possible to perform a classification task to track objects.

4.2.1 Different Model Hyperparameter Configurations

From the conceptualization of the model in Section 3, there are two values that must be defined previous to the training of the network. In this section, we try to analyze what impact each change can have on the final computational cost of the model. The first one corresponds to the size limit of the objects found in a context window. An optimal value has been defined by a dataset-specific analysis, but in Table 3 we measured how the inference time changes based on this value N. By only varying this value and keeping total frames fixed at 3 and cutout size to 64, we can see that lowering the sequence size from 100 to 50 barely yields an increase in FPS, but increasing this value to 200 shows a big slowdown.

Table 2. Impact of Total Frames on FPS

Total Frames	FPS	Delta (%)
1	62.5	-
2	58.83	-5.87
3	**43.5**	-30.4
4	33	-47.2

Based on our tests presented in Tables 2, 3 and 4, what affects the FPS the most is the context window used for each inference, which is logical, as it makes so the model has to go through more images and objects before the inference. However, keeping too low of a context window can affect the model's results. The values of this test are contained in Table 2.

Finally, altering the size of the objects before they are averaged behaves similarly to the change in sequence size. In Table 4, the difference between 32 and 64-sized objects is very small, but between 64 and 128, we can see an 18.6% decrease in FPS. This means we can get a small speed-up

Table 3. Impact of Sequence Size on FPS

Sequence Size	FPS	Delta (%)
100	**43.5**	-
50	45.16	3.81
200	33.8	-22.3

by resizing to a lower resolution. However, doing so could cause a bigger loss in the ID assigning step, as specific features would be harder to learn.

Table 4. Impact of Cutout Size on FPS

Cutout Size	FPS	Delta (%)
64	**43.5**	-
32	45.45	4.48
128	35.41	-18.6

4.3 Comparative Results

In Table 5, results are presented for 4 main MOT metrics: MOTA (Multiple Object Tracking Accuracy), HOTA (High Order Tracking Accuracy), IDF1 (Identification F1-Score). All definitions of the previous three were implemented based on Luiten *et al*. [2020], and inference time in Frames per Second (FPS). In this case, all models were tested on a machine with the following configuration: Ryzen 5 3600 processor, 32GB of RAM, and a Nvidia RTX 3060.

Due to the time complexity cost of training such models, it becomes hard to evaluate these configurations exhaustively. The main objective of the research was to allow proper usage of an efficient transformer-based end-to-end MOT model, we decided to use the previous FPS results to select the final configuration, which was trained and evaluated. These results are displayed in Table 5. These results show how this approach was effective at being an efficient model, surpassing all other models, while having a loss of around 10% versus other models in accuracy metrics. Specifically, the models that are based on transformers, like MOTR, MOTRv2, TransTrack, and Trackformer, were surpassed by a big margin in terms of inference speed, by losing at most 14% in MOTA vs MOTRv2. This shows that this approach is valid to solve MOT efficiently, but still can see improvements more focused on extracting better information from the whole pipeline.

Considering these results, we can notice that the metric deficit between the best model in the MOTA and IDF1 accuracy is very similar. Since IDF1 indicates the performance on the specific task of data association, this shows that the biggest issue in the current model lies in its ability to re-identify the objects over time. In the next section, we explore in more detail some of these results.

4.4 Visualization

Here we show some examples of model predictions to discuss the results and what was traded off for the inference speed. In Figure 9, there are three frames, (a), (b) and (c). All of these are example images taken from MOT17. In frame (a) we can see the model detected two people. One has ID=1

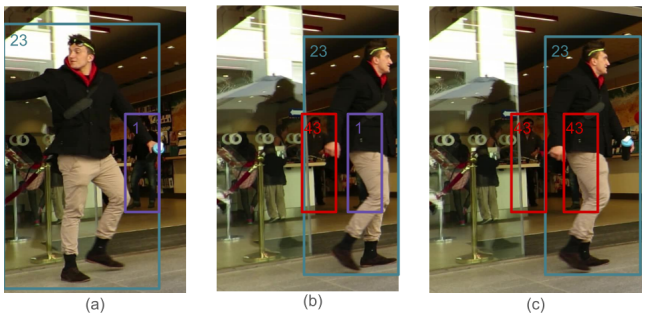

Figure 9. Some frames with predictions from the model.

and the other has ID=23. Some frames later, another person enters the scene. We can see this in frames (b) and (c). It is tagged as ID=23 in frame (b). We can also see that the model kept the tag of object 1, even through an almost complete occlusion. But in frame (c), we can see that it switched this ID for the ID of the new person, who was close to him.

This is a recurring mistake the model tends to make, which justifies most of the lower metrics it achieves, compared to similar models. This can be associated with the method used to assign IDs to each object, which is described in Section 3.5. Since the tracking head treats the result as a classification problem, and each object will go through this head, it's not impossible to guarantee that the model will not predict the same object ID twice in the same frame, which should be impossible.

5 Conclusion and Future Work

The results show that the introduced model offers an innovative approach, achieving efficient task performance with low inference time. However, improvements are needed in its metrics to compete with established state-of-the-art models. Given its lightweight nature, it emerges as a strong candidate for solutions requiring efficiency. The model's design limitation, such as the maximum of 100 objects per window, is adjustable, providing flexibility for various applications. Another limitation was the choice of a classification-like head for tracking the objects. While providing faster inference speed, it introduced the duplicated ID problem, which presents a big challenge in the current model architecture.

In terms of application scenarios, this model provides unmatched processing speeds, this being a good option, even though it presents lower metrics scores besides the FPS. From the qualitative analysis of the results, the issues with the re-identification of objects can present a challenge for the practical usage of this solution. Still, it can present a base work for this type of model, focusing now on inference time without sacrificing as much performance on the other metrics.

Future work includes exploring different data representations at the model's input, possibly employing alternative encoders to the Vision Transformer for enhanced results. Investigating new output data representations and refining the tracking stage to better handle ID assignments by considering object similarities are promising avenues. The model's adaptability also warrants testing against diverse datasets in

Table 5. Table with comparative results

Model	MOTA (%)	HOTA (%)	IDF1 (%)	FPS
StrongSORT	79.6	64.4	79.5	7.1
BoT-SORT	**80.6**	64.6	79.5	6.6
FairMOT	69.2	59.3	73.3	26.8
ByteTrack	76.0	63.1	79.0	29.6
SparseTrack	81.0	**65.1**	**80.1**	19.9
MOTR	73.4	-	68.6	6.9
MOTRv2	78.6	62.0	75.0	7.5
TransTrack	74.5	54.1	63.9	12
TrackFormer	62.3	-	57.6	7.4
CenterTrack	67.8	52.2	64.7	17.5
OneTrack	64.7	56.3	64.5	**43.5**

various domains. These tests can include higher object per frame window configuration (from the value of 100 used in our tests), which can attest to the scalability of this model.

The disparity in frame representation between convolutional feature extraction and linear projection poses a challenge. A dedicated loss function, tailored to simultaneously address both detection and tracking aspects, could further optimize the model's performance. Another challenge can be evaluating how this model in particular performs with critical occlusion scenarios, although this requires specific data which was unavailable at the time of this work.

We believe this paper brought another paradigm of implementing such models while using transformer-based models. One which could not produce fast responses, limiting its practical use in real-life cases and applications. Further improvements to this model can allow it to perform even better over time, with improved training data for specific domains, and further working on the actual model and its development.

6 Acronyms Table

Table 6. Acronyms Reference

Acronym	Meaning
MOT	Multiple Object Tracking
MOTA	Multiple Object Tracking Accuracy
HOTA	Higher Order Tracking Accuracy
IDF1	Identification F1-Score
FPS	Frames per Second
ID	Object identification number
IoU	Intersection Over Union
ViT	Vision Transformer
OAV	Object Averaged Vector
DETR	Detection Transformer

Declarations

Authors' Contributions

All authors contributed equally to this article. All authors read and approved the final manuscript.

Competing interests

The authors declare that they have no competing interests.

Availability of data and materials

The dataset used for training and evaluation is currently available at https://motchallenge.net/data/MOT17.

References

Aharon, N., Orfaig, R., and Bobrovsky, B.-Z. (2022). Bot-sort: Robust associations multi-pedestrian tracking.

Bewley, A., Ge, Z., Ott, L., Ramos, F., and Upcroft, B. (2016). Simple online and realtime tracking. In *2016 IEEE International Conference on Image Processing (ICIP)*. IEEE. DOI: 10.1109/icip.2016.7533003.

Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., and Zagoruyko, S. (2020). End-to-end object detection with transformers.

Deng, J., Dong, W., Socher, R., Li, L.-J., Li, K., and Fei-Fei, L. (2009). Imagenet: A large-scale hierarchical image database. In *2009 IEEE Conference on Computer Vision and Pattern Recognition*, pages 248–255. DOI: 10.1109/CVPR.2009.5206848.

Deng, J., Guo, J., Yang, J., Xue, N., Kotsia, I., and Zafeiriou, S. (2022). Arcface: Additive angular margin loss for deep face recognition. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 44(10):5962–5979. DOI: 10.1109/TPAMI.2021.3087709.

Dosovitskiy, A., Beyer, L., Kolesnikov, A., Weissenborn, D., Zhai, X., Unterthiner, T., Dehghani, M., Minderer, M., Heigold, G., Gelly, S., Uszkoreit, J., and Houlsby, N. (2021). An image is worth 16x16 words: Transformers for image recognition at scale.

Du, Y., Zhao, Z., Song, Y., Zhao, Y., Su, F., Gong, T., and Meng, H. (2023). Strongsort: Make deepsort great again.

Ge, Z., Liu, S., Wang, F., Li, Z., and Sun, J. (2021). Yolox: Exceeding yolo series in 2021.

Girshick, R., Donahue, J., Darrell, T., and Malik, J. (2014). Rich feature hierarchies for accurate object detection and semantic segmentation. Available at:https://openaccess.thecvf.com/content_cvpr_2014/html/Girshick_Rich_Feature_Hierarchies_2014_CVPR_paper.html.

Liu, Z., Ning, J., Cao, Y., Wei, Y., Zhang, Z., Lin, S., and Hu, H. (2021). Video swin transformer. Available at:https://openaccess.thecvf.com/content/CVPR2022/html/Liu_Video_Swin_Transformer_CVPR_2022_paper.html.

Loshchilov, I. and Hutter, F. (2019). Decoupled weight decay regularization.

Luiten, J., Osep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., and Leibe, B. (2020). Hota: A higher order metric for evaluating multi-object tracking. *International Journal of Computer Vision*, 129(2):548–578. DOI: 10.1007/s11263-020-01375-2.

Meinhardt, T., Kirillov, A., Leal-Taixe, L., and Feichtenhofer, C. (2022). Trackformer: Multi-object tracking with transformers.

Milan, A., Leal-Taixé, L., Reid, I., Roth, S., and Schindler, K. (2016). MOT16: A benchmark for multi-object tracking. *arXiv:1603.00831 [cs]*. arXiv: 1603.00831. DOI: 10.48550/arXiv.1603.00831.

Mostafa, R., Baraka, H., and Bayoumi, A. (2022). Lmot: Efficient light-weight detection and tracking in crowds. *IEEE Access*, 10:83085–83095. DOI: 10.1109/ACCESS.2022.3197157.

Pitie, F., Berrani, S.-A., Kokaram, A., and Dahyot, R. (2005). Off-line multiple object tracking using candidate selection and the viterbi algorithm. In *IEEE International Conference on Image Processing 2005*, volume 3, pages III–109. DOI: 10.1109/ICIP.2005.1530340.

Redmon, J., Divvala, S., Girshick, R., and Farhadi, A. (2016). You only look once: Unified, real-time object detection. Available at:https://www.cv-foundation.org/openaccess/content_cvpr_2016/html/Redmon_You_Only_Look_CVPR_2016_paper.html.

Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., and Savarese, S. (2019). Generalized intersection over union: A metric and a loss for bounding box regression. Available at:https://openaccess.thecvf.com/content_CVPR_2019/html/Rezatofighi_Generalized_Intersection_Over_Union_A_Metric_and_a_Loss_for_CVPR_2019_paper.html.

Schroff, F., Kalenichenko, D., and Philbin, J. (2015). Facenet: A unified embedding for face recognition and clustering. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. IEEE. DOI: 10.1109/cvpr.2015.7298682.

Sun, P., Cao, J., Jiang, Y., Zhang, R., Xie, E., Yuan, Z., Wang, C., and Luo, P. (2021). Transtrack: Multiple object tracking with transformer.

Wang, C.-Y., Bochkovskiy, A., and Liao, H.-Y. M. (2022). Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. Available at:https://openaccess.thecvf.com/content/CVPR2023/html/Wang_YOLOv7_Trainable_Bag-of-Freebies_Sets_New_State-of-the-Art_for_Real-Time_Object_Detectors_CVPR_2023_paper.html.

Wang, Y.-H., Hsieh, J.-W., Chen, P.-Y., Chang, M.-C., So, H. H., and Li, X. (2023). Smiletrack: Similarity learning for occlusion-aware multiple object tracking.

Wojke, N., Bewley, A., and Paulus, D. (2017). Simple online and realtime tracking with a deep association metric.

Zeng, F., Dong, B., Zhang, Y., Wang, T., Zhang, X., and Wei, Y. (2022). Motr: End-to-end multiple-object tracking with transformer.

Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., and Wang, X. (2022). Bytetrack: Multi-object tracking by associating every detection box.

Zhang, Y., Wang, C., Wang, X., Zeng, W., and Liu, W. (2021). Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129(11):3069–3087. DOI: 10.1007/s11263-021-01513-4.

Zhang, Y., Wang, T., and Zhang, X. (2023). Motrv2: Bootstrapping end-to-end multi-object tracking by pretrained object detectors. Available at:https://openaccess.thecvf.com/content/CVPR2023/html/Zhang_MOTRv2_Bootstrapping_End-to-End_Multi-Object_Tracking_by_Pretrained_Object_Detectors_CVPR_2023_paper.html.

Zhou, X., Koltun, V., and Krähenbühl, P. (2020). Tracking objects as points. *ECCV*. DOI: 10.1007/978-3-030-58548-8_2.