APPTracker: Improving Tracking Multiple Objects in Low-Frame-Rate Videos

Tao Zhou
Zhejiang University
zhoutao2015@zju.edu.cn

Wenhan Luo
Sun Yat-sen University
whluo.china@gmail.com

Zhiguo Shi*
Zhejiang University
shizg@zju.edu.cn

Jiming Chen
Zhejiang University
cjm@zju.edu.cn

Qi Ye†
Zhejiang University
qi.ye@zju.edu.cn

ABSTRACT

Multi-object tracking (MOT) in the scenario of low-frame-rate videos is a promising solution for deploying MOT methods on edge devices with limited computing, storage, power, and transmitting bandwidth. 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 correct displacement estimations of objects whose visibility flip within adjacent frames. 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. Then we design a two-stage association policy where displacement estimations or historical motion cues are leveraged in the corresponding stage according to APP predictions. Our method, with little additional computational overhead, 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.

CCS CONCEPTS

• **Computing methodologies** → *Artificial Intelligence*; *Computer vision.*

*Zhiguo Shi is with the College of Information Science and Electronic Engineering, Zhejiang University, Hangzhou, China, and also with the International Joint Innovation Center, Zhejiang University, Haining, China.
†Corresponding Author Qi Ye. Qi Ye is with the College of Control Science and Engineering, the State Key Laboratory of Industrial Control Technology, Zhejiang University, and also with the Key Laboratory of Collaborative Sensing and Autonomous Unmanned Systems of Zhejiang Province.

MM '22, October 10–14, 2022, Lisboa, Portugal

ACM ISBN 978-1-4503-9203-7/22/10…$15.00
https://doi.org/10.1145/3503161.3548162

KEYWORDS

Multi-object tracking, low-frame-rate videos, occlusion handling

ACM Reference Format:
Tao Zhou, Wenhan Luo, Zhiguo Shi, Jiming Chen, and Qi Ye. 2022. APPTracker: Improving Tracking Multiple Objects in Low-Frame-Rate Videos. In *Proceedings of the 30th ACM International Conference on Multimedia (MM '22), October 10–14, 2022, Lisboa, Portugal.* ACM, New York, NY, USA, 11 pages. https://doi.org/10.1145/3503161.3548162

1 INTRODUCTION

In multi-object tracking (MOT) [24], objects of interest are usually required to be detected and tracked in videos recorded with a frame rate of about 30 FPS. The problem has been studied for decades and tremendous progress has been achieved. However, deploying the existing methods on edge devices encounters challenges due to the constraints in power, computing, storage, and transmitting bandwidth. A potential way to solve the challenges 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 [2, 27, 35] that assume objects moving slightly between frames thus fail in low-frame-rate cases (shown in Fig. 1a). Besides, the larger displacement leads to the failure of motion models (like the Kalman filter [13]) with zero-velocity initialization assumptions. Thus, as shown in Fig. 1b, the effectiveness of methods [4, 47] which heavily rely on motion models is limited in low-frame-rate cases.

Second, the appearances and visibilities 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 occluded in the current frame. The influence is two-fold. 1) The large difference in appearance and the different occlusion states of the same target between frames make the appearance similarity measurement less discriminative (shown in Fig. 1c[1]). 2) The objects that appear or disappear around other continuously visible objects result in further noise in the similarity matrix.

[1]To evaluate the performance of the re-identification head of FairMOT [48], we disable the Kalman filter inside the original FairMOT method. Details can be found in Section 4.5.

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

Figure 1: Tracking results of modern trackers and our method in low-frame-rate videos. Bounding boxes with different colors represent different identities. 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 zoom in.

A model with a vision-based motion estimator together with a sufficient receptive field is expected to address the first challenge. The recent CenterTrack [49] 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) [8], it can handle large displacements between frames. However, the local nature makes CenterTrack suffer from the second challenge. It fails to give correct displacement estimations for objects having visibility flips across frames, 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, implying that the displacement estimation of this object is not reliable. The APP is built on the CenterTrack architecture by adding the APP head on top of the output from the backbone, in parallel with other detection and displacement prediction heads. In Fig. 1e, we highlight recognized appearing objects by indicating the predicted APP scores on the right top of corresponding bounding boxes.

Training a robust APP head is challenging. The ratio of emerging samples is fairly low (less than 10% among all instances) on the popularly used MOT17 dataset [26]. Thus, we propose a data augmentation strategy, leveraging a large-scale static image dataset [32] by randomly erasing objects in images to create emerging samples. Further, to overcome the local weakness of [49] and address the second challenge, we propose a two-stage matching policy according to the predictions from the APP head. In the first stage, we link non-emerging detections to tracklets with reliable displacement estimations. In the second stage, we drop the unreliable displacement estimations of emerging detections and link these detections by extending remaining tracklets with motion cues. In this two-stage manner, the noise inside the similarity matrix involved by visibility flips is reduced and the local nature is overcome by dropping unreliable displacement estimations and leveraging historical motion cues. In Fig. 1e, we employ white arrows to indicate motion estimations.

We conduct comprehensive experiments on the MOT17 [26] and MOT20 [10] datasets, demonstrating the robustness of our method in low-frame-rate cases especially in preserving identities of objects involving occlusions. With little additional computational overhead, our method achieves consistent improvement in IDF1 [30] score compared to the baseline [49] 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.
- A two-stage matching policy is proposed to reduce the noise involved in the similarity matrix and to leverage historical motion cues.
- Experimental results on public datasets in various low-frame-rate settings demonstrate the effectiveness and robustness of the proposed method.

2 RELATED WORK

2.1 Tracking by Detection

Most modern multi-object tracking methods follow the tracking-by-detection paradigm. These methods can be roughly grouped into online ones [6, 11, 12, 14, 19, 40, 41, 47, 50], which extend the tracklet at each time step, and offline ones [5, 9, 15, 23, 45], where the tracklets are updated after processing a batch of frames. In the tracking-by-detection paradigm, an object detector [7, 21, 28, 29] is firstly adopted to find all objects in each frame. Then, detections in different frames are associated by comparing the similarity of their features extracted through motion models [13, 31], re-identification (re-ID) models [14, 19, 40], or graph neural networks [5, 9, 12]. As detection is separated from the association, and the appearance features are usually discarded or extracted by an extra model during the association stage, the effectiveness or efficiency is thus limited.

2.2 Joint Detection and Tracking

To reduce the computational cost, recent works [2, 25, 27, 34, 38, 39, 42, 48, 49] make effort to accomplish both tasks with a single network. For example, Tracktor++ [2] explores the bounding box regression head of an object detector, such as Faster R-CNN [29], to predict the position of an object in the next frame, thereby converting a detector into a tracker. Chained-Tracker [27] takes two adjacent frames as input and regresses a pair of bounding boxes for the same target. CenterTrack [49] takes two adjacent frames together with a heatmap rendered from the tracked object centers as input and outputs detections together with their displacement

estimations. JDE [39] and FairMOT [48] jointly accomplish object detection and re-identification in a single network. JDE [39] uses the architecture of the Feature Pyramid Network (FPN) [20] and outputs a dense prediction map containing box classification results, box regression coefficients, and a dense embedding map. FairMOT [48] adopts a similar manner but replaces the backbone network with an encoder-decoder network [44]. Its detector and appearance embedding extractor act in a center-based manner. The fairness of the detection task and the re-identification task is studied in FairMOT. All these methods are designed to track objects in normal frame rate videos. However, their robustness with low-frame-rate inputs has not been well studied.

2.3 Tracking Objects in Challenging Scenarios

With the development of multi-object trackers, recent works [2, 31, 33, 36, 46, 47] start to focus on tracking multiple objects in challenging scenarios. For example, Tracktor++ [2] evaluates the performance of several state-of-the-art methods for tracking objects of small size, low visibilities, and long-term occlusions. Besides, they evaluate the robustness of their model with low-frame-rate inputs. ByteTrack [47], different from most tracking-by-detection methods which only associate detections whose scores are higher than a threshold, proposes to associate low score detections with motion cues. ArTIST [31] proposes a probabilistic autoregressive motion model to deal with long-term occlusions. Close to our approach, Tokmakov et al. [37] extends the CenterTrack [49] architecture from pairs of frames as input to arbitrary video sequences by injecting a convolutional gated recurrent unit (ConvGRU) [1]. During training, it supervises the displacement predictions of almost all objects observed in history, rather than only supervising those of objects which are both visible in adjacent frames. It thus addresses the temporally local weakness of CenterTrack [49]. However, as pointed out by the author [37], 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.

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 the objects of interest, and to assign a unique and consistent identity to each object across frames, outputing 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 [49], we define the notations in a local perspective for simplicity. In particular, we employ $\mathbb{L}^{t-1} = \{l_1^{t-1}, l_2^{t-1}, \cdots\}$ to denote tracklet set in the time step $t-1$, where each tracklet $l_j^{t-1} = (\mathbf{p}, \mathbf{s}, \mathbf{v}, w, id)$ is described by its center location $\mathbf{p} \in \mathbb{R}^2$, size $\mathbf{s} \in \mathbb{R}^2$, velocity estimation $\mathbf{v} \in \mathbb{R}^2$, 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 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 [43], we propose a solution with the following advances. Fig. 2 shows an overview of our method. Firstly, we build an APP head on top of the CenterTrack architecture [49] to recognize unreliable displacement estimations in Section 3.1. To improve the robustness of APP predictions, we propose a new augmentation strategy to pretrain the APP head on static image data in Section 3.2. Secondly, to reduce the noise in the similarity matrix involved by visibility flips and leverage historical motion cues, detections are associated with tracklets in a two-stage manner in Section 3.3.

3.1 Learning to Recognize Appearing Objects

We build our method on top of the CenterTrack [49] 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 three inputs first go through three individual 7×7 convolutional layers and then are accumulated and sent into a backbone network. Several individual heads are then adopted to produce the low-resolution heatmap $\hat{Y}^t \in [0, 1]^{\frac{W}{R} \times \frac{H}{R} \times 1}$, the size map $\hat{S}^t \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$, and the displacement map $\hat{D}^t \in \mathbb{R}^{\frac{W}{R} \times \frac{H}{R} \times 2}$, etc. $R = 4$ is the downsampling factor. The center location $\mathbf{p}^t \in \mathbb{R}^2$, box size $\mathbf{s}^t \in \mathbb{R}^2$, and corresponding displacement $\mathbf{d}^t \in \mathbb{R}^2$ of each detection o_i^t is then decoded from these maps. The displacement estimations $\hat{\mathbf{d}}^t$ are used to warp detection centers $\hat{\mathbf{p}}^t$ in I^t to their locations in I^{t-1} by calculating $\hat{\mathbf{p}}^t - \hat{\mathbf{d}}^t$. The detections $\mathbb{O}^t$ are then linked to tracklets $\mathbb{L}^{t-1}$ by matching the warped detection centers with the tracklet centers in a greedy manner, resulting in $\mathbb{L}^t$.

However, in the training stage of [49], 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 estimations 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.

Formally, our model, takes the set of $\{H^{t-1}, I^{t-1}, I^t\}$ as input and outputs all the estimations as [49] does, together with an extra APP map $A^t \in [0, 1]^{\frac{W}{R} \times \frac{H}{R} \times 1}$. We generate the ground-truth heatmap A^t by rendering Gaussian-shaped peaks [18] into A^t at the centers of emerging objects in frame t. Some visualized cases are shown in Fig. 3. Given a ground-truth heatmap A^t, we supervise the APP head with a training objective based on the focal loss [18, 21]:

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

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.

For the components shared with [49], we follow the same settings in training losses. The overall training objective is:

$$\mathcal{L} = \lambda_a \mathcal{L}_a + \mathcal{L}_{\mathbf{ct}}, \quad (2)$$

where $\mathcal{L}_{\mathbf{ct}} = \lambda_{\mathbf{p}} \mathcal{L}_{\mathbf{p}} + \lambda_{\mathbf{s}} \mathcal{L}_{\mathbf{s}} + \lambda_{\mathbf{d}} \mathcal{L}_{\mathbf{d}} + \lambda_{\mathbf{off}} \mathcal{L}_{\mathbf{off}}$ is the training objective of original CenterTrack model. $\lambda_a, \lambda_{\mathbf{p}}, \lambda_{\mathbf{s}}, \lambda_{\mathbf{d}}, \lambda_{\mathbf{off}}$ are hyperparameters used to balance the contribution of each loss component.

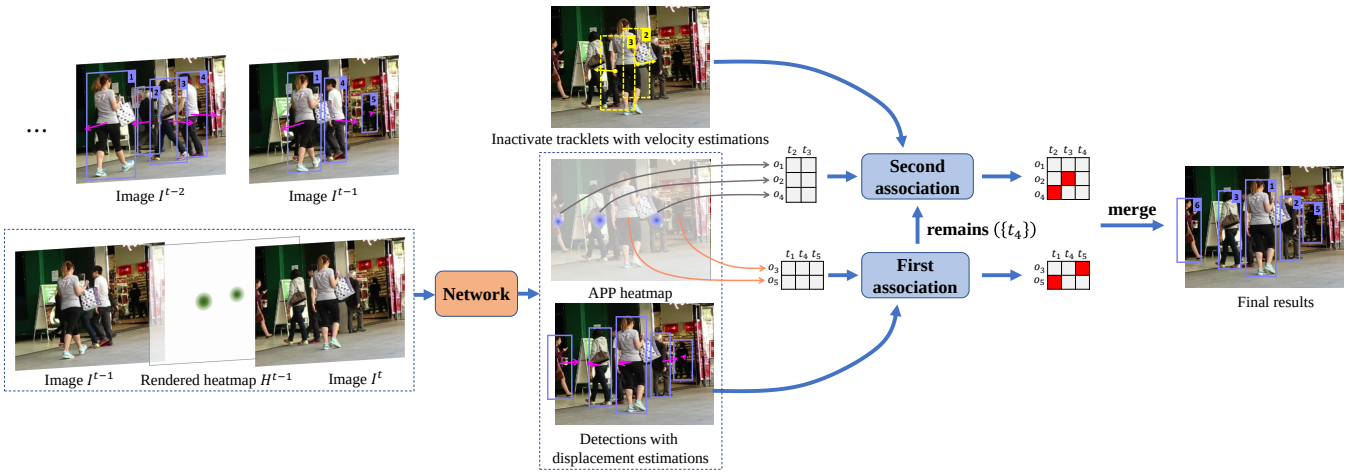


Figure 2: Overview of our method. Our model takes the same input set as [49], outputting detections and displacement estimations, together with a predicted APP heatmap. According to the APP predictions, the detections in frame I^t are grouped into emerging ones $\{o_1, o_2, o_4\}$ and non-emerging ones $\{o_3, o_5\}$. Note that the displacement estimations of emerging objects are not reliable. Then, the association is performed in a two-stage manner. In the first stage, 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. In the second stage, we associate remaining tracklets $\{t_2, t_3, t_4\}$ to remaining detections $\{o_1, o_2, o_4\}$ by extending tracklets through their historical velocity estimations. In this way, unreliable displacement estimations of emerging detections $\{o_1, o_2, o_4\}$ are recognized and dropped, and the identities of tracklets $\{t_2, t_3\}$ occluded in frame $t-1$ are preserved by motion cues.

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 Training Strategies

3.2.1 Training on Videos. We use the MOT17 [26] dataset to train our model. This dataset annotates the identity, bounding box, class, and visibility for each object of interest in each frame. For a given object i in frame t, we determine its appear label a_i^t by

$$a_i^t = \begin{cases} 1 & \text{if } vis_i^t > \tau_{vis} \text{ and } vis_i^{t-1} < \tau_{vis} \\ 0 & \text{otherwise} \end{cases}, \tag{3}$$

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 object i in frame t and $t-1$, respectively. After the label a_i^t is determined, we use the rendering function in [18] to render the ground-truth heatmap A^t.

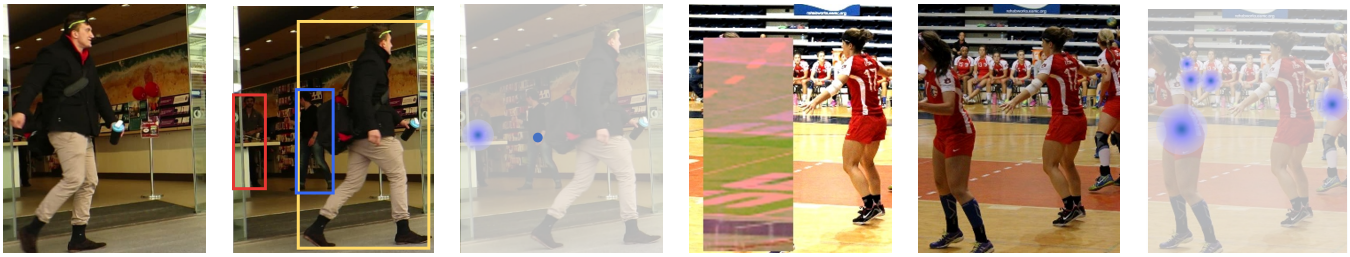
However, the visibility annotations provided by the MOT17 dataset are noisy. For a given object, its visibility is determined based on the intersection-over-union (IoU) of its bounding box against other bounding boxes. As shown in Fig. 3b, some annotated visibilities are incorrect, leading to noisy labels for the detection head and the APP head. To address it, the predictions of objects annotated with low visibility are ignored in training losses by adding a circular mask on the ground-truth detection heatmap and the ground-truth APP heatmap (shown in Fig. 3c). Further details on the circular masking policy are provided in the supplement.

When training our model on video data, low-frame-rate videos are generated by extracting frames from original videos through a fixed interval n_d. After that, for a given current frame I^t, another frame at time k is randomly sampled as I^{t-1} if $|k-t| < 3$, where t and k are frame indexes in low-frame-rate videos.

3.2.2 Training on Static Images. Training the APP head on MOT17 alone cannot provide satisfying estimations for the emergence of objects as the ratio of emerging samples is fairly low, typically less than 10% among all instances. In CenterTrack [49], a single image is randomly scaled and translated to create a pair of neighboring frames so that the network can be pretrained 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. In particular, 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 at one time and a new patch bg_{mixed} is generate by blending bg_1 and bg_2:

$$bg_{mixed} = \gamma bg_1 + (1-\gamma) bg_2, \tag{4}$$

where γ is a random weight following the uniform distribution of $[0.3, 0.7]$. Several other augmentations, such as random flipping, intensity jittering, and color channel shuffling are further adopted. More details can be found in the supplement. Fig. 3d to Fig. 3f visualize some training samples generated by the proposed randomly erasing strategy. As shown in Section 4.4, this augmentation strategy remarkably improves the robustness of the APP head.



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

Figure 3: Visualization of training samples. A pair of images I^{t-1} (a) and I^t (b) are sampled from the MOT17 dataset. The person inside the red box in I^t is not visible in the previous frame I^{t-1}. A positive response is thus placed on the ground-truth APP heatmap A^t (c). Besides, although the person inside the blue box in I^t is highly visible, its visibility is annotated to zero by the MOT17 dataset, as the blue box is covered by the orange one. A circular mask is placed on the ground-truth APP heatmap (c) to ignore the predictions of objects annotated with low visibilities. When training on static image data, the previous frame I^{t-1} (d) is simulated by performing randomly scaling, translating, and erasing on the current frame I^t (e). The corresponding ground-truth APP heatmap A^t is shown in (f).

3.3 Two-stage Matching with APP

To reduce the noise in the similarity matrix resulting from visibility flipped objects and to leverage historical motion cues, we introduce a two-stage matching strategy according to the appearing predictions from the APP head.

The pseudo code for the two-stage matching is shown in Algorithm 1. At each time step t, before matching, we group the detections $\mathbb{O}^t$ into emerging ones $\mathbb{O}^t_{ap}$ and otherwise $\mathbb{O}^t_{ac}$ by comparing their APP scores with a threshold τ_{ap}. Meanwhile, the tracklet set $\mathbb{L}^{t-1}$ is grouped into two subsets, active subset $\mathbb{L}^{t-1}_{ac}$ and inactive subset $\mathbb{L}^{t-1}_{inac}$ according to whether the tracklets have been matched in frame $t-1$. As a result, the input consists of a set of emerging objects $\mathbb{O}^t_{ap}$, a set of otherwise detections $\mathbb{O}^t_{ac}$, a set of active tracklets $\mathbb{L}^{t-1}_{ac}$, and a set of inactive tracklets $\mathbb{L}^{t-1}_{inac}$.

In the first stage (line 2 to 9 in Algorithm 1), we link non-emerging detections $\mathbb{O}^t_{ac}$ to active tracklets $\mathbb{L}^{t-1}_{ac}$. Center locations of non-emerging detections $\mathbb{O}^t_{ac}$ are warped to those in frame $t-1$ by their displacement estimations. The warped detection centers are then matched with centers of active tracklets $\mathbb{L}^{t-1}_{ac}$ by Hungarian Algorithm [17], taking the Euclidean distance as the similarity measurement. Each matched tracklets l is updated by the corresponding detection o. Note that the velocities of matched tracklets are updated by the displacement estimations (line 7 in Algorithm 1).

In the second stage (line 10 to 17 in Algorithm 1), we link the remaining tracklets ($\mathbb{L}^{t-1}_{inac} \cup \mathbb{L}^{t-1}_{rem1}$) to the remaining detections ($\mathbb{O}^t_{ap} \cup \mathbb{O}^t_{rem1}$). Center locations of tracklets ($\mathbb{L}^{t-1}_{inac} \cup \mathbb{L}^{t-1}_{rem1}$) are warped to those in frame t by their velocity estimations. 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 (line 15 in Algorithm 1).

Finally, we update the locations of the remained tracklets $\mathbb{L}^{t-1}_{rem2}$ with a constant velocity assumption (line 18 to 21 in Algorithm 1) and initialize new tracklets for remained detections $\mathbb{O}^t_{rem2}$ with zero velocities (line 22 to 25 in Algorithm 1).

4 EXPERIMENTS

4.1 Datasets and Evaluation Metrics

Datasets. We use two widely-used benchmarks, MOT17 [26] and MOT20 [10], 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[32]. Following common practices in [47–49], 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 1 to n_d.

Evaluation Metrics. We use the IDF1 score [30] and the CLEAR metrics [3], including MOT Accuracy (MOTA), number of identity switches (IDs), false positives (FP), and false negatives (FN). MOTA is computed based on FP, FN, and IDs. As the numbers of FP and FN are usually much greater than IDs, MOTA is dominated by detection performance. Instead, we focus on the IDF1 [30] score and IDs in our experiments, which evaluate the ability of identity preservation and focus more on the association performance.

4.2 Implementation Details

For the components of our model shared with CenterTrack [49], we follow their architecture and training details. In particular, we use DLA-34 [44] as the network backbone. We adopt the Adam [16] optimizer with a learning rate of $1.25e-4$ and a batch size of 32. The input images are resized and padded to the resolution of 960×544. In addition to the data augmentation operations adopted by CenterTrack [49], we adopt the random erasing operation proposed in Section 3.2.2 when pretraining the model on the CrowdHuman dataset. The model is firstly pretrained on the CrowdHuman dataset

Algorithm 1: Pseudo Code of Cascade 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 $\mathbb{L}^{t}$

```
/* group detections beasd on APP scores */
Initialization: L^t ← ∅

/* Stage 1: association with displacements */
associate O^t_ac and L^{t-1}_ac by estimated displacements
O^t_rem1 ← remaining objects from O^t_ac
L^{t-1}_rem1 ← remaining objects from L^{t-1}_ac
for l in (L^{t-1}_ac \ L^{t-1}_rem1) do
    l.location ← o.location
    l.velocity ← o.displacement
    L^t ← L^t ∪ {l}
end

/* Stage 2: association with velocities */
associate (O^t_ap ∪ O^t_rem1) and (L^{t-1}_inac ∪ L^{t-1}_rem1) by velocities
O^t_rem2 ← remaining objects from (O^t_ap ∪ O^t_rem1)
L^{t-1}_rem2 ← remaining objects from (L^{t-1}_inac ∪ L^{t-1}_rem1)
for l in (L^{t-1}_inac ∪ L^{t-1}_rem1 \ L^{t-1}_rem2) do
    l.location ← o.location
    l.velocity ← mean displacement
    L^t ← L^t ∪ {l}
end

/* update inactive tracklets */
for l in L^{t-1}_rem2 do
    l.location ← l.location + l.velocity
    L^t ← L^t ∪ {l}
end

/* initialize new tracklets */
for o in O^t_rem2 do
    o.velocity ← 0
    L^t ← L^t ∪ {o}
end
```

for 160 epochs. The learning rate is dropped by a factor of 10 at the 140^{th} epoch. Then we fine-tune the model on the MOT17 dataset by 80 epochs and drop the learning rate by a factor of 10 at the 60^{th} epoch. During training, the visibility threshold τ_{vis} is 0.25 and the loss weight λ_a is 1. During inferring, the APP score threshold τ_{ap} is 0.4. We follow the default settings of other hyper-parameters from CenterTrack [49]. We provide further details in the supplement.

4.3 Appear Prediction

We first investigate the benefit of engaging the APP head in low-frame-rate cases. In this study, the low-frame-rate inputs are generated by extracting frames from original videos through a fixed interval of $n_d = 10$.

Table 1: Comparison between several model variants on the MOT17 validation set. "Pred" is short for predictions. "Pred-inv" represents deriving predictions by temporally flipping the order of the input frames.

Appear	Disappear	IDF1(↑)	IDs(↓)	FP(↓)	FN(↓)
-	-	64.7	2304	1819	14824
Pred	-	65.6	**2050**	1817	14822
-	Pred	64.9	2466	**1812**	**14817**
-	Pred-inv	65.2	2246	1817	14822
Pred	Pred-inv	**65.7**	2103	1815	14820

By comparing the first two rows in Table 1, a reduction of 11% on IDs is observed by using the appearing prediction and the two-stage matching policy. Further, in Fig. 4, we analyze its improvement especially in preserving the identities of objects involved in occlusions.

In particular, 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. 4 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 yellow bars to indicate the ratio of visibility flips r_{vflip} calculated by

$$r_{vflip} = \frac{n_{vflip}}{n_{gt}}, \tag{5}$$

where n_{vflip} is the sum of visibility flips for all objects of interest, 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

$$r_{IDs} = \frac{IDs}{n_{vflip}}, \tag{6}$$

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. 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. Qualitative results can be found in the supplement.

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.

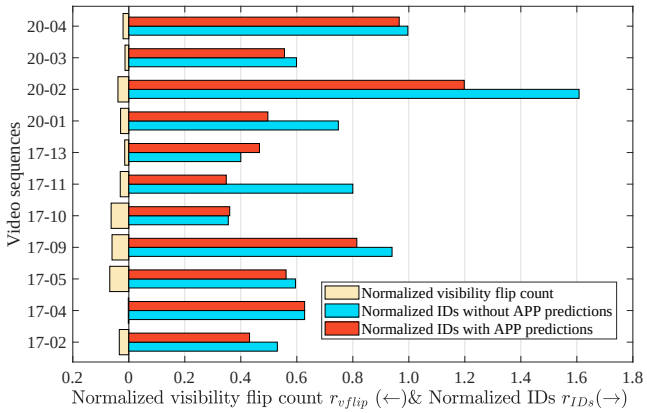


Figure 4: The performance improvement on preserving identities of objects involved in occlusions.

We first train a DISAPP head following the same settings as those for training the APP head. However, the IDs (third row in Table 1) even becomes higher 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 pretrained together with the backbone network on the COCO dataset [22]. 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 accurate "disappearing" predictions from the APP head by 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 bring a reduction of 2.5% on IDs (the fourth row in Table 1). By comparing the second and the fourth rows, 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 predictions of emerging detections in the current frame may lead them to compete for the tracklets against the correct detections, resulting in the identity switch twice. Instead, at the current time step, the model does not predict displacements for tracked objects in the previous frame. No competition is thus involved by disappeared objects. As a result, we only keep the APP head in the following experiments, since no extra improvement is observed by engaging DISAPP predictions.

4.4 Ablation Study

Table 2. Further, we ablate other components of our method, including the proposed data augmentation policy, the proposed two-stage matching strategy, and the optical-flow-based motion estimator.

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 remarkable reduction of 512 in IDs (second row in Table 2).

Table 2: Variants evaluated on the MOT17 validation set. "Aug." indicates the proposed data augmentation policy. "Hght." stands for engaging height distance in association. "Thresh." represents using step-wise thresholds in cascade matching.

Aug.	Hght.	Thresh.	IDF1(↑)	IDs(↓)	FP(↓)	FN(↓)
			63.8	2562	2064	15620
✓			65.6	2050	1817	14822
✓	✓		67.0	1940	1819	14824
✓		✓	69.4	1731	**1801**	**14806**
✓	✓	✓	**70.3**	**1686**	1802	14807

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	IDF1(↑)	IDs(↓)	FP(↓)	FN(↓)
Normal	Opt. + Static	69.5	353	2018	14528
Normal	Opt. + Constant	68.7	326	**2005**	**14515**
Normal	Kalman Filter	**72.0**	**321**	2011	14543
Low	Opt. + Static	64.6	3971	**1785**	**14790**
Low	Opt. + Constant	**70.3**	**1686**	1802	14807
Low	Kalman Filter	60.4	5679	1966	15034

Height Gating. While CenterTrack [49] drops the similarity of box sizes during association, we pick it up by rejecting matchings with unreasonable height distances. Formally, the height distance d^h_{ij} of a detection bounding box i and a tracklet bounding box j is calculated by

$$d^h_{ij} = \frac{|h_i - h_j|}{max(h_i, h_j)}, \tag{7}$$

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 $\tau_h = 0.33$ are rejected.

Step-wise Matching Thresholds. Originally, we follow CenterTrack to reject an assignment (o^t_i, l^{t-1}_j) if the Euclidean distance between their centers (after warping) is larger than $\kappa = \sqrt{w^{t-1}_j h^{t-1}_j}$, where w^{t-1}_j and h^{t-1}_j are the width and height of tracklet l^{t-1}_j. To avoid ambiguous assignments in the first stage, a step-wise threshold policy is adopted by setting the thresholds to $\frac{2}{3}\kappa$ and κ in the first and the second matching stages, respectively.

Motion Models. We evaluate the effectiveness of a Kalman filter and the optical-flow-based motion estimator that we adopt from CenterTrack with different frame rate inputs in Table 3. For the optical-flow-based estimator, we also evaluate two motion models, a static model, and a constant velocity model, which are adopted to leverage historical motion cues, as introduced in Section 3.3. For low-frame-rate evaluations, we extract frames from original videos through a fixed interval of $n_d = 10$. We have the following findings. First, though the static model performs reasonably well in normal-frame-rate videos due to the large overlap of the same

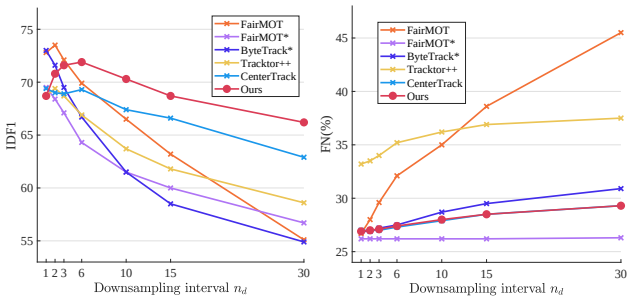


Figure 5: Comparison between our method and several state-of-the-art methods on the MOT17 validation set with different frame rate inputs. The IDF1 score (a) and the FN (b) are reported.

target between frames, it fails in low-frame-rate cases. Second, although the Kalman filter gets a better IDF1 score of 72.0% in normal-frame-rate cases, the remarkable advantages of our method are demonstrated when the frame rate becomes lower. We suspect that, in normal-frame-rate cases, the motion information contains a large portion of noise caused by the jitter of detections. Kalman filter can filter out such measurement noise while the optical-flow-based estimator and the constant velocity model are influenced by such noise. However, the influence of such noise is reduced as the displacements of objects become larger in low-frame-rate videos. Without a good initial velocity estimation (the velocity of an object is usually initialized to zero) or a good velocity measurement, the Kalman filter fails to chase objects in low-frame-rate cases.

Full Method. Based on the analysis above, we adopt the augmentation policy, height gating, step-wise matching thresholds, and the constant velocity assumption as our full method, whose performance is shown in the last row in Table 2.

4.5 Comparison with SOTA Methods

In this section, we compare our method with several existing representative methods in the settings of different frame rate inputs: FairMOT [48] (re-identification-based method), ByteTrack [47] (motion-model-based method), Tracktor++ [2] (detector-based method), CenterTrack [49] (optical-flow-based method). All these methods, except FairMOT, are evaluated with the same detection set. Evaluating FairMOT with private detections may harm the extracting of re-identification embeddings. The detector inside FairMOT is thus kept during evaluation, which performs slightly better than ours. Due to the space limit, We report the IDF1 score and the FN in Fig. 5. Full results can be found in the supplement.

FairMOT. FairMOT [48] jointly accomplishes the tasks of object detection and re-identification in a single network. Besides, it adopts a Kalman filter to exploit motion cues. Associations with low overlaps are rejected. To reduce false alarms, it allows an object to spawn a new tracklet only if it is continuously visible 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 test the performance of the re-identification head of FairMOT [48], we disable the Kalman filter together with the strict spawning strategy inside the original FairMOT method. While FN stops increasing, the drops in IDF1 indicate the re-identification performance suffers from the large appearance variations of objects in low-frame-rate cases.

ByteTrack*. ByteTrack [47] proposes to associate low score detections with motion cues. Similar to FairMOT, Kalman filter and the strict spawning strategy are adopted. To avoid unexpected increases in FN, the strict spawning strategy is disabled when evaluating the ByteTrack in our experiments. The method is evaluated with the detections from our model. Though its performance in normal-frame-rate cases is good, it drops more quickly than other methods since no other information (e.g., appearance embeddings, optical flows) except motion cues is leveraged in the association.

Tracktor++. Tracktor++ [2], 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 non-maximum suppression (NMS) operations inside the Tracktor++. As its primary assumption does not hold when the frame rate becomes low, the IDF1 decreases quickly.

CenterTrack. To evaluate CenterTrack [49], we use the same detection set from our model but use its original tracking management. Compared with other methods, the performance drop of CenterTrack is not that significant. With a temporally local displacement estimator and a large receptive field, it is more robust to large displacements and large camera motions in low-frame-rate cases.

Ours. Enabled by the proposed APP head and the two-stage matching strategy, our method overcomes the local weakness of CenterTrack [49], thus achieving consistent improvement in IDF1 score compared to the CenterTrack [49] and outperforms the state-of-the-art methods in low-frame-rate cases.

5 CONCLUSION

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 flips in low-frame-rate cases. A two-stage matching policy is further proposed to reduce the noise inside the similarity matrix before matching and to leverage historical motion cues. Experiments on public datasets with various frame rate settings verify the effectiveness and robustness of our method.

ACKNOWLEDGMENTS

This work was supported in part by NSFC under Grants 62088101, U21A20456, 62103372, the 5G Open Laboratory of Hangzhou Future Sci-Tech City and the Fundamental Research Funds for the Central Universities.

REFERENCES

[1] Nicolas Ballas, Li Yao, Chris Pal, and Aaron Courville. 2015. Delving deeper into convolutional networks for learning video representations. *arXiv preprint arXiv:1511.06432* (2015).

[2] Philipp Bergmann, Tim Meinhardt, and Laura Leal-Taixe. 2019. Tracking without bells and whistles. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 941–951.

[3] Keni Bernardin and Rainer Stiefelhagen. 2008. Evaluating multiple object tracking performance: the clear mot metrics. *EURASIP Journal on Image and Video Processing* 2008 (2008), 1–10.

[4] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. 2016. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*. IEEE, 3464–3468.

[5] Guillem Brasó and Laura Leal-Taixé. 2020. Learning a neural solver for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 6247–6257.

[6] Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. 2021. Transmot: Spatial-temporal graph transformer for multiple object tracking. *arXiv preprint arXiv:2104.00194* (2021).

[7] Xuangeng Chu, Anlin Zheng, Xiangyu Zhang, and Jian Sun. 2020. Detection in crowded scenes: One proposal, multiple predictions. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 12214–12223.

[8] Jifeng Dai, Haozhi Qi, Yuwen Xiong, Yi Li, Guodong Zhang, Han Hu, and Yichen Wei. 2017. Deformable convolutional networks. In *Proceedings of the IEEE international conference on computer vision*. 764–773.

[9] Peng Dai, Renliang Weng, Wongun Choi, Changshui Zhang, Zhangping He, and Wei Ding. 2021. Learning a proposal classifier for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 2443–2452.

[10] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. 2020. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003* (2020).

[11] Song Guo, Jingya Wang, Xinchao Wang, and Dacheng Tao. 2021. Online multiple object tracking with cross-task synergy. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 8136–8145.

[12] Jiawei He, Zehao Huang, Naiyan Wang, and Zhaoxiang Zhang. 2021. 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*. 5299–5309.

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

[14] Chanho Kim, Li Fuxin, Mazen Alotaibi, and James M Rehg. 2021. 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*. 9553–9562.

[15] Chanho Kim, Fuxin Li, Arridhana Ciptadi, and James M Rehg. 2015. Multiple hypothesis tracking revisited. In *Proceedings of the IEEE international conference on computer vision*. 4696–4704.

[16] Diederik P Kingma and Jimmy Ba. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

[17] Harold W Kuhn. 1955. The Hungarian method for the assignment problem. *Naval research logistics quarterly* 2, 1-2 (1955), 83–97.

[18] Hei Law and Jia Deng. 2018. Cornernet: Detecting objects as paired keypoints. In *Proceedings of the European conference on computer vision (ECCV)*. 734–750.

[19] Chao Liang, Zhipeng Zhang, Yi Lu, Xue Zhou, Bing Li, Xiyong Ye, and Jianxiao Zou. 2020. Rethinking the competition between detection and reid in multi-object tracking. *arXiv preprint arXiv:2010.12138* (2020).

[20] Tsung-Yi Lin, Piotr Dollár, Ross Girshick, Kaiming He, Bharath Hariharan, and Serge Belongie. 2017. Feature pyramid networks for object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2117–2125.

[21] Tsung-Yi Lin, Priya Goyal, Ross Girshick, Kaiming He, and Piotr Dollár. 2017. Focal loss for dense object detection. In *Proceedings of the IEEE international conference on computer vision*. 2980–2988.

[22] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. 2014. Microsoft coco: Common objects in context. In *European conference on computer vision*. Springer, 740–755.

[23] Wenhan Luo, Björn Stenger, Xiaowei Zhao, and Tae-Kyun Kim. 2018. Trajectories as topics: Multi-object tracking by topic discovery. *IEEE Transactions on Image Processing* 28, 1 (2018), 240–252.

[24] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, and Tae-Kyun Kim. 2021. Multiple object tracking: A literature review. *Artificial Intelligence* 293 (2021), 103448.

[25] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. 2021. Trackformer: Multi-object tracking with transformers. *arXiv preprint arXiv:2101.02702* (2021).

[26] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. 2016. MOT16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831* (2016).

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

[28] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 2016. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 779–788.

[29] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. 2015. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems* 28 (2015).

[30] Ergys Ristani, Francesco Solera, Roger Zou, Rita Cucchiara, and Carlo Tomasi. 2016. Performance measures and a data set for multi-target, multi-camera tracking. In *European conference on computer vision*. Springer, 17–35.

[31] Fatemeh Saleh, Sadegh Aliakbarian, Hamid Rezatofighi, Mathieu Salzmann, and Stephen Gould. 2021. Probabilistic tracklet scoring and inpainting for multiple object tracking. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 14329–14339.

[32] Shuai Shao, Zijian Zhao, Boxun Li, Tete Xiao, Gang Yu, Xiangyu Zhang, and Jian Sun. 2018. CrowdHuman: A Benchmark for Detecting Human in a Crowd. *arXiv preprint arXiv:1805.00123* (2018).

[33] Daniel Stadler and Jurgen Beyerer. 2021. Improving multiple pedestrian tracking by track management and occlusion handling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 10958–10967.

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

[35] ShiJie Sun, Naveed Akhtar, XiangYu Song, HuanSheng Song, Ajmal Mian, and Mubarak Shah. 2020. Simultaneous detection and tracking with motion modelling for multiple object tracking. In *European Conference on Computer Vision*. Springer, 626–643.

[36] Ramana Sundararaman, Cedric De Almeida Braga, Eric Marchand, and Julien Pettre. 2021. Tracking pedestrian heads in dense crowd. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3865–3875.

[37] Pavel Tokmakov, Jie Li, Wolfram Burgard, and Adrien Gaidon. 2021. Learning to track with object permanence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*. 10860–10869.

[38] Qiang Wang, Yun Zheng, Pan Pan, and Yinghui Xu. 2021. Multiple object tracking with correlation learning. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 3876–3886.

[39] Zhongdao Wang, Liang Zheng, Yixuan Liu, Yali Li, and Shengjin Wang. 2020. Towards real-time multi-object tracking. In *European Conference on Computer Vision*. Springer, 107–122.

[40] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. 2017. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*. IEEE, 3645–3649.

[41] Jiarui Xu, Yue Cao, Zheng Zhang, and Han Hu. 2019. Spatial-temporal relation networks for multi-object tracking. In *Proceedings of the IEEE/CVF international conference on computer vision*. 3988–3998.

[42] Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan, Daniela Rus, and Xavier Alameda-Pineda. 2021. Transcenter: Transformers with dense queries for multiple-object tracking. *arXiv preprint arXiv:2103.15145* (2021).

[43] Tianwei Yin, Xingyi Zhou, and Philipp Krahenbuhl. 2021. Center-based 3d object detection and tracking. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*. 11784–11793.

[44] Fisher Yu, Dequan Wang, Evan Shelhamer, and Trevor Darrell. 2018. Deep layer aggregation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*. 2403–2412.

[45] Yang Zhang, Hao Sheng, Yubin Wu, Shuai Wang, Wei Ke, and Zhang Xiong. 2020. Multiplex labeling graph for near-online tracking in crowded scenes. *IEEE Internet of Things Journal* 7, 9 (2020), 7892–7902.

[46] Yang Zhang, Hao Sheng, Yubin Wu, Shuai Wang, Wei Ke, and Zhang Xiong. 2020. Multiplex labeling graph for near-online tracking in crowded scenes. *IEEE Internet of Things Journal* 7, 9 (2020), 7892–7902.

[47] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. 2021. ByteTrack: Multi-Object Tracking by Associating Every Detection Box. *arXiv preprint arXiv:2110.06864* (2021).

[48] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. 2021. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision* 129, 11 (2021), 3069–3087.

[49] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. 2020. Tracking objects as points. In *European Conference on Computer Vision*. Springer, 474–490.

[50] Zongwei Zhou, Wenhan Luo, Qiang Wang, Junliang Xing, and Weiming Hu. 2020. Distractor-aware discrimination learning for online multiple object tracking. *Pattern Recognition* 107 (2020), 107512.

A FURTHER IMPLEMENTATION DETAILS

CrowdHuman Dataset and the Random Erasing Strategy. The training split of the CrowdHuman dataset [32] contains 15000 images characterized by high density and heavy occlusion. As shown in Fig. 6, both the visible-region bounding box and the full-body (amodal) bounding box for each human instance are provided. We follow the training settings in [49], using the input resolution of 512×512, and applying the same augmentations (random flipping, intensity jittering, random shift, random scaling, etc.) to the additional input H^{t-1} as [49]. To simulate the larger displacements in low-frame-rate cases, we enlarge the x-axis random shift ratio in [49] to 0.1, but keep the y-axis random shift ratio as 0.05. To erase a visible human instance, we fill the corresponding visible-region bounding box with a mixed background patch, as introduced in Section 3.2.2 of the main paper. However, we require the model to regress the full-body bounding box for each object to align with MOT17 [26].

As the objects in the Crowdhuman dataset are usually partially occluded, erasing a visible objects may destroy several surrounding objects. To minimize destroying other objects and to avoid confusable "appearing" labels, we first check the overlap area ratios between objects and determine whether an object can be erased. An object is allowed to be erased only if it overlaps with surrounding objects slightly (less than 15%) or heavily enough (greater than 85%). Among all instances, about 15% visible objects are randomly erased.

Figure 6: An illustrative example of visible-region bounding box and full-body bounding box annotations.

MOT Datasets and the Circular Masking Policy. For the MOT17 [26] and MOT20 [10] datasets, we use the input resolution of 960×544. No random erasing is adopted. Individual models are trained and evaluated on each dataset. For a given frame, each object is annotated with a full-body bounding box, a tracking identity, and a visibility ratio. CenterTrack [49] drops annotations of objects with low visibility. As a result, positive detection responses on these objects are punished by the loss function. However, as introduced in Section 3.2.1 of the main paper, the visibility annotations are noisy. Dropping these annotations may lead to ambiguous supervision. To avoid it, we ignore the predictions on objects annotated with low visibility. Since no visible-region bounding box annotations are available on MOT17 or MOT20, we place circular masks on the ground-truth detection heatmap Y^t and the ground-truth appearing heatmap A^t at the center locations of instances with low visibility annotations, so that predictions on other instances are not influenced. Fig. 3c in the main paper shows an example. Formally, the prediction on the pixel (x, y) is ignored if

$$exp(-\frac{(x-x_i)^2+(y-y_i)^2}{2\sigma_i^2}) > 0.3, \tag{8}$$

where (x_i, y_i) is the ground-truth center location of a low visible object i, σ_i is a function of the object size [18]. As shown in Table 4, the remarkable reduction in FN and the improvement in IDF1 score validate the effectiveness of the above masking policy. The experiments are conducted with low-frame-rate inputs of the sampling interval $n_d = 10$.

Table 4: Comparison between models with or without the circular masking policy on the MOT17 validation set (n_d = 10).

Circular mask	MOTA(↑)	IDF1(↑)	MT(↑)	ML(↓)	FP(↓)	FN(↓)	IDs(↓)
×	61.9	68.3	39.5	21.8	1741	16998	1537
✓	65.5	70.3	43.5	18.2	1802	14807	1686

Table 5: 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

B BUFFER SIZE

A proper buffer size, above which an inactive tracklet is terminated, is required for preserving the identities of objects involved in occlusions. In Fig. 7, we search the optimal buffer size in low-frame-rate cases ($n_d = 10$). According to Fig. 7, we adopt the buffer size of 10 for low-frame-rate videos of the sampling interval $n_d = 10$. Table 5 lists the buffer sizes we adopted for different n_d settings when evaluating our method and state-of-the-art methods in Section 4.5 of the main paper.

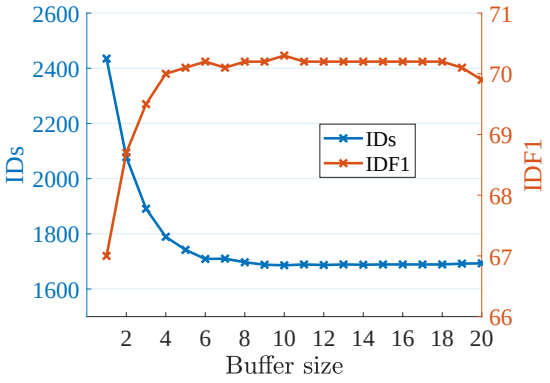


Figure 7: Tracking performance with different buffer size settings on the MOT17 validation set (n_d = 10).

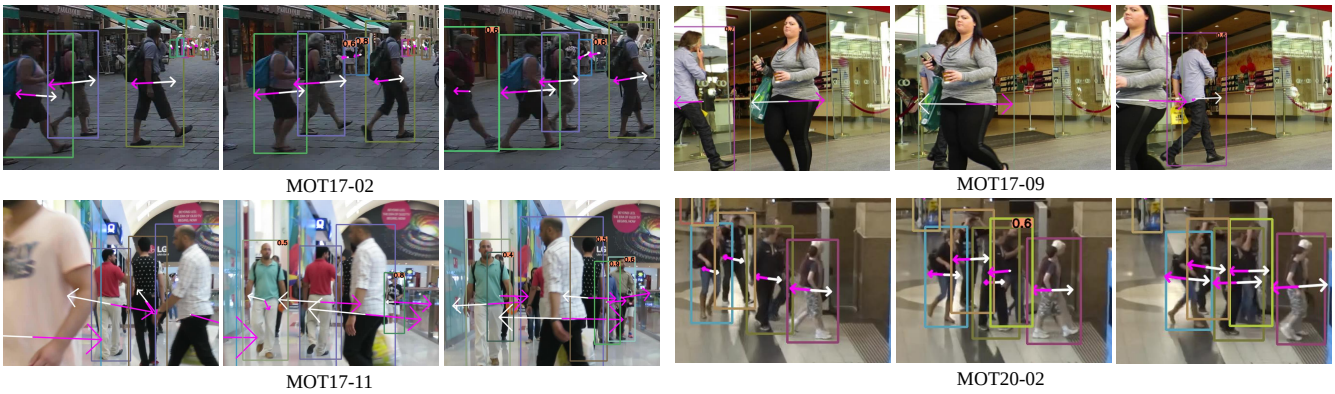


Figure 8: Qualitative results of our method. Each row shows three consecutive frames. Bounding boxes with different colors represent different identities. We highlight recognized appearing objects by indicating the predicted APP scores on the right top of corresponding bounding boxes. Pink arrows indicate displacement estimations, which are not reliable for appearing objects. White arrows indicate the updated velocity estimations. Best viewed in color and by zoom in.

C FULL RESULTS OF SOTA METHODS

We list the full results of evaluating our method and state-of-the-art methods with different frame settings in Table 6, Table 7, and Table 8.

The metrics include MOT accuracy (MOTA) [3], identity F1 score (IDF1) [30], mostly tracked (MT), mostly lost (ML), false positives (FP), false negatives (FN), and 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. The MOTA is calculated by:

$$MOTA = 1 - \frac{\sum_t (FN_t + FP_t + IDs_t)}{\sum_t GT_t}, \tag{9}$$

where t is the frame index, FN_t, FP_t, IDs_t, and GT_t are the number of false negatives, false positives, identity switches, and ground-truth bounding boxes in frame I_t, respectively. Note that a lower value of IDs, such as the fourth entry (Tracktor++) in Table 6, does not necessarily indicate a better association performance if it is accompanied by a much higher FN.

All these methods, except FairMOT, are evaluated with our detection set. Our detector, enabled by the circular masking policy, performs better than that of the baseline [49]. As mentioned in Section 4.5 of the main paper, evaluating FairMOT with other detection sets may harm the extracting of re-identification embeddings. FairMOT is thus evaluated with its own detection set.

Table 6: Full results of evaluating methods on the MOT17 validation set ($n_d = 1$).

Method	MOTA(↑)	IDF1(↑)	MT(↑)	ML(↓)	FP(↓)	FN(↓)	IDs(↓)
FairMOT [48]	**69.1**	72.8	42.2	15.6	**1977**	14439	298
FairMOT* [48]	68.4	69.5	**42.8**	**14.5**	2369	**14138**	541
ByteTrack* [47]	68.8	**73.0**	42.5	19.8	2047	14497	262
Tracktor++ [2]	58.3	68.7	36.6	19.8	4427	17946	**146**
CenterTrack [49]	68.7	69.4	41.3	18.6	2016	14526	347
Ours	68.7	68.7	41.3	18.9	2005	14515	326

Table 7: Full results of evaluating methods on the MOT17 validation set ($n_d = 10$).

Method	MOTA(↑)	IDF1(↑)	MT(↑)	ML(↓)	FP(↓)	FN(↓)	IDs(↓)
FairMOT [48]	58.3	66.5	25.8	28.6	**1107**	18577	2436
FairMOT* [48]	61.1	61.5	**46.3**	**13.8**	2393	**14150**	4454
ByteTrack* [47]	58.1	61.5	41.7	20.6	1729	15175	5282
Tracktor++ [2]	49.4	63.7	32.8	21.5	4761	19080	2824
CenterTrack [49]	65.0	67.4	43.7	18.4	1788	14793	1941
Ours	**65.5**	**70.3**	43.5	18.2	1802	14807	**1686**

Table 8: Full results of evaluating methods on the MOT17 validation set ($n_d = 30$).

Method	MOTA(↑)	IDF1(↑)	MT(↑)	ML(↓)	FP(↓)	FN(↓)	IDs(↓)
FairMOT [48]	44.5	55.1	25.0	36.2	**922**	23580	4274
FairMOT* [48]	55.5	56.7	**51.4**	**15.7**	2389	**14195**	7434
ByteTrack* [47]	49.4	54.9	42.8	25.4	1438	15668	8502
Tracktor++ [2]	42.7	58.6	36.7	25.1	4815	18599	5051
CenterTrack [49]	59.3	62.9	46.3	22.2	1563	14833	4205
Ours	**59.8**	**66.2**	46.3	22.2	1591	14861	**3890**

D QUALITATIVE RESULTS

In Fig. 8, we visualize four cases selected from the validation sets of MOT17 and MOT20. It can be observed that emerging objects are accurately recognized and the identities of objects involved in occlusions are preserved.