Collaborative Tracking Learning for Frame-Rate-Insensitive Multi-Object Tracking

Yiheng Liu, Junta Wu*, Yi Fu
ByteDance Inc.
lyh156@mail.ustc.edu.cn, juntawu@163.com, emmafu1013@gmail.com

Abstract

Multi-object tracking (MOT) at low frame rates can reduce computational, storage and power overhead to better meet the constraints of edge devices. Many existing MOT methods suffer from significant performance degradation in low-frame-rate videos due to significant location and appearance changes between adjacent frames. To this end, we propose to explore collaborative tracking learning (ColTrack) for frame-rate-insensitive MOT in a query-based end-to-end manner. Multiple historical queries of the same target jointly track it with richer temporal descriptions. Meanwhile, we insert an information refinement module between every two temporal blocking decoders to better fuse temporal clues and refine features. Moreover, a tracking object consistency loss is proposed to guide the interaction between historical queries. Extensive experimental results demonstrate that in high-frame-rate videos, ColTrack obtains higher performance than state-of-the-art methods on large-scale datasets Dancetrack and BDD100K, and outperforms the existing end-to-end methods on MOT17. More importantly, ColTrack has a significant advantage over state-of-the-art methods in low-frame-rate videos, which allows it to obtain faster processing speeds by reducing frame-rate requirements while maintaining higher performance. Code will be released at https://github.com/yolomax/ColTrack

1. Introduction

The goal of multi-object tracking (MOT) is to estimate bounding boxes and identities of objects of interest in videos. In high-frame-rate videos, the velocities of objects are slow, which makes the difference between adjacent frames small. State-of-the-art MOT methods [2, 28, 24, 34, 33, 37, 16, 30] achieve impressive results in the high-frame-rate situation. However, limited by storage, computing, and network bandwidth, low-frame-rate videos are very common. In low-frame-rate videos, the difference between adjacent frames is larger, which degrades the performance of existing methods.

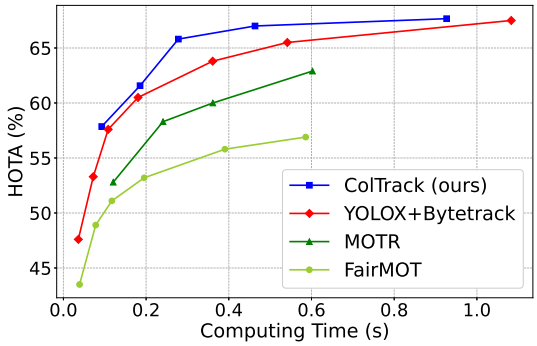

Figure 1. The comparison of the computing time for different methods to achieve the required HOTA score when tracking in a one-second video. These data are calculated based on the HOTA score and FPS of different methods on the MOT17 validation set. ColTrack still maintains high HOTA scores when tracking at low frame rates, so it achieves faster tracking speed by reducing the frame rate requirement while ensuring a high HOTA score.

The challenges caused by low-frame-rate videos are manifold. First, the displacements of objects between adjacent frames become larger, which even leads to no overlapping of the object boxes. This requires the model to match targets over a larger range, which includes more noisy objects. Furthermore, the position estimation error of the motion model (*e.g.* the Kalman filter [14]) is amplified and leads to significant performance degradation of the Kalman filter-based methods, *e.g.*, Bytetrack [33] and FairMOT [34]. Second, objects have severe appearance changes between adjacent frames. The viewpoints, visibilities, and poses of the objects change greatly. In addition, the sudden occlusion causes the objects to lose key appearance features rapidly. This greatly challenges some methods [34, 24] that rely on appearance features.

Some methods focusing on the detection of emerging objects [36] or the adjustment of training strategies [12] are proposed to improve the MOT performance at low frame

*First Author and Second Author contribute equally to this work.

rates, while these methods do not fundamentally solve the problems of unreliable features and large displacements in low-frame-rate videos. The end-to-end MOT methods [16, 30] use the deformable attention-based DETR-like detection model [8, 39] to match objects in the current frame based on the queries from the last frame. The utilization of deformable attention allows the model to adaptively find targets in a larger range. This helps to alleviate the problem caused by the large displacements. However, the matching of objects in this way heavily depends on the quality of the queries, which cannot be guaranteed due to the unreliable features in low-frame-rate videos.

In this paper, we propose collaborative tracking learning (ColTrack) for frame-rate-insensitive MOT, which is an end-to-end MOT approach. ColTrack utilizes multiple historical queries belonging to the same object as the collaborative tracking queries to track the same target. These queries contain descriptions of the same target at different moments. Their combination effectively alleviates the impact of unreliable features. However, the introduction of multiple historical queries to track the same target is against the one-to-one matching strategy of the DETR-like detection architecture. This causes the model to not only lose the capability of inhibiting duplicate predictions, but also fail to train with the bipartite matching loss [8].

To address these issues, we propose an information refinement module (IRM) and insert it between every two temporal blocking decoders to enable the information fusion between collaborative tracking queries while retaining the capability of inhibiting duplicate predictions. IRM contains an information removal branch and an information addition branch to assist queries to decide how to refine themselves based on temporal clues. Furthermore, we propose a tracking object consistency loss (TOCLoss), which requires each tracking query to collect discriminative features from other historical queries for the correct tracking. The joint use of these modules enables ColTrack to achieve more stable performance at low frame rates and better track difficult targets at high frame rates.

As shown in Fig. 1, ColTrack further increases the processing speed of the video by reducing the frame rate requirement while ensuring higher accuracy. In contrast, existing methods [33, 34, 30] require higher frame rates to achieve high accuracy, which results in more video frames being processed and lower processing speed.

To summarize, our contributions are as follows:

- We propose a query-based end-to-end model ColTrack that uses the collaborative tracking of multiple historical queries to achieve stable performance even at low frame rates.
- We further devise a IRM module to allow each query to better fuse information based on temporal cues. The proposed TOCLoss guides queries to collect valuable clues from other historical queries.
- ColTrack not only outperforms state-of-the-art methods on large-scale datasets under high frame rates but also achieves higher and more stable performance under low frame rates. This allows it to obtain a higher equivalent FPS by reducing the frame rate requirement.

2. Related Works

Classical MOT Methods. Most classical MOT methods follow the tracking-by-detection paradigm by detecting the object-bounding boxes first and then tracking objects by data association. For example, SORT [4] , DeepSORT [25], and ByteTrack [33] all follow this paradigm. They use Kalman filters to model tracks and update the underlying locations or features at each time step. JDE [24], FairMOT [34], and Unicorn [27] further explore the MOT system that jointly learns object detection and appearance embedding with a shared model.

Transformer-Based MOT Methods. Recently, the transformer has been applied in various computer vision tasks and achieved great success. TransTrack [22] introduces a query-key mechanism based on transformer architecture. It uses object features from the last frame as queries and tracks existing targets by associating bounding box locations. Trackformer [16] and MOTR [30] follow the DETR structure and both introduce autoregressive track queries to the transformer decoder to achieve implicit data association between frames. TransMOT [9] augments the transformer with spatial-temporal graphs to enhance the modeling capabilities of spatial relationships, which builds a new tracking-by-attention paradigm to MOT.

MOT Methods with Historical Features. Temporal information is crucial for MOT as it is essentially a video analysis task. MTrack [28] extracts discriminative representation to track objects in occlusion scenarios. It obtains the weighted feature representation of the trajectory according to the cosine similarity of historical features. GTR [38] matches current detection results with tracked objects by calculating the similarity between current features and multiple historical features. The interaction between historical features in these methods is limited, and the complementary information between features is not fully exploited.

MeMOT [6] designs the instance feature memory banks to generate a better track query for each object. However, fusing information from multiple historical queries into one query through only one module inevitably leads to information loss. To avoid these problems, we introduce multiple historical queries as collaborative tracking queries to jointly participate in the tracking. We allow multiple interactions of historical queries to fully integrate information.

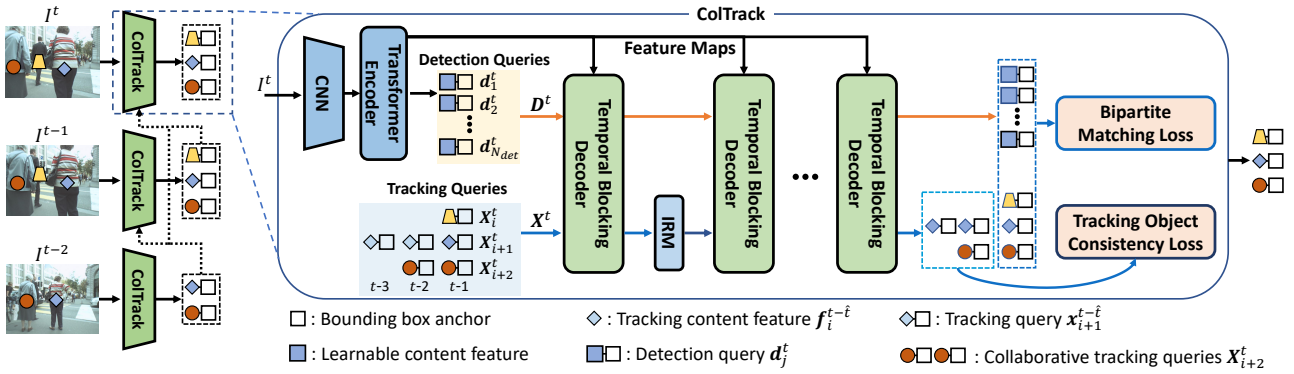

Figure 2. The overall workflow of ColTrack. The transformer encoder provides image feature maps and detection queries of emerging objects. Multiple historical queries of each tracked object constitute its collaborative tracking queries for the joint tracking of it. The combined queries are fed into multiple temporal blocking decoders to iteratively refine the predictions. An information refinement module (IRM) is inserted between every two decoders for collaborative tracking queries belonging to the same target to integrate temporal clues and refine themselves. The tracking object consistency loss guides the consistent tracking of historical queries to the corresponding target.

MOT at Low Frame Rates. The tracking in low-frame-rate videos is a big challenge for the MOT task due to the large difference between adjacent frames. APPTracker [36] adds an appear predictor (APP) head into CenterTrack [37] architecture to detect objects that newly appear in the current frame. Since CenterTrack [37] cannot predict the correct displacements for objects having visibility flips across frames, APP improves the performance of CenterTrack at low frame rates. However, APPTracker focuses more on the detection of new objects, which limits its performance in low-frame-rate videos.

FraMOT [12] directly introduces frame rate cues to the association module to handle the low-frame-rate case and applies tracking patterns to reduce the gap between the training phase and the inference one. However, the tracking patterns need to be updated periodically, which severely slows down the training. Besides, the frame rate information can only provide coarse clues for matching, which makes it unable to fundamentally solve the problems caused by a low frame rate.

Different from these methods, our method is specifically devised to address the large appearance and location change problem when tracking at a low frame rate. This enables our method to achieve satisfactory performance in both high-frame-rate and low-frame-rate videos.

3. Our Method

3.1. Overview

As shown in Fig. 2, the proposed collaborative tracking learning model (ColTrack) for frame-rate-insensitive MOT is built on the encoder-decoder Transformer [32] architecture. Given a sequence of video frames $\{\mathbf{I}^1, \mathbf{I}^2, \cdots, \mathbf{I}^T\}$, ColTrack tracks objects of interest in each frame and predicts their class and bounding boxes. For the video frame $\mathbf{I}^t$, the CNN model extracts its features, which are then fed to the transformer encoder to provide feature maps and N_{det} candidate target anchors. Each anchor $\hat{\mathbf{b}}_j^t$ is the predicted bounding box of one object. These anchors together with a set of learnable content features constitute N_{det} detection queries $\mathbf{D}^t = \{\mathbf{d}_j^t\}_{j=1:N_{\text{det}}}$. Detection queries are used to detect new objects appearing in the current frame.

Similar to the existing methods [16, 30, 6], ColTrack uses the queries output from previous video frames as the tracking queries to track the tracked targets. But unlike these methods that only construct one tracking query for each tracked target, ColTrack utilizes multiple historical features of each tracked target to construct collaborative tracking queries for collaborative tracking in the current frame. The tracking queries $\mathbf{X}^t$ and detection queries $\mathbf{D}^t$ are combined and fed to subsequent multiple temporal blocking decoders to iteratively refine features and bounding boxes. An information refinement module (IRM) is inserted between two adjacent decoders to allow information fusion of historical features belonging to the same target.

The output queries of each decoder are sent to the bipartite matching loss and the tracking object consistency loss to guide the training of the model (in Fig. 2, the losses of the middle decoders are not drawn for brevity). The queries of the matched targets output by the last decoder are sent to subsequent frames as new historical queries. After sequentially processing each frame, ColTrack obtains the tracking results of the entire video.

3.2. Collaborative Tracking Queries

To enhance the model's tracking at a low frame rate, we propose a collaborative tracking method based on historical features. For the i^{th} tracked object, we store its historical features $\mathbf{F}^t = \{\mathbf{f}_i^{t-\hat{t}}\}_{\hat{t}=1:N_i^t}$, $N_i^t \leq N_{\max}$. $N_{\max}$ is the max memory size and N_i^t is the number of historical features.

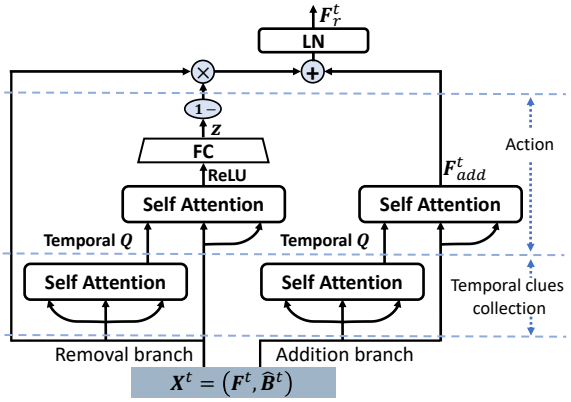

Figure 3. The architecture of the information refinement module (IRM). It mainly includes two branches: the information removal branch and the information addition branch. Each branch is mainly composed of temporal clues collection layer and action module.

$\mathbf{f}_i^{t-\hat{t}} \in \mathbb{R}^d$ is the output content feature of the corresponding target in $t-\hat{t}$ frame. Considering that usually the latest location prediction $\hat{\mathbf{b}}_i^{t-1}$ is closer to the location of the target in the current frame than $\{\hat{\mathbf{b}}_i^{t-\hat{t}}\}_{\hat{t}=2}^{N_i^t}$, we adopt $\hat{\mathbf{b}}_i^{t-1}$ as the target anchor, which is combined with each historical feature $\mathbf{f}_i^{t-\hat{t}}$ to construct the tracking query $\mathbf{x}_i^{t-\hat{t}}$. The N_i^t queries $\mathbf{X}_i^t = \{\mathbf{x}_i^{t-\hat{t}}\}_{\hat{t}=1:N_i^t}$ are considered as collaborative tracking queries and jointly track the same target in the current frame. They are initialized with different historical features and the same anchor $\hat{\mathbf{b}}_i^{t-1}$.

The benefits of the introduction of collaborative tracking queries are manifold. First, this approach provides more abundant descriptions of objects to alleviate the impact of unreliable features in low-frame-rate videos. Multiple historical queries containing different temporal clues directly participate in the tracking of the target in the current frame. During the iterative refinement of multiple decoders and IRM modules, each historical query adaptively collects valuable clues based on the tracking results of other historical queries to obtain a more accurate feature description for better tracking in the next decoder. The direct participation of historical queries enables the contained valuable information to be mined better.

Second, this approach is more conducive to training a frame-rate-insensitive MOT model. ColTrack requires historical features of the same target with different time spans from the current frame to track the target correctly. This is equivalent to training the model to track targets at different frame rates. Older features usually have a larger appearance difference from the current target. This requires ColTrack to be able to extract more robust features, and the refinement module IRM should be able to better integrate effective information from other historical features to deal with the impact of frame rate.

3.3. Information Refinement Module

As demonstrated in [8], benefit from the self-attention mechanism over the activations in decoders, the DETR-like detection model [8, 39, 32] can discard the non-maximum suppression (NMS) post-processing. This means that the decoder suppresses the case where multiple queries detect the same object to inhibit duplicate predictions. But in our method, collaborative tracking queries of the same object have to track the same object, which is against the design of the decoder. To address this issue, we propose temporal blocking decoders, which avoid the interaction between historical tracking queries of the same target by modifying the attention mask of the multi-head self-attention module in the decoder.

The temporal interaction blocking capability provided by the temporal blocking decoder avoids mutual inhibition between collaborative tracking queries of the same target, but it also makes it impossible to exchange information between collaborative tracking queries to refine themselves. To solve this problem, we devise a new information refinement module (IRM) to allow the interaction between collaborative tracking queries and refine them. As shown in Fig. 3, the multi-head self-attention layer is the main component of the IRM. The input of IRM is the N^t collaborative tracking queries of all the tracked objects, where $N^t = \sum_i N_i^t$. The output is the refined features. We modify the attention mask to avoid the interaction of features from different objects.

Since the main contribution of IRM is to allow collaborative tracking queries belonging to the same target to interact and update information, then for each query, it is necessary to decide how to remove old information and what new information to add. Inspired by this, the IRM we devised mainly consists of two branches: an information removal branch and an information addition branch. Each branch contains two parts, *i.e.* the temporal clues collection part and the action part.

A single multi-head self-attention layer of the transformer is not able to compute any cross-correlations between the queries [8]. This is because a single self-attention layer can only organize information once based on the similarities of features. The model cannot see the global information to decide how to output. Therefore, we add a multi-head self-attention module as the temporal clues collection part to collect global temporal clues for each tracking query.

For the information addition branch, the action part uses the collected global temporal clues as queries and uses the original content features as keys and values to generate the content features $\mathbf{F}_{\text{add}}^t \in \mathbb{R}^{N^t \times d}$ that needs to be added. Compared with the information addition branch, the information removal branch has one more fully connected layer followed by a sigmoid layer to map the embedding to a gating vector $\mathbf{z} \in \mathbb{R}^{N^t \times d_{\text{head}}}$. d_{head} is the head number of the multi-head self-attention module. We divide each con-

tent feature into d_{head} groups, and use the gating vector $\mathbf{z}$ to control the degree of deletion of each group of features. Then, we formulate the refinement of content features as

$$\mathbf{F}_r^t = \text{LN}(2\mathbf{F}^t \times (1-\mathbf{z}) + \mathbf{F}_{\text{add}}^t)\,, \tag{1}$$

where $\text{LN}(\cdot)$ is the layer normalization [1]. $\mathbf{F}_r^t \in \mathbb{R}^{N^t \times d}$ is the refined content features. $\mathbf{F}^t \times (1-\mathbf{z})$ denotes the reserved features. We double it to increase its weight. The refined content features $\mathbf{F}_r^t$ are combined with the corresponding anchors to form the refined tracking queries, which are sent to the next temporal blocking decoder. The combination of the temporal blocking decoders and IRM avoids the duplicate predictions problem and ensures effective information interaction between historical tracking queries belonging to the same target.

Besides, we insert IRM between every two decoders, which allows multiple interactions between historical queries. After each decoder, each query completes a target detection in the current frame and obtains new features and position estimation. During multiple interactions through IRMs, collaborative tracking queries continuously exchange varied new observations to obtain a more comprehensive description of the target. Collaborative tracking based on multiple information interactions is very important for stable tracking at a low frame rate.

3.4. Tracking Object Consistency Loss

In the existing transformer-based end-to-end MOT methods [30, 6, 16], the bipartite matching loss [8] is adopted to train the network. In our method, for each tracked target, due to the introduction of collaborative tracking queries, predictions of multiple queries belonging to the same target are matched with the same ground truth. This prevents us from directly using the one-to-one matching strategy to calculate the bipartite matching loss. Therefore, we propose a tracking object consistency loss (TOCLoss) to handle the training of the collaborative tracking queries. Then, the bipartite matching loss $\mathcal{L}_{\text{bip}}^t$ and the TOCLoss $\mathcal{L}_{\text{toc}}^t$ make up the overall training objective $\mathcal{L}$, which is formulated as

$$\mathcal{L} = \sum\nolimits_{t=1}^{T} (\mathcal{L}_{\text{bip}}^t + \mathcal{L}_{\text{toc}}^t)\,. \tag{2}$$

For collaborative tracking queries $\mathbf{X}_i^t = \{\mathbf{x}_i^{t-\hat{t}}\}_{\hat{t}=1:N_i^t}$ of one target, we use $\{\hat{\mathbf{y}}_i^{t-\hat{t}}\}_{\hat{t}=1:N_i^t}$ to denote their predictions in the current frame. Each prediction $\hat{\mathbf{y}}_i^{t-\hat{t}}$ contains the predicted class probabilities $\hat{\mathbf{p}}_i^{t-\hat{t}}$ and box prediction $\hat{\mathbf{b}}_i^{t-\hat{t}}$. We use ψ^t to denote the identity set of all tracked objects.

As shown in Fig. 2, the tracking predictions $\{\hat{\mathbf{y}}_i^{t-1}\}_{i\in\psi^t}$ of the latest tracking queries of all tracks are combined with the predictions of the detection queries to participate in the bipartite matching. The mapping π^t between predictions and ground truth objects is determined either via track identities or costs based on object class and bounding box similarity [16]. Then, we calculate the bipartite matching loss for them by mapping π^t, which is denoted as $\mathcal{L}_{\text{bip}}^t$. Since the additional predictions $\{\hat{\mathbf{y}}_i^{t-\hat{t}}\}_{\hat{t}=2:N_i^t}$ of each track are separated from $\mathcal{L}_{\text{bip}}^t$, $\mathcal{L}_{\text{bip}}^t$ satisfies the one-to-one matching requirement.

As for the remaining predictions $\{\hat{\mathbf{y}}_i^{t-\hat{t}}\}_{\hat{t}=2:N_i^t}$ of the i^{th} track, since they and $\hat{\mathbf{y}}_i^{t-1}$ have the same identity, they can share the mapping π^t between them and the ground truth objects. Then, we define the loss of each prediction as

$$\mathcal{L}_i^t(\hat{\mathbf{y}}_i^{t-\hat{t}}, \pi^t) = \begin{cases} -\log \hat{\mathbf{p}}_i^{t-\hat{t}}(\pi^t(i)) + \mathcal{L}_{\text{b}}(\hat{\mathbf{b}}_i^{t-\hat{t}}, \pi^t) & \text{if } i \in \pi^t\,, \\ -\log \hat{\mathbf{p}}_i^{t-\hat{t}}(0) & \text{otherwise}\,, \end{cases} \tag{3}$$

where $\hat{\mathbf{p}}_i^{t-\hat{t}}(\pi^t(i))$ is the predicted probability of the assigned class obtained from mapping π^t. $\hat{\mathbf{p}}_i^{t-\hat{t}}(0)$ is probability of background class. $\mathcal{L}_{\text{b}}(\hat{\mathbf{b}}_i^{t-\hat{t}}, \pi^t)$ is the bounding box loss [32]. Then, the tracking object consistency loss $\mathcal{L}_{\text{toc}}^t$ of the remaining historical tracking queries is formulated as

$$\mathcal{L}_{\text{toc}}^t = \frac{\sum_{i\in\psi^t} \sum_{\hat{t}=2}^{N_i^t} \mathcal{L}_i^t(\hat{\mathbf{y}}_i^{t-\hat{t}}, \pi^t)}{\sum_{t=1}^{T} N_{\text{his}}^t}\,, \tag{4}$$

where N_{his}^t is the number of queries that are assigned ground truth objects by mapping $\pi^t(i)$ among the remaining historical tracking queries, which is formulated as

$$N_{\text{his}}^t = \sum_{i\in\pi^t} |\{\hat{\mathbf{y}}_i^{t-\hat{t}}\}_{\hat{t}=2:N_i^t}|\,. \tag{5}$$

In the inference stage, for a track, given the predictions of all collaborative tracking queries, only the output $\hat{\mathbf{y}}_i^{t-1}$ of query $\mathbf{x}_i^{t-1}$ is adopted as the predicted location and score of the target in the current frame. The remaining historical queries provide temporal cues to assist tracking and their final output is ignored. We consider the object to appear when the score is greater than a threshold σ, otherwise, the object is lost. The queries of lost targets are kept for up to N_{keep} frames.

4. Experiment

4.1. Experimental Setup

Datasets. We evaluate our method on three popular multi-object tracking datasets, including MOT17 [11], Dancetrack [21] and BDD100K [29]. We use the private detection protocol for MOT17. Following [33, 34, 30], the CrowdHuman dataset [20] is added to build the joint dataset when training on the MOT17 and Dancetrack. MOT17 does not provide a validation set. In the ablation studies part, we use the first half of each video in the training set for training and the last half for validation following [33, 34, 30].

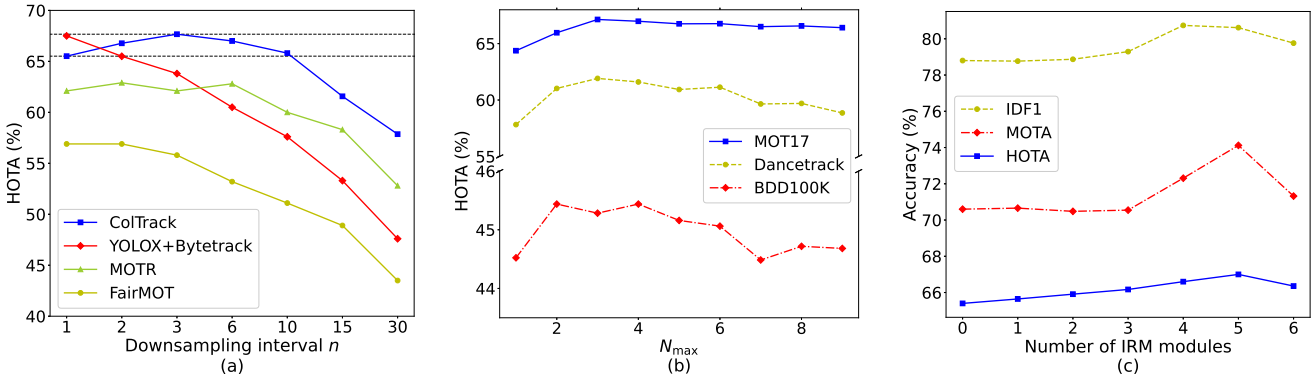

Figure 4. (a) Performance comparison of ColTrack with existing methods on videos at different frame rates on the validation set of MOT17. n represents that the videos are downsampled at a sampling interval of n. (b) Performance of ColTrack under different $N_{\max}$ in low-frame-rate videos ($n = 6$). $N_{\max}$ is the maximum number of collaborative historical queries of each track. (c) Performances of ColTrack with different numbers of IRMs on the validation set of MOT17 under low frame rate ($n = 6$).

Evaluation Protocols. We follow the standard evaluation protocols including CLEAR metrics (MOTA, IDs) [3], Identity F1 Score (IDF1) [19], and HOTA [15]. MOTA focuses more on the detection performance, while IDF1 focuses more on the association performance. HOTA [15] balances the impact of detection and data association well. Therefore, we take HOTA as the main metric.

Implementation Details. ColTrack extends the DETR-like deformable detection model DINO [32] and takes ResNet50 [13] as the CNN feature extractor. The tracking-by-detection version of the baseline method is represented as Baseline+Bytetrack, which directly trains a detection model and uses Bytetrack [33] to associate objects. The model is trained for 40 epochs. The end-to-end (E2E) version of the baseline method is denoted as Baseline+E2E, which directly takes the tracking results of the last frame as the tracking queries [16, 30] in the current frame. To reduce the GPU memory usage and increase the video clip length during training, we use the detection model trained by Baseline+Bytetrack to initialize the CNN and encoders of Baseline+E2E and ColTrack, whose parameters are frozen. The decoders and 300 learnable query embeddings are trained from scratch. The model is trained 60 epochs on MOT17, 40 epochs on Dancetrack and 20 epochs on BDD100K.

Following [33], the input frames are resized to 1440×800. The data augmentation includes multi-scale training, Mosaic [5] and Mixup [31]. All the models are trained by AdamW algorithm with an initial learning rate of 1×10^{-4} and weight decay of 1×10^{-4}. The learning rate is scaled by $\times 0.1$ during the last 10 epochs. The batch size is 8 video clips and each has 4 frames. $N_{\text{det}} = 300, d_{\text{head}} = 8, \sigma = 0.6, N_{\text{keep}} = 5, N_{\max} = 3$. Following [33], we measure FPS with FP16-precision and batch size of 1 on a single V100 GPU.

To obtain videos under different frame rates, following [36], we sample frames at a fixed interval n from the original dataset to obtain low-frame-rate videos. n varies from 1 to 30. The larger n, the lower the frame rate, and the larger difference between adjacent frames.

	ColTrack	YOLOX+ Bytetrack	FairMOT	MOTR
FPS*	10.8	27.7	25.6	8.3
FPS$^{\text{eq}}$	32.4	27.7	25.6	16.6
HOTA	67.7	67.5	56.9	62.9

Table 1. FPS and the highest HOTA score comparison of different methods on the validation set of MOT17. FPS* denotes the FPS we obtained by reproducing the methods on the same machine. FPS$^{\text{eq}}$ is the equivalent FPS, which is obtained by multiplying FPS* by the downsampling interval n when each method obtains the highest HOTA score. The corresponding maximum HOTA score is also listed.

4.2. Effect of Frame Rate on MOT

Performance at low frame rates. In Fig. 4(a), we compare the performance of ColTrack with existing methods [33, 34, 30] at different frame rates on MOT17 dataset.

MOTR [30] has stable HOTA when n is less than 6. The HOTA of ColTrack is stable when n is varied from 1 to 10. This is benefited from the deformable attention-based end-to-end architecture, which relies on content features to track the target. This approach has more advantageous in a low-frame-rate situation that has large displacements.

Compared to MOTR, the HOTA performance and frame rate robustness of ColTrack are significantly better, which benefits from the introduction of collaborative tracking queries and the exquisite design of each module. They provide more temporal cues for the model to obtain more comprehensive and accurate descriptions of objects.

For YOLOX+Bytetrack [33] and ReID-based FairMOT [34], the ablation models use the officially provided weight. Both of them show a rapid performance drop when

Method	MOT17		Dancetrack			BDD100K		
	n=1	n=10	n=1	n=6	n=10	n=1	n=6	n=10
IDF1								
APPTracker	68.7	70.3	-	-	-	-	-	-
BL+Bytetrack	77.5	64.4	47.1	31.6	27.3	41.3	22.6	20.3
BL+E2E	77.6	74.8	50.6	55.5	49.9	51.0	50.7	47.4
ColTrack	78.1	78.6	54.6	61.6	51.3	54.0	52.7	51.4
MOTA								
APPTracker	68.7	65.5	-	-	-	-	-	-
BL+Bytetrack	75.3	61.0	89.4	72.2	62.3	29.4	14.6	13.2
BL+E2E	73.8	64.9	88.9	86.3	79.7	36.1	34.7	30.6
ColTrack	76.5	68.7	86.6	86.5	80.7	40.0	37.0	35.7
HOTA								
FairMOT	56.9	51.1	37.6	26.3	23.4	-	-	-
Bytetrack	67.5	57.6	46.1	32.3	29.4	-	-	-
OC-SORT	66.1	56.1	52.2	35.9	30.3	-	-	-
MOTR	62.1	60.0	51.7	52.2	47.3	-	-	-
BL+Bytetrack	65.2	55.9	45.8	30.8	26.8	33.7	22.1	21.3
BL+E2E	64.9	62.5	55.6	58.4	52.8	42.3	43.0	41.0
ColTrack	65.5	65.8	57.9	61.9	53.7	45.0	45.3	44.6

Table 2. Performance comparison of different methods on videos at different frame rates on the validation set of three datasets. BL denotes Baseline. APPTracker [36] is an existing method devised for MOT under low frame rates.

the frame rate is lowered. This is because these methods all rely on the Kalman filter and IOU matching, which are unreliable for fast-moving objects.

Equivalent FPS. The HOTA accuracy of ColTrack when $n = 3$ is higher than that of YOLOX+Bytetrack when $n = 1$. Therefore, as shown in Table. 1, although ColTrack doesn't have the highest FPS, it achieves a higher equivalent FPS by reducing the frame rate requirement. This allows it to process a video in a shorter time while ensuring high accuracy. Baseline+E2E has an 11.3 FPS running speed. Comparing the FPS of Baseline+E2E and ColTrack, it can be seen that the calculation overhead caused by the introduction of historical queries and several IRM modules is small. This is because these only slightly affect the calculation of the decoder part, and the calculations of CNN and encoders are not affected.

Verification under various scenarios. Further, in Table. 2, we compare the performances of different methods in videos at varied frame rates on three datasets. Compared with APPTracker[36] devised for MOT in low-frame-rate videos, ColTrack achieves higher performance on both low-frame-rate videos as well as high-frame-rate videos. APPTracker pays more attention to the detection and association of emerging targets and does not fully solve the problems in the follow-up tracking of fast-moving targets.

Compared with Baseline+Bytetrack and Baseline+E2E, ColTrack has a significant performance improvement. The erratic movements and similar appearance of the dancers in Dancetrack make tracking difficult for the classical association method ByteTrack [33] based on Kalman filtering. Meanwhile, the large driving dataset BDD100K with multiple categories contains more fast-moving objects. This makes object tracking very challenging in its low-frame-rate videos. Benefiting from the introduction of collaborative tracking queries, ColTrack still has satisfactory performance when n is varied from 1 to 10 on BDD100K.

Method	MOT17			Dancetrack		
	HOTA	IDF1	MOTA	HOTA	IDF1	MOTA
BL+Bytetrack	59.3	69.1	66.5	30.8	31.6	72.2
BE (BL+E2E)	64.5	78.1	67.3	58.4	55.5	86.3
BE+CTQ	65.1	78.5	70.4	59.3	57.1	86.1
BE+CTQ+IRM (rem)	66.1	79.2	71.7	60.0	57.3	86.0
BE+CTQ+IRM (add)	66.6	79.9	71.1	60.6	58.6	86.6
BE+CTQ+IRM	66.7	80.0	72.2	61.0	58.9	**87.6**
BE+CTQ+IRM+TOC	**67.0**	**80.6**	**74.1**	**61.9**	**61.6**	86.5

Table 3. Ablation study of the components in ColTrack on the validation set of MOT17 and DanceTrack with downsampling interval $n = 6$. BL means Baseline. BE means BL+E2E. CTQ denotes collaborative tracking queries. IRM is the information refinement module. IRM (add) means only the addition branch in IRM is kept. IRM (rem) means only the removal branch is kept. TOC denotes the tracking object consistency loss (TOCLoss).

4.3. Ablation Study

In Table. 3, we analyze the impact of each module on the model performance under low frame rates ($n = 6$). In BL+E2E+CTQ, although collaborative tracking queries are used, there is no interaction between historical queries of the same track. They can only assist the model training by interacting with queries of other tracks as negative samples through temporal blocking decoders. Therefore, the improvement BL+E2E+CTQ brings is very small.

When IRM is adopted, the performance of the model is significantly improved. This is because IRM enables collaborative tracking queries to interact with each other and refine themselves with temporal clues. We also analyze the impact of the information removal branch and the information addition branch in IRM on the model performance. Experimental results show that having both information removal and information addition capabilities helps the model refine features better.

After adding TOCLoss, the performance of the model is further improved. TOCLoss forces collaborative tracking queries to refine themselves with IRM to better track corresponding targets. Then, they provide better temporal clues during iterative refinement in the following decoders.

In Fig. 4(b) we analyze the effect of the max number of collaborative tracking queries $N_{\max}$ on the tracking performance of ColTrack. The results indicate that ColTrack achieves best performances when $N_{\max} = 3$. A too large $N_{\max}$ results in too many historical features being included, which introduces too much noise.

Analysis of the location and number of IRMs. We also compare the performance of ColTrack with the differ-

Method	Source	HOTA	IDF1	MOTA	IDs
TraDes [26]	CVPR'21	52.7	63.9	69.1	3555
FairMOT [34]	IJCV'21	59.3	72.3	73.7	3303
MTrack [28]	CVPR'22	-	73.5	72.1	2028
Unicorn [27]	ECCV'22	61.7	75.5	77.2	5379
YOLOX+Bytetrack [33]	ECCV'22	63.1	77.3	80.3	2196
OC-SORT [7]	CVPR'23	**63.2**	77.5	78.0	1950
P3AFormer (Swin) [35]	ECCV'22	-	**78.1**	**81.2**	1893
E2E methods					
CenterTrack [37]	ECCV'20	52.2	64.7	67.8	3039
Chained-tracker [18]	ECCV'20	49.0	57.4	66.6	5529
TransTrack [22]	arXiv'20	-	63.9	74.5	3663
TrackFormer [16]	CVPR'22	-	68.0	74.1	2829
MeMOT [6]	CVPR'22	56.9	69.0	72.5	2724
MOTR [30]	ECCV'22	57.8	68.6	73.4	2439
ColTrack	-	61.0	73.9	78.8	**1881**

Table 4. Performance comparison between ColTrack and the state-of-the-art methods under the "private detector" protocol on MOT17 test set.

Method	Source	HOTA	IDF1	MOTA	AssA
FairMOT [34]	IJCV'21	39.7	40.8	82.2	23.8
TransTrack [22]	arXiv'20	45.5	45.2	88.4	27.5
CenterTrack [37]	ECCV'20	41.8	35.7	86.8	22.6
TraDes [26]	CVPR'21	43.3	41.2	86.2	25.4
QDTrack [17]	CVPR'21	54.2	50.4	87.7	36.8
YOLOX+Bytetrack [33]	ECCV'22	47.7	53.9	89.6	32.1
MOTR [30]	ECCV'22	54.2	51.5	79.7	40.2
OC-SORT [7]	CVPR'23	55.1	54.2	89.4	38.0
MOTRv2 [23]	CVPR'23	69.9	71.7	91.9	59.0
MOTRv2 [23] (+val+ens)	CVPR'23	73.4	76.0	92.1	64.4
ColTrack	-	72.6	74.0	92.1	62.3
ColTrack (+val)	-	**75.3**	**77.3**	**92.2**	**66.9**

Table 5. Evaluation results on the test set of Dancetrack. *+val* means adding the validation set for training. *+ens* denotes test ensemble.

ent number of IRM modules in Fig. 4(c). 6 decoders can add up to 6 IRM modules. We gradually remove the corresponding IRM in the shallow decoder layers. The experimental results indicate that removing the IRM before the first decoder layer and adding an IRM before each of the following decoder layers achieves the best performance. This is because queries have not interacted with the features of the current frame before passing through the first decoding layer, which makes tracking queries unable to adjust the fusion of temporal information according to the current frame. After the first decoder, more IRM modules allow queries to have more opportunities to communicate new observations with each other to gather more valuable information.

4.4. Comparison to the State-of-the-art Methods

In Table. 4, Table. 5 and Table. 6, we compare ColTrack with the state-of-the-art methods on three datasets. MOT17 is a small dataset containing only 7 training videos and 7 testing videos. Although such a small amount of data is difficult to train an end-to-end model to learn temporal relationship modeling, ColTrack still outperforms existing end-to-end methods and achieves comparable performance to existing tracking-by-detection methods. Dancetrack containing 100 videos is a large challenging dataset due to the irregular movements, similar clothing, and severe occlusions of the dancers. ColTrack outperforms all methods on Dancetrack. This is benefited by our introduction of collaborative tracking queries, which provide more abundant descriptions of targets and makes ColTrack less susceptible to similar appearances and occlusions. BDD100K containing 2000 driving videos is also a challenging dataset due to complex scenes and more fast-moving objects. ColTrack also achieves the best performance on BDD100K, especially for the IDF1 metric that focuses more on association performance.

Method	Source	Split	mMOTA	mIDF1	IDs
Yu *et al*. [29]	CVPR'20	val	25.9	44.5	8315
DeepBlueAI [10]	CVPRC'20	val	26.9	-	13366
QDTrack [17]	CVPR'21	val	36.6	50.8	6262
MOTR [30]	ECCV'22	val	32.0	43.5	**3493**
Unicorn(ResNet) [27]	ECCV'22	val	35.1	-	-
YOLOX+Bytetrack [33]	ECCV'22	val	39.4	48.9	27902
ColTrack	-	val	**40.0**	**54.0**	3741
Yu *et al*. [29]	CVPR'20	test	26.3	44.7	14674
DeepBlueAI [10]	CVPRC'20	test	31.6	38.7	25186
QDTrack [17]	CVPR'21	test	35.5	52.3	10790
ColTrack	-	test	**40.4**	**56.0**	**6249**

Table 6. Comparison of the state-of-the-art methods on BDD100K.

These sufficient experimental results show that on more challenging datasets, our method not only achieves higher performance than existing methods under high frame rates but also better tracks objects in low-frame-rate videos. This fully proves that ColTrack is a frame-rate-insensitive model. It achieves faster processing speed while ensuring high performance at different frame rates.

5. Conclusion

In this paper, we propose a collaborative tracking learning method (ColTrack) to address the challenges introduced by low frame rates in multi-object tracking (MOT). By introducing multiple historical queries to track the same target, rich temporal clues are used to obtain more comprehensive and accurate descriptions of the targets. We carefully devise the temporal blocking decoders and the information refinement module (IRM) such that the model allows collaborative tracking queries to better integrate the information while retaining the ability to inhibit duplicate predictions. Meanwhile, the proposed tracking object consistency loss (TOCLoss) forces each historical query to integrate valuable clues from other queries for the correct tracking. Thanks to the collaboration of these modules, ColTrack outperforms existing methods and achieves faster processing speeds on more challenging datasets Dancetrack and BDD100K at both high and low frame rates.

References

[1] Jimmy Lei Ba, Jamie Ryan Kiros, and Geoffrey E Hinton. Layer normalization. *arXiv preprint arXiv:1607.06450*, 2016.

[2] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. Tracking without bells and whistles. In *CVPR*, pages 941–951, 2019.

[3] Keni Bernardin and Rainer Stiefelhagen. Evaluating multiple object tracking performance: the clear mot metrics. *JIVP*, 2008:1–10, 2008.

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

[5] Alexey Bochkovskiy, Chien-Yao Wang, and Hong-Yuan Mark Liao. Yolov4: Optimal speed and accuracy of object detection. *arXiv preprint arXiv:2004.10934*, 2020.

[6] Jiarui Cai, Mingze Xu, Wei Li, Yuanjun Xiong, Wei Xia, Zhuowen Tu, and Stefano Soatto. Memot: multi-object tracking with memory. In *CVPR*, pages 8090–8100, 2022.

[7] Jinkun Cao, Xinshuo Weng, Rawal Khirodkar, Jiangmiao Pang, and Kris Kitani. Observation-centric sort: Rethinking sort for robust multi-object tracking. *arXiv preprint arXiv:2203.14360*, 2022.

[8] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *ECCV*, pages 213–229. Springer, 2020.

[9] Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. Transmot: Spatial-temporal graph transformer for multiple object tracking. In *WACV*, pages 4870–4880, 2023.

[10] CodaLab Competition. Cvpr 2020 bdd100k mot challenge. https://competitions.codalab.org/competitions/24910. Online; accessed 19. Jul. 2022.

[11] Patrick Dendorfer, Aljosa Osep, Anton Milan, Konrad Schindler, Daniel Cremers, Ian Reid, Stefan Roth, and Laura Leal-Taixé. Motchallenge: A benchmark for single-camera multiple target tracking. *IJCV*, 129(4):845–881, 2021.

[12] Weitao Feng, Lei Bai, Yongqiang Yao, Fengwei Yu, and Wanli Ouyang. Towards frame rate agnostic multi-object tracking. *arXiv preprint arXiv:2209.11404*, 2022.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

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

[15] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *IJCV*, 129:548–578, 2021.

[16] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. In *CVPR*, pages 8844–8854, 2022.

[17] 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, 2021.

[18] Jinlong Peng, Changan Wang, Fangbin Wan, Yang Wu, Yabiao Wang, Ying Tai, Chengjie Wang, Jilin Li, Feiyue Huang, and Yanwei Fu. Chained-tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking. In *ECCV*, pages 145–161. Springer, 2020.

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

[20] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. Crowdhuman: A benchmark for detecting human in a crowd. *arXiv preprint arXiv:1805.00123*, 2018.

[21] 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 20993–21002, 2022.

[22] Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple object tracking with transformer. *arXiv preprint arXiv:2012.15460*, 2020.

[23] MOTRv2: Bootstrapping End to-End Multi-Object Tracking by Pretrained Object Detectors. Quasi-dense similarity learning for multiple object tracking. In *CVPR*, 2023.

[24] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. Towards real-time multi-object tracking. In *ECCV*, pages 107–122. Springer, 2020.

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

[26] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to detect and segment: An online multi-object tracker. In *CVPR*, pages 12352–12361, 2021.

[27] Bin Yan, Yi Jiang, Peize Sun, Dong Wang, Zehuan Yuan, Ping Luo, and Huchuan Lu. Towards grand unification of object tracking. In *ECCV*, pages 733–751. Springer, 2022.

[28] En Yu, Zhuoling Li, and Shoudong Han. Towards discriminative representation: multi-view trajectory contrastive learning for online multi-object tracking. In *CVPR*, pages 8834–8843, 2022.

[29] Fisher Yu, Haofeng Chen, Xin Wang, Wenqi Xian, Yingying Chen, Fangchen Liu, Vashisht Madhavan, and Trevor Darrell. Bdd100k: A diverse driving dataset for heterogeneous multitask learning. In *CVPR*, pages 2636–2645, 2020.

[30] Fangao Zeng, Bin Dong, Yuang Zhang, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. In *ECCV*, pages 659–675. Springer, 2022.

[31] Hongyi Zhang, Moustapha Cisse, Yann N Dauphin, and David Lopez-Paz. mixup: Beyond empirical risk minimization. *arXiv preprint arXiv:1710.09412*, 2017.

[32] Hao Zhang, Feng Li, Shilong Liu, Lei Zhang, Hang Su, Jun Zhu, Lionel M. Ni, and Heung-Yeung Shum. Dino: Detr with improved denoising anchor boxes for end-to-end object detection, 2022.

[33] 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*, pages 1–21. Springer, 2022.

[34] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *IJCV*, 129:3069–3087, 2021.

[35] Zelin Zhao, Ze Wu, Yueqing Zhuang, Boxun Li, and Jiaya Jia. Tracking objects as pixel-wise distributions. In *ECCV*, pages 76–94. Springer, 2022.

[36] Tao Zhou, Wenhan Luo, Zhiguo Shi, Jiming Chen, and Qi Ye. Apptracker: Improving tracking multiple objects in low-frame-rate videos. In *ACM MM*, pages 6664–6674, 2022.

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

[38] Xingyi Zhou, Tianwei Yin, Vladlen Koltun, and Philipp Krähenbühl. Global tracking transformers. In *CVPR*, pages 8771–8780, 2022.

[39] Xizhou Zhu, Weijie Su, Lewei Lu, Bin Li, Xiaogang Wang, and Jifeng Dai. Deformable detr: Deformable transformers for end-to-end object detection. *arXiv preprint arXiv:2010.04159*, 2020.