Noname manuscript No.
(will be inserted by the editor)

APPTracker+: Displacement Uncertainty for Occlusion Handling in Low-Frame-Rate Multiple Object Tracking

Tao Zhou · Qi Ye · Wenhan Luo · Haizhou Ran · Zhiguo Shi · Jiming Chen

Received: date / Accepted: date

Abstract Multi-object tracking (MOT) in the scenario of low-frame-rate videos is a promising solution to better meet the computing, storage, and transmitting bandwidth resource constraints of edge devices. Tracking with a low frame rate poses particular challenges in the association stage as objects in two successive frames typically exhibit much quicker variations in locations, velocities, appearances, and visibilities than those in normal frame rates. In this paper, we observe severe performance degeneration of many existing association strategies caused by such variations. Though optical-flow-based methods like CenterTrack can handle the large displacement to some extent due to their large receptive field, the temporally local nature makes them fail to give reliable displacement estimations of objects that newly appear in the current frame (i.e., not visible in the previous frame). To overcome the local nature of optical-flow-based methods, we propose an online tracking method by extending the CenterTrack architecture with a new head, named APP, to recognize unreliable displacement estimations. Further, to capture the fine-grained and private unreliability of each displacement estimation, we extend the binary APP predictions to displacement uncertainties. To this end, we reformulate the displacement estimation task via Bayesian deep learning tools. With APP predictions, we propose to conduct association in a multi-stage manner where vision cues or historical motion cues are leveraged in the corresponding stage. By rethinking the commonly used bipartite matching algorithms, we equip the proposed multi-stage association policy with a hybrid matching strategy conditioned on displacement uncertainties. Our method shows robustness in preserving identities in low-frame-rate video sequences. Experimental results on public datasets in various low-frame-rate settings demonstrate the advantages of the proposed method.

Keywords Multi-object tracking, Low-frame-rate videos, Occlusion handling, Uncertainty

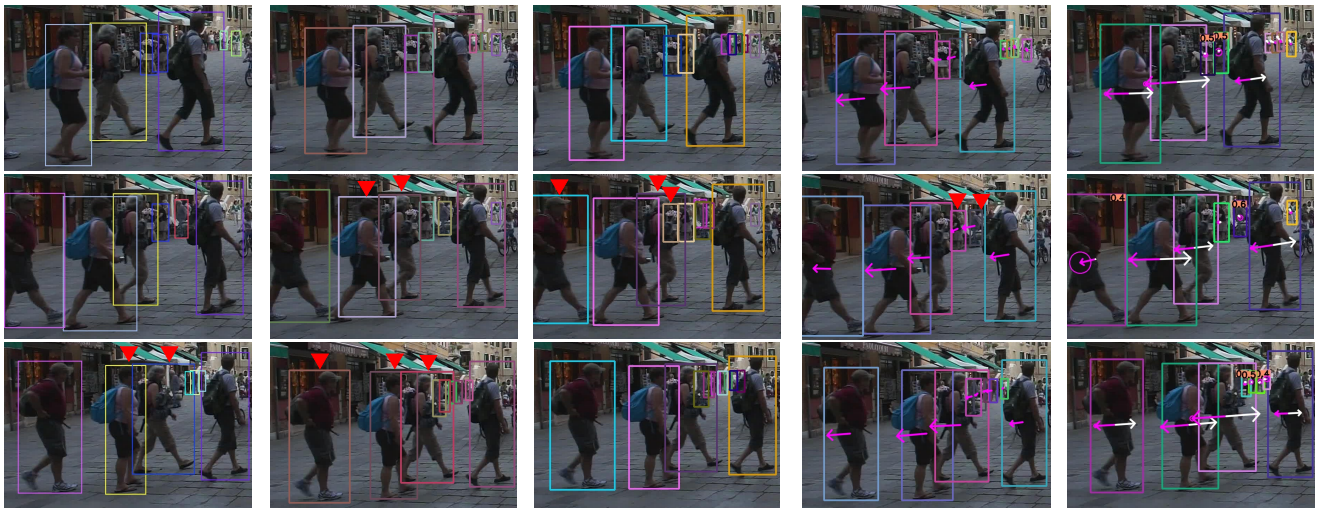
Tao Zhou, Qi Ye, Haizhou Ran, Zhiguo Shi, and Jiming Chen
Zhejiang University, China
E-mail: {zhoutao2015, qi.ye, haizhou.ran, shizg, cjm}@zju.edu.cn

Wenhan Luo
Hong Kong University of Science and Technology, Hong Kong
E-mail: whluo.china@gmail.com

1 Introduction

In multi-object tracking (MOT) [45], objects of interest are usually required to be detected and tracked in videos recorded with a real-time frame rate. The problem has been studied for decades and tremendous progress has been achieved. However, collecting real-time videos from massive edge cameras and performing inferences on these videos can be constrained by the limitations of computing, storage, and transmitting bandwidth. A potential way to solve the constraints is to decrease the capturing frame rate, but the decreased frame rate causes various difficulties in tracking multiple objects robustly.

Specifically, as shown in Fig. 1, there are several challenges in tracking multiple objects in low-frame-rate videos. First, the displacement of objects between frames becomes larger; in some cases, two detections of the same target in two consecutive frames even have no overlap. Methods [50,3,61] that assume objects moving slightly between frames and thus fail in low-frame-



(a) Tracktor++ (b) ByteTrack (c) FairMOT (d) CenterTrack (e) Ours

Fig. 1: Tracking results of modern trackers and our method in low-frame-rate videos. Identities are coded by color. Each column shows three consecutive frames. We use red triangles to highlight identity switches. While modern trackers fail to handle the quicker variations in locations, appearances, and visibilities of objects in low-frame-rate cases, our method shows robustness in preserving identities. Best viewed in color and by zooming in.

rate cases (shown in Fig. 1a). Besides, the larger displacement leads to the failure of motion models (like the Kalman filter [27]) using zero-velocity initialization. Thus, as shown in Fig. 1b, the effectiveness of methods [76,5] which heavily rely on motion models is limited in low-frame-rate cases.

Second, the visibilities and appearances of objects change more abruptly. In high-frame-rate videos, objects are occluded gradually. While in low-frame-rate videos, one highly visible object in the previous frame may be extremely occluded in the current frame. The influence is two-fold. 1) The larger difference in occlusion states and appearances of the same target between frames makes the appearance less discriminative (shown in Fig. 1c[1]). 2) The large difference in detection entries between frames results in further noise in the association matrix.

A model with a vision-based motion estimator together with a sufficient receptive field is expected to address the first challenge. The recent CenterTrack [81] architecture is therefore selected. It takes two adjacent frames as input and outputs detections and displacement estimations (indicated by pink arrows in Fig. 1d). Without seeking historical information beyond two frames, CenterTrack tracks objects from a temporally local perspective. Enabled by a deep deformable convolutional neural network (DCN) [16], it can handle large displacements between frames. However, the temporally local nature makes CenterTrack suffer from the second challenge. It fails to give correct displacement estimations for discontinuously visible targets, which results in identity switches (shown in Fig. 1d).

In this paper, we study the challenges above and propose an online tracker, APPTracker+, to track multiple objects in low-frame-rate videos, aiming at enhancing the ability to preserve identities when objects are involved in occlusions. To this end, we first propose an appear predictor (APP) to detect objects that newly appear in the current frame (i.e., not visible in the previous frame), akin to detecting anomalies of optical flows. For an appearing object, the predictor produces a positive response, indicating that the displacement estimation of this object is not reliable. In Fig. 1e, we highlight recognized appearing objects by indicating the predicted APP scores at the top right of the corresponding bounding boxes.

Apart from newly appearing objects, the displacement estimation of each non-appearing object is not equally reliable, especially those being partially occluded. To capture such fine-grained unreliability for each displacement estimation, the model is further extended to provide displacement uncertainties in addition to the binary APP predictions. Specifically, the uncertainty of the displacement captures the residual between the ground-truth displacement and the estimated one. To this end, we reformulate the displacement estimation task as a heteroscedastic regression task [35,29], which assumes that observation noise can vary across instances. In Fig. 1e, the radius of the circle at the end

[1] To evaluate the performance of the re-identification head of FairMOT [77], we disable the Kalman filter inside the original FairMOT method. Details can be found in Sec.4.6.

of the arrow represents the uncertainty of the displacement estimation (best viewed by zooming in).

Training a robust APP head is challenging. For instance, the ratio of emerging samples is fairly low (less than 10% among all instances) on the popularly used MOT17 dataset [48]. Thus, we propose a data augmentation strategy, leveraging a large-scale static image dataset [58] by randomly erasing objects in images to create emerging samples.

With APP predictions and displacement uncertainties, we propose a multi-stage matching policy during association. We use the APP prediction to determine the switch between using vision cues (i.e., displacement estimation) or motion cues (where we adopt a constant velocity assumption) for the association. Further, we rethink the commonly used bipartite matching algorithms, i.e., the greedy matching [81], and Hungarian matching [33] and experimentally reveal that greedy matching is better at handling discrete detection noise (e.g., missed detections) because greedy matching is more robust to outliers, while Hungarian matching is more adept at dealing with association noise of inliers (e.g., small errors in displacement estimates). Based on these findings, we propose a hybrid matching policy conditioned on displacement uncertainties. In this multi-stage manner, the noise inside the association matrix caused by unreliable vision cues is reduced, and the local nature is overcome by leveraging historical motion cues. In Fig. 1e, we employ white arrows to indicate motion cues.

This paper extends our previous conference version [80] in several aspects. (1) We extend the idea of binary APP predictions to displacement uncertainties by reformulating the regression task via Bayesian deep learning tools. This enables the model to capture the fine-grained unreliability of each displacement estimation. (2) We conduct a rethink of greedy matching and Hungarian matching. Oracle experiments reveal the different strengths of these two matching strategies under detection noise and association noise. An uncertainty-conditioned hybrid matching strategy is proposed inspired by this observation. (3) More experimental studies and in-depth analyses are provided. Besides, we update the results with a recent metric, HOTA [43], as recommended by the MOT community.

We conduct comprehensive experiments on MOT17 [48], MOT20 [18], and KITTI [22] datasets, demonstrating the robustness of our method in low-frame-rate cases especially in preserving identities of objects involving occlusions. Our method achieves consistent improvement in IDF1 [56] score and AssA [43] compared to the baseline [81] and outperforms the state-of-the-art methods in low-frame-rate cases. To summarize, our contributions are as follows.

- We study the challenges of tracking multiple objects in low-frame-rate videos and design an APP head accompanied by a novel augmentation strategy to robustly detect unreliable displacement predictions due to the visibility flips of objects in low-frame-rate videos.
- The idea of binary APP prediction is extended to fine-grained displacement uncertainty by employing Bayesian deep learning tools.
- An APP-conditioned multi-stage matching policy is proposed that leverages vision cues and motion cues. It is further extended to fuse displacement uncertainty based on the rethink of the two bipartite matching algorithms.
- Experimental results on public datasets in various low-frame-rate settings demonstrate the effectiveness and robustness of the proposed method.

2 Related Work

Most modern multi-object tracking methods follow the tracking-by-detection paradigm. These methods can be roughly grouped into online ones [24,30,25,37,83,67,76,68,14], which extend the tracklet at each time step, and offline ones [17,7,75,44,31], where the tracklets are updated after processing a batch of frames. In the tracking-by-detection paradigm, an object detector [53,21,54,82] is firstly adopted to find all objects. The association of detections across different frames is commonly modeled as a bipartite graph matching problem, which can be solved using Hungarian matching [33] or greedy matching [81]. It has also been formulated as a graph optimization problem where each detection is taken as a graph node [7,6,46,25]. Motion cues, appearance cues, and their combinations are commonly adopted during association.

Motion cues are usually captured by motion models, which can be classified as filter-based ones and model-based ones. For example, Kalman filter [27] is a classic filter-based motion model widely used in MOT [5,67,77,76], often in conjunction with the constant velocity assumption. To deal with unexpected camera motion, Yoon et al. [69] exploit the structural constraints between objects, and Bergmann et al. [3] align frames through image registration [19] in the image preprocessing stage. Model-based methods work in a data-driven manner. In addition to the historical motion of individual targets, these methods also model the interactions between targets to enhance trajectory prediction [1,57,46,51]. However, a standalone motion model

does not provide robust association in low-frame-rate cases due to the large and crossed displacements of targets between adjacent frames. Some motion methods also involve visual inputs in motion predictions. Tracktor++ [3] takes the bounding box from the previous frame as a proposal for the current frame and refines it using the regression head [54] to achieve identity propagation. It assumes a high overlap between objects in adjacent frames, which does not hold in low-frame-rate cases. CenterTrack [81] looks at two adjacent frames and estimates the displacement of each target, similar to sparse optical-flow estimation. Its effectiveness decreases when the target is not continuously visible.

Appearance provides rich visual cues in MOT. For example, DeepSORT [67] and POI [71] adopt separate appearance embedding models to help the association. However, the separate training and inference of the detector and appearance embedding model results in high computational costs and suboptimal performance. To reduce the computational cost, JDE [66] and FairMOT [77] make efforts to accomplish the detection and appearance embedding with a single model. JDE [66] uses the architecture of the Feature Pyramid Network (FPN) [38] and outputs a dense prediction map containing detection information and appearance embeddings. FairMOT [77] adopts an encoder-decoder network [72] as the backbone and studies the fairness of the detection task and the appearance embedding task. Recently, Quasi-Dense [49] and MTrack [70] involve contrastive learning to enhance the discrimination of appearance representations.

To achieve better fusion and interaction between appearance and motion, some methods [25,7,6,46] model the association problem via graph neural networks. Recently, facilitated by query-based object detectors [12, 84], some methods [60,47,73,78] involve the attention mechanism, tracking multiple objects by query propagation. For example, MOTR [73] and Trackformer [47] extend the architecture of Deformable DETR [84] by involving track queries. Each query implicitly and jointly encodes the appearance and motion information of a tracked instance. These queries are used to detect the same object in subsequent frames. By adopting the anchor formulation of queries, MOTRv2 [78] further benefits from the pre-trained object detector YOLOX [21]. Generally, the attention mechanism yields better temporal and spatial modeling but leads to higher computational overhead.

Occlusion is a long-standing challenge in MOT. Partial occlusion results in low detection confidence and perturbed appearance embeddings. To address it, ByteTrack [76], different from most tracking-by-detection methods which only associate detections whose confidence is higher than a threshold, proposes to associate low-confidence detections with motion cues. MTrack [70] enables the network to adaptively sample appearance embeddings from the unoccluded regions of the target. Due to the dynamic environment, objects undergo a large appearance variation in a long-period extreme occlusion [51]. When re-activating a track after long-term occlusion, OC-SORT [9] proposes a re-update stage that uses virtual observations on the occluded time steps to prevent error accumulation. Close to our approach, Tokmakov et al. [64] extend the CenterTrack [81] architecture from pairs of frames as input to arbitrary video sequences by injecting a convolutional gated recurrent unit (ConvGRU) [2], so that enhancing the tracking of occluded targets. However, as pointed out by the author [64], their model is data-hungry and the receptive field is limited by the depth of the ConvGRU, which may limit its robustness in low-frame-rate cases. In low-frame-rate videos, the visibility of objects between adjacent frames changes more frequently and dramatically, introducing greater noise in the association process. To address it, we propose to identify newly appeared objects and adopt a multi-stage matching strategy conditioned on whether the object is a newly appeared one.

Tracking objects at low frame rates was studied [36, 15,74] before the era of deep learning and starts to receive attention [20,42] again recently. Early works [36, 15,74] made efforts to manually construct discriminative features and improve association modules (like particle filter [26]) to address quick variations in appearance and motion. Recent work, FraMOT [20] engages frame rate cues during association and introduces a periodic training scheme to reduce the impact of the frame rate gap between training and inference. ColTrack [42] follows the query-based MOT framework [73] to handle large displacement and explores adopting more historical queries to improve query quality. Our method captures large displacement in a flow-based manner [81] and learns to identify the reliability of flow estimates, thereby enabling switching between visual cues and historical motion cues.

Compared with improving model efficiency [10,11], the primary motivation behind studying low-frame-rate tracking is the unavailability of normal-frame-rate videos. Consider a scenario where videos are captured by massive edge cameras and transmitted to a central computing node for inference. Reducing the frame rate of the video sequence can reduce computational, communication, and storage costs, but introduces additional challenges as analyzed above. Tackling these challenges constitutes the motivation behind our study.

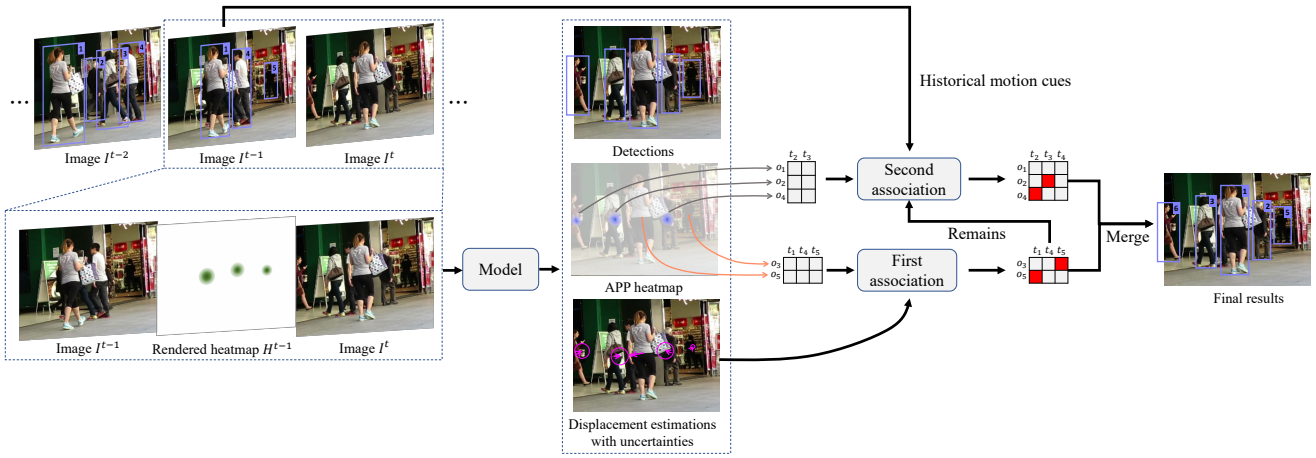


Fig. 2: Overview of our method. Our model takes the same input set as [81], outputting detections, an APP heatmap, and displacement estimations with uncertainties. APP predictions group the detections in frame I^t into emerging ones $\{o_1, o_2, o_4\}$ and non-emerging ones $\{o_3, o_5\}$. Note that the displacement estimations of emerging objects are not reliable and have high uncertainties. The association is performed in a two-stage manner. First, we associate non-emerging detections $\{o_3, o_5\}$ to tracklets $\{t_1, t_4, t_5\}$ visible in frame I^{t-1} by warping detection centers through their displacement estimations. Second, remaining tracklets $\{t_2, t_3, t_4\}$ are associated to remaining detections $\{o_1, o_2, o_4\}$ by extending tracklets through their historical velocity estimations.

3 Proposed Method

Given a sequence of video frames $\{I^1, \cdots, I^i, \cdots, I^n\}$, where $I^i \in \mathbb{R}^{W \times H \times 3}$, the multi-object tracking task is to detect all objects of interest and to assign a unique and consistent identity to each object across frames, outputting a set of tracklets $\mathbb{L} = \{l_1^{t_{s_1}:t_{e_1}}, l_2^{t_{s_2}:t_{e_2}}, \cdots\}$. Each tracklet $l_j^{t_{s_j}:t_{e_j}}$ contains a set of bounding boxes, starting at the time step t_{s_j} and terminating at t_{e_j}. Following [81], we define the notations under a local perspective for simplicity. Specifically, we use $\mathbb{L}^{t-1} = \{l_1^{t-1}, l_2^{t-1}, \cdots\}$ to denote tracklet set in the time step $t-1$, where each tracklet $l_j^{t-1} = (\boldsymbol{p}, \boldsymbol{s}, \boldsymbol{v}, w, id)$ is described by its center location $\boldsymbol{p} \in \mathbb{R}^2$, size $\boldsymbol{s} \in \mathbb{R}^4$ (for $left, top, right, bottom$), velocity estimation $\boldsymbol{v} \in \mathbb{R}^2$, detection confidence $w \in [0, 1]$, and unique identity $id \in \mathbb{I}$. Given the tracklet set $\mathbb{L}^{t-1}$ and the t^{th} frame I^t, we first detect the objects $\mathbb{O}^t = \{o_1^t, o_2^t, \cdots\}$ in the frame I^t, and then determine their identities by associating them with the tracklet set $\mathbb{L}^{t-1}$.

To address the local nature of CenterTrack [81], we propose a solution with the following advances. Fig. 2 shows an overview of our method and Fig. 3 shows the network architecture of our model. Firstly, we build an APP head on top of the CenterTrack architecture [81] to recognize unreliable displacement estimations in Sec. 3.1. Further, the idea of binary APP prediction is extended to fine-grained displacement uncertainty in Sec. 3.2. To improve the robustness of APP predictions, we propose a new augmentation strategy that enables the APP head to be pre-trained together with the rest of the model on static image data in Sec. 3.3. Finally, to reduce the influence of unreliable vision cues and to leverage historical motion cues, detections are associated with tracklets in a multi-stage manner conditioned on APP and uncertainty predictions in Sec. 3.4.

3.1 Learning to Recognize Appearing Objects

We build our method on top of the CenterTrack [81] architecture. CenterTrack takes two adjacent frames $\{I^{t-1}, I^t\}$ together with a heatmap H^{t-1} rendered from the object centers in frame I^{t-1} as input. The output consists of a heatmap $\hat{Y}^t \in [0, 1]^{\frac{W}{R} \times \frac{H}{R} \times 1}$, an offset map $\hat{F}^t \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$, a size map $\hat{S}^t \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 4}$, and a displacement map $\hat{D}^t \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$. $R = 4$ is the downsampling factor. The center location $\boldsymbol{p}^t \in \mathbb{R}^2$, box size $\boldsymbol{s}^t \in \mathbb{R}^4$ (for $left, top, right, bottom$), and corresponding displacement $\boldsymbol{d}^t \in \mathbb{R}^2$ of each detection o_i^t is then decoded from these maps. The displacement estimations $\hat{\boldsymbol{d}}^t$ are used to warp detection centers $\hat{\boldsymbol{p}}^t$ in I^t to their locations in I^{t-1} by calculating $\hat{\boldsymbol{p}}^t + \hat{\boldsymbol{d}}^t$. The detections $\mathbb{O}^t$ are then linked to tracklets $\mathbb{L}^{t-1}$ by matching the warped detection centers with the tracklet centers, resulting in $\mathbb{L}^t$.

However, in the training stage of [81], the displacement estimation head is only supervised by objects that are highly visible in both adjacent frames I^{t-1} and I^t. Given this, in the inference stage, the displacement es-

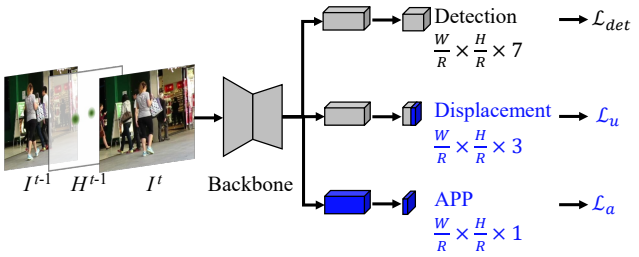


Fig. 3: The network architecture of APPTracker+. Modules inherited from CenterTrack are indicated in gray, while the difference is highlighted in blue. We merge the representation of the detection branch for simplicity.

timations corresponding to objects that newly appear in the current frame I^t are unreliable. Such unreliable estimations increase the probability of incorrect associations.

To bridge the gap between training and inferring, we propose to recognize unreliable displacement estimations by detecting objects newly appearing in the current frame. Intuitively, it is akin to detecting anomalies of optical flows. To achieve it, we build an APP head, in parallel with other detection and displacement estimation heads.

We denote the output of the APP head as $\hat{A}^t \in [0,1]^{\frac{W}{R}\times\frac{H}{R}\times 1}$. The ground-truth APP heatmap A^t is generated by rendering Gaussian-shaped peaks [34] into A^t at the centers of emerging objects in frame t. Some visualized cases are shown in Fig. 4. Given a ground-truth heatmap A^t, we supervise the APP head with a training objective based on the focal loss [39,34]:

$$\mathcal{L}_a = \frac{1}{N}\sum_{xy}\begin{cases}(1-\hat{A}^t_{xy})^{\alpha} log(\hat{A}^t_{xy}) & \text{if } A^t_{xy}=1\\ (1-A^t_{xy})^{\beta}(\hat{A}^t_{xy})^{\alpha} log(1-\hat{A}^t_{xy}) & \text{otherwise}\end{cases}, \tag{1}$$

where x and y enumerate all pixels in frame I^t, N is the number of emerging objects, and $\alpha=2$ and $\beta=4$ are hyperparameters of the focal loss.

During the inferring stage, a detection o^t is regarded as an emerging one if its APP score a^t is greater than a threshold τ_{ap}. Its displacement estimation $\mathbf{d}^t$ is dropped as it is not reliable.

3.2 Displacement Uncertainty

The proposed APP head formulates the unreliability recognition task as a binary classification task, classifying detections into emerging ones and non-emerging ones. It assumes that displacement estimations for any non-emerging detections are equally reliable. This coarse assumption is improved in this section.

To obtain a fine-grained unreliability score (also named as "uncertainty score" blow) for each displacement estimation, we take the idea of heteroscedastic regression [35,29] and treat each displacement as a random variable under the Gaussian distribution. The displacement estimation task is no longer to regress deterministic values but to estimate the parameters of individual Gaussian distributions. Specifically, for a detection o_i, its displacement distribution can be represented as $p(\boldsymbol{d}_i)=\mathcal{N}(\boldsymbol{\mu}_i,\sigma_i^2)$, where $\boldsymbol{\mu}_i=[d_x,d_y]$ contains 2-axis displacements d_x and d_y. We let the 2-axis displacements share the same σ_i for simplicity. To this end, the original 2-channel displacement output $\hat{D}^t\in\mathbb{R}^{\frac{W}{R}\times\frac{H}{R}\times 2}$ is extended to a 3-channel one $\hat{D}^t\in\mathbb{R}^{\frac{W}{R}\times\frac{H}{R}\times 3}$ where each channel correspondingly represents d_x, d_y, and σ.

Given a ground-truth displacement $\boldsymbol{d}_i$ and its estimated distribution parameters $(\hat{\boldsymbol{\mu}}_i,\hat{\sigma}_i)$, the objective of the reformulated displacement estimation task is to maximize the likelihood

$$p(\boldsymbol{d}_i|\hat{\boldsymbol{\mu}}_i,\hat{\sigma}_i)=\frac{1}{\sqrt{2\pi}\hat{\sigma}_i}e^{-\frac{(\boldsymbol{d}_i-\hat{\boldsymbol{\mu}}_i)^2}{2\hat{\sigma}_i^2}}. \tag{2}$$

By taking the negative logarithm form of Eq. 2 and neglecting constant items, the training loss of the new displacement estimation task is designed as

$$\mathcal{L}_u=\frac{1}{N}\sum_{i=1}^{N}\frac{1}{2\hat{\sigma}_i^2}\|\boldsymbol{d}_i-\hat{\boldsymbol{\mu}}_i\|^2+\frac{1}{2}\log\hat{\sigma}_i^2, \tag{3}$$

where N is the number of instances, and $\|\cdot\|^2$ represents the L_2 norm.

With the above formulation, the parameter $\hat{\sigma}$, also named "displacement uncertainty", represents the fine-grained unreliability of displacement estimations. Learning such displacement uncertainty does not require explicit labels. By setting the derivative of $\mathcal{L}_u$ with respect to $\hat{\sigma}$ to zero, we find that the optimum value of $\mathcal{L}_u$ is achieved when $\hat{\sigma}^2=\|\boldsymbol{d}_i-\hat{\boldsymbol{\mu}}_i\|^2$. This means that the loss $\mathcal{L}_u$ enables the uncertainty $\hat{\sigma}$ to capture the residual between the ground truth $\boldsymbol{d}_i$ and the estimated one $\hat{\boldsymbol{\mu}}_i$. To make $\hat{\sigma}$ positive, we apply exponential mapping to the output of the uncertainty channel.

We name the APPTracker with the reformulated displacement estimation head as "APPTracker+". The overall training objective of APPTracker+ is:

$$\mathcal{L}=\lambda_a\mathcal{L}_a+\lambda_u\mathcal{L}_u+\mathcal{L}_{det}, \tag{4}$$

where $\mathcal{L}_{det}=\lambda_p\mathcal{L}_p+\lambda_s\mathcal{L}_s+\lambda_{off}\mathcal{L}_{off}$ is the training objective of the detection task inherited from original CenterTrack [81]. $\lambda_a,\lambda_u,\lambda_p,\lambda_s,\lambda_{off}$ are hyper-parameters used to balance the contribution of each loss component.

(a) (b) (c) (d) (e) (f)

Fig. 4: Training samples. (a) and (b) are a pair of annotated frames I^{t-1}, I^t sampled from the MOT17 dataset. Noisy visibility annotations can be observed on the person inside the blue box in (b). (c) is the corresponding ground-truth APP heatmap A^t. When training on static image data, the previous frame (d) is a replica of an original image (e) with augmentations including erasing, scaling, and translating. The corresponding ground-truth APP heatmap A^t is shown in (f).

3.3 Training Strategies

3.3.1 Training on Videos

MOT17 [48] and MOT20 [18] are commonly used MOT benchmarks. They annotate the identity, bounding box, class, and visibility for each object of interest. Given an object i in frame I^t, we determine its appear label a_i^t by

$$a_i^t = \begin{cases} 1 & \text{if } vis_i^t > \tau_{vis} \text{ and } vis_i^{t-1} < \tau_{vis} \\ 0 & \text{otherwise} \end{cases}, \tag{5}$$

where τ_{vis} is the threshold above which an object is regarded as a visible one, vis_i^t and vis_i^{t-1} are the visibility annotations of the object i in frame t and $t-1$, respectively. After the label a_i^t is determined, we use the rendering function in [34] to render the ground-truth heatmap A^t.

However, the visibility annotations provided by MOT17 and MOT20 are noisy. The visibility of an object is determined based on the intersection-over-union (IoU) of its bounding box against other bounding boxes. As shown in Fig. 4b (the man inside the blue box), such box-level visibility annotation may lead to noisy supervision for detection and APP prediction. To address it, we ignore the predictions for objects annotated as low visibility by placing a circular mask on the ground-truth detection heatmap Y^t and the ground-truth APP heatmap A^t (shown in Fig. 4c). Specifically, the prediction at (x, y) is ignored if $exp(-\frac{(x-x_i)^2+(y-y_i)^2}{2\sigma_i^2}) > \tau$, where (x_i, y_i) is the ground-truth center location of a low-visible object i, $\tau \in [0, 1]$ is a hyperparameter, and σ_i is a function w.r.t. the object size [34]. A smaller value of τ leads to a mask with a larger radius, which better excludes outliers but may impact surrounding objects. We set $\tau = 0.3$ in experiments.

3.3.2 Training on Static Images

Training the APP head on MOT17 alone cannot provide robust APP predictions as the ratio of emerging samples is fairly low, typically less than 10% among all instances. In CenterTrack [81], a static image is first duplicated and then randomly scaled and translated to create a fake adjacent frame pair so that the network can be pre-trained on static image datasets. Inspired by this, we propose a novel augmentation strategy, random erasing, to pretrain the APP head simultaneously with other heads on static image data. Specifically, a visible object in the original image is erased by placing a background patch on it. The background patch is randomly sampled from the background region within the same image. To avoid ambiguity in optical flow, the patch should look different from the region where it is sampled from. Thus, we sample two different background patches bg_1 and bg_2, then a new patch bg_{mixed} is generate by blending bg_1 and bg_2:

$$bg_{mixed} = \gamma bg_1 + (1-\gamma) bg_2, \tag{6}$$

where γ is randomly sampled from a uniform distribution $\mathcal{U}(0.3, 0.7)$. Several other augmentations, such as flipping, intensity jittering, and color channel shuffling are further applied on bg_{mixed}. Fig. 4d to Fig. 4f visualize an example generated by the proposed random erasing strategy.

3.3.3 Supervising Each Displacement Estimation

To prevent unreasonable supervision signals, CenterTrack [81] only places supervision on displacement estimations of continuously visible objects. However, by introducing uncertainty, the network can reduce the contribution of outliers to the loss by assigning them high uncertainty scores. That is, the model becomes more robust to outliers. Facilitated by this, APPTracker+

supervises displacement estimation of each object visible in frame I^t, even if it was occluded in frame I^{t-1}. It promotes the utilization of all instances rather than discarding low-visibility ones. When training on MOT datasets [48,18], in case a target is out of view in frame I^{t-1}, we fabricate a "virtual detection" for it to supplement the displacement supervision. Specifically, we compute the distance from the target's center to the image's four boundaries, and the boundary with the shortest distance is considered the entry boundary. A virtual detection in frame I^{t-1} is fabricated by shifting the real detection in frame I^t out of that boundary.

Another advantage of introducing uncertainty is that it eliminates the dependence on explicit visibility labels. This makes the uncertainty estimation usable even when trajectories are not fully annotated throughout its lifecycle, while the APP prediction branch cannot be trained in such cases. It is validated by our evaluations on KITTI [22] without visibility labels.

Algorithm 1 Multi-stage Matching.

Data: The emerging detection set $\mathbb{O}_{ap}^t$, non-emerging detection set $\mathbb{O}_{ac}^t$, active tracklet set $\mathbb{L}_{ac}^{t-1}$, inactive tracklet set $\mathbb{L}_{inac}^{t-1}$
Result: The tracklet set $\mathbb{L}^t$

1 Initialization: $\mathbb{L}^t \leftarrow \emptyset$

/* Stage 1: association with displacements */
2 $\mathbb{M}_1, \mathbb{O}_{rem1}^t, \mathbb{L}_{rem1}^{t-1} \leftarrow$ Hybrid_Match($\mathbb{O}_{ac}^t$, $\mathbb{L}_{ac}^{t-1}$)
3 **for** (o, l) *in* $\mathbb{M}_1$ **do**
4 | $l.location \leftarrow o.location$
5 | $l.velocity \leftarrow o.displacement$
6 | $\mathbb{L}^t \leftarrow \mathbb{L}^t \cup \{l\}$
7 **end**

/* Stage 2: association with velocities */
8 $\mathbb{M}_2, \mathbb{O}_{rem2}^t, \mathbb{L}_{rem2}^{t-1} \leftarrow$ Hungarian_Match($\mathbb{O}_{ap}^t \cup \mathbb{O}_{rem1}^t$, $\mathbb{L}_{inac}^{t-1} \cup \mathbb{L}_{rem1}^{t-1}$)
9 **for** (o, l) *in* $\mathbb{M}_2$ **do**
10 | $l.location \leftarrow o.location$
11 | $l.velocity \leftarrow$ *mean displacement*
12 | $\mathbb{L}^t \leftarrow \mathbb{L}^t \cup \{l\}$
13 **end**

/* update inactive tracklets */
14 **for** l *in* $\mathbb{L}_{rem2}^{t-1}$ **do**
15 | $l.location \leftarrow l.location + l.velocity$
16 | $\mathbb{L}^t \leftarrow \mathbb{L}^t \cup \{l\}$
17 **end**

/* initialize new tracklets */
18 **for** o *in* $\mathbb{O}_{rem2}^t$ **do**
19 | $o.velocity \leftarrow 0$
20 | $\mathbb{L}^t \leftarrow \mathbb{L}^t \cup \{o\}$
21 **end**

3.4 Multi-stage Association

With APP predictions and displacement uncertainties, we propose to associate detections and tracklets in a multi-stage manner. For clarity, in Sec. 3.4.1, we first introduce how the APP prediction determines the switch between using visual cues or motion cues during association. Further, in Sec. 3.4.2, we conduct a rethink of the commonly used bipartite graph matching algorithms and propose an uncertainty-conditioned hybrid matching policy in Sec. 3.4.3.

3.4.1 Two-stage Association Conditioned on APP

To reduce the noise in the association matrix resulting from unreliable vision cues and to leverage historical motion cues, we introduce a two-stage matching strategy conditioned on APP predictions, whose pseudocode is shown in Algorithm 1.

At each time step t, before matching, we group the detections $\mathbb{O}^t$ into emerging ones $\mathbb{O}_{ap}^t$ and non-emerging ones $\mathbb{O}_{ac}^t$ by comparing their APP scores with a threshold τ_{ap}. Meanwhile, the tracklet set $\mathbb{L}^{t-1}$ is grouped into two subsets, active tracklets $\mathbb{L}_{ac}^{t-1}$ and inactive ones $\mathbb{L}_{inac}^{t-1}$ according to whether the tracklets were visible in frame $t-1$.

In the first stage (lines 2-7), we link non-emerging detections $\mathbb{O}_{ac}^t$ to active tracklets $\mathbb{L}_{ac}^{t-1}$. Center locations of detections $\mathbb{O}_{ac}^t$ are warped by their displacement estimations. The warped detection centers are then matched with centers of tracklets $\mathbb{L}_{ac}^{t-1}$ by hybrid matching (detailed in Sec. 3.4.3), taking the Euclidean distance as the similarity measurement. We use $\mathbb{M}_1, \mathbb{O}_{rem1}^t, \mathbb{L}_{rem1}^{t-1}$ to denote matched detection-tracklet pairs, remaining detections, and remaining tracklets, respectively. Then each matched tracklets l is updated by the matched detection o (lines 3-7). Note that the velocities of matched tracklets are updated by the displacement estimations.

In the second stage (lines 8-13), we link the remaining tracklets ($\mathbb{L}_{inac}^{t-1} \cup \mathbb{L}_{rem1}^{t-1}$) to the remaining detections ($\mathbb{O}_{ap}^t \cup \mathbb{O}_{rem1}^t$). In this stage, we warp tracklets ($\mathbb{L}_{inac}^{t-1} \cup \mathbb{L}_{rem1}^{t-1}$) by their velocity estimations to match with detections. In case an inactive tracklet is matched in this stage, it is re-birthed and its velocity is updated by the mean displacement during the invisible period.

Finally, we update the locations of the remained tracklets $\mathbb{L}_{rem2}^{t-1}$ with a constant velocity assumption (lines 14-17) and spawn new tracklets for remaining detections $\mathbb{O}_{rem2}^t$ with zero velocities (lines 18-21).

3.4.2 Rethink of Bipartite Graph Matching Algorithms

Assigning new detections to historical tracklets is known as a bipartite graph matching problem. In modern MOT methods, this is commonly solved by Hungarian matching [33] or sometimes by the greedy assignment [81]. Specifically, the Hungarian matching minimizes the global matching cost while the greedy assignment iteratively determines the assignment that minimizes the local cost. Due to the detection noise (caused by occlusions and occasional detection errors) and association noise (like unreliable displacement estimations), it is hard to assert which matching algorithm works better for the MOT task.

In this section, we conduct several oracle comparisons between the two matching algorithms where we decouple the detection noise from the association noise[1].

Specifically, given ground-truth detections and ground-truth displacements, we first simulate detection noise by randomly discarding detections[2]. As shown in Fig. 5a, greedy matching consistently outperforms Hungarian matching. It reveals that greedy matching is better at handling discrete detection noise (e.g., missed detections) because greedy matching is more robust to outliers.

Secondly, we simulate association noise by injecting a random perturbation $\tilde{\boldsymbol{\mu}}_i = [\tilde{dx}, \tilde{dy}] = [r\cos\theta, r\sin\theta]$ to each ground-truth displacement $[dx, dy]$, where r is a given radius and θ is randomly sampled from a uniform distribution $\mathcal{U}(0, 2\pi)$. No ground-truth detections are discarded this time. Fig. 5b plots the tracking performance under different noise radius settings. We find that Hungarian matching outperforms greedy matching this time, revealing that Hungarian matching is more adept at dealing with association noise of inliers (e.g., small errors in displacement estimates).

Finally, to compare their performance in the presence of both detection noise and association noise, the third oracle experiment is conducted using the detection and displacements estimated by APPTracker. Again, we inject the random perturbation $\tilde{\boldsymbol{\mu}}_i = [\tilde{dx}, \tilde{dy}] = [r\cos\theta, r\sin\theta]$ into each displacement estimation $\hat{\boldsymbol{\mu}}_i = [\hat{dx}, \hat{dy}]$. A performance intersection can be observed in Fig. 5, which inspires us to develop the uncertainty-conditioned hybrid matching strategy detailed in Sec. 3.4.3.

[1] All the oracle experiments are conducted on the MOT17 dataset with a 1/10 frame rate.

[2] To eliminate association noise, we provide ground-truth displacements for all detections, even if they were discarded in the previous frame.

Algorithm 2 Hybrid Matching.

Function `Hybrid_Matching`($\mathbb{O}_{ac}^{t}, \mathbb{L}_{ac}^{t-1}$)**:**

Data: The non-emerging detection set $\mathbb{O}_{ac}^{t}$ and active tracklet set $\mathbb{L}_{ac}^{t-1}$

Result: Matched pairs $\mathbb{M}_1$, remaining detections $\mathbb{O}_{rem1}^{t}$, and remaining tracklets $\mathbb{L}_{rem1}^{t-1}$

Set uncertainty threshold τ_u

$\mathbb{O}_{ac\_low}^{t}, \mathbb{O}_{ac\_high}^{t} \leftarrow$ split $\mathbb{O}_{ac}^{t}$ by τ_u

$\mathbb{M}_G, \mathbb{O}_{rem0}^{t}, \mathbb{L}_{rem0}^{t-1} \leftarrow$ Greedy_Match($\mathbb{O}_{ac\_low}^{t}$, $\mathbb{L}_{ac}^{t-1}$)

$\mathbb{M}_H, \mathbb{O}_{rem1}^{t}, \mathbb{L}_{rem1}^{t-1} \leftarrow$ Hungarian_Match($\mathbb{O}_{ac\_high}^{t} \cup \mathbb{O}_{rem0}^{t}$, $\mathbb{L}_{rem0}^{t-1}$)

$\mathbb{M}_1 \leftarrow \mathbb{M}_G \cup \mathbb{M}_H$

return $\mathbb{M}_1, \mathbb{O}_{rem1}^{t}, \mathbb{L}_{rem1}^{t-1}$

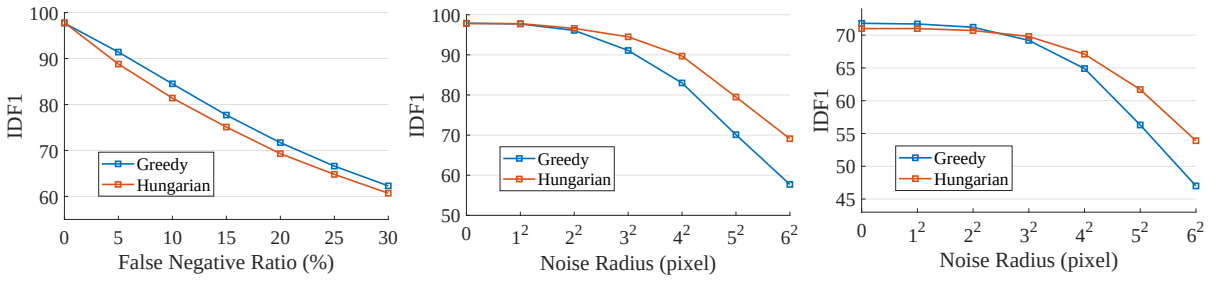
3.4.3 Hybrid Matching Conditioned on Uncertainty

The oracle comparisons elaborated in Sec. 3.4.2 imply that hybridizing the greedy matching and the Hungarian matching may lead to better tracking performance. However, selecting the appropriate detection subsets for each matching strategy is challenging because it may filter out correct assignments. We find that the fine-grained displacement uncertainty can be taken as a good indicator. According to the oracle experiment in Fig. 5c, we empirically split the non-emerging detection set $\mathbb{O}_{ac}^{t}$ into two parts, $\mathbb{O}_{ac\_low}^{t}$ and $\mathbb{O}_{ac\_high}^{t}$, with an uncertainty threshold $\tau_u = 8$ by default. Then we sequentially perform greedy matching and Hungarian matching. This process is specified in Algorithm 2.

4 Experiments

4.1 Datasets and Evaluation Metrics

MOT. We mainly use two widely-used benchmarks, MOT17 [48] and MOT20 [18], to conduct the experiments. MOT17 contains 7 training sequences and 7 testing sequences. The videos are captured by stationary or moving cameras from various viewpoints. The video frame rate varies from 14 to 30 FPS. MOT20 contains 4 training sequences and 4 testing sequences of crowded scenes captured by stationary cameras at 25 FPS. Besides, we pretrain our model on the CrowdHuman dataset[58]. Following common practices in [81, 77, 76], we split each training sequence in MOT17 and MOT20 into two halves for experiments, and use the first half for training and the rest for validation. We generate low-frame-rate videos by extracting frames from original videos through fixed intervals. The intervals will be specified in each experiment. For a given original video and a given sampling interval n_d, we generate n_d low-frame-rate videos with the first frame index varying from 0 to $n_d - 1$. We use $n_d = 10$ throughout the



(a) (b) (c)

Fig. 5: Oracle comparison between the greedy matching and the Hungarian matching. (a) Perturbated ground-truth detection & clean ground-truth displacement. (b) Clean ground-truth detection & perturbated ground-truth displacement. (c) Clean estimated detection & perturbated estimated displacement.

experiments on MOT17 and MOT20 unless mentioned otherwise.

KITTI. We also evaluate our method on the KITTI [22] 2D MOT benchmark. The benchmark comprises 29 test sequences with hidden annotations, totaling over 10,000 frames. Videos were captured at a frame rate of 10 FPS using a monocular camera mounted on a moving vehicle. We submit our predicted results to the official benchmark where we do not downsample the frame rate of videos.

Evaluation Metrics. We use the IDF1 score [56] and the CLEAR metrics [4], including MOT Accuracy (MOTA), mostly tracked (MT), mostly lost (ML), false positives (FP), false negatives (FN), and the number of identity switches (IDs). An object is regarded as a mostly tracked (MT) one if it is tracked for at least 80% of its life span (whether the assigned identity changes or not). Instead, if an object is tracked for less than 20% of its life span, it is regarded as a mostly lost (ML) one. Compared to our previous conference paper [80], we update the results with a recent metric, HOTA [43], where DetA evaluates the detection performance, AssA evaluates the association performance, and HOTA is the geometric mean of DetA and AssA.

4.2 Implementation Details

For the components of our model shared with CenterTrack [81], we follow their architecture and training details. Specifically, we use DLA-34 [72] as the network backbone. We adopt the Adam [32] optimizer with an initial learning rate of $1.25e-4$ and a batch size of 32. The loss weights are set as $\lambda_a = 1$, $\lambda_u = 1$, $\lambda_p = 1$, $\lambda_s = 0.1$, and $\lambda_{off} = 1$.

For experiments on MOT17 [48] and MOT20 [18], the model is pre-trained on CrowdHuman [58] for 180 epochs with the input resolution of 512×512. The learning rate decays to $1.25e-5$ at the 140^{th} epoch. We adopt the random erasing operation proposed in Sec. 3.3.2 during pre-training. In practice, erasing an individual object within a crowd may destroy surrounding objects. To minimize such impact, we first check the box-level overlap between objects. An object is allowed to be erased only if it overlaps with surrounding objects slightly (less than 15%). Among all instances, about 12% visible objects are erased. We fine-tune the model on MOT17 [48] (or MOT20 [18]) by 90 epochs with the input resolution of 960×544. The learning rate decays to $1.25e-5$ at the 60^{th} epoch. The erasing operation is disabled during finetuning. Frame pairs are randomly sampled within a maximum frame gap of 30, where pairs in temporally reverse order are allowed. We set the visibility threshold $\tau_{vis} = 0.25$ during training and the APP threshold $\tau_{ap} = 0.4$ during inferring.

For experiments on KITTI [22], we adopt an identical training recipe as our baseline method CenterTrack [81]. Specifically, we use the pre-trained vehicle detection module released by CenterTrack, which was pre-trained on nuScenes [8] for 140 epochs and we fine-tune it on the KITTI training set for another 70 epochs. The uncertainty-based displacement estimation branch was trained from scratch on the KITTI training set. In KITTI, not every target is annotated throughout its lifecycle. This introduces difficulties in training the binary APP prediction branch because it requires explicit labels determined by the visibility changes of the targets across adjacent frames. Therefore, when conducting experiments on KITTI, we remove the APP prediction branch but retain the uncertainty estimation. With the absence of the APP branch, we cannot determine whether a target is continuously visible. In contrast to using $\tau_u = 8$ in MOT17 and MOT20, we adopt a more conservative uncertainty threshold of $\tau_u = 20$ for KITTI to split the two stages of hybrid matching. This makes the tracker inclined towards greedy matching, which has been observed in Sec. 3.4.2 to better handle changes in detection entries.

Table 1: Comparison between several model variants on the MOT17 validation set (1/10-frame-rate). "inv" represents deriving DISAPP predictions by temporally flipping the order of the input frames.

APP	DISAPP	HOTA↑	DetA↑	AssA↑	IDF1↑	IDs(%)↓	FP(%)↓	FN(%)↓	MOTA↑
-	-	53.4	55.8	51.6	64.7	4.5	3.0	27.9	64.6
✓	-	54.2	55.9	53.0	65.9	**3.9**	3.0	27.9	**65.2**
-	✓	53.6	55.8	52.0	65.1	4.7	3.0	27.9	64.4
-	✓(inv)	53.8	55.9	52.3	65.4	4.3	3.0	27.9	64.8
✓	✓(inv)	**54.3**	**56.0**	**53.2**	**66.0**	4.0	3.0	27.9	65.1

4.3 Appear Prediction

We first investigate the benefit of engaging the APP head in 1/10-frame-rate cases. By comparing the first two rows in Table 1, a reduction of about 13% on IDs (from 4.5% to 3.9%) is observed by using the appearing prediction and the multi-stage matching policy. Further, in Fig. 6, we highlight our improvement in preserving the identities of objects involved in occlusions.

Specifically, objects of interest are selected following two standards. 1) Before being occluded, the object should be continuously visible for at least two frames. This is the minimal sufficient time to estimate its velocity. 2) The visibility should flip at least twice, one for becoming occluded and the other for appearing again. In practice, we introduce two thresholds τ_{high} and τ_{low} to derive reliable visibility flipping counts. For an object i with the visible ratio $vis_i^{t_1} > \tau_{high}$ in frame I_{t_1}, and with $vis_i^{t_2} < \tau_{low}$ in a future frame I_{t_2}, its visibility is regarded to flip once, and verse visa. We set $\tau_{high} = 0.4$ and $\tau_{low} = 0.15$.

Fig. 6 plots the IDs metric values achieved by models with and without APP head in each validation video of MOT17 and MOT20 datasets. The models are trained on MOT17 and MOT20 individually. Besides, we use purple bars to indicate the ratio of visibility flips r_{vflip} calculated by

$$r_{vflip} = \frac{n_{vflip}}{n_{gt}}, \tag{7}$$

where n_{vflip} is the sum of visibility flips for all objects of interest, and n_{gt} is the number of all ground-truth bounding boxes. For better comparison, we normalize the IDs by the number of visibility flips. Formally, the normalized IDs r_{IDs} is calculated by

$$r_{IDs} = \frac{IDs}{n_{vflip}}, \tag{8}$$

where IDs is the count of identity switches of all objects of interest. r_{IDs} ranges from 0 to 2, since one incorrect association may lead to two IDs at the most (one target steals the identity of another one). Our approach shows improvement on most videos, especially on crowded ones, such as MOT17-09, MOT17-11, MOT20-01, and MOT20-02. For videos with stochastic and catastrophic camera motions, such as MOT17-05, MOT17-10, and MOT17-13, the constant velocity assumption fails to estimate correct motion even though the emerging objects are precisely recognized.

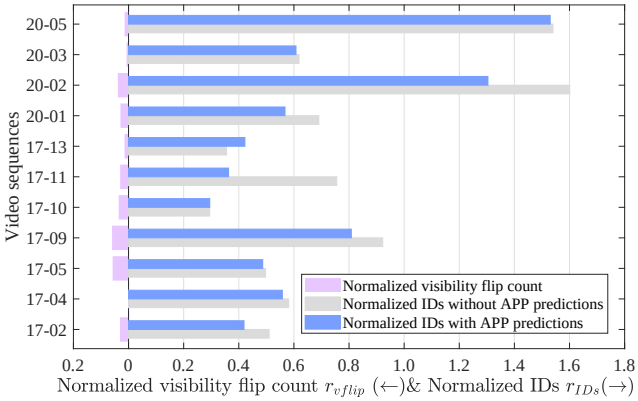
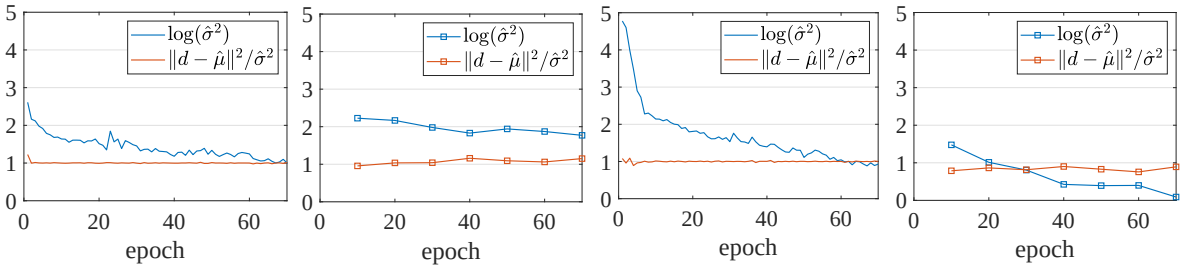


Fig. 6: The performance improvement on preserving identities of objects involved in occlusions.

Model Variants. Symmetrically to the APP head, we also study the benefit of adopting a "disappearing" prediction (DISAPP) head in Table 1. The DISAPP head is trained to recognize objects which are visible in the previous frame I^{t-1} but are invisible in the current frame I^t. However, as is stated below, how to obtain robust "disappearing" predictions is not that straightforward.

We first train a DISAPP head following the same settings as those for training the APP head. However, it leads to a higher IDs (third row in Table 1) than that of the baseline (first row in Table 1). The reason lies in that the information of the two input frames $\{I^{t-1}, I^t\}$ is not fairly utilized by the model. Specifically, before being accumulated and sent into the backbone network, I^t first goes through a 7×7 convolution layer whose weights are pre-trained together with the backbone network on the COCO dataset [40]. However, I^{t-1} first goes through another 7×7 convolution layer whose weights are randomly initialized.

To address this issue, we explore another way to derive "disappearing" predictions from the APP head by



(a) Train. on MOT17 [48] (b) Val. on MOT17 [48] (c) Train. on KITTI [22] (d) Val. on KITTI [22]

Fig. 7: Quantitative analysis of uncertainty estimation.

Table 2: Ablation study on the MOT17 validation set (1/10-frame-rate). "Aug." indicates the proposed data augmentation policy. "Hght." stands for height gating. "Thresh." represents using stage-specific thresholds in multi-stage matching.

APP	Aug.	Hght.	Thresh.	HOTA↑	DetA↑	AssA↑	IDF1↑	IDs(%)↓	FP(%)↓	FN(%)↓	MOTA↑
✓				54.2	**56.4**	52.6	65.1	4.4	3.1	**27.6**	65.0
✓	✓			54.2	55.9	53.0	65.9	3.9	3.0	27.9	65.2
✓	✓	✓		55.3	56.0	55.1	67.6	3.6	3.0	27.9	65.5
✓	✓		✓	56.7	56.2	57.8	69.7	3.4	2.9	27.9	65.8
✓	✓	✓	✓	**57.5**	56.3	**59.2**	**70.8**	**3.3**	**2.9**	27.9	**65.9**

temporally flipping the order of the input frames I^{t-1} and I^t. No re-training is required since we also sample frame pairs with inverse time orders during training. With such a strategy, the DISAPP predictions (the fourth row in Table 1) bring a reduction of about 4% on IDs (from 4.5% to 4.3%). By comparing the second and the fourth rows in Table 1, we can find the improvement engaged by the DISAPP head is not as remarkable as that engaged by the APP head. The reason is that the unreliable displacement estimations of emerging detections may lead them to compete for the tracklets against the correct detections, resulting in the identity switch twice. Instead, such competition is not involved in disappearing objects. As a result, we only keep the APP head in the following experiments, since engaging DISAPP predictions yields minor gains.

4.4 Uncertainty Estimation

In this section, we provide a quantitative analysis of uncertainty estimation. According to Eq. (3) in the main text, the displacement estimation loss $\mathcal{L}_u$ contains two items, $\frac{\|\boldsymbol{d}_i-\hat{\boldsymbol{u}}_i\|^2}{\hat{\sigma}_i^2}$ and $\log\hat{\sigma}_i^2$, for each instance i. As an illustration, we separately plot the curves of these two items during training the model on MOT17 [48] and KITTI [22]. As shown in Figure 7, the ratio $\frac{\|\boldsymbol{d}_i-\hat{\boldsymbol{u}}_i\|^2}{\hat{\sigma}_i^2}$ on the validation set of both dataset remains close to 1, indicating that the predicted uncertainty $\hat{\sigma}_i^2$ effectively captures the residual between the displacement estimation $\hat{\boldsymbol{u}}_i$ and the ground truth $\boldsymbol{d}_i$. Besides, from the validation loss, we can also observe a negative correlation between these two items. The validation loss may be smaller than the training loss. This is because there is data augmentation during the training phase, while it is not present during the validation phase.

4.5 Ablation Study

Sec. 4.3 has discussed some model variants and demonstrated the advancement of involving APP predictions without bells and whistles. In this subsection, we ablate other components of our designs. For clarity and fairness, experiments in Table 1, Table 2 and Table 3 do not involve uncertainty or uncertainty-conditioned hybrid matching (we use Hungarian matching instead). These two components are ablated in Table 4.

Random Erasing Augmentation. As shown in the first row in Table 2, the APP head trained only on the MOT17 dataset fails to generate robust predictions. The proposed data augmentation policy engages a reduction of about 11% in IDs (from 4.4% to 3.9%).

Height Gating. In addition to velocity, the size of the object also contains historical motion cues. While CenterTrack [81] drops the similarity of box sizes during association, we pick it up by rejecting matchings with unreasonable height changes. Formally, the height distance d_{ij}^h of a detection bounding box i and a tracklet

Table 3: Comparison of different motion models with different frame rate inputs on the MOT17 validation set. "Opt." stands for the optical-flow-based motion estimator.

Frame rate	Motion model	DetA↑	AssA↑	IDF1↑
Normal	Opt. + Static	56.6	58.7	70.8
	Opt. + Constant	**56.7**	57.2	69.2
	Kalman Filter	56.6	**60.0**	**72.0**
Low	Opt. + Static	56.2	58.6	70.3
	Opt. + Constant	**56.3**	**59.2**	**70.8**
	Kalman Filter	54.2	49.6	60.6

bounding box j is calculated by

$$d_{ij}^{h} = \frac{|h_i - h_j|}{max(h_i, h_j)}, \tag{9}$$

where h_i and h_j are the height of the detection bounding box i and that of the tracklet bounding box j, respectively. Assignments with a height distance larger than a threshold τ_h are rejected. We use $\tau_h = 0.33$ throughout but it can be fine-tuned according to different frame rates.

Stage-specific Matching Thresholds. Originally, we follow CenterTrack to reject an assignment (o_i^t, l_j^{t-1}) if the Euclidean distance between their centers (after warping) is larger than $\kappa = \sqrt{w_j^{t-1} h_j^{t-1}}$, where w_j^{t-1} and h_j^{t-1} are the width and height of tracklet l_j^{t-1}. To restrict the association quality in the first stage, a stage-specific threshold policy is adopted by setting the thresholds to $\frac{3}{5}\kappa$ and κ in the first and the second matching stages, respectively.

Motion Models. Table 3 compares the performance of the Kalman filter and the optical-flow-based displacement estimator with normal-frame-rate inputs and 1/10-frame-rate inputs. For the optical-flow-based estimator, we evaluate two motion models, a static model, and a constant velocity model. We have the following findings. First, the static model performs reasonably well in normal-frame-rate videos due to the little movement of the same target between frames. Second, though the Kalman filter gets a better IDF1 score of 72.0% in normal-frame-rate cases, it fails as the frame rate goes lower. We suspect that, in normal-frame-rate cases, motion estimations are perturbated by detection jitters. Kalman filter can filter out such noise while the constant velocity model is heavily influenced. However, it goes the opposite when the frame rate decreases. Without a good initial velocity estimation (the velocity of an object is usually initialized to zero), the Kalman filter fails to chase objects in low-frame-rate cases.

Uncertainty-Conditioned Hybrid Matching. By comparing the 1st, 2nd, and 5th rows in Table 4, the proposed uncertainty-conditioned hybrid matching strategy is witnessed to work better than using Hungarian matching or greedy matching alone. Note that, as uncertainty only contributes to hybrid matching, the 1st and 2nd rows in Table 4 are thus marked as "w/o Unctt.". Furthermore, we investigate whether using hybrid matching alone could bring benefits. For this purpose, we set a baseline where we mimic the operation introduced in Sec. 3.4.3 but splitting the non-emerging detection set $\mathbb{O}_{ac}^t$ based on the matching cost (i.e., the Euclidean distance), rather than based on the uncertainty score. As indicated in the 3rd row in Table 4, this receives no performance improvement, thus validating the necessity of the uncertainty.

Complementarity of APP and Uncertainty. By comparing the last two rows in Table 4, we find that the fine-grained displacement uncertainty, as expected, almost covers the function of APP. However, binary APP classification is capable of distinguishing whether a target is a newly appeared one, whereas continuous uncertainty scores can not achieve this. Though APP brings marginal performance improvement, we retain the APP branch in APPTracker+ due to its insightful handling of occlusions.

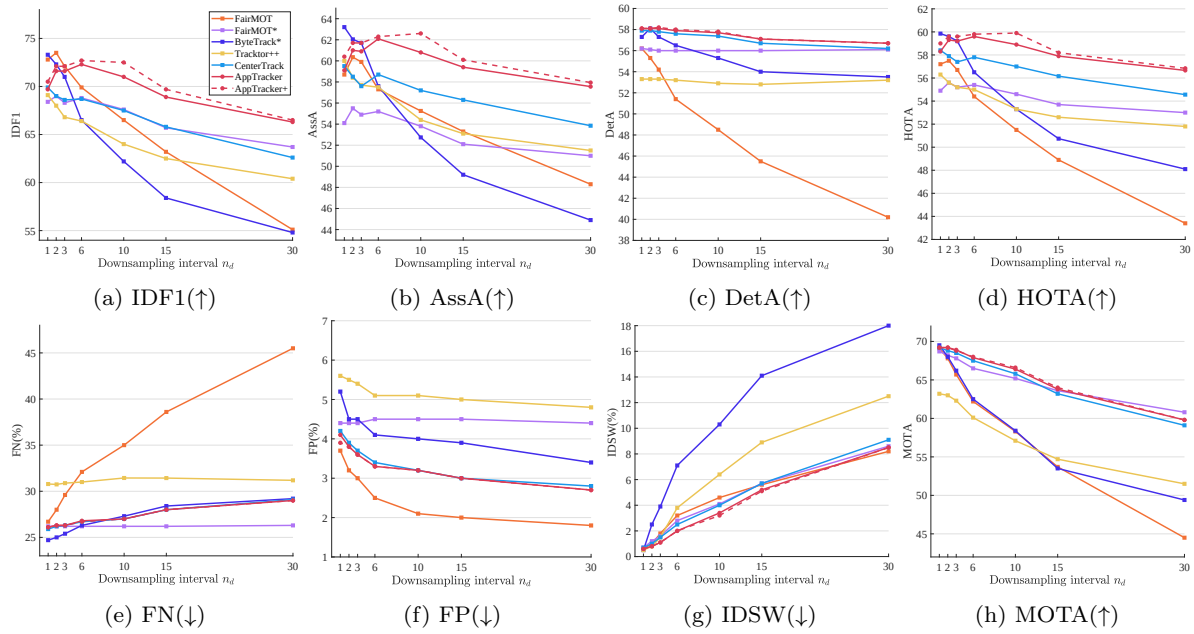
4.6 Results on MOTChallenge

In this section, we compare our method with several existing representative methods: FairMOT [77] (re-identification-based method), ByteTrack [76] (motion-model-based method), Tracktor++ [3] (detector-based method), and CenterTrack [81] (optical-flow-based method). All these methods, except FairMOT, are evaluated with the same detection set for fairness, while evaluating FairMOT with external detections may harm the extracting of re-identification embeddings. Fig. 8 provides an overview of performance curves with different frame rate inputs. Among all these metrics, we refer readers to the comparisons in association metrics (e.g., IDF1 and AssA) because 1) these methods are compared using the same detection set and the differences in detection metrics (e.g., DetA, FP, and FN) should be small; 2) comprehensive metrics like MOTA and HOTA consider both association metrics and detection metrics (MOTA=100% − FN − FP − IDSW; HOTA=$\sqrt{\text{AssA} \cdot \text{DetA}}$), exhibiting less pronounced differences when using the same detection set. Table 5 and Table 6 report detailed results in 1/10-frame-rate cases. Please refer to Sec. A in the appendix for more detailed experimental setups of each baseline method.

FairMOT. FairMOT [77] jointly accomplishes the tasks of object detection and re-identification in a single network. In addition, it adopts a Kalman filter to

Table 4: Ablation study of the uncertainty-conditioned hybrid matching on the MOT17 validation set (1/10-frame-rate). "Unctt." is short for displacement uncertainty.

Methods	Matching	HOTA↑	DetA↑	AssA↑	IDF1↑	IDs(%)↓	FP(%)↓	FN(%)↓	MOTA↑
APPTracker+ (w/o Unctt.)	Hungarian	58.9	57.7	60.8	71.1	3.4	3.0	27.0	66.5
APPTracker+ (w/o Unctt.)	Greedy	59.5	57.8	62.0	71.8	3.5	3.0	27.0	66.5
APPTracker+ (w/o Unctt.)	Hybrid	59.5	57.8	61.9	71.9	3.3	3.0	27.0	66.7
APPTracker+ (w/o APP)	Hybrid	59.7	57.8	62.3	72.3	3.2	3.0	27.0	66.8
APPTracker+	Hybrid	**59.9**	**57.8**	**62.6**	**72.5**	**3.2**	**3.0**	**27.0**	**66.8**



(a) IDF1(↑) (b) AssA(↑) (c) DetA(↑) (d) HOTA(↑)

(e) FN(↓) (f) FP(↓) (g) IDSW(↓) (h) MOTA(↑)

Fig. 8: Results on MOT17 validation set with different frame rate inputs. Arrows indicate the direction of better performance.

exploit motion cues. Associations with low overlaps are rejected. To avoid tracking occasional false alarms, it allows an object to spawn a new tracklet only if it is continuously tracked for at least two frames (except objects in the first frame). The rejection policy and the strict spawning policy lead to an intolerable increase in FN in low-frame-rate cases.

FairMOT*. To evaluate the re-identification performance of FairMOT [77], we disable the Kalman filter together with the strict spawning strategy, resulting in a lite version, FairMOT*. The reported results for FairMOT* are obtained after performing a grid search of hyperparameters. We observed that the default parameter settings of FairMOT are not suitable for low-frame-rate scenarios. For more details, please refer to Sec. C.1 in the appendix. Compared to other methods, FairMOT* exhibits a slower performance degradation at extremely low frame rates (i.e., $\leq 1/15$), since the multi-object tracking task at such low frame rates transforms into a ReID task.

ByteTrack and ByteTrack*. ByteTrack [76] proposes to associate low-score detections with motion cues. Similar to FairMOT, the Kalman filter, and the strict spawning strategy are adopted, leading to additional FN in low-frame-rate cases. To avoid unexpected increases in FN, the strict spawning strategy is disabled in our experiments (denoted by ByteTrack*). The method is evaluated with the detections from our model. Though it performs well in normal-frame-rate cases, it fails in low-frame-rate cases with the failure of the motion model. We also investigate a GIoU [55]-version of ByteTrack in the appendix but receive minor performance gain.

Tracktor++. Tracktor++ [3], assuming that bounding boxes of the same target have a large overlap between frames, propagates the identity of a region proposal to the next frame by bounding box regression. To evaluate Tracktor++, we take the detection set generated by our model as input. However, degeneration in detection performance is observed due to subsequent detection refinement operations inside the Track-

Table 5: Full results of evaluating methods on the MOT17 validation set (1/10-frame-rate). Runtimes are obtained on a single Nvidia RTX3090 GPU with a single thread on an Intel Xeon Gold 6326 CPU. +D means detection time.

Method	Time(ms)	HOTA(↑)	DetA(↑)	AssA(↑)	MOTA(↑)	IDF1(↑)	MT(↑)	ML(↓)	FP(↓)	FN(↓)	IDs(↓)
FairMOT [77]	40	51.8	48.8	55.6	58.3	66.5	25.8	28.6	**1107**	18577	2436
FairMOT* [77]	40	55.0	56.0	54.6	65.5	68.2	**46.1**	**14.0**	2421	**14153**	2041
ByteTrack [76]	4+D	52.3	49.4	56.9	56.3	62.6	28.2	30.7	1417	18996	2705
ByteTrack* [76]	4+D	53.3	55.3	52.7	58.4	62.2	45.0	16.9	2105	14474	5477
Tracktor++ [3]	≥666+D	53.3	52.9	54.4	57.1	64.0	42.2	16.9	2675	16562	3373
CenterTrack [81]	37	57.0	57.4	57.2	65.8	67.5	45.9	16.4	1676	14282	2129
APPTracker [80]	39	58.9	57.7	60.8	66.4	71.0	45.6	16.2	1716	14321	1781
APPTracker+	39	**59.9**	**57.8**	**62.6**	**66.6**	**72.5**	45.6	16.3	1691	14316	**1670**

Table 6: Full results of evaluating methods on the MOT20 validation set (1/10-frame-rate). +D means detection time.

Method	Time(ms)	HOTA(↑)	DetA(↑)	AssA(↑)	MOTA(↑)	IDF1(↑)	MT(↑)	ML(↓)	FP(↓)	FN(↓)	IDs(↓)
FairMOT [77]	71	56.0	62.6	50.4	74.6	70.1	62.4	6.0	22284	100643	33186
FairMOT* [77]	71	58.3	**66.6**	51.2	79.6	72.5	**78.3**	**4.0**	35298	**64714**	25240
ByteTrack [76]	24+D	47.5	52.1	43.8	55.7	59.5	37.7	13.0	67571	176062	27328
ByteTrack* [76]	24+D	47.7	54.1	42.5	57.1	59.0	44.3	9.6	71122	156621	35020
Tracktor++ [3]	≥833+D	47.2	56.3	39.7	63.1	57.8	45.5	8.6	13898	166970	44755
CenterTrack [81]	73	58.1	64.9	52.2	77.9	70.8	65.2	7.4	15228	102575	17694
APPTracker [80]	80	60.6	65.4	56.3	78.1	74.0	64.9	7.4	**13852**	103348	16612
APPTracker+	80	**61.4**	65.4	**57.7**	**78.7**	**75.3**	65.2	7.3	14779	102562	**13109**

Table 7: Cross-dataset evaluation. The models are trained on MOT17 but are evaluated on MOT20 (1/10-frame-rate).

Method	HOTA(↑)	DetA(↑)	AssA(↑)	MOTA(↑)	IDF1(↑)	MT(↑)	ML(↓)	FP(↓)	FN(↓)	IDs(↓)
FairMOT [77]	40.4	49.0	34.0	57.0	53.3	34.3	11.8	**26545**	185825	52640
FairMOT* [77]	22.3	**52.6**	10.0	43.1	24.4	**49.9**	**8.5**	39056	**147260**	163787
ByteTrack [76]	36.8	42.4	32.7	51.1	50.1	26.3	16.7	51166	209233	35220
ByteTrack* [76]	38.6	47.1	32.2	53.4	51.7	34.6	12.3	58707	178057	48497
Tracktor++ [3]	35.4	46.3	27.6	52.4	46.0	33.7	10.3	34064	197411	59567
CenterTrack [81]	40.9	49.9	34.0	63.4	54.9	44.3	11.4	31589	159302	33038
APPTracker [80]	44.2	50.3	39.5	64.0	59.5	43.7	11.8	30392	161069	28950
APPTracker+	**44.8**	50.5	**40.4**	**64.3**	**60.4**	44.4	11.6	31014	160457	**27243**

tor++. As its primary assumption does not hold when the frame rate becomes low, the performance decreases quickly.

CenterTrack. CenterTrack [81] serves as our baseline. It is evaluated given detections and displacements estimated by our model, but using its own association logic. Compared with other methods, the performance drop of CenterTrack is less significant. With a large receptive field, it is more robust to large target motions and camera motions in low-frame-rate cases.

APPTracker and APPTracker+. Enabled by the proposed APP head and the multi-stage matching strategy, APPTracker overcomes the local weakness of CenterTrack [81], achieving consistent improvement in IDF1 score compared to the CenterTrack [81] and outperforms the state-of-the-art methods in low-frame-rate cases. With the proposed extensions, APPTracker+ shows further improvements.

Cross-Dataset Evaluation. To avoid potential identity leakage risks, we additionally conduct a cross-dataset experiment where we first pre-train the models on CrowdHuman [58] then finetune the models on half of MOT17 [48] and finally test them on the MOT20 [18] validation set. These three datasets have no identity overlaps. As shown in Table 7, FairMOT demonstrates better detection generalization but limited ReID generalization, possibly due to occlusion and domain gaps. According to Table 7, despite the domain gap, our method maintains a relative advantage in association metrics including AssA, IDF1, and IDs. Results for more frame-rates can be found in Sec. B in the appendix.

Runtime. Utilizing a single Nvidia RTX3090 GPU with a single thread on an Intel Xeon Gold 6326 CPU, our method achieved 25FPS and 12.5FPS on MOT17 and MOT20, respectively, with an input resolution of

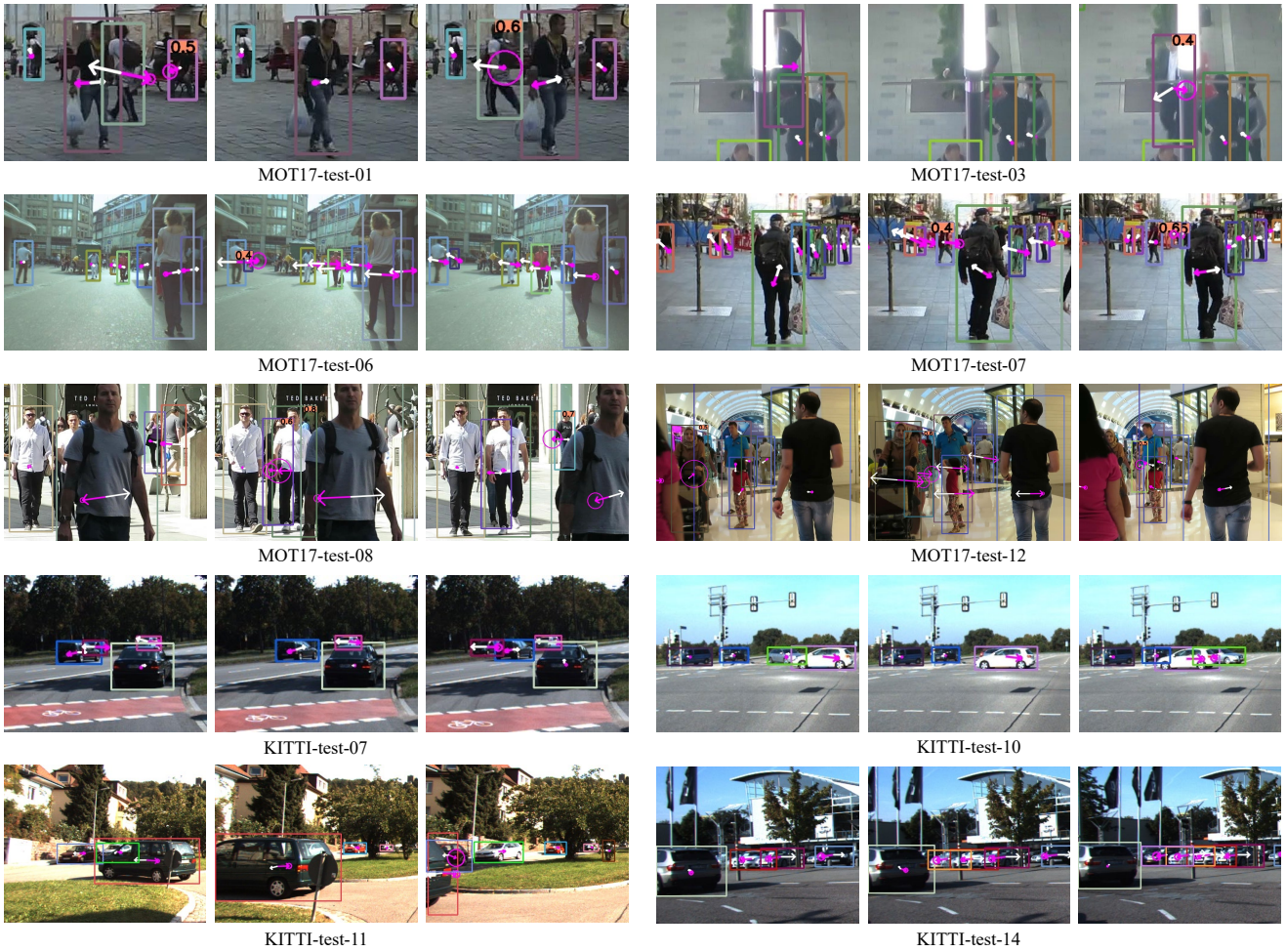


Fig. 9: Qualitative results. Each case shows three consecutive frames. Identities are coded by color. We highlight recognized appearing objects by indicating the predicted APP scores on the right top of corresponding bounding boxes. Pink arrows indicate displacement estimations, with a circle at the end of each arrow representing the uncertainty. White arrows indicate velocity estimations. Best viewed in color and by zooming in. Some video sequences were captured by moving cameras. Therefore, the actual motion direction of the target may differ from what appears visually. For further clarification, please kindly check our demo video. There is no APP prediction when running on KITTI. For more details please refer to the document.

$960 * 544$. When the model operates on videos with a 1/10 frame rate, this is equivalent to achieving an inference speed of 250FPS on MOT17.

Table 8: Results on KITTI [22] test set. Runtimes are obtained from the leaderboard. +D means detection time.

Method	Time	HOTA(↑)	DetA(↑)	AssA(↑)	MOTA(↑)	FP(↓)	FN(↓)	IDs(↓)
MASS [28]	10+D	68.3	72.9	64.5	84.6	4145	786	353
TuSimple [13]	-	71.6	72.6	71.1	86.3	3656	759	292
SMAT [23]	100+D	71.9	72.1	72.1	83.6	5254	**175**	198
TrackMPNN [52]	50+D	72.3	74.7	70.6	87.3	2577	1298	481
CenterTrack [81]	47	73.0	75.6	71.2	88.8	2703	886	254
LGM [65]	80+D	73.1	74.6	72.3	87.6	**2249**	1568	448
APPTracker+	50	75.2	75.6	75.4	89.1	3242	334	**176**
OC-SORT [9]	1.4+D	76.5	77.3	76.4	90.3	2685	407	250
PermaTrack [64]	100	78.0	78.3	78.4	91.3	2320	402	258
RAM [63]	90	**79.5**	**78.8**	**80.9**	**91.6**	2094	583	210

4.7 Results on KITTI

In Table 8, we report the results on the test set where we compare our method with state-of-the-art monocular-camera-based 2D tracking methods. Following Centertrack [81], we submitted our results with flip testing. As shown in Table 8, our method achieves a 4.2 improvement in association accuracy (AssA) compared to our baseline method, CenterTrack [81], while maintaining similar detection accuracy (DetA). Our method without flip testing runs at 34 ms.

It is worth noting that OC-SORT [9], PermTrack [64], and RAM [63] involved an additional synthetic tracking dataset ParallelDomain [64] during training the detector, which is 40 times larger

than KITTI. Since our work mainly aims to improve association performance, we would emphasize the fair comparison with TrackMPNN [52], CenterTrack [81], and LGM [65], where TrackMPNN [52] and LGM [65] adopted the detection set from CenterTrack [81].

5 Conclusion and Discussion

In this paper, we have studied the challenges of tracking multiple objects in low-frame-rate videos. An online tracker is proposed to address these challenges. Based on a tracker with a local displacement estimator, we design an APP head together with a novel augmentation strategy to robustly filter out unreliable displacement estimations due to frequent visibility changes in low-frame-rate cases. The idea of the APP head is further extended to capture fine-grained displacement uncertainty by reformulating the regression task. With APP predictions and displacement uncertainties, a multi-stage matching policy is proposed to reduce the noise inside the association matrix and to leverage historical motion cues. Experiments on public datasets with various frame rate settings verify the effectiveness and robustness of our method.

Limitations. Though our method studies the multi-object tracking at low frame rates and achieves competitive results, there exist some limitations: (1) Our approach shows reduced effectiveness at extremely low frame rates ($< 1/15$ frame rate) due to inherent limitations in optical-flow-based tracking. (2) Our method fails to handle cases where more than one target is simultaneously occluded at the same location.

Future work. Currently, our model jointly learns detection and displacement estimation. On the engineering side, one may decouple the model into separate detection and displacement estimation models, allowing for more flexible deployment and independent optimization. This would further enable each component to benefit from more advanced studies in their respective fields, i.e., detection [21,41], flow/correspondence estimation [62,59], etc. We leave these as our future work.

Data Availability Statement

The datasets analyzed during the current study are available in the CrowdHuman repository, MOTChallenge repository, and KITTI repository, with the links as http://www.crowdhuman.org/, https://motchallenge.net/data/MOT17/, https://motchallenge.net/data/MOT20/, and https://www.cvlibs.net/download.php?file=data_tracking_image_2.zip.

References

1. Alahi, A., Goel, K., Ramanathan, V., Robicquet, A., Fei-Fei, L., Savarese, S.: Social lstm: Human trajectory prediction in crowded spaces. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 961–971 (2016) 3
2. Ballas, N., Yao, L., Pal, C., Courville, A.: Delving deeper into convolutional networks for learning video representations. arXiv preprint arXiv:1511.06432 (2015) 4
3. Bergmann, P., Meinhardt, T., Leal-Taixe, L.: Tracking without bells and whistles. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 941–951 (2019) 1, 3, 4, 13, 14, 15, 20
4. Bernardin, K., Stiefelhagen, R.: Evaluating multiple object tracking performance: the clear mot metrics. EURASIP Journal on Image and Video Processing **2008**, 1–10 (2008) 10
5. Bewley, A., Ge, Z., Ott, L., Ramos, F., Upcroft, B.: Simple online and realtime tracking. In: 2016 IEEE international conference on image processing (ICIP), pp. 3464–3468. IEEE (2016) 2, 3
6. Brasó, G., Cetintas, O., Leal-Taixé, L.: Multi-object tracking and segmentation via neural message passing. International Journal of Computer Vision **130**(12), 3035–3053 (2022) 3, 4
7. Brasó, G., Leal-Taixé, L.: Learning a neural solver for multiple object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 6247–6257 (2020) 3, 4
8. Caesar, H., Bankiti, V., Lang, A.H., Vora, S., Liong, V.E., Xu, Q., Krishnan, A., Pan, Y., Baldan, G., Beijbom, O.: nuscenes: A multimodal dataset for autonomous driving. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 11621–11631 (2020) 10
9. Cao, J., Pang, J., Weng, X., Khirodkar, R., Kitani, K.: Observation-centric sort: Rethinking sort for robust multi-object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9686–9696 (2023) 4, 16
10. Cao, Z., Huang, Z., Pan, L., Zhang, S., Liu, Z., Fu, C.: Tctrack: Temporal contexts for aerial tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14798–14808 (2022) 4
11. Cao, Z., Huang, Z., Pan, L., Zhang, S., Liu, Z., Fu, C.: Towards real-world visual tracking with temporal contexts. IEEE Transactions on Pattern Analysis and Machine Intelligence (2023) 4
12. Carion, N., Massa, F., Synnaeve, G., Usunier, N., Kirillov, A., Zagoruyko, S.: End-to-end object detection with transformers. In: European conference on computer vision, pp. 213–229. Springer (2020) 4
13. Choi, W.: Near-online multi-target tracking with aggregated local flow descriptor. In: Proceedings of the IEEE international conference on computer vision, pp. 3029–3037 (2015) 16
14. Chu, P., Wang, J., You, Q., Ling, H., Liu, Z.: Transmot: Spatial-temporal graph transformer for multiple object tracking. In: Proceedings of the IEEE/CVF Winter Conference on Applications of Computer Vision, pp. 4870–4880 (2023) 3
15. Chuang, M.C., Hwang, J.N., Williams, K., Towler, R.: Tracking live fish from low-contrast and low-frame-rate stereo videos. IEEE Transactions on Circuits and Systems for Video Technology **25**(1), 167–179 (2014) 4

16. Dai, J., Qi, H., Xiong, Y., Li, Y., Zhang, G., Hu, H., Wei, Y.: Deformable convolutional networks. In: Proceedings of the IEEE international conference on computer vision, pp. 764–773 (2017) 2
17. Dai, P., Weng, R., Choi, W., Zhang, C., He, Z., Ding, W.: Learning a proposal classifier for multiple object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 2443–2452 (2021) 3
18. Dendorfer, P., Rezatofighi, H., Milan, A., Shi, J., Cremers, D., Reid, I., Roth, S., Schindler, K., Leal-Taixé, L.: Mot20: A benchmark for multi object tracking in crowded scenes. arXiv preprint arXiv:2003.09003 (2020) 3, 7, 8, 9, 10, 15, 20, 21, 23, 24, 25
19. Evangelidis, G.D., Psarakis, E.Z.: Parametric image alignment using enhanced correlation coefficient maximization. IEEE transactions on pattern analysis and machine intelligence **30**(10), 1858–1865 (2008) 3
20. Feng, W., Bai, L., Yao, Y., Yu, F., Ouyang, W.: Towards frame rate agnostic multi-object tracking. International Journal of Computer Vision **132**(5), 1443–1462 (2024) 4
21. Ge, Z., Liu, S., Wang, F., Li, Z., Sun, J.: Yolox: Exceeding yolo series in 2021. arXiv preprint arXiv:2107.08430 (2021) 3, 4, 17
22. Geiger, A., Lenz, P., Urtasun, R.: Are we ready for autonomous driving? the kitti vision benchmark suite. In: 2012 IEEE conference on computer vision and pattern recognition, pp. 3354–3361. IEEE (2012) 3, 8, 10, 12, 16, 24, 25
23. Gonzalez, N.F., Ospina, A., Calvez, P.: Smat: Smart multiple affinity metrics for multiple object tracking. In: Image Analysis and Recognition: 17th International Conference, ICIAR 2020, Póvoa de Varzim, Portugal, June 24–26, 2020, Proceedings, Part II 17, pp. 48–62. Springer (2020) 16
24. Guo, S., Wang, J., Wang, X., Tao, D.: Online multiple object tracking with cross-task synergy. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8136–8145 (2021) 3
25. He, J., Huang, Z., Wang, N., Zhang, Z.: Learnable graph matching: Incorporating graph partitioning with deep feature learning for multiple object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 5299–5309 (2021) 3, 4
26. Isard, M., Blake, A.: Condensation—conditional density propagation for visual tracking. International journal of computer vision **29**(1), 5–28 (1998) 4
27. Kalman, R.E.: A new approach to linear filtering and prediction problems (1960) 2, 3, 20, 22
28. Karunasekera, H., Wang, H., Zhang, H.: Multiple object tracking with attention to appearance, structure, motion and size. IEEE Access **7**, 104423–104434 (2019) 16
29. Kendall, A., Gal, Y.: What uncertainties do we need in bayesian deep learning for computer vision? Advances in neural information processing systems **30** (2017) 2, 6
30. Kim, C., Fuxin, L., Alotaibi, M., Rehg, J.M.: Discriminative appearance modeling with multi-track pooling for real-time multi-object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 9553–9562 (2021) 3
31. Kim, C., Li, F., Ciptadi, A., Rehg, J.M.: Multiple hypothesis tracking revisited. In: Proceedings of the IEEE international conference on computer vision, pp. 4696–4704 (2015) 3
32. Kingma, D.P., Ba, J.: Adam: A method for stochastic optimization. arXiv preprint arXiv:1412.6980 (2014) 10
33. Kuhn, H.W.: The hungarian method for the assignment problem. Naval research logistics quarterly **2**(1-2), 83–97 (1955) 3, 9
34. Law, H., Deng, J.: Cornernet: Detecting objects as paired keypoints. In: Proceedings of the European conference on computer vision (ECCV), pp. 734–750 (2018) 6, 7
35. Le, Q.V., Smola, A.J., Canu, S.: Heteroscedastic gaussian process regression. In: Proceedings of the 22nd international conference on Machine learning, pp. 489–496 (2005) 2, 6
36. Li, Y., Ai, H., Yamashita, T., Lao, S., Kawade, M.: Tracking in low frame rate video: A cascade particle filter with discriminative observers of different life spans. IEEE transactions on pattern analysis and machine intelligence **30**(10), 1728–1740 (2008) 4
37. Liang, C., Zhang, Z., Zhou, X., Li, B., Zhu, S., Hu, W.: Rethinking the competition between detection and reid in multiobject tracking. IEEE Transactions on Image Processing **31**, 3182–3196 (2022) 3
38. Lin, T.Y., Dollár, P., Girshick, R., He, K., Hariharan, B., Belongie, S.: Feature pyramid networks for object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2117–2125 (2017) 4
39. Lin, T.Y., Goyal, P., Girshick, R., He, K., Dollár, P.: Focal loss for dense object detection. In: Proceedings of the IEEE international conference on computer vision, pp. 2980–2988 (2017) 6
40. Lin, T.Y., Maire, M., Belongie, S., Hays, J., Perona, P., Ramanan, D., Dollár, P., Zitnick, C.L.: Microsoft coco: Common objects in context. In: European conference on computer vision, pp. 740–755. Springer (2014) 11
41. Liu, S., Zeng, Z., Ren, T., Li, F., Zhang, H., Yang, J., Li, C., Yang, J., Su, H., Zhu, J., et al.: Grounding dino: Marrying dino with grounded pre-training for open-set object detection. arXiv preprint arXiv:2303.05499 (2023) 17
42. Liu, Y., Wu, J., Fu, Y.: Collaborative tracking learning for frame-rate-insensitive multi-object tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9964–9973 (2023) 4
43. Luiten, J., Osep, A., Dendorfer, P., Torr, P., Geiger, A., Leal-Taixé, L., Leibe, B.: Hota: A higher order metric for evaluating multi-object tracking. International Journal of Computer Vision **129**, 548–578 (2021) 3, 10, 22
44. Luo, W., Stenger, B., Zhao, X., Kim, T.K.: Trajectories as topics: Multi-object tracking by topic discovery. IEEE Transactions on Image Processing **28**(1), 240–252 (2018) 3
45. Luo, W., Xing, J., Milan, A., Zhang, X., Liu, W., Kim, T.K.: Multiple object tracking: A literature review. Artificial Intelligence **293**, 103448 (2021) 1
46. Ma, C., Yang, F., Li, Y., Jia, H., Xie, X., Gao, W.: Deep human-interaction and association by graph-based learning for multiple object tracking in the wild. International Journal of Computer Vision **129**, 1993–2010 (2021) 3, 4
47. Meinhardt, T., Kirillov, A., Leal-Taixe, L., Feichtenhofer, C.: Trackformer: Multi-object tracking with transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 8844–8854 (2022) 4
48. Milan, A., Leal-Taixé, L., Reid, I., Roth, S., Schindler, K.: Mot16: A benchmark for multi-object tracking. arXiv preprint arXiv:1603.00831 (2016) 3, 7, 8, 9, 10, 12, 15, 20, 21, 22, 24, 25
49. Pang, J., Qiu, L., Li, X., Chen, H., Li, Q., Darrell, T., Yu, F.: Quasi-dense similarity learning for multiple object tracking. In: Proceedings of the IEEE/CVF conference

on computer vision and pattern recognition, pp. 164–173 (2021) 4
50. Peng, J., Wang, C., Wan, F., Wu, Y., Wang, Y., Tai, Y., Wang, C., Li, J., Huang, F., Fu, Y.: Chained-tracker: Chaining paired attentive regression results for end-to-end joint multiple-object detection and tracking. In: European conference on computer vision, pp. 145–161. Springer (2020) 1
51. Qin, Z., Zhou, S., Wang, L., Duan, J., Hua, G., Tang, W.: Motiontrack: Learning robust short-term and long-term motions for multi-object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 17939–17948 (2023) 3, 4
52. Rangesh, A., Maheshwari, P., Gebre, M., Mhatre, S., Ramezani, V., Trivedi, M.M.: Trackmpnn: A message passing graph neural architecture for multi-object tracking. arXiv preprint arXiv:2101.04206 (2021) 16, 17
53. Redmon, J., Divvala, S., Girshick, R., Farhadi, A.: You only look once: Unified, real-time object detection. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 779–788 (2016) 3
54. Ren, S., He, K., Girshick, R., Sun, J.: Faster r-cnn: Towards real-time object detection with region proposal networks. Advances in neural information processing systems **28** (2015) 3, 4, 20
55. Rezatofighi, H., Tsoi, N., Gwak, J., Sadeghian, A., Reid, I., Savarese, S.: Generalized intersection over union: A metric and a loss for bounding box regression. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 658–666 (2019) 14, 24
56. Ristani, E., Solera, F., Zou, R., Cucchiara, R., Tomasi, C.: Performance measures and a data set for multi-target, multi-camera tracking. In: European conference on computer vision, pp. 17–35. Springer (2016) 3, 10
57. Saleh, F., Aliakbarian, S., Rezatofighi, H., Salzmann, M., Gould, S.: Probabilistic tracklet scoring and inpainting for multiple object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 14329–14339 (2021) 3
58. Shao, S., Zhao, Z., Li, B., Xiao, T., Yu, G., Zhang, X., Sun, J.: Crowdhuman: A benchmark for detecting human in a crowd. arXiv preprint arXiv:1805.00123 (2018) 3, 9, 10, 15
59. Sun, J., Shen, Z., Wang, Y., Bao, H., Zhou, X.: Loftr: Detector-free local feature matching with transformers. In: Proceedings of the IEEE/CVF conference on computer vision and pattern recognition, pp. 8922–8931 (2021) 17
60. Sun, P., Cao, J., Jiang, Y., Zhang, R., Xie, E., Yuan, Z., Wang, C., Luo, P.: Transtrack: Multiple object tracking with transformer. arXiv preprint arXiv:2012.15460 (2020) 4
61. Sun, S., Akhtar, N., Song, X., Song, H., Mian, A., Shah, M.: Simultaneous detection and tracking with motion modelling for multiple object tracking. In: European Conference on Computer Vision, pp. 626–643. Springer (2020) 1
62. Teed, Z., Deng, J.: Raft: Recurrent all-pairs field transforms for optical flow. In: Computer Vision–ECCV 2020: 16th European Conference, Glasgow, UK, August 23–28, 2020, Proceedings, Part II 16, pp. 402–419. Springer (2020) 17
63. Tokmakov, P., Jabri, A., Li, J., Gaidon, A.: Object permanence emerges in a random walk along memory. In: International Conference on Machine Learning, pp. 21506–21519. PMLR (2022) 16
64. Tokmakov, P., Li, J., Burgard, W., Gaidon, A.: Learning to track with object permanence. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 10860–10869 (2021) 4, 16
65. Wang, G., Gu, R., Liu, Z., Hu, W., Song, M., Hwang, J.N.: Track without appearance: Learn box and tracklet embedding with local and global motion patterns for vehicle tracking. In: Proceedings of the IEEE/CVF International Conference on Computer Vision, pp. 9876–9886 (2021) 16, 17
66. Wang, Z., Zheng, L., Liu, Y., Li, Y., Wang, S.: Towards real-time multi-object tracking. In: European Conference on Computer Vision, pp. 107–122. Springer (2020) 4
67. Wojke, N., Bewley, A., Paulus, D.: Simple online and realtime tracking with a deep association metric. In: 2017 IEEE international conference on image processing (ICIP), pp. 3645–3649. IEEE (2017) 3, 4
68. Xu, J., Cao, Y., Zhang, Z., Hu, H.: Spatial-temporal relation networks for multi-object tracking. In: Proceedings of the IEEE/CVF international conference on computer vision, pp. 3988–3998 (2019) 3
69. Yoon, J.H., Lee, C.R., Yang, M.H., Yoon, K.J.: Structural constraint data association for online multi-object tracking. International Journal of Computer Vision **127**, 1–21 (2019) 3
70. Yu, E., Li, Z., Han, S.: Towards discriminative representation: Multi-view trajectory contrastive learning for online multi-object tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 8834–8843 (2022) 4
71. Yu, F., Li, W., Li, Q., Liu, Y., Shi, X., Yan, J.: Poi: Multiple object tracking with high performance detection and appearance feature. In: Computer Vision–ECCV 2016 Workshops: Amsterdam, The Netherlands, October 8-10 and 15-16, 2016, Proceedings, Part II 14, pp. 36–42. Springer (2016) 4
72. Yu, F., Wang, D., Shelhamer, E., Darrell, T.: Deep layer aggregation. In: Proceedings of the IEEE conference on computer vision and pattern recognition, pp. 2403–2412 (2018) 4, 10
73. Zeng, F., Dong, B., Zhang, Y., Wang, T., Zhang, X., Wei, Y.: Motr: End-to-end multiple-object tracking with transformer. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXVII, pp. 659–675. Springer (2022) 4
74. Zhang, X., Hu, W., Xie, N., Bao, H., Maybank, S.: A robust tracking system for low frame rate video. International Journal of Computer Vision **115**, 279–304 (2015) 4
75. Zhang, Y., Sheng, H., Wu, Y., Wang, S., Ke, W., Xiong, Z.: Multiplex labeling graph for near-online tracking in crowded scenes. IEEE Internet of Things Journal **7**(9), 7892–7902 (2020) 3
76. Zhang, Y., Sun, P., Jiang, Y., Yu, D., Weng, F., Yuan, Z., Luo, P., Liu, W., Wang, X.: Bytetrack: Multi-object tracking by associating every detection box. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXII, pp. 1–21. Springer (2022) 2, 3, 4, 9, 13, 14, 15, 20, 22, 24
77. Zhang, Y., Wang, C., Wang, X., Zeng, W., Liu, W.: Fairmot: On the fairness of detection and re-identification in multiple object tracking. International Journal of Computer Vision **129**(11), 3069–3087 (2021) 2, 3, 4, 9, 13, 14, 15, 20, 21, 22

78. Zhang, Y., Wang, T., Zhang, X.: Motrv2: Bootstrapping end-to-end multi-object tracking by pretrained object detectors. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pp. 22056–22065 (2023) 4
79. Zheng, L., Bie, Z., Sun, Y., Wang, J., Su, C., Wang, S., Tian, Q.: Mars: A video benchmark for large-scale person re-identification. In: Computer Vision–ECCV 2016: 14th European Conference, Amsterdam, The Netherlands, October 11-14, 2016, Proceedings, Part VI 14, pp. 868–884. Springer (2016) 22
80. Zhou, T., Luo, W., Shi, Z., Chen, J., Ye, Q.: Apptracker: Improving tracking multiple objects in low-frame-rate videos. In: Proceedings of the 30th ACM International Conference on Multimedia, pp. 6664–6674 (2022) 3, 10, 15
81. Zhou, X., Koltun, V., Krähenbühl, P.: Tracking objects as points. In: European Conference on Computer Vision, pp. 474–490. Springer (2020) 2, 3, 4, 5, 6, 7, 9, 10, 12, 13, 15, 16, 17, 20, 22, 24, 25
82. Zhou, X., Wang, D., Krähenbühl, P.: Objects as points. arXiv preprint arXiv:1904.07850 (2019) 3
83. Zhou, Z., Luo, W., Wang, Q., Xing, J., Hu, W.: Distractor-aware discrimination learning for online multiple object tracking. Pattern Recognition **107**, 107512 (2020) 3
84. Zhu, X., Su, W., Lu, L., Li, B., Wang, X., Dai, J.: Deformable detr: Deformable transformers for end-to-end object detection. arXiv preprint arXiv:2010.04159 (2020) 4

Appendix A Detailed Experimental Setups

This section elaborates on the detailed setups of experiments on MOT17 [48] and MOT20 [18].

FairMOT [77]. It is a ReID-based model and it randomly samples frames during training, making its training stage frame-rate-agnostic.

ByteTrack [76]. The tracking module of ByteTrack only includes a Kalman filter without learnable parameters. In our experiments, we evaluated ByteTrack with the same detection set obtained from our model.

Tracktor [3]. The trainable module of Tracktor is an object detector [54] and thus is frame-rate-agnostic. In the inference stage, tracking is achieved by propagating the bounding boxes from the previous frame to the next frame as proposals and leveraging the denoising capability of the detector's regression branch to obtain the refined boxes.

APPTracker/APPTracker+. Our model took two adjacent frames as input and was trained on 1/10-frame-rate videos. Following CenterTrack [81], the time interval between the two input frames is uniformly sampled from {-20, -10, 0, 10, 20} where pairs in temporally reverse order are allowed.

CenterTrack [81]. To ensure a fair comparison on the same detection set, CenterTrack is evaluated using detections and displacements estimated by our model, while employing its own association logic. The simulated performance of CenterTrack slightly exceeds that reported by the original authors, as shown in Table 9. This is due to the different ignoring policies on low-visible objects during training, as illustrated in Section 3.3.1 of the main text.

Table 9: Results on MOT17 validation set (normal-frame-rate).

CenterTrack version	MOTA(↑)	IDF1(↑)	MT(↑)	ML(↓)	FP(↓)	FN(↓)	IDs(↓)
Reported in [81]	66.2	69.4	39.5	22.1	**3.9%**	29.5%	**0.4%**
Simulated by our model	**69.1**	**69.8**	**45.7**	**15.0**	4.3%	**26.0%**	0.6%

Buffer size. Table 10 lists the buffer sizes (above which an inactive tracklet is terminated) we adopted for different frame rate settings.

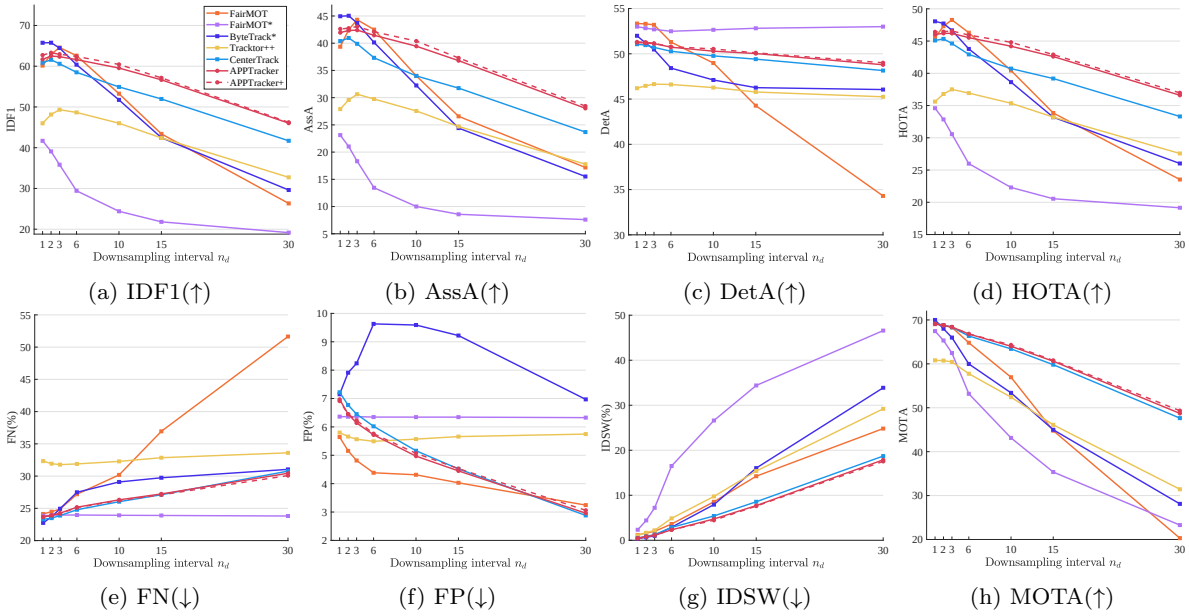
Table 10: Buffer sizes for different settings of n_d.

Sampling interval n_d	1	2	3	6	10	15	30
Buffer size	30	15	10	10	10	6	3

For each method, only one model is trained on each dataset and is tested at different frame rates.

Appendix B Cross-Dataset Evaluation with Multi-Frame-Rate Settings

Fig. 10 shows the multi-frame-rate experiments for the cross-dataset evaluation (trained on MOT17 [48] and tested on MOT20 [18]) where we maintained our original experimental setup, using the same detection set for all methods except FairMOT and FairMOT*. As shown in Fig. 10, the shape of the curves approximately aligns with the previous results on the MOT17 validation set except the ReID-based method denoted by FairMOT*. Besides, we note that (1) in high-frame-rate videos, our method slightly falls behind ByteTrack [76] and FairMOT [77], mainly due to the advantages of the Kalman filter [27] in high-frame-rate scenarios, as has been analyzed in Table 3 of the main text. This gap can be mitigated on the engineering side by switching the motion model according to target frame rates; (2) the detection performance of our model is slightly affected by frame rate. This is because our detection branch is learned jointly with the displacement estimation branch which shares the input of two adjacent frames. On the engineering side, one may address this influence by decoupling the detection and displacement estimation modules.



(a) IDF1(↑) (b) AssA(↑) (c) DetA(↑) (d) HOTA(↑)

(e) FN(↓) (f) FP(↓) (g) IDSW(↓) (h) MOTA(↑)

Fig. 10: Cross-dataset evaluation (trained on MOT17 [48] and tested on MOT20 [18]) with multiple frame rate downsampling factor n_d (x-axis label). Arrows indicate the direction of better performance.

Appendix C Further Investigations on Re-Identification Models

C.1 FairMOT

In this section, we conduct a comprehensive investigation of FairMOT [77]. The conclusions can be summarized in two points: (1) FairMOT's excessive confidence in historical appearance embeddings leads to significant additional failures in low-frame-rate videos. (2) The differences in detection entries are one of the main reasons for the performance degradation of ReID-based trackers in low-frame-rate videos, and it is not well addressed by adjusting the matching threshold. We elaborate on the details below. To investigate the ReID performance of FairMOT, in the following experiments, we disable the Kalman filter together with the strict spawning strategy.

Impact of matching threshold. We first study the impact of adjusting the matching threshold τ. Given two appearance embeddings, $\boldsymbol{e}_a$ and $\boldsymbol{e}_b$, FairMOT takes the cosine distance $1 - \frac{\boldsymbol{e}_a \cdot \boldsymbol{e}_b}{\|\boldsymbol{e}_a\|_2 \|\boldsymbol{e}_b\|_2}$ as the matching cost, which ranges from 0 to 2. According to Table 11, adjusting the matching threshold does not receive significant performance improvements. We set the matching threshold τ to 0.5 throughout our experiments (which was officially set to 0.4 in normal-frame-rate cases).

Impact of online appearance embedding update. Further, we conducted additional investigations on FairMOT and found that its excessive confidence in historical appearance embeddings leads to significant additional failures in low-frame-rate videos. Specifically, FairMOT [77] smoothly updates the latest appearance embeddings of tracklets. In a new time step t, the registered appearance embedding $\boldsymbol{e}^t_{\text{smooth}}$ of an associated tracklet is updated by

$$\boldsymbol{e}^t_{\text{smooth}} = \alpha \boldsymbol{e}^{t-1}_{\text{smooth}} + (1-\alpha)\boldsymbol{e}^t, \tag{10}$$

where α is a smoothness factor and is set to 0.9 by default, indicating the low confidence on the new appearance observation $\boldsymbol{e}^t$. We performed a grid search on α for various frame rate settings in Table 12 and plot the curves of $\alpha = \{0.0, 0.3, 0.6, 0.9\}$ in Figure 11

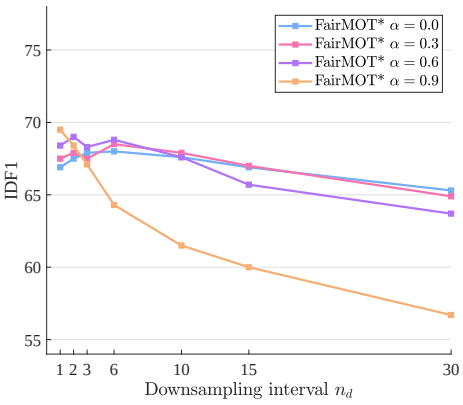


Fig. 11: IDF1 scores *vs.* α. Results on MOT17 validation set (1/10 frame rate).

Table 11: IDF1 score vs. matching threshold τ on MOT17 validation set (1/10-frame-rate). We annotate the official parameter setting with underline and the maximum performance gains from parameter tuning.

Dataset	Matching threshold τ											
	0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	2.0
MOT17 [48]	7.0	32.4	49.8	59.2	61.4	**61.5**(+0.1)	59.3	58.4	57.5	56.1	55.7	55.8

Table 12: IDF1 score vs. α and n_d. Results are obtained on MOT17 validation set. We annotate the official parameter setting with underline and the maximum performance gains from parameter tuning.

		Frame rate downsampling factor n_d						
		1	2	3	6	10	15	30
α	0.0	66.9	67.5	67.9	68.0	67.6	66.9	**65.3**(+8.6)
	0.1	67.4	68.2	67.9	68.2	67.7	66.9	65.2
	0.2	67.2	68.3	67.5	68.4	68.1	**67.0**	65.2
	0.3	67.5	67.9	67.5	68.5	67.9	**67.0**(+7.0)	64.9
	0.4	67.6	67.8	67.2	69.0	67.9	66.5	64.7
	0.5	68.2	67.7	67.5	**69.2**(+4.9)	**68.2**(+6.7)	66.3	64.3
	0.6	68.4	69.0	68.3	68.8	67.6	65.7	63.7
	0.7	69.0	69.9	68.9	68.0	66.4	64.8	62.7
	0.8	68.2	**70.4**(+2.0)	**69.0**(+1.9)	67.5	65.1	63.1	60.6
	0.9	**69.5**(+0.0)	68.4	67.1	64.3	61.5	60.0	56.7
	1.0	49.0	49.4	49.0	49.1	49.3	50.4	52.0

for ease of observation. We use a fixed α of 0.6 in the main text to avoid over-tailored hyperparameters.

According to Table 12, the optimal values for parameter α are roughly distributed along the diagonal. This is reasonable, considering that targets exhibit larger appearance changes as the frame rate goes lower. According to Figure 8(a) and Figure 8(b) of the main text, while FairMOT* achieves comparable IDF1 scores to CenterTrack [81], there is still a gap in terms of AssA. This is because the IDF1 score encourages more about estimating the total number of unique objects in a scene than about good detection or association [43]. Without the Kalman filter and the strict spawning strategy, FairMOT* tends to transfer identities between targets rather than spawning new tracks with new identities, which is preferred by the IDF1 score. Another weakness of using smaller values of α is that it may compromise long-term re-identification across occlusions.

Finally, we once again conducted a grid search of matching threshold τ with $\alpha = 0.5$. As shown in Table 13, using a small value of α makes the performance more sensitive to the matching threshold τ. However, compared to the default threshold τ of 0.4, parameter tuning still brings minor gains.

Impact of difference in detection entries. Further, we visualize the typical failure cases of the ReID-based tracking in Figure 12, where we find that the differences in detection entries caused by occlusion are one of the main reasons for the performance degradation of ReID-based trackers in low-frame-rate videos. Considering that identifying the detection variations is one of the main ideas in our work, we believe our work has the potential to enhance the performance of ReID-based trackers in low-frame-rate scenarios.

C.2 DeepSORT

Besides FairMOT [77], we also study the effectiveness of DeepSORT with a pre-trained ReID model and our detection set. Specifically, we follow the same DeepSORT implementation adopted by ByteTrack [76], where the ReID model was trained on a large-scale ReID dataset, MARS [79], containing over 1,100,000 images of 1,261 pedestrians. We provide the results on the MOT17 [48] validation set in Fig. 13. To study the tracking performance through ReID alone, we also include a version with the Kalman filter [27] turned off (denoted as DeepSORT*). Additionally, we experiment with different matching thresholds[1] τ, but receive no benefit compared to the default setting of $\tau = 0.1$.

For comparison, we attach the results of APPTracker+ (our method) and FairMOT* in Fig. 13, but set them in gray to avoid potential unfair comparisons, as APPTracker+ and FairMOT* are trained on half of the MOT17 data. From Fig. 13, the pre-trained model

[1] DeepSORT rejects matches with $(1 - \cos(\mathbf{e_a}, \mathbf{e_b})) > \tau$, where $\mathbf{e_a}$ and $\mathbf{e_b}$ are the appearance embeddings of a pair of detection and track, respectively. Unlike FairMOT [77], where the memorized appearance embedding of a track is updated by new observations with a fixed weight α at each time step, DeepSORT uses extra memory to store the appearance embeddings of all previous observations for each track. Therefore, the matching threshold τ is the only hyperparameter that needs to be tuned for DeepSORT.

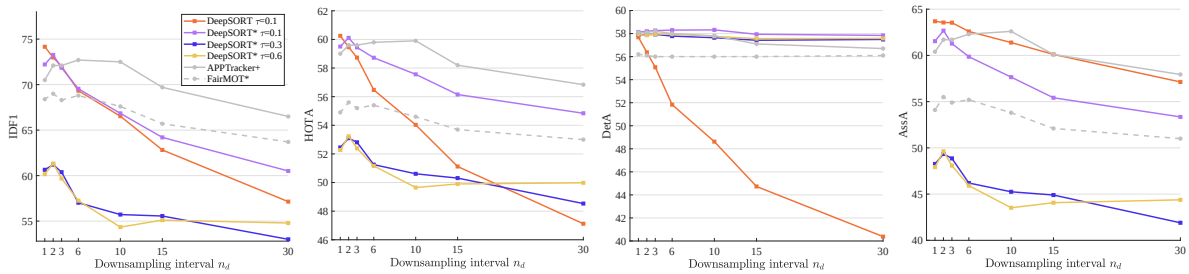
Fig. 12: Typical failure cases of FairMOT* (1/10 frame rate). In the first row of each case, we present cosine similarity heatmaps of appearance embeddings and annotate the query object in the first frame. The second and third rows provide tracking results from FairMOT* and our method, respectively. The appearance embedding is updated frame-by-frame with the historical embedding weight $\alpha = 0$. We highlight the recognized appearing objects in our results by indicating the predicted APP scores on the right top of corresponding bounding boxes.

demonstrates better appearance discrimination capability compared to FairMOT*, especially at high frame rates.

Further, in Fig. 14, we provide results on the MOT20 [18] validation set, which is characterized by crowded scenes and poor lighting conditions. Similarly, we attach the results of APPTracker+ and FairMOT*, which are trained on half of the MOT17 data. Shown in Fig. 14, under crowded and poor lighting conditions, (1) the pre-trained ReID also experiences reduced effectiveness, and (2) by comparing the results of DeepSORT and DeepSORT*, the motion model is found to consistently bring tracking performance improvements (as shown by the IDF1 score and the association accuracy, AssA), but lead to a huge decrease in DetA at low frame rates.

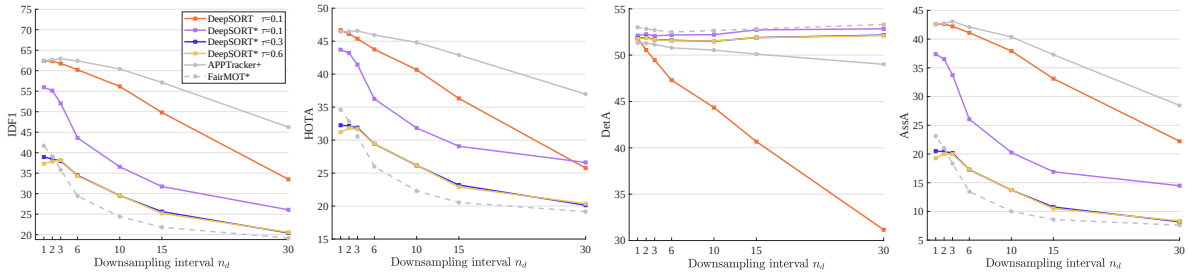
Table 13: IDF1 score vs. matching threshold τ (1/10-frame-rate). We annotate the official parameter setting with underline and the maximum performance gains from parameter tuning.

Dataset	α	Matching threshold τ 0.0	0.1	0.2	0.3	0.4	0.5	0.6	0.7	0.8	0.9	1.0	2.0
MOT17	0.9	7.0	32.4	49.8	59.2	61.4	**61.5**(+0.1)	59.3	58.4	57.5	56.1	55.7	55.8
MOT17	0.5	7.0	42.8	63.2	**69.3**(+0.3)	69.0	68.2	66.2	64.2	63.0	61.5	60.5	60.1



(a) IDF1 (b) HOTA (c) DetA (d) AssA

Fig. 13: Results on MOT17 validation set.



(a) IDF1 (b) HOTA (c) DetA (d) AssA

Fig. 14: Results on MOT20 validation set (APPTracker+ and FairMOT* are trained on MOT17).

Appendix D ByteTrack with GIoU

In Table 14, we report the results of adopting GIoU [55] in the Kalman-filter-based tracker, ByteTrack [76], with various matching threshold settings but receive minor performance gain. The failure of the motion model in low-frame-rate videos is mainly due to (1) the failure of zero-velocity initialization, and (2) quick changes in target velocity.

Appendix E Computational Cost

We reported the FLOPs, MACs, and parameter count in Table 15. Compared with CenterTrack [81], our method introduces 7.7% extra FLOPs/MACs and 0.7% extra parameters when running on MOT17 [48] and MOT20 [18]. With the absence of the APP prediction head when running on KITTI [22], our method only introduces negligible additional computational cost.

Table 14: IDF1 score vs. matching threshold on MOT17 validation set (1/10-frame-rate). The matching distance is calculated by 1-IoU or 1-GIoU, and the matching threshold constrains the upper bound for the matching distance. We annotate the official parameter setting with underline.

Method	Matching threshold												
	0.8	0.9	1.0	1.1	1.2	1.3	1.4	1.5	1.6	1.7	1.8	1.9	2.0
ByteTrack (IoU)	62.2	**62.8**	59.8	-	-	-	-	-	-	-	-	-	-
ByteTrack (GIoU)	61.4	**62.7**	62.4	61.9	61.8	61.4	60.5	59.5	58.6	56.1	55.2	52.2	49.5

Table 15: Computational cost.

Dataset	Input Size	Method	FLOPs(G)	MACs(G)	Params(M)
MOT17 [48] & MOT20 [18]	960*544	CenterTrack [81]	125.16	62.40	19.96
		Ours	134.84	67.23	20.10
KITTI [22]	1280*384	CenterTrack [81]	108.67	54.17	19.81
		Ours	108.68	54.18	19.81