Multiple Object Tracking as ID Prediction

Supplementary Material

A. Overview

In the supplementary material, we primarily:

1. State more experimental details, in Appendix B.
2. Discuss concerns regarding the introduction of static images for joint training, in Appendix C.
3. Provide additional experimental and visualization results, in Appendix D.

B. Experimental Details

Due to space constraints in the main text, we could not provide a comprehensive account of all experimental details. In this section, we will describe the specific details related to the training (Appendix B.1), inference (Appendix B.2), and ablation experiments (Appendix B.3).

B.1. Training

Settings. In each training iteration, we need to sample $T + 1$ frames, as mentioned in Sec. 4.2. Similar to previous works [8, 31, 35] that employ multi-frame training, we adopt random sampling intervals to enhance the diversity of training data. However, continuously increasing the sampling interval may make training samples excessively challenging. This could cause a discrepancy between training data and the inference video sequences, ultimately adversely affecting the model's performance. In our experiments, we set the random sampling interval to range from 1 to 4 by default.

For the final training strategies, we have chosen the following approaches: On DanceTrack [28], we train MOTIP for 10 epochs on the train set and drop the learning rate by a factor of 10 at the 5-th and 9-th epoch. On SportsMOT [6], we train our model for 13 epochs on the train set and drop the learning rate by a factor of 10 at the 8-th and 12-th epoch. On BFT [40], we train the model for 22 epochs while drop the learning rate at the 16-th and 20-th epoch. To expedite the convergence, we use COCO [14] pre-trained weights and perform detection pre-training on the corresponding datasets. This serves as the initialization for the DETR part of MOTIP. Our typical hardware setup involves 8 NVIDIA RTX 4090 GPUs, with the batch size of each GPU set to 1.

Parallelization. Recent tracking-by-query methods [8, 31, 35], which also use DETRs as their frameworks, process multi-frame video sequences in a manner similar to RNNs during training. For instance, when processing a five-frame video clip, the model needs to perform five sequential forward passes of the DETR component, one frame at a time. Since the DETR architecture accounts for the majority of computational cost, this processing approach fails to leverage the parallel processing capabilities of the GPU. In contrast, our MOTIP decouples detection and association components, allowing it to detect targets in all frames at once during training, as illustrated in Fig. 1. Meanwhile, since our ID Decoder is a transformer decoder structure, it can also achieve parallelism by leveraging the attention masks [29]. Therefore, our method can attain high parallelism on the GPU during training, improving GPU utilization and enabling efficient training.

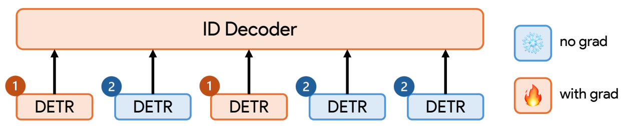


Figure 1. Illustration of the parallelized training of MOTIP, using a five-frame demo. Since the detection process for each frame is independent, all DETRs in a sequence can perform forward simultaneously, which is GPU-friendly. In our implementation, we divide all DETRs into two forward passes (as shown in numbers 1 and 2) since we only backpropagate gradients for a subset of them, as described in Sec. 4.2.

```
# N                 - detected objects in the current frame.
# K                 - the capacity of the ID dictionary.
# id_scores         - [N, K+1],
#                     confidence of N objects in the current frame,
#                     on K+1 different ID labels,
#                     the first K ones represent normal IDs,
#                     the last one represents `newborn`.
# tracked_ids       - [M, ],
#                     previously tracked ID labels.
# score_thresh      - minimum acceptable ID confidence score.


# Step 1: get the candidate ID label and score of each target
object_max_scores, object_id_labels = torch.max(id_scores, dim=1)

# Step 2: get the max score of each candidate ID label
id_max_scores = dict()
for score, id in zip(object_max_scores, object_id_labels):
    if id != K + 1:        # except the `newborn` token
       if id not in id_max_scores:
           id_max_scores[id] = score
       else:
           id_max_scores[id] = max(id_max_scores[id], score)

# Step 3: output the final ID assignment results (without duplication)
final_ids = list()         # will be a N-length list at the end of the code
for i in range(len(object_max_scores)):
    score, id = object_max_scores[i], object_id_labels[i]
    if id not in tracked_ids:
        final_ids.append(K + 1)   # not in the tracked IDs, is newborn
    elif score < score_thresh:
        final_ids.append(K + 1)   # not exceed the threshold, is newborn
    elif score < id_max_scores[id]:
        final_ids.append(K + 1)   # not the highest-score one, is newborn
    else:
        final_ids.append(id)      # normal case, accept this ID

# final_ids is the final results of the ID assignment process
```

Figure 2. Python-like pseudocode for the core of our ID assignment process.

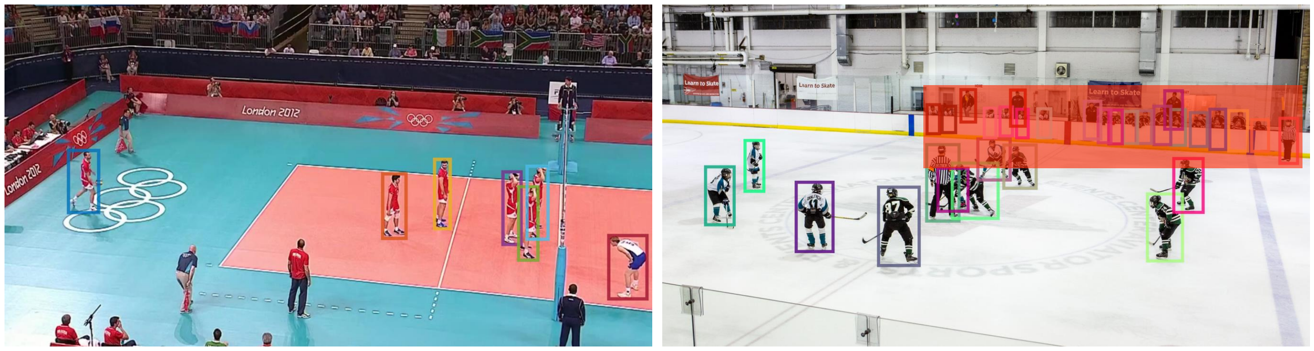

(b) SportsMOT (a) CrowdHuman

Figure 3. **Visualizing the different annotation standards between SportsMOT [6] and CrowdHuman [27].** (a) In SportsMOT, only athletes are annotated, excluding referees and spectators, or any other people. (b) Since CrowdHuman aims to detect all humans, it additionally includes annotations for crowds outside the sports venues, as shown by the area covered in the red mask.

B.2. Inference

As discussed in Sec. 3.4, during inference, we utilize a straightforward ID assignment strategy. We apply an approach similar to classification tasks, selecting the prediction with the highest confidence score for each object as the final accepted result. This simplifies our inference process, eliminating the need for more intricate allocation strategies. Although this approach seems feasible, most tracking evaluation approaches [1, 17, 25] cannot handle duplicate IDs within the same frame. Therefore, we need to introduce an additional rule to handle this situation: when duplicate IDs appear in the final results, we select the one with the highest confidence and label the others as newborn objects. This simple patch completely avoids the occurrence of duplicate ID labels within the same frame. The pseudocode for the aforementioned assignment process is provided in Fig. 2.

B.3. Ablation Studies

Default Settings. As stated in Sec. 4.4, we train the models on the train set of DanceTrack [28] and subsequently evaluate on its official validation set to conduct our ablation experiments. Unless otherwise stated, all experiments are implemented without using trajectory augmentation techniques to ensure a fair comparison, *i.e.*, $\lambda_{\text{occ}} = \lambda_{\text{sw}} = 0.0$. To reduce the computational cost, we shortened the sampling length of the video sequence to $T = 19$. As for the inference, we leverage $\lambda_{det} = \lambda_{new} = 0.5$, while $\lambda_{id} = 0.1$ for simplicity. Other unspecified details are consistent with the default setups, as referenced in Sec. 4.2, Appendix B.1 and Appendix B.2.

***Re-ID* Pipeline.** As shown in Tab. 5, we construct a *re-id* pipeline to compare with our in-context ID prediction approach. During the training process, we refer to FairMOT [36], a well-known joint detection and embedding method. Interestingly, it also utilized the cross-entropy function for supervision. This similarity undoubtedly provides a suitable competitor for our method. In practice, we treat each trajectory as a class and assign it a unique ID label that remained consistent throughout the entire training process. After obtaining the output embeddings from the model, we employ a linear projection as the classification head and use the corresponding ID labels to supervise the classification results of the targets. Although both approaches use classification supervision, our procedures for defining ID labels differ significantly, as discussed in Sec. 3.1. During inference, the *re-id* pipeline applies cosine similarity to calculate the cost matrix for ID assignment, which is a widely adopted strategy in ReID-based methods [18, 33, 36].

***Contra* Pipeline.** Inspired by contrastive learning methods like CLIP [24], recent work [22] has employed contrastive learning to supervise the object embeddings of different trajectories, aiming to learn distinguishable features. Therefore, we employ the infoNCE loss [22, 24] to supervise the model as a comparative method, which is denoted as *contra* pipeline in Tab. 5. In experiments, we also tune some hyperparameters to improve the performance (about 3.0 HOTA) for a thorough comparison. During inference, we also use the cosine similarity matrix. In the above two comparative pipelines, we observe that the tracking performance is similar when the similarity threshold is below 0.5. Therefore, we retain the similarity threshold at 0.1 in our ablation experiments for simplicity.

One-Stage and Two-Stage Training In Tab. 5, we establish two different training strategies: *one-stage* and *two-stage*. The former uses a combined loss function to supervise the entire network, like Eq. (3). In contrast, two-stage training divides the model into two sequentially trained parts. First, the DETR component is trained using detection supervision. Then, the trained weights are frozen, and the ob-

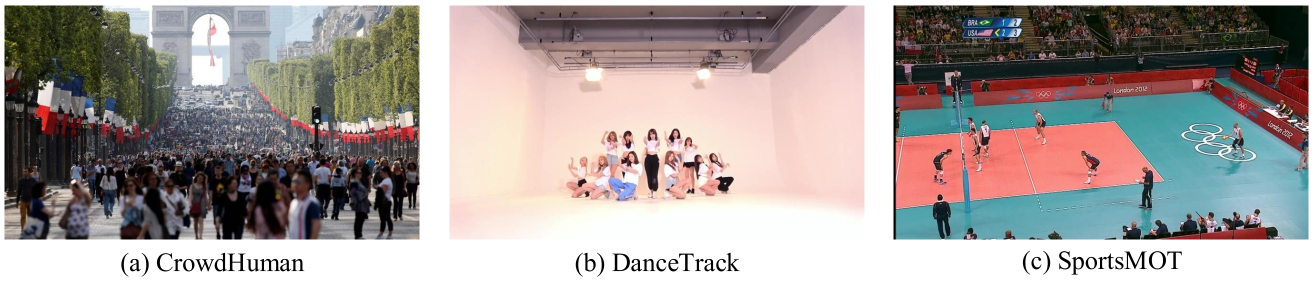
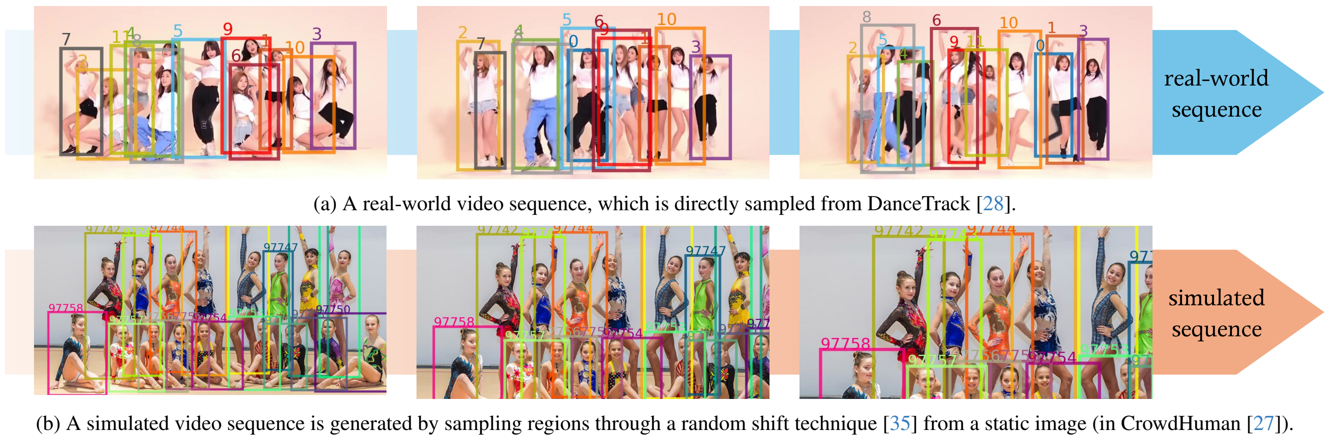


Figure 4. **Illustrate the inconsistent scenario characteristics from different datasets.** (a) humans in high-density scenarios from CrowdHuman [27]. (b) DanceTrack [28] aims to track indoor dancers. (c) SportsMOT [6] is chiefly concerned with the tracking of sports events.



real-world sequence
(a) A real-world video sequence, which is directly sampled from DanceTrack [28].
simulated sequence
(b) A simulated video sequence is generated by sampling regions through a random shift technique [35] from a static image (in CrowdHuman [27]).

Figure 5. Illustrating two distinct approaches of video sequence acquisition: *real-world vs. simulated* sequences. The latter is tantamount to transform the objects by mere translational and scaling transformations, which intuitively seems overly simplistic for a tracking model.

ject association part is trained separately. This two-stage training ensures the consistency of the object embeddings produced by DETR, thereby providing a fair testbed for the three different pipelines.

C. Rethinking Joint Training with Images

Some recent work [6, 31, 34, 38] has opted to use additional image detection datasets, such as CrowdHuman [27], for joint training. For methods [31, 34, 38] requiring multi-frame training, they commonly use random shifting to simulate short video clips from single images to meet the training requirements. While this kind of joint training can indeed improve tracking performance, we argue it hinders the sustained advancement of MOT methods, particularly for end-to-end models that focus on temporal information. The main impacts are concentrated in three areas: inconsistent scene characteristics (Appendix C.1), inconsistent annotation standards (Appendix C.2), and overly simplistic video simulations (Appendix C.3).

C.1. Inconsistent Scenario Characteristics

As discussed in Sec. 3.1 of CrowdHuman [27], this dataset aims to be diverse for real-world scenarios. To this end, various different keywords were used to collect data from Google Image search. In contrast, existing MOT benchmarks [6, 20, 28] predominantly focus on specific scenarios. For instance, SportsMOT [6] primarily collects high-quality videos from professional sports events, while DanceTrack [28] crawls network videos, including mostly group dancing. Consequently, in CrowdHuman, some scenes may never appear in specific MOT datasets. As illustrated in Fig. 4, some crowded scenes of CrowdHuman are virtually absent in both DanceTrack and SportsMOT. Additionally, the CrowdHuman dataset also encompasses some scenes under atypical low-light conditions and wide-angle lens perspectives, which significantly deviate from the distribution of the target datasets, like DanceTrack and SportsMOT.

Although the inconsistencies across these scenarios did not adversely impact the performance on target benchmarks yet, there are still some concerns based on common consensus in deep learning training. Using out-of-domain data is inherently a double-edged sword. While it can boost performance, it may also cause the model astray from its intended application scenarios. This typically calls for a careful adjustment of the training data ratio during training to maintain this delicate balance. We argue this would lead researchers to spend an excessive amount of unnecessary effort on tuning the hyperparameters required for training. While additional detection datasets are still necessary for some traditional, smaller datasets (like MOT15 [10], MOT17 [20]), many recently proposed benchmarks already contain sufficient training data to fully unlock the potential of most models. Introducing extra image data, in this case, would be redundant.

C.2. Inconsistent Annotation Standards

Across different datasets, even if the category labels seem the same, the focal points may vary. A notable example is the distinction between the CrowdHuman [27] and SportsMOT [6] datasets. CrowdHuman aims to detect every visible person in the images, whereas SportsMOT focuses solely on the athletes in the videos. This results in differences in the annotation protocols between these two datasets. As illustrated in Fig. 3, compared to SportsMOT, CrowdHuman includes additional annotations for spectators and referees in sports scenes. Using these two datasets for joint training can confuse the model because of the different annotation standards. Specifically, it becomes unclear if people who are not athletes should be considered as positive detections, thereby impairing the final performance. Some more sophisticated engineering designs have been used to address this issue. For example, MixSort [6] employs different combinations of training data at multiple stages and finally performs fine-tuning exclusively on SportsMOT. However, the optimal joint training strategy can vary for different models while dealing with this issue. We argue that customizing multi-stage joint training strategies would divert researchers' efforts toward engineering tricks rather than general tracking solutions. Therefore, we believe that when the amount of training data is sufficient to validate the effectiveness of the method, there is no need to introduce additional training data. This helps avoid the complexity of adjusting training strategies.

C.3. Simplistic Video Simulations

Recent research [8, 26, 35] has demonstrated that multi-frame training is highly beneficial for developing a more robust tracking model. When incorporating detection datasets like CrowdHuman [27] for joint training, random shifting is employed to sample different regions of the same image to generate video clips. For each target, this is equivalent to continuously applying a translation and scaling operation at a constant ratio, as shown in Fig. 5b. The resulting sequence will have very small differences between frames, lacking features such as deformation, occlusion, and changes in relative position that are present in real-world sequences (as shown in Fig. 5a). This phenomenon becomes more pronounced as the sampling length increases. Because the usable area of the static image remains constant, the shifting scale of each step must be reduced to ensure validity, making adjacent frames even more similar. Recent work [7, 8, 26, 35] has increasingly focused on the application of temporal information in tracking and has benefited from long-term sequence training. However, the overly simplistic method of video simulation can contaminate the distribution of training data, thereby severely impairing model performance. This might impede researchers from delving deeper into the exploration of temporal information. While more complex and diverse video simulation approaches can help alleviate this issue, this goes beyond the scope of general MOT methods. For more details, you can refer to some related studies [11–13].

C.4. Discussions

Our MOTIP is an end-to-end learnable approach, which encounters additional challenges during joint training, as discussed in Appendix C.1 and Appendix C.2. Additionally, we utilize long-sequence training to handle temporal information, which means that video data generated through image simulation does not sufficiently benefit our model, as discussed in Appendix C.3. For example, during joint training of SportsMOT [6] and CrowdHuman [27], single-stage training can cause issues for the detector due to annotation inconsistencies. On the other hand, multi-stage training struggles to address the problem of model forgetting. This challenge is not unique to our model. As the computer vision community evolves, more end-to-end and long-term modeling approaches will become available for multi-object tracking. Complex joint training might shift the focus towards engineering implementations rather than the research of more generalized tracking methods. Therefore, we suggest that when the dataset is sufficiently large, it may be preferable to avoid introducing extra data for training. This approach allows for a more focused effort on addressing the various challenges in multiple object tracking.

D. More Results

D.1. MOT17

Although MOT17 [20] is widely recognized as an important pedestrian tracking dataset, its limited amount of training data has been noted by many works [8, 26] to be inadequate for training modern models, especially end-to-end approaches. Since it only contains 7 video sequences for training, current methods [35, 37] always incorporate extra detection datasets [10, 27] for joint training to ensure data

Methods	HOTA	DetA	AssA	MOTA	IDF1
heuristic:					
CenterTrack [41]	52.2	53.8	51.0	67.8	64.7
QDTrack [21]	53.9	55.6	52.7	68.7	66.3
GTR [42]	59.1	61.6	57.0	75.3	71.5
FairMOT [36]	59.3	60.9	58.0	73.7	72.3
DeepSORT [30]	61.2	63.1	59.7	78.0	74.5
SORT [2]	63.0	64.2	62.2	80.1	78.2
ByteTrack [37]	63.1	64.5	62.0	80.3	77.3
OC-SORT [4]	63.2	63.2	63.4	78.0	77.5
C-BIoU [32]	64.1	64.8	63.7	81.1	79.7
MotionTrack [23]	65.1	65.4	65.1	81.1	80.1
end-to-end:					
TrackFormer [19]	/	/	/	74.1	68.0
MeMOT [3]	56.9	/	55.2	72.5	69.0
MOTR [35]	57.2	58.9	55.8	71.9	68.4
MOTRv2† [38]	57.6	58.1	57.5	70.1	70.3
MeMOTR [8]	58.8	59.6	58.4	72.8	71.5
MOTIP (ours)	**59.3**	**62.0**	**57.0**	**75.3**	**71.3**
CO-MOT [31]	60.1	59.5	60.6	72.6	72.7
MOTRv3 [34]	60.2	62.0	56.9	75.5	71.2

Table 1. Performance comparison with state-of-the-art methods on MOT17 [20]. The best performance among the end-to-end methods is marked in **bold**. The results shown in gray font indicate unfair comparisons due to network structure, as we detailed in Appendix D.1. MOTRv2† refers to the results of MOTRv2 [38] after removing additional heuristic post-processing algorithms, as derived from [34].

diversity. Nonetheless, some studies [8] have shown that the lack of diversity makes models prone to overfitting on training data, resulting in insufficient generalization capabilities. Under the same settings, end-to-end methods face more severe problems compared to heuristic algorithms, because additional datasets are insufficient for models to learn optimal tracking strategies, as we discussed in Appendix C.3. We argue these compromises and issues might divert research from fundamental tracking solutions, causing an overemphasis on engineering details. For this reason, we chose some more modern and diverse datasets in our main text, such as DanceTrack [28], SportsMOT [6], and BFT [40], to ensure the model is well-trained.

Nevertheless, we still present the state-of-the-art comparison on MOT17 [20] in Tab. 1. To handle crowded scenes, we modify several hyperparameters, such as setting the capacity of the ID dictionary (K) to 200. Compared to MOTR [35], which also uses the standard Deformable DETR framework, our MOTIP shows a significant performance improvement (59.3 HOTA *vs.* 57.2 HOTA). It should be noted that some of the methods (in gray font) in Tab. 1 are not a fair comparison with ours: MeMOTR [8] uses DAB-Deformable DETR [15] as the framework, while CO-MOT [31] customizes the reference points in Deformable DETR. MOTRv2† employs an additional YOLOX detector [9] as the proposal generator. MOTRv3 utilizes a more powerful backbone, ConvNeXT-Base [16]. Nevertheless, compared to these latest work, our method still demonstrates competitive performance. However, there is still a significant gap between our method and the state-of-the-art heuristic algorithms. On the one hand, heuristic algorithms have been continuously customized and developed over the past decade for these linear motion scenarios. In contrast, end-to-end approaches lack this human-crafted prior knowledge and still require time to mature. On the other hand, as previously discussed, overly homogeneous training data is detrimental to learnable methods. Therefore, we look forward to diverse and large-scale pedestrian tracking datasets to better explore and evaluate end-to-end general tracking methods.

Methods	FP32	FP16
MOTR [35]	13.1 FPS	/
MOTIP (ours)	12.7 FPS	22.8 FPS

Table 2. Comparison of inference speed. The experiments are conducted on a single NVIDIA RTX A5000 GPU. Using FP16 precision, MOTIP can achieve near real-time performance.

D.2. Inference Speed

Based on the analysis of our network structure, although we introduce an ID Decoder structure, its computational cost during inference is negligible compared to that required by Deformable DETR [43]. In Tab. 2, we compare our method with another query-based approach that also uses Deformabe DETR. The results show that MOTIP and MOTR [35] have similar inference speeds, supporting our perspective. Additionally, we are surprised to find that at FP16 precision, MOTIP can achieve nearly real-time inference speed, indicating its feasibility for practical applications. To address real-time considerations in the future, Deformable DETR could be replaced with some recent real-time DETR frameworks [5, 39].

D.3. Visualization of ID Decoder

As discussed in Sec. 4.4, we believe MOTIP is more flexible and intelligent in handling historical trajectory information compared to heuristic-based Re-ID methods. In this section, we use some visualizations to elucidate this explanation. In Fig. 6, we present a case where target 5 is not visible from frames 638 to 641 and reappears in frame 642. When a target disappears, it is always accompanied by severe occlusion issues. For example, as shown in Fig. 6, although the detector can correctly identify target 5 in frame 636, she is

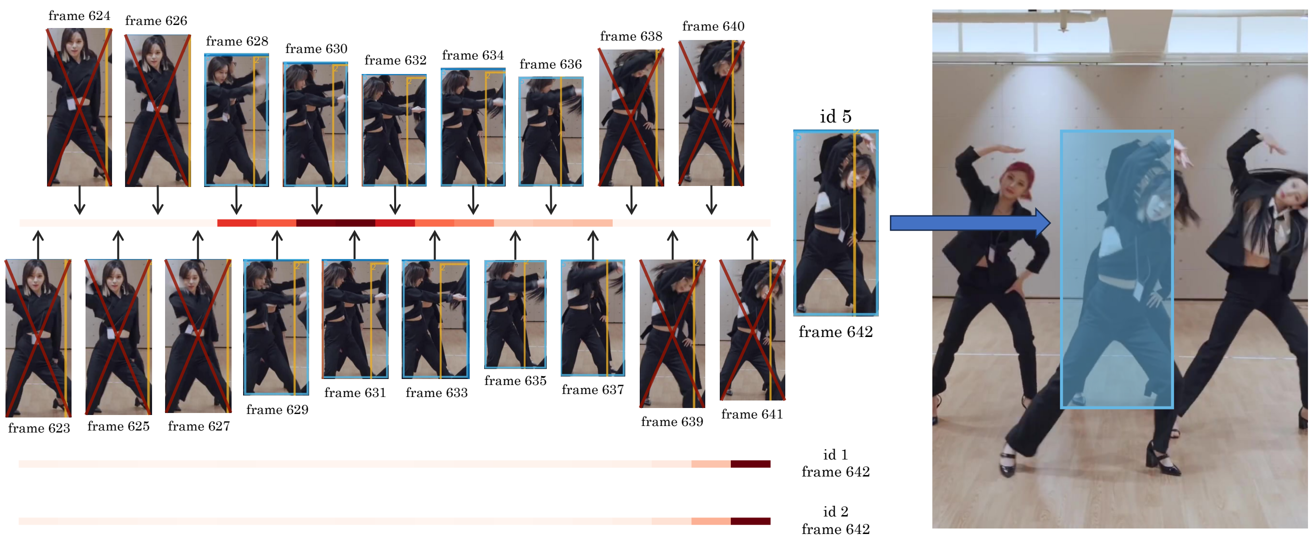


Figure 6. **Visualization of the cross-attention scores in the ID Decoder.** We show the response intensity between a target and its corresponding historical tracklets, with darker shades indicating stronger responses. Object 5 is occluded from frame 638 to 641 and reappears in frame 642. The other two objects, 1 and 2, remain visible during these 20 frames. The targets marked with a red cross indicate that they are not visible in the current frame. For a more comprehensive example, please refer to Fig. 7.

almost completely occluded by the dancer standing in front. This severe occlusion can render the target features unreliable. In traditional heuristic algorithms, this issue cannot be dynamically identified and addressed because the matching rules are manually fixed. However, our MOTIP can make dynamically optimal choices in such situations based on its cross-attention structure. As shown in Fig. 6, object 5 selects the targets in frames 630 and 631 as more reliable features, avoiding the pitfalls of unreliable ones (like in frame 640 and 641). In contrast, targets 1 and 2, which are not occluded throughout, will select the features closest to the current frame as they are the most similar. For a more detailed illustration, please refer to Fig. 7. We believe that compared to manually crafted rules based on experience, this flexible dynamic decision-making learned directly from data can help the model make accurate choices in challenging scenarios.

D.4. Number of ID Decoder Layers

As a key component, we investigate the impact of different numbers of layers in the ID Decoder on the final tracking performance in Tab. 3. Overall, as the number of layers increases, the final tracking performance improves gradually (from 54.3 to 60.5 HOTA). We believe this is because more decoding layers allow for elaborate modeling and further refinements of the ID allocations, enabling the model to handle more complicated situations. However, empirical evidence suggests an excessive number of network layers may lead to difficulties in model convergence, thereby increasing the training burden. At the same time, the improvements brought by increasing decoding layers exhibit diminishing marginal returns. Based on the above considerations, as mentioned in Sec. 4.2, we select a 6-layer structure as our default configuration.

Dec Layers	HOTA	DetA	AssA	MOTA	IDF1
1	54.3	75.5	39.3	84.6	51.9
3	57.6	**75.8**	44.0	**85.8**	58.3
6	59.5	75.3	47.2	85.6	61.1
9	**60.5**	75.3	**48.9**	85.7	**62.4**
12	60.3	75.0	48.7	85.1	61.7

Table 3. Ablation experiments on the number of layers in the proposed ID Decoder. The gray background is the choice for our final experiment.

D.5. Previous Results

This paper has an earlier version at arXiv:2403.16848v1. Although the model structure remains unchanged, we updated the codebase and some hyperparameters, resulting in improved tracking performance in Sec. 4.3 compared to the earlier version. We suggest that, for subsequent studies, comparing either of these two results based on their respective code frameworks is reasonable and acceptable.

References

[1] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: The CLEAR MOT metrics.

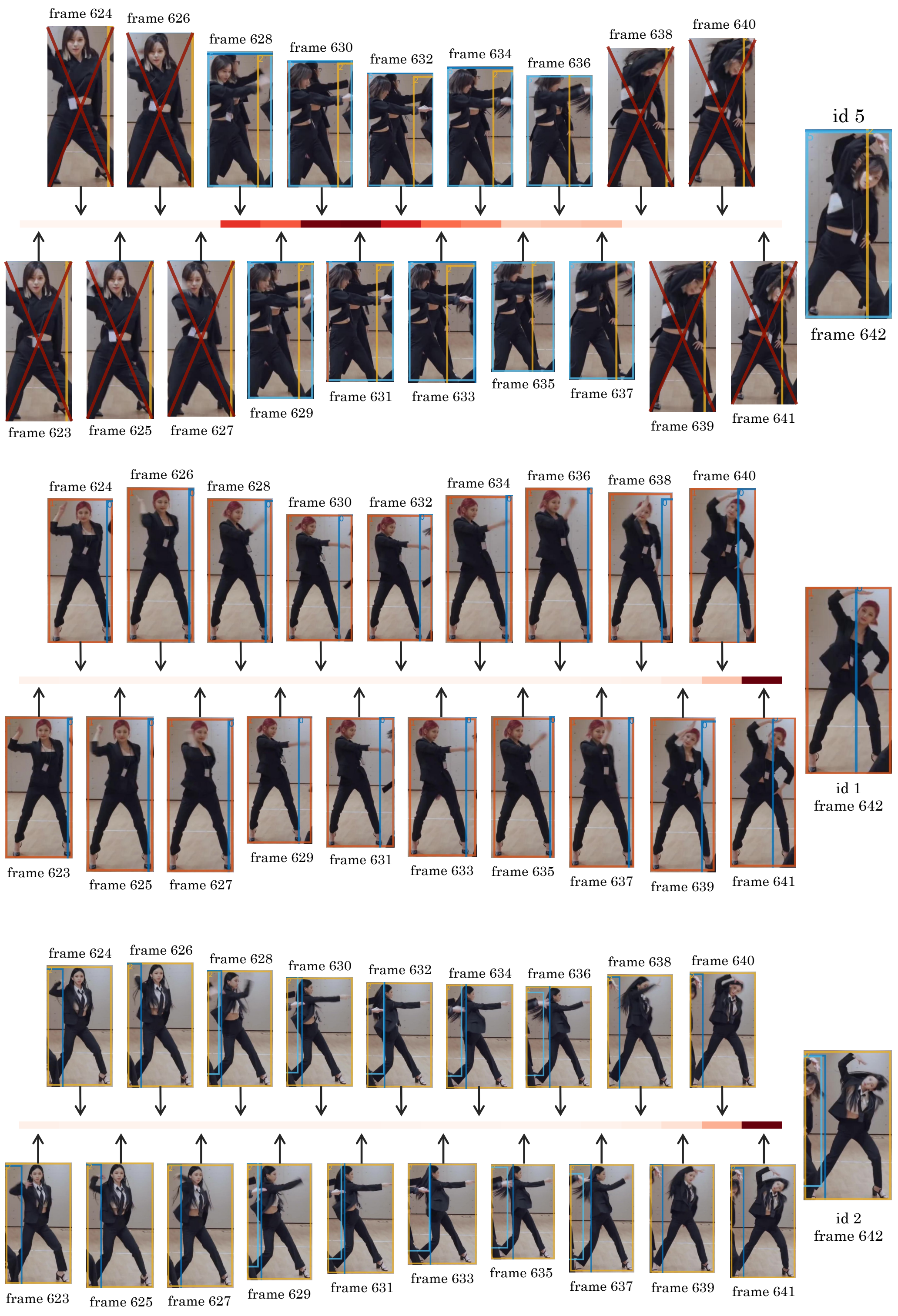


Figure 7. **A more comprehensive illustration of the example in Fig. 6.** Unlike object 5, targets 1 and 2 are clearly visible in all frames, and therefore, naturally select the closest results as reliable features.

EURASIP J. Image Video Process., 2008, 2008. 2

[2] Alex Bewley, ZongYuan Ge, Lionel Ott, Fabio Tozeto Ramos, and Ben Upcroft. Simple online and realtime tracking. In *ICIP*, pages 3464–3468. IEEE, 2016. 5

[3] Jiarui Cai, Mingze Xu, Wei Li, Yuanjun Xiong, Wei Xia, Zhuowen Tu, and Stefano Soatto. Memot: Multi-object tracking with memory. In *CVPR*, pages 8080–8090. IEEE, 2022. 5

[4] Jinkun Cao, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang, and Kris Kitani. Observation-centric SORT: rethinking SORT for robust multi-object tracking. *CoRR*, abs/2203.14360, 2022. 5

[5] Qiang Chen, Xiangbo Su, Xinyu Zhang, Jian Wang, Jiahui Chen, Yunpeng Shen, Chuchu Han, Ziliang Chen, Weixiang Xu, Fanrong Li, Shan Zhang, Kun Yao, Errui Ding, Gang Zhang, and Jingdong Wang. LW-DETR: A transformer replacement to YOLO for real-time detection. *CoRR*, abs/2406.03459, 2024. 5

[6] Yutao Cui, Chenkai Zeng, Xiaoyu Zhao, Yichun Yang, Gangshan Wu, and Limin Wang. SportsMOT: A large multi-object tracking dataset in multiple sports scenes. In *ICCV*, 2023. 1, 2, 3, 4, 5

[7] Patrick Dendorfer, Vladimir Yugay, Aljosa Osep, and Laura Leal-Taixé. Quo vadis: Is trajectory forecasting the key towards long-term multi-object tracking? In *NeurIPS*, 2022. 4

[8] Ruopeng Gao and Limin Wang. MeMOTR: Long-term memory-augmented transformer for multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 9901–9910, 2023. 1, 4, 5

[9] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. YOLOX: exceeding YOLO series in 2021. *CoRR*, abs/2107.08430, 2021. 5

[10] Laura Leal-Taixé, Anton Milan, Ian D. Reid, Stefan Roth, and Konrad Schindler. Motchallenge 2015: Towards a benchmark for multi-target tracking. *CoRR*, abs/1504.01942, 2015. 4

[11] Pengxiang Li, Zhili Liu, Kai Chen, Lanqing Hong, Yunzhi Zhuge, Dit-Yan Yeung, Huchuan Lu, and Xu Jia. Trackdiffusion: Multi-object tracking data generation via diffusion models. *CoRR*, abs/2312.00651, 2023. 4

[12] Siyuan Li, Tobias Fischer, Lei Ke, Henghui Ding, Martin Danelljan, and Fisher Yu. Ovtrack: Open-vocabulary multiple object tracking. In *CVPR*, pages 5567–5577. IEEE, 2023.

[13] Siyuan Li, Lei Ke, Martin Danelljan, Luigi Piccinelli, Mattia Segù, Luc Van Gool, and Fisher Yu. Matching anything by segmenting anything. In *CVPR*, pages 18963–18973. IEEE, 2024. 4

[14] Tsung-Yi Lin, Michael Maire, Serge J. Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C. Lawrence Zitnick. Microsoft COCO: common objects in context. In *ECCV (5)*, pages 740–755. Springer, 2014. 1

[15] Shilong Liu, Feng Li, Hao Zhang, Xiao Yang, Xianbiao Qi, Hang Su, Jun Zhu, and Lei Zhang. DAB-DETR: dynamic anchor boxes are better queries for DETR. In *ICLR*. OpenReview.net, 2022. 5

[16] Zhuang Liu, Hanzi Mao, Chao-Yuan Wu, Christoph Feichtenhofer, Trevor Darrell, and Saining Xie. A convnet for the 2020s. In *CVPR*, pages 11966–11976. IEEE, 2022. 5

[17] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip H. S. Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. HOTA: A higher order metric for evaluating multi-object tracking. *Int. J. Comput. Vis.*, 129(2):548–578, 2021. 2

[18] Gerard Maggiolino, Adnan Ahmad, Jinkun Cao, and Kris Kitani. Deep oc-sort: Multi-pedestrian tracking by adaptive re-identification. *arXiv preprint arXiv:2302.11813*, 2023. 2

[19] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixé, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *CVPR*, pages 8834–8844. IEEE, 2022. 5

[20] Anton Milan, Laura Leal-Taixé, Ian D. Reid, Stefan Roth, and Konrad Schindler. MOT16: A benchmark for multi-object tracking. *CoRR*, abs/1603.00831, 2016. 3, 4, 5

[21] Jiangmiao Pang, Linlu Qiu, Xia Li, Haofeng Chen, Qi Li, Trevor Darrell, and Fisher Yu. Quasi-dense similarity learning for multiple object tracking. In *CVPR*, pages 164–173. Computer Vision Foundation / IEEE, 2021. 5

[22] Pierre-François De Plaen, Nicola Marinello, Marc Proesmans, Tinne Tuytelaars, and Luc Van Gool. Contrastive learning for multi-object tracking with transformers. In *WACV*, pages 6853–6863. IEEE, 2024. 2

[23] Zheng Qin, Sanping Zhou, Le Wang, Jinghai Duan, Gang Hua, and Wei Tang. Motiontrack: Learning robust short-term and long-term motions for multi-object tracking. In *CVPR*, pages 17939–17948. IEEE, 2023. 5

[24] Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. Learning transferable visual models from natural language supervision. In *ICML*, pages 8748–8763. PMLR, 2021. 2

[25] Ergys Ristani, Francesco Solera, Roger S. Zou, Rita Cucchiara, and Carlo Tomasi. Performance measures and a data set for multi-target, multi-camera tracking. In *ECCV Workshops (2)*, pages 17–35, 2016. 2

[26] Mattia Segù, Luigi Piccinelli, Siyuan Li, Yung-Hsu Yang, Bernt Schiele, and Luc Van Gool. Samba: Synchronized set-of-sequences modeling for multiple object tracking. *CoRR*, abs/2410.01806, 2024. 4

[27] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. Crowdhuman: A benchmark for detecting human in a crowd. *CoRR*, abs/1805.00123, 2018. 2, 3, 4

[28] Peize Sun, Jinkun Cao, Yi Jiang, Zehuan Yuan, Song Bai, Kris Kitani, and Ping Luo. Dancetrack: Multi-object tracking in uniform appearance and diverse motion. In *CVPR*, pages 20961–20970. IEEE, 2022. 1, 2, 3, 5

[29] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N. Gomez, Lukasz Kaiser, and Illia Polosukhin. Attention is all you need. In *NIPS*, pages 5998–6008, 2017. 1

[30] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *ICIP*, pages 3645–3649. IEEE, 2017. 5

[31] Feng Yan, Weixin Luo, Yujie Zhong, Yiyang Gan, and Lin Ma. Bridging the gap between end-to-end and non-end-to-end multi-object tracking, 2023. 1, 3, 5

[32] Fan Yang, Shigeyuki Odashima, Shoichi Masui, and Shan Jiang. Hard to track objects with irregular motions and similar appearances? make it easier by buffering the matching space. In *WACV*, pages 4788–4797. IEEE, 2023. 5

[33] Mingzhan Yang, Guangxin Han, Bin Yan, Wenhua Zhang, Jinqing Qi, Huchuan Lu, and Dong Wang. Hybrid-sort: Weak cues matter for online multi-object tracking. *CoRR*, abs/2308.00783, 2023. 2

[34] En Yu, Tiancai Wang, Zhuoling Li, Yuang Zhang, Xiangyu Zhang, and Wenbing Tao. Motrv3: Release-fetch supervision for end-to-end multi-object tracking. *CoRR*, abs/2305.14298, 2023. 3, 5

[35] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. MOTR: end-to-end multiple-object tracking with transformer. In *ECCV (27)*, pages 659–675. Springer, 2022. 1, 3, 4, 5

[36] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *Int. J. Comput. Vis.*, 129(11):3069–3087, 2021. 2, 5

[37] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Fucheng Weng, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. In *ECCV (22)*, pages 1–21. Springer, 2022. 4, 5

[38] Yuang Zhang, Tiancai Wang, and Xiangyu Zhang. Motrv2: Bootstrapping end-to-end multi-object tracking by pretrained object detectors, 2022. 3, 5

[39] Yian Zhao, Wenyu Lv, Shangliang Xu, Jinman Wei, Guanzhong Wang, Qingqing Dang, Yi Liu, and Jie Chen. Detrs beat yolos on real-time object detection. In *CVPR*, pages 16965–16974. IEEE, 2024. 5

[40] Guangze Zheng, Shijie Lin, Haobo Zuo, Changhong Fu, and Jia Pan. Nettrack: Tracking highly dynamic objects with a net. In *CVPR*, pages 19145–19155. IEEE, 2024. 1, 5

[41] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *ECCV (4)*, pages 474–490. Springer, 2020. 5

[42] Xingyi Zhou, Tianwei Yin, Vladlen Koltun, and Philipp Krähenbühl. Global tracking transformers. In *CVPR*, pages 8761–8770. IEEE, 2022. 5

[43] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable DETR: deformable transformers for end-to-end object detection. In *ICLR*. OpenReview.net, 2021. 5