OBSERVATION-CENTRIC SORT: RETHINKING SORT FOR ROBUST MULTI-OBJECT TRACKING

Anonymous authors
Paper under double-blind review

ABSTRACT

Recent advances in object detection and re-identification have greatly improved the performance of Multi-Object Tracking (MOT) methods, but progress in motion modeling has been limited. The motion model is a key component of many MOT methods and is commonly used to predict an object's future position. However, mainstream motion models in MOT naively assume that object motion is linear. They rely on detections on each frame as the observation value to supervise motion models. However, in practice, the observations can be noisy and even missing, especially in crowded scenes, which greatly degrade the performance of existing MOT methods. In this work, we show that a simple filtering-based motion model can still obtain state-of-the-art tracking performance if proper care is given to missing observations and noisy estimates. We emphasize the role of observations when recovering tracks from being lost and reducing the error accumulated by the assumption of linear motion when the target is lost. In contrast to the popular motion-based method SORT, which is estimation-centric, we name our method **O**bservation-**C**entric **SORT** (OC-SORT). It remains simple, online, and real-time but improves robustness over occlusion and non-linear motion. It achieves state-of-the-art on multiple MOT benchmarks, including MOT17, MOT20, KITTI, head tracking, and especially DanceTrack where the object motion is highly non-linear.

1 INTRODUCTION

We aim to develop a motion model-based multi-object tracking (MOT) method that is robust to occlusion. Most existing motion model-based algorithms assume that the tracking targets have a constant velocity, which is called linear motion assumption. This assumption breaks in many practical scenarios, but it still works because when the interval between time steps is small enough, the motion in this short period can be reasonably approximated as linear. In this work, we are motivated by the fact that most of the errors from motion model-based tracking methods occur when occlusion and non-linear motion happen together. To mitigate the adverse effects caused thereby, we first rethink current motion models and recognize some limitations. Then, we propose to address them for more robust tracking performance, especially in occlusions.

As the main branch of motion model-based tracking, filtering-based methods assume a motion function to predict the state of objects on future time steps, which are called state "estimations". Besides estimations, they leverage an observation model, such as an object detector, to derive the state "observations" of target objects. Observations usually serve as the auxiliary information to adjust the parameters in filters and the trajectories are still extended by the state estimations. Among this line of works, the most widely used one is SORT [4], which uses a Kalman filter (KF) to estimate object states. We argue that estimations are more likely to be unreliable under occlusion compared to the observations from modern detectors. Therefore, we present a different perspective that, instead of being centric to estimations, we put observations in a more important role in prolonging tracks.

We begin with rethinking SORT and recognizing its **three limitations**: (1) although the high frame rate is the key to approximating the object motion as linear, it also amplifies the model's sensitivity to the noise of state estimations. Specifically, between consecutive frames of a high frame-rate video, we demonstrate that the noise of displacement of the object can be of the same magnitude as the actual object displacement, leading to the estimated object velocity by KF suffering from a large variance. (2) The noise of state estimations by KF is further accumulated through the time when there is no observation matched to existing trajectories. We prove that the error accumulation of

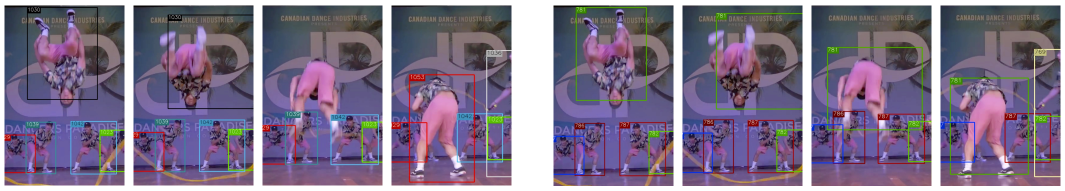

(a) SORT **(b)** The proposed OC-SORT

Figure 1: Samples from the results on DanceTrack [54]. SORT and OC-SORT use the same detection results. On the third frame, SORT encounters an ID switch for the backflip target while ours not.

object position estimations by KF is of square-order with respect to the time of the target's being untracked. (3) Given the development of modern detectors, single-frame detections usually have lower noise than the state estimations propagated along time steps. However, SORT is designed centric to the state estimations and observation is not necessary for updating KF parameters.

To relieve the negative effect of these limitations, we propose **two main innovations** in this work: (1) we design a module to use object state observations to reduce the accumulated error during the track's being lost in a backtrack fashion. We name it Observation-centric Online Smoothing (OOS). To be precise, on the time step that an inactive (untracked) track is re-associated with an observation, we first build a virtual trajectory by interpolating from the current step back to the previous step at which this object becomes untracked. Along this virtual trajectory, we recalculate the parameters of KF to reduce the accumulated error during the period of being untracked. (2) Under the assumption of linear motion, we have not just the speed consistency but also the direction consistency. So we incorporate the direction consistency of tracks in the cost matrix for the association. This new term suggests under the linear motion assumption, the object should move in constant momentum. So we name it Observation-Centric Momentum (OCM). We also provide analytical justification for the noise of velocity direction estimation in practice.

The proposed method, named as **O**bservation-**C**entric **SORT** or **OC-SORT** in short, remains simple, online, real-time and significantly improves robustness over occlusion and non-linear motion. Our contributions are summarized as the following: (1) we recognize, analytically and empirically, three limitations of SORT: sensitivity to the noise of state estimations, error accumulation over time, and being estimation-centric; (2) we propose OC-SORT for robust tracking under occlusion and non-linear motion with two main innovations: OOS and OCM. (3) our OC-SORT achieves new state-of-the-art performance on modern MOT benchmarks.

2 Related Works

Motion Models. Many recent MOT algorithms [4,10,62,71,68] use motion models. Typically, these motion models use Bayesian estimation [34] to predict the next state by maximizing a posterior estimation. As one of the most classic motion models, the Kalman filter (KF) [30] is a recursive Bayes filter that follows a typical predict-update cycle. The true state is assumed to be an unobserved Markov process, and the measurements are observations of a hidden Markov model [43]. Given that the linear motion assumption limits KF, follow-up works like Extended KF [51] and Unscented KF [28] were proposed to handle non-linear motion with first-order and third-order Taylor approximation. However, they still rely on approximating the Gaussian prior assumed by KF. On the other hand, particle filters [22] deal with a non-linear motion by sampling-based posterior estimation but require exponential order of computation. In Bayesian Estimation, filtering uses only past data, while the process of smoothing requires data points on both past and future time steps to adjust the data on a step in between. Fixed-lag smoother [1] and fixed-interval smoother [5,16,45] are studied to improve sequential data fitting after having built the whole trajectory. Though such smoothers are popular in general Bayesian Estimation, it can be hardly applied in video object tracking because they typically require the state on future steps to gain a better estimation on a previous time step, thus making the process not online anymore. Our proposed online smoothing is to discard accumulated error on historical steps and keeps our method an online algorithm.

Multi-object Tracking is traditionally approached from probabilistic perspectives, e. g. joint probabilistic association [2]. And modern video object tracking is usually built upon modern object detectors [46,48,70]. SORT [4] adopts the Kalman filter for motion-based multi-object tracking given observations from deep detectors. DeepSORT [62] further introduces deep visual features [50,23] into object association in the framework of SORT. Re-id-based object association[62,41,69] has also become popular since then but falls short when scenes are crowded and objects are represented coarsely (e.g bounding boxes), or object appearance is not distinguishable. More recently, transformers [57] have been introduced to MOT [38,67,53] to learn deep representations from both visual information and object trajectories. However, their performance still has a significant gap between state-of-the-art tracking-by-detection methods in terms of accuracy and speed.

3 Rethinking SORT

In this section, we review SORT [4] and Kalman filter. We recognize some of their limitations, which become significant with occlusion or non-linear object motion and motivate our study.

3.1 Background

Kalman filter (KF) [30] is a linear estimator for dynamical systems discretized in the time domain. KF only requires the state estimations on the previous time step and the current measurement (observation) to estimate the target state on the next time step. The filter maintains two variables, the posterior state estimate $\mathbf{x}$, and the posterior estimate covariance matrix $\mathbf{P}$ of the state. In the task of object tracking, we describe the KF process with the state transition model $\mathbf{F}$, the observation model $\mathbf{H}$, the process noise $\mathbf{Q}$ and the observation noise $\mathbf{R}$. At each step t, given observations $\mathbf{z}_t$, KF works in an alternation of "predict" and "update" stages:

$$\text{(predict)} \begin{cases} \hat{\mathbf{x}}_{t|t-1} = \mathbf{F}_t \hat{\mathbf{x}}_{t-1|t-1}, \\ \mathbf{P}_{t|t-1} = \mathbf{F}_t \mathbf{P}_{t-1|t-1} \mathbf{F}_t^\top + \mathbf{Q}_t \end{cases}, \quad \text{(update)} \begin{cases} \mathbf{K}_t = \mathbf{P}_{t|t-1} \mathbf{H}_t^\top (\mathbf{H}_t \mathbf{P}_{t|t-1} \mathbf{H}_t^\top + \mathbf{R}_t)^{-1}, \\ \hat{\mathbf{x}}_{t|t} = \hat{\mathbf{x}}_{t|t-1} + \mathbf{K}_t (\mathbf{z}_t - \mathbf{H}_t \hat{\mathbf{x}}_{t|t-1}), \\ \mathbf{P}_{t|t} = (\mathbf{I} - \mathbf{K}_t \mathbf{H}_t) \mathbf{P}_{t|t-1} \end{cases} \tag{1}$$

In practice, observations are often absent on some time steps, e. g. the target object is occluded in multi-object tracking. In such cases, we cannot update the KF parameters by the update operation as in Eq. 1 anymore. To address this, a way is to use the estimations $\hat{\mathbf{x}}_{t|t-1}$ directly as posterior. The philosophy is to trust estimations when no observations are available to supervise them. However, we will see that this estimation-centric mechanism can cause trouble for MOT.

SORT [4] is a multi-object tracker built upon KF. The KF's state $\mathbf{x}$ in SORT is defined as $\mathbf{x} = [u, v, s, r, \dot{u}, \dot{v}, \dot{s}]^\top$, where (u, v) is the 2D coordinates of the object center in the image. s is the bounding box scale (area) and r is the bounding box aspect ratio. The aspect ratio r is assumed to be constant. The other three variables, $\dot{u}$, $\dot{v}$ and $\dot{s}$ are the corresponding time derivatives. The observation is a bounding box $\mathbf{z} = [u, v, w, h, c]^\top$ with object center position (u, v), object width w and height h and the detection confidence c respectively. SORT assumes linear motion of the target:

$$u_{t+1} = u_t + \dot{u}_t \Delta t, \quad v_{t+1} = v_t + \dot{v}_t \Delta t. \tag{2}$$

When the time difference between two steps is constant during the transition, e. g. , the video frame rate is constant, we can set $\Delta t = 1$. When the video frame rate is high, SORT works well even when the object motion is non-linear globally, (e. g. dancing, fencing, wrestling) because the motion of the target object can be well approximated as linear within short time intervals.

3.2 Limitations of SORT

In this section, we identify three main limitations of SORT which are connected. This analysis lays the foundation of our proposed method.

3.2.1 Sensitive to State Noise

Now we prove that SORT is sensitive to the noise from KF's state estimations. To begin with, it is reasonable to assume that the estimated object center position follows $u \sim \mathcal{N}(\mu_u, \sigma_u^2)$ and $v \sim \mathcal{N}(\mu_v, \sigma_v^2)$, where (μ_u, μ_v) is the underlying true position. Then, if we assume that the state noises are independent on different steps, by Eq.2, the estimated object speed between two time steps, $t \longrightarrow t + \Delta t$, is

$$\dot{u} = \frac{u_{t+\Delta t} - u_t}{\Delta t}, \qquad \dot{v} = \frac{v_{t+\Delta t} - v_t}{\Delta t}, \tag{3}$$

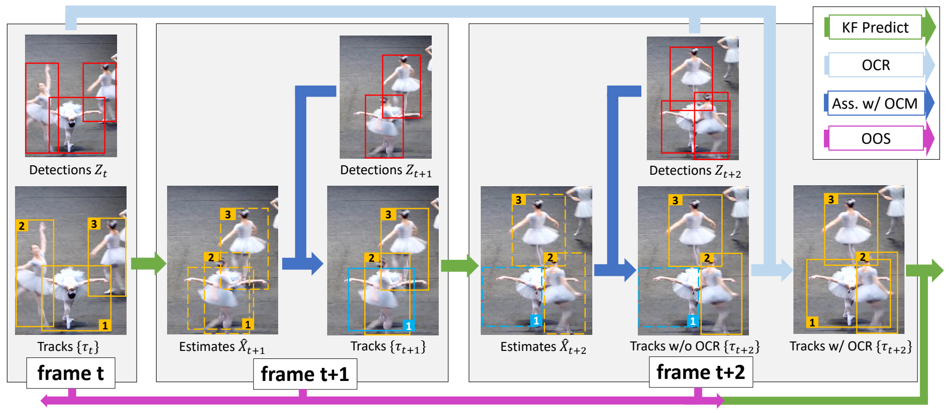


Figure 2: The pipeline of our proposed OC-SORT. The red boxes are detections, orange boxes are active tracks, blue boxes are untracked tracks, dashed boxes are the estimates from KF. During association OCM is used to add the velocity consistency cost. The target #1 is lost on the frame t+1 because of occlusion. But on the next frame, it is recovered by referring to its observation of the frame t by OCR. It being re-tracked triggers OOS from t to t+2 for the parameters of its KF.

making the noise of estimated speed $\delta_{\dot{u}} \sim \mathcal{N}(0, \frac{2\sigma_u^2}{(\Delta t)^2})$, $\delta_{\dot{v}} \sim \mathcal{N}(0, \frac{2\sigma_v^2}{(\Delta t)^2})$. Therefore, estimating the speed between consecutive frames, i. e. $\Delta t = 1$, maximizes the noise.

Moreover, for most multi-object tracking scenarios, the target object displacement is only a few pixels between consecutive frames. For instance, the average displacement is 1.93 pixels and 0.65 pixels along the image width and height for the MOT17 [40] training dataset. In such a case, even if the estimated position has a shift of only a single pixel, it causes a significant variation in the estimated speed. In general, the variance of the speed estimation can be of the same magnitude as the speed itself or even greater. In most cases, this will not make a massive impact as the shift is only of few pixels from the ground truth on the next time step, and the observations, whose variance is independent of time period, will be able to supervise the state estimations from the KF motion model. However, we will see that the sensitivity introduces significant problems in practice because of the error accumulation across multiple time steps when no observation is available for KF update.

3.2.2 Temporal Error Magnification

For analysis above in Eq. 3, we assume the noise of object state is i.i.d on different time steps. This is reasonable for object detections but not for the estimations from KF. This is because KF's estimations always rely on its estimations on previous time steps. The effect is usually minor because KF can use observation to supervise the state estimations in the update stage to avoid its parameters deviating from the true value too far away. However, when no observations are provided to KF, it cannot use observation to supervise the update of its parameters anymore. It simply uses its own priori state estimation $\hat{\mathbf{x}}_{t|t-1}$ to replace $\mathbf{z}_t$ during update. Consider a track is occluded on the time steps between t and $t+T$ and the noise of speed estimate follows $\delta_{\dot{u}_t} \sim \mathcal{N}(0, 2\sigma_u^2)$, $\delta_{\dot{v}_t} \sim \mathcal{N}(0, 2\sigma_v^2)$ for SORT. Till the step $t+T$, state estimation would be

$$u_{t+T} = u_t + T\dot{u}_t, \qquad v_{t+T} = v_t + T\dot{v}_t, \tag{4}$$

whose noise follows $\delta_{u_{t+T}} \sim \mathcal{N}(0, 2T^2\sigma_u^2)$ and $\delta_{v_{t+T}} \sim \mathcal{N}(0, 2T^2\sigma_v^2)$. So without the supervision from observation, the estimates from the linear motion assumption of KF result in a square-order error accumulation with respect to time. Given σ_v and σ_u is of the same magnitude as object displacement between consecutive frames, the noise of final object position (u_{t+T}, v_{t+T}) is of the same magnitude as the object size. For instance, the size of pedestrians close to the camera on MOT17 is around 50×300 pixels. So even assuming the variance of position estimates to be around 1 pixel, 10-frame occlusion can accumulate a shift of final position estimates as large as the object size. Such error magnification leads to a major accumulation of errors when the scenes are crowded.

3.2.3 Estimation-Centric

The aforementioned limitations come from a fundamental property of SORT that KF is designed to be estimation-centric. The external observations serve only to assist in the propagation of KF

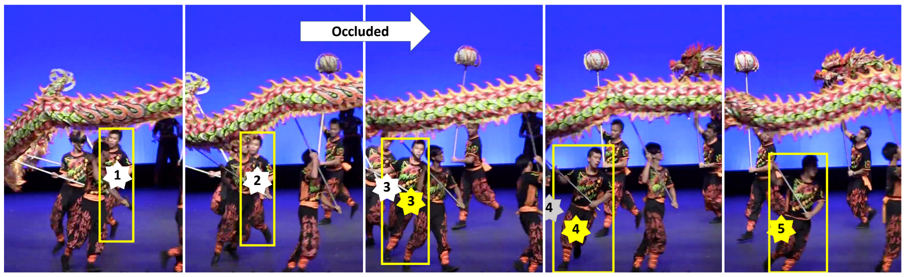


Figure 3: Example of how Observation-centric Online Smoothing reduces the error accumulation when a track is broken. The target is occluded between the second and the third time step and the tracker finds it back at the third step. Yellow boxes are the state observations by the detector. White stars are the estimated centers without OOS. Yellow stars are the estimated centers fixed by OOS. The gray star on the fourth step is the estimated center without OOS and fails to match observations.

trajectory. A key difference between state estimates and observations is that we can assume that the observations by a object detector in each frame are affected by i.i.d. noise $\delta_{\mathbf{z}} \sim \mathcal{N}(0, {\sigma'}^2)$. Modern object detectors use object visual features [50,48], which are ignored by KF when making estimates, making it safe to assume $\sigma' < \sigma_u$ and $\sigma' < \sigma_v$. Additionally, the variance of detections will not be accumulated over time which happens to KF's estimates. Therefore, a robust multi-object tracker under occlusion should give more importance to the observations than the KF's state estimates.

4 Observation-Centric SORT

In this section, we introduce the proposed Observation-Centric Sort (OC-SORT). To address the limitations of SORT discussed above, we use the momentum of the object moving into the association stage and develop a pipeline with less noise and more robustness over occlusion and non-linear motion. The key is to design the tracker as **observation-centric** instead of **estimation-centric**. If a track is recovered from being untracked, we use an Observation-centric Online Smoothing (OOS) strategy to counter the accumulated error during the untracked period. OC-SORT also adds an Observation-Centric Momentum (OCM) term in the association cost. Please refer to Algorithm 1 in Appendix for the pseudo-code of OC-SORT. The pipeline is shown in Fig. 2.

4.1 Observation-centric Online Smoothing (OOS)

In practice, even if an object can be associated again by SORT after a period of being untracked, it is probably lost again because its KF parameters have already deviated far away from the correct due to the temporal error magnification. To alleviate this problem, we propose Observation-centric Online Smoothing (OOS) to reduce the accumulated error. Once a track is associated with an observation again after a period of being untracked, we would backtrack the period of being lost and re-calculate the parameters of KF along a virtual trajectory during this period. This virtual trajectory is generated with the help of state observations before and after the untracked period. For example, by denoting the last observation before being untracked as $\mathbf{z}_{t_1}$ and the observation triggering the re-association as $\mathbf{z}_{t_2}$, the virtual trajectory is denoted as

$$\hat{\mathbf{z}}_t = Traj_{\text{virtual}}(\mathbf{z}_{t_1}, \mathbf{z}_{t_2}, t), t_1 < t < t_2. \tag{5}$$

Along this virtual trajectory, we can start from the state at t_1 to backtrack and refresh the filter parameters by alternating between the stages of prediction and update (Eq. 1). Given the supervision of the states on the virtual trajectory, the error raised in state estimation would not be accumulated anymore. The refreshed state estimations along the virtual trajectory follow $\hat{\mathbf{x}}_t = \mathbf{F}_t\hat{\mathbf{x}}_{t-1} + \mathbf{K}_t(\hat{\mathbf{z}}_t - \mathbf{H}_t\mathbf{F}_t\hat{\mathbf{x}}_{t-1})$. And other KF parameters are updated correspondingly. This operation is not Bayesian smoothing [5,16,45], which is widely used for offline data series post-processing, as it only uses data up to the current time step and does not change previous tracking results. Furthermore, it performs on observations instead of filter parameters directly. To stress on the fundamental difference of our designed process from offline smoothing, we call it "online smoothing".

4.2 Observation-Centric Momentum (OCM)

The linear motion model assumes a consistent velocity. However, this assumption often does not hold due to the non-linear motion of objects and state noise. In a reasonably short time, we can approximate the motion as linear but the noise still prevents us from leveraging the consistency of velocity direction. We propose a way to reduce the noise and add the velocity consistency (momentum) term in association. Given N existing tracks and M detections, the association cost is

$$C(\hat{\mathbf{X}}, \mathbf{Z}) = C_{\text{IoU}}(\hat{\mathbf{X}}, \mathbf{Z}) + \lambda C_v(\hat{\mathbf{X}}, \mathbf{Z}, \mathbf{V}), \tag{6}$$

where $\hat{\mathbf{X}} \in \mathbb{R}^{N\times 7}$ and $\mathbf{Z} \in \mathbb{R}^{M\times 5}$ are the sets of object state estimates and observations. $\mathbf{V} \in \mathbb{R}^N$ contains the directions of existing tracks calculated by two previous observations of time difference Δt. $C_{\text{IoU}}(\cdot,\cdot)$ calculates the negative pairwise IoU (Intersection over Union) and $C_v(\cdot,\cdot)$ calculates the consistency of i) track directions and ii) direction formed by a track's historical observation and the new observations. In our implementation, we use the difference of trajectory direction in radians, namely $\Delta\theta$, as the term C_v. θ is calculated by linking observations on two time steps. The detailed discussion about this is provided in Appendix A. λ is a weighting factor. We use observations on a track for direction calculation to avoid error accumulation in state estimations, but there is still a choice about which two observations we should choose.

We now analyze the relation between the variance of direction estimation and the choice of time steps. Assuming we calculate the trajectory direction by observations on two steps, t_1 and t_2, on which the underlying true object positions are $(\mu_{u_{t_1}}, \mu_{v_{t_1}})$ and $(\mu_{u_{t_2}}, \mu_{v_{t_2}})$. And the estimations of positions as (u_{t_1}, v_{t_1}) and (u_{t_2}, v_{t_2}). The movement direction is $\theta = arctan(\frac{\mu_{v_{t_1}} - \mu_{v_{t_2}}}{\mu_{u_{t_1}} - \mu_{u_{t_2}}})$. Noting $w = u_{t_1} - u_{t_2}$, $y = v_{t_1} - v_{t_2}$, and $z = \frac{y}{w}$, under the assumptions of noise assumption mentioned before, we can derive a closed-form probability density function of the distribution of z as

$$p(z) = \frac{g(z)e^{\frac{g(z)^2 - \alpha r(z)^2}{2\beta^2 r(z)^2}}}{\sqrt{2\pi}\sigma_w\sigma_y r(z)^3}\left[\Phi\left(\frac{g(z)}{\beta r(z)}\right) - \Phi\left(-\frac{g(z)}{\beta r(z)}\right)\right] + \frac{\beta e^{-2\alpha/\beta}}{\pi\sigma_w\sigma_y r(z)^2}, \tag{7}$$

which is explained in detail in Appendix A. By analyzing the property of this distribution, we reach a conclusion that, under the linear-motion model, the scale of noise z's estimation is negatively correlated to the time difference between the two observation points, i. e. $\Delta t = t_2 - t_1$ (see Appendix A for more details). But, on the other hand, the trajectory can only be approximated as linear within a short time interval, so the time difference should be held not too large to avoid the collapse of linear approximation. This requires a trade-off in practice.

Besides OOS and OCM, we also find it empirically helpful to check a track's last presence when re-covering it from being lost. We thus propose a heuristic Observation-Centric Recovery (OCR) technique here as a secondary module. Once a track is still untracked after the normal association stage, OCR asks to associate the last observation of this track to the observations on the new-coming step to handle the case of an object stopping or being occluded for a short time interval.

5 Experiments

5.1 Experimental Setup

Datasets. We evaluate our method on multiple multi-object tracking datasets including MOT17 [40], MOT20 [13], KITTI [20], DanceTrack [54] and HeadTrack [55] (in Appendix). MOT17 [40] and MOT20 [13] are for pedestrian tracking, where targets mostly move linearly, while scenes in MOT20 are more crowded. KITTI [20] is for pedestrian and car tracking with a relatively low frame rate of 10FPS. DanceTrack [54] is a recently proposed dataset for human tracking. In this dataset, object localization is easy, but the object motion is highly non-linear. Furthermore, the objects have a close appearance, severe occlusion, and frequent crossovers. Considering our goal is to improve tracking robustness in occlusion and non-linear object motion, we would emphasize the comparison between OC-SORT and previous methods on DanceTrack in the following experiments.

Implementations. For a fair comparison, we directly apply the object detections from existing baselines. For MOT17, MOT20, and DanceTrack, we use the publicly available YOLOX [19] detector weights by ByteTrack [68]. For KITTI [20], we use the detections from PermaTrack [56] publicly available in the official release. For OOS, we generate the virtual trajectory during occlusion with the constant-velocity assumption. Therefore, Eq. 5 is adopted as

$$\hat{\mathbf{z}}_t = \mathbf{z}_{t_1} + \frac{t - t_1}{t_2 - t_1}(\mathbf{z}_{t_2} - \mathbf{z}_{t_1}), t_1 < t < t_2. \tag{8}$$

For OCM, the velocity direction is calculated using the observations three time steps apart, i. e. $\Delta t = 3$. The direction difference is measured by the absolute difference of angles in radians. We set $\lambda = 0.2$ in Eq. 6. Following the common practice of SORT, we set the detection confidence threshold at 0.4 for MOT20 and 0.6 for other datasets. The IoU threshold during association is 0.3.

Metrics. We adopt HOTA [37] as the main metric as it maintains a better balance between the accuracy of object detection and association [37]. We also emphasize AssA and IDF1 to evaluate the association performance. Some other metrics we report, such as MOTA, are highly related to detection performance; this can lead to fair comparison only when all methods use the same detections for tracking– this setting is referred to as "public tracking".

Table 1: Results on MOT17-test with the private detections. ByteTrack and ours share detections.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP(10^4)↓	FN(10^4)↓	IDs↓	Frag↓	AssA↑	AssR↑
FairMOT [69]	59.3	73.7	72.3	2.75	11.7	3,303	8,073	58.0	63.6
TransCt [66]	54.5	73.2	62.2	2.31	12.4	4,614	9,519	49.7	54.2
TransTrk [53]	54.1	75.2	63.5	5.02	8.64	3,603	4,872	47.9	57.1
GRTU [59]	62.0	74.9	75.0	3.20	10.8	1,812	**1,824**	62.1	65.8
QDTrack [41]	53.9	68.7	66.3	2.66	14.66	3,378	8,091	52.7	57.2
MOTR [67]	57.2	71.9	68.4	2.11	13.6	2,115	3,897	55.8	59.2
PermaTr [56]	55.5	73.8	68.9	2.90	11.5	3,699	6,132	53.1	59.8
TransMOT [11]	61.7	76.7	75.1	3.62	9.32	2,346	7,719	59.9	66.5
ByteTrack [68]	63.1	**80.3**	77.3	2.55	8.37	2,196	2,277	62.0	**68.2**
Ours	**63.2**	78.0	**77.5**	**1.51**	10.8	1,950	2,040	**63.2**	67.5

Table 2: Results on MOT20-test with private detections. ByteTrack and ours share detections.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP(10^4)↓	FN(10^4)↓	IDs↓	Frag↓	AssA↑	AssR↑
FairMOT [69]	54.6	61.8	67.3	10.3	8.89	5,243	7,874	54.7	60.7
TransCt [66]	43.5	58.5	49.6	6.42	14.6	4,695	9,581	37.0	45.1
Semi-TCL [35]	55.3	65.2	70.1	6.12	11.5	4,139	8,508	56.3	60.9
CSTrack [36]	54.0	66.6	68.6	2.54	14.4	3,196	7,632	54.0	57.6
GSDT [60]	53.6	67.1	67.5	3.19	13.5	3,131	9,875	52.7	58.5
TransMOT [11]	61.9	77.5	75.2	3.42	**8.08**	1,615	2,421	60.1	66.3
ByteTrack [68]	61.3	**77.8**	75.2	2.62	8.76	1,223	1,460	59.6	66.2
Ours	**62.1**	75.5	**75.9**	**1.80**	10.8	**913**	**1,198**	**62.0**	**67.5**

5.2 Benchmark Results

Here we report the benchmark results on multiple datasets. We put all methods that use the shared detection results in a block at the bottom of each table.

MOT17 and MOT20. We report OC-SORT's performance on MOT17 and MOT20 in Table 1 and Table 2 using private detections. To make a fair comparison, we use the same detection as ByteTrack [68]. OC-SORT achieves performance comparable to other state-of-the-art methods. Our gains are especially significant in MOT20 under severe pedestrian occlusion, setting a state-of-the-art HOTA of 62.1. As our method is designed to be simple for better generalization, we do not use adaptive detection thresholds as in ByteTrack. If we set the detection threshold adaptive as ByteTrack does, OC-SORT will achieve higher MOTA scores (80.5 on MOT17 and 78.1 on MOT20) but we prefer to maintain the implementation of OC-SORT clean and generalizable and share a same setting of hyper-parameters cross datasets. But we still inherit its linear interpolation for a fair comparison. To more clearly discard the variance from the detector, we also perform public tracking on MOT17 and MOT20, which is reported in Table 12 and Table 13 in Appendix C. OC-SORT still outperforms the existing state-of-the-art in public tracking settings.

DanceTrack. To evaluate OC-SORT under challenging non-linear object motion, we report results on the DanceTrack in Table 3. OC-SORT sets a new state-of-the-art, outperforming the baselines by a great margin under non-linear object motions. Although OC-SORT has better association performance, association metrics such as HOTA, IDF1, and AssA are still lower than the results on previous datasets. This suggests that the difficulty of tracking objects on DanceTrack is very challenging. We compare the tracking results of SORT and OC-SORT under extreme non-linear situations in Fig.1 and more samples are available in Fig. 6 in Appendix E. We also visualize the output trajectories by OC-SORT and SORT on randomly selected DanceTrack videos clips in Fig. 7

Table 3: Results on DanceTrack test set. Methods in the bottom block use the same detections.

Tracker	HOTA↑	DetA↑	AssA↑	MOTA↑	IDF1↑
CenterTrack [71]	41.8	78.1	22.6	86.8	35.7
FairMOT [69]	39.7	66.7	23.8	82.2	40.8
QDTrack [41]	45.7	72.1	29.2	83.0	44.8
TransTrk [53]	45.5	75.9	27.5	88.4	45.2
TraDes [63]	43.3	74.5	25.4	86.2	41.2
MOTR [67]	54.2	73.5	40.2	79.7	51.5
SORT [4]	47.9	72.0	31.2	91.8	50.8
DeepSORT [62]	45.6	71.0	29.7	87.8	47.9
ByteTrack [68]	47.3	71.6	31.4	89.5	52.5
Ours	54.6	80.4	40.2	89.6	54.6
Ours + Linear Interp	**54.9**	**81.9**	**40.4**	**92.2**	**54.9**

Table 4: Results on KITTI test set. HP indicates adding the lost detections during initializing tracks. Our method uses the same detections as PermaTr [56]

	Car					Pedestrian				
Tracker	HOTA↑	MOTA↑	AssA↑	IDs↓	Frag↓	HOTA↑	MOTA↑	AssA↑	IDs↓	Frag↓
IMMDP [64]	68.66	82.75	69.76	211	181	-	-	-	-	-
SMAT [21]	71.88	83.64	72.13	198	294	-	-	-	-	-
TrackMPNN [44]	72.30	87.33	70.63	481	237	39.40	52.10	35.45	626	669
MPNTrack [6]	-	-	-	-	-	45.26	46.23	47.28	397	1,078
CenterTr [71]	73.02	88.83	71.18	254	227	40.35	53.84	36.93	425	618
LGM [58]	73.14	87.60	72.31	448	**164**	-	-	-	-	-
TuSimple [10]	71.55	86.31	71.11	292	218	45.88	57.61	47.62	246	651
PermaTr [56]	**77.42**	**90.85**	**77.66**	275	271	47.43	65.05	43.66	483	703
Ours	74.64	87.81	74.52	257	318	52.95	62.00	57.81	**181**	**598**
Ours + HP	76.54	90.28	76.39	250	280	**54.69**	**65.14**	**59.08**	184	609

Table 5: Ablation study on MOT17 val set and DanceTrack-val set.

			MOT17-val				DanceTrack-val			
OOS	OCM	OCR	HOTA↑	AssA↑	MOTA↑	IDF1↑	HOTA↑	AssA↑	MOTA↑	IDF1↑
			64.9	66.8	74.6	76.9	47.8	31.0	**88.2**	48.3
✓			66.3	68.0	74.7	77.2	48.5	32.2	87.2	49.8
✓	✓		66.4	**69.0**	74.6	**77.8**	**52.1**	35.0	87.3	50.6
✓	✓	✓	**66.5**	68.9	**74.9**	77.7	**52.1**	**35.3**	87.3	**51.6**

Table 6: Ablation study on the trajectory hypothesis used in OOS.

	MOT17-val				DanceTrack-val			
	HOTA↑	AssA↑	MOTA↑	IDF1↑	HOTA↑	AssA↑	MOTA↑	IDF1↑
Const. Speed	**66.5**	**68.9**	**74.9**	**77.7**	**52.1**	**35.3**	**87.3**	**51.6**
GPR	63.1	65.2	74.0	75.7	49.5	33.7	86.7	49.6
Linear Regression	64.3	66.5	74.2	76.0	49.3	33.4	86.2	49.2
Const. Acceleration	66.2	67.9	74.7	77.4	51.3	34.8	87.0	50.9

in Appendix E. As we focus on improving multi-object tracking in occlusion and non-linear motion cases, the results on DanceTrack are strong evidence of the efficiency of OC-SORT.

KITTI. In Table 4 we report the results on the KITTI dataset. For a fair comparison, we adopt the detector weights by PermaTr [56] and report its performance in the table as well. Then, we run OC-SORT given the shared detections. As initializing SORT's track requires continuous tracking across several frames ("minimum hits"), we observe that the results not recorded during the track initialization make a significant difference. To address this, we do offline head padding (HP) post-processing by writing these entries back after finishing the online tracking stage. The results on KITTI show an essential shortcoming of OC-SORT that it highly relies on the IoU matching for the association. As a result, when the object velocity is high or the frame rate is low, the IoU of object bounding boxes between consecutive frames can be very low or even zero. This phenomenon poses a significant challenge to our method. But still, in contrast to the baseline car tracking performance, OC-SORT improves pedestrian tracking performance to a new state-of-the-art.

We believe that the results shown on multiple benchmarks have demonstrated the efficiency of our proposed OC-SORT. We note that we use a shared parameter stack for different datasets, and carefully tuning the parameters might further boost the performance. For example, the adaptive detection threshold is proven useful in previous work [68]. Besides the association performance discussed above, we also care about the inference speed of tracking algorithms. As different methods report results on different detectors and running environments, it is hard to compare them directly. Therefore, we only report the inference speed of OC-SORT. Given off-the-shelf detections, the association stage of OC-SORT runs at 793 FPS on an Intel i9-9980XE CPU @ 3.00GHz.

5.3 Ablation Study

Component Ablation. We ablate the contribution of proposed modules in OC-SORT on the validation sets of MOT17 and DanceTrack in Table 5. The splitting of MOT17 follows a popular convention [71]. The results demonstrate the efficiency of the proposed modules in OC-SORT.

Table 7: Influence of choice of Δt for estimating direction in OCM.

	MOT17-val				DanceTrack-val			
	HOTA↑	AssA↑	MOTA↑	IDF1↑	HOTA↑	AssA↑	MOTA↑	IDF1↑
$\Delta t = 1$	66.1	67.5	74.9	76.9	51.3	34.3	87.1	51.3
$\Delta t = 2$	66.3	68.0	**75.0**	77.3	**52.2**	**35.4**	87.2	51.4
$\Delta t = 3$	**66.5**	**68.9**	74.9	**77.7**	52.1	35.3	87.3	51.6
$\Delta t = 6$	66.0	67.5	74.6	76.9	52.1	**35.4**	**87.4**	**51.8**

Virtual Trajectory in OOS. For simplicity, we follow the naive hypothesis of constant speed in Eq 8 to generate virtual trajectory in OOS. There are other alternatives like constant acceleration, regression-based fitting such as Linear Regression (LR) or Gaussian Process Regression (GPR), and Near Constant Acceleration Model (NCAM) [27]. The results of comparing these choices are shown in Table 6. For GPR, we use the RBF kernel [9] $k(\mathbf{x}, \mathbf{x}') = \exp\left(-\frac{||\mathbf{x}-\mathbf{x}'||^2}{50}\right)$. We provide more studies on the kernel configuration in Appendix B. The results show that local hypotheses such as Constant Speed/Acceleration perform much better than global hypotheses such as LR and GPR. This is probably because, as virtual trajectory generation happens in an online fashion, it is hard to get a reliable fit using only limited data points on historical time steps.

Δt in OCM. As discussed in Section 4, there is a trade-off when choosing the time difference Δt in OCM. A large Δt has better robustness over the noise under the linear motion assumption. However, in practice, a large Δt is likely to discourage approximating object motion as linear. Therefore, we study the influence of varying Δt in Table 7. Our results agree with our analysis that increasing Δt from $\Delta t = 1$ can boost the association performance. It is believed to be effective in relieving the impact of noise on direction estimation. Keeping increasing Δt higher than the bottleneck instead hurts the performance because of the difficulty of maintaining the approximation of linear motion.

5.4 Limitations

Our experiments reveal some limitations of OC-SORT. For example, when the video has a low frame rate or the object motion is fast, such as cars in KITTI, the proposed method falls short in matching objects by only using IoU and trajectory direction consistency. Nevertheless, SORT also has the same limitation. Adding other cues such as center distance [71] or appearance similarity has been demonstrated [62] efficient to solve this.

6 Conclusion

We analyze the popular motion-based SORT tracker and point out its intrinsic limitations from the use of the Kalman filter. These limitations play significant roles to hurt tracking accuracy when the tracker fails to gain observations for supervision - likely caused by unreliable detectors, occlusion, or fast and non-linear target object motion. To address these issues, we propose Observation-Centric SORT (OC-SORT). OC-SORT is more robust to occlusion and non-linear object motion while still being simple, online, and real-time. Our proposed method is motivated by both analytical and empirical findings and focuses on leveraging observations more confidently in the interaction with Kalman filter. In our experiments on multiple popular tracking datasets, OC-SORT significantly outperforms the state of the art. Our gains are especially significant for multi-object tracking under severe occlusion and on objects with dramatic non-linear motion.

REFERENCES

[1] Brian DO Anderson and John B Moore. *Optimal filtering*. Courier Corporation, 2012. 2

[2] Yaakov Bar-Shalom, Fred Daum, and Jim Huang. The probabilistic data association filter. *IEEE Control Systems Magazine*, 29(6):82–100, 2009. 2

[3] Sumit Basu, Irfan Essa, and Alex Pentland. Motion regularization for model-based head tracking. In *Proceedings of 13th International Conference on Pattern Recognition*, volume 3, pp. 611–616. IEEE, 1996. 18

[4] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pp. 3464–3468. IEEE, 2016. 1, 2, 3, 8

[5] Gerald J Bierman. *Factorization methods for discrete sequential estimation*. Courier Corporation, 2006. 2, 5

[6] Guillem Brasó and Laura Leal-Taixé. 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. 8, 18

[7] Jinkun Cao, Hongyang Tang, Hao-Shu Fang, Xiaoyong Shen, Cewu Lu, and Yu-Wing Tai. Cross-domain adaptation for animal pose estimation. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 9498–9507, 2019. 18

[8] Ming-Fang Chang, John Lambert, Patsorn Sangkloy, Jagjeet Singh, Slawomir Bak, Andrew Hartnett, De Wang, Peter Carr, Simon Lucey, Deva Ramanan, et al. Argoverse: 3d tracking and forecasting with rich maps. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 8748–8757, 2019. 18

[9] Yin-Wen Chang, Cho-Jui Hsieh, Kai-Wei Chang, Michael Ringgaard, and Chih-Jen Lin. Training and testing low-degree polynomial data mappings via linear svm. *Journal of Machine Learning Research*, 11(4), 2010. 9

[10] Wongun Choi. Near-online multi-target tracking with aggregated local flow descriptor. In *Proceedings of the IEEE international conference on computer vision*, pp. 3029–3037, 2015. 2, 8

[11] Peng Chu, Jiang Wang, Quanzeng You, Haibin Ling, and Zicheng Liu. Transmot: Spatial-temporal graph transformer for multiple object tracking. *arXiv preprint arXiv:2104.00194*, 2021. 7

[12] Peng Dai, Renliang Weng, Wongun Choi, Changshui Zhang, Zhangping He, and Wei Ding. 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. 18

[13] Patrick Dendorfer, Hamid Rezatofighi, Anton Milan, Javen Shi, Daniel Cremers, Ian Reid, Stefan Roth, Konrad Schindler, and Laura Leal-Taixé. Mot20: A benchmark for multi object tracking in crowded scenes. *arXiv preprint arXiv:2003.09003*, 2020. 6, 19

[14] David Duvenaud. *Automatic model construction with Gaussian processes*. PhD thesis, University of Cambridge, 2014. 17

[15] Hao-Shu Fang, Jinkun Cao, Yu-Wing Tai, and Cewu Lu. Pairwise body-part attention for recognizing human-object interactions. In *Proceedings of the European conference on computer vision (ECCV)*, pp. 51–67, 2018. 18

[16] D Fraser and J Potter. The optimum linear smoother as a combination of two optimum linear filters. *IEEE Transactions on automatic control*, 14(4):387–390, 1969. 2, 5

[17] Juergen Gall, Angela Yao, Nima Razavi, Luc Van Gool, and Victor Lempitsky. Hough forests for object detection, tracking, and action recognition. *IEEE transactions on pattern analysis and machine intelligence*, 33(11):2188–2202, 2011. 18

[18] Damien Garreau, Wittawat Jitkrittum, and Motonobu Kanagawa. Large sample analysis of the median heuristic. *arXiv preprint arXiv:1707.07269*, 2017. 18

[19] Zheng Ge, Songtao Liu, Feng Wang, Zeming Li, and Jian Sun. Yolox: Exceeding yolo series in 2021. *arXiv preprint arXiv:2107.08430*, 2021. 6, 19

[20] Andreas Geiger, Philip Lenz, Christoph Stiller, and Raquel Urtasun. Vision meets robotics: The kitti dataset. *The International Journal of Robotics Research*, 32(11):1231–1237, 2013. 6

[21] Nicolas Franco Gonzalez, Andres Ospina, and Philippe Calvez. Smat: Smart multiple affinity metrics for multiple object tracking. In *International Conference on Image Analysis and Recognition*, pp. 48–62. Springer, 2020. 8

[22] Fredrik Gustafsson, Fredrik Gunnarsson, Niclas Bergman, Urban Forssell, Jonas Jansson, Rickard Karlsson, and P-J Nordlund. Particle filters for positioning, navigation, and tracking. *IEEE Transactions on signal processing*, 50(2):425–437, 2002. 2

[23] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pp. 770–778, 2016. 3

[24] David V Hinkley. On the ratio of two correlated normal random variables. *Biometrika*, 56(3): 635–639, 1969. 15

[25] Andrea Hornakova, Roberto Henschel, Bodo Rosenhahn, and Paul Swoboda. Lifted disjoint paths with application in multiple object tracking. In *International Conference on Machine Learning*, pp. 4364–4375. PMLR, 2020. 18

[26] Andrea Hornakova, Timo Kaiser, Paul Swoboda, Michal Rolinek, Bodo Rosenhahn, and Roberto Henschel. Making higher order mot scalable: An efficient approximate solver for lifted disjoint paths. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 6330–6340, 2021. 18

[27] Andrew H Jazwinski. *Stochastic processes and filtering theory*. Courier Corporation, 2007. 9

[28] Simon J Julier and Jeffrey K Uhlmann. New extension of the kalman filter to nonlinear systems. In *Signal processing, sensor fusion, and target recognition VI*, volume 3068, pp. 182–193. International Society for Optics and Photonics, 1997. 2, 16

[29] Simon J Julier and Jeffrey K Uhlmann. Unscented filtering and nonlinear estimation. *Proceedings of the IEEE*, 92(3):401–422, 2004. 16

[30] Rudolf Emil Kalman et al. Contributions to the theory of optimal control. *Bol. soc. mat. mexicana*, 5(2):102–119, 1960. 2, 3, 16

[31] Kris M Kitani, Brian D Ziebart, James Andrew Bagnell, and Martial Hebert. Activity forecasting. In *European conference on computer vision*, pp. 201–214. Springer, 2012. 18

[32] Jonathan Ko and Dieter Fox. Gp-bayesfilters: Bayesian filtering using gaussian process prediction and observation models. *Autonomous Robots*, 27(1):75–90, 2009. 16

[33] Parth Kothari, Sven Kreiss, and Alexandre Alahi. Human trajectory forecasting in crowds: A deep learning perspective. *IEEE Transactions on Intelligent Transportation Systems*, 2021. 18

[34] Erich L Lehmann and George Casella. *Theory of point estimation*. Springer Science & Business Media, 2006. 2

[35] Wei Li, Yuanjun Xiong, Shuo Yang, Mingze Xu, Yongxin Wang, and Wei Xia. Semi-tcl: Semi-supervised track contrastive representation learning. *arXiv preprint arXiv:2107.02396*, 2021. 7

[36] Chao Liang, Zhipeng Zhang, Yi Lu, Xue Zhou, Bing Li, Xiyong Ye, and Jianxiao Zou. Rethinking the competition between detection and reid in multi-object tracking. *arXiv preprint arXiv:2010.12138*, 2020. 7

[37] Jonathon Luiten, Aljosa Osep, Patrick Dendorfer, Philip Torr, Andreas Geiger, Laura Leal-Taixé, and Bastian Leibe. Hota: A higher order metric for evaluating multi-object tracking. *International journal of computer vision*, 129(2):548–578, 2021. 7

[38] Tim Meinhardt, Alexander Kirillov, Laura Leal-Taixe, and Christoph Feichtenhofer. Trackformer: Multi-object tracking with transformers. *arXiv preprint arXiv:2101.02702*, 2021. 3, 18

[39] Stan Melax, Leonid Keselman, and Sterling Orsten. Dynamics based 3d skeletal hand tracking. In *Proceedings of the ACM SIGGRAPH Symposium on Interactive 3D Graphics and Games*, pp. 184–184, 2013. 18

[40] Anton Milan, Laura Leal-Taixé, Ian Reid, Stefan Roth, and Konrad Schindler. Mot16: A benchmark for multi-object tracking. *arXiv preprint arXiv:1603.00831*, 2016. 4, 6

[41] Jiangmiao Pang, Linlu Qiu, Xia Li, Haofeng Chen, Qi Li, Trevor Darrell, and Fisher Yu. 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. 3, 7, 8, 18

[42] Dezhi Peng, Zikai Sun, Zirong Chen, Zirui Cai, Lele Xie, and Lianwen Jin. Detecting heads using feature refine net and cascaded multi-scale architecture. *arXiv preprint arXiv:1803.09256*, 2018. 18

[43] Lawrence Rabiner and Biinghwang Juang. An introduction to hidden markov models. *ieee assp magazine*, 3(1):4–16, 1986. 2

[44] Akshay Rangesh, Pranav Maheshwari, Mez Gebre, Siddhesh Mhatre, Vahid Ramezani, and Mohan M Trivedi. Trackmpnn: A message passing graph neural architecture for multi-object tracking. *arXiv preprint arXiv:2101.04206*, 2021. 8

[45] Herbert E Rauch, F Tung, and Charlotte T Striebel. Maximum likelihood estimates of linear dynamic systems. *AIAA journal*, 3(8):1445–1450, 1965. 2, 5

[46] Joseph Redmon, Santosh Divvala, Ross Girshick, and Ali Farhadi. 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. 2

[47] Steven Reece and Stephen Roberts. An introduction to gaussian processes for the kalman filter expert. In *2010 13th International Conference on Information Fusion*, pp. 1–9. IEEE, 2010. 16

[48] Shaoqing Ren, Kaiming He, Ross Girshick, and Jian Sun. Faster r-cnn: Towards real-time object detection with region proposal networks. *Advances in neural information processing systems*, 28, 2015. 2, 5

[49] Toby Sharp, Cem Keskin, Duncan Robertson, Jonathan Taylor, Jamie Shotton, David Kim, Christoph Rhemann, Ido Leichter, Alon Vinnikov, Yichen Wei, et al. Accurate, robust, and flexible real-time hand tracking. In *Proceedings of the 33rd annual ACM conference on human factors in computing systems*, pp. 3633–3642, 2015. 18

[50] Karen Simonyan and Andrew Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 3, 5

[51] Gerald L Smith, Stanley F Schmidt, and Leonard A McGee. *Application of statistical filter theory to the optimal estimation of position and velocity on board a circumlunar vehicle*. National Aeronautics and Space Administration, 1962. 2, 16

[52] Daniel Stadler and Jurgen Beyerer. Improving multiple pedestrian tracking by track management and occlusion handling. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 10958–10967, 2021. 18

[53] Peize Sun, Jinkun Cao, Yi Jiang, Rufeng Zhang, Enze Xie, Zehuan Yuan, Changhu Wang, and Ping Luo. Transtrack: Multiple object tracking with transformer. *arXiv preprint arXiv:2012.15460*, 2020. 3, 7, 8

[54] Peize Sun, Jinkun Cao, Yi Jiang, Zehuan Yuan, Song Bai, Kris Kitani, and Ping Luo. Dancetrack: Multi-object tracking in uniform appearance and diverse motion. *arXiv preprint arXiv:2111.14690*, 2021. 2, 6, 19

[55] Ramana Sundararaman, Cedric De Almeida Braga, Eric Marchand, and Julien Pettre. Tracking pedestrian heads in dense crowd. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pp. 3865–3875, 2021. 6, 17, 18, 19

[56] Pavel Tokmakov, Jie Li, Wolfram Burgard, and Adrien Gaidon. Learning to track with object permanence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 10860–10869, 2021. 6, 7, 8

[57] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 3

[58] Gaoang Wang, Renshu Gu, Zuozhu Liu, Weijie Hu, Mingli Song, and Jenq-Neng Hwang. 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. 8

[59] Shuai Wang, Hao Sheng, Yang Zhang, Yubin Wu, and Zhang Xiong. A general recurrent tracking framework without real data. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pp. 13219–13228, 2021. 7

[60] Yongxin Wang, Kris Kitani, and Xinshuo Weng. Joint object detection and multi-object tracking with graph neural networks. In *2021 IEEE International Conference on Robotics and Automation (ICRA)*, pp. 13708–13715. IEEE, 2021. 7

[61] Christopher Williams and Carl Rasmussen. Gaussian processes for regression. *Advances in neural information processing systems*, 8, 1995. 16

[62] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pp. 3645–3649. IEEE, 2017. 2, 3, 8, 9

[63] Jialian Wu, Jiale Cao, Liangchen Song, Yu Wang, Ming Yang, and Junsong Yuan. Track to detect and segment: An online multi-object tracker. In *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, pp. 12352–12361, 2021. 8

[64] Yu Xiang, Alexandre Alahi, and Silvio Savarese. Learning to track: Online multi-object tracking by decision making. In *Proceedings of the IEEE international conference on computer vision*, pp. 4705–4713, 2015. 8

[65] Yuliang Xiu, Jiefeng Li, Haoyu Wang, Yinghong Fang, and Cewu Lu. Pose flow: Efficient online pose tracking. *arXiv preprint arXiv:1802.00977*, 2018. 18

[66] Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan, Daniela Rus, and Xavier Alameda-Pineda. Transcenter: Transformers with dense queries for multiple-object tracking. *arXiv preprint arXiv:2103.15145*, 2021. 7, 18

[67] Fangao Zeng, Bin Dong, Tiancai Wang, Xiangyu Zhang, and Yichen Wei. Motr: End-to-end multiple-object tracking with transformer. *arXiv preprint arXiv:2105.03247*, 2021. 3, 7, 8

[68] Yifu Zhang, Peize Sun, Yi Jiang, Dongdong Yu, Zehuan Yuan, Ping Luo, Wenyu Liu, and Xinggang Wang. Bytetrack: Multi-object tracking by associating every detection box. *arXiv preprint arXiv:2110.06864*, 2021. 2, 6, 7, 8, 9

[69] Yifu Zhang, Chunyu Wang, Xinggang Wang, Wenjun Zeng, and Wenyu Liu. Fairmot: On the fairness of detection and re-identification in multiple object tracking. *International Journal of Computer Vision*, 129(11):3069–3087, 2021. 3, 7, 8, 17, 18, 19

[70] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Objects as points. *arXiv preprint arXiv:1904.07850*, 2019. 2

[71] Xingyi Zhou, Vladlen Koltun, and Philipp Krähenbühl. Tracking objects as points. In *European Conference on Computer Vision*, pp. 474–490. Springer, 2020. 2, 8, 9, 18

A VELOCITY DIRECTION VARIANCE IN OCM

In this section, we work on the setting of linear motion with noisy states. We provide proof that the trajectory direction estimation has a smaller variance if the two states we use for the estimation have a larger time difference. We assume the motion model is $\mathbf{x}_t = f(t) + \epsilon$ where ϵ is gaussian noise and the ground-truth center position of the target is (μ_{u_t}, μ_{v_t}) at time step t. Then, estimated on two steps t_1 and t_2, the true motion direction between these two points is

$$\theta = arctan(\frac{\mu_{v_{t_1}} - \mu_{v_{t_2}}}{\mu_{u_{t_1}} - \mu_{u_{t_2}}}), \tag{9}$$

which is a constant if our estimation incurs zero noise. And we have $\mu_{v_{t_1}} - \mu_{v_{t_2}} \propto t_1 - t_2$, $\mu_{u_{t_1}} - \mu_{u_{t_2}} \propto t_1 - t_2$. As the detection results do not suffer from the error accumulation due to propagating along Markov process as Kalman filter does, we can assume the states from observation suffers some i.i.d. noise, i.e., $u_t \sim \mathcal{N}(\mu_{u_t}, \sigma_u^2)$ and $v_t \sim \mathcal{N}(\mu_{v_t}, \sigma_v^2)$. We now analyze the noise of the estimated $\tilde{\theta} = \frac{v_{t_1} - v_{t_2}}{u_{t_1} - u_{t_2}}$ by two observations on the trajectory. Because the function of $arctan(\cdot)$ is monotone over the whole real field, we can study $tan\tilde{\theta}$ instead which simplifies the analysis. We denote $w = u_{t_1} - u_{t_2}$, $y = v_{t_1} - v_{t_2}$, and $z = \frac{y}{w}$, first we can see that y and w jointly form a Gaussian distribution:

$$\begin{bmatrix} y \\ w \end{bmatrix} \sim \mathcal{N}\left(\begin{bmatrix} \mu_y \\ \mu_w \end{bmatrix}, \begin{bmatrix} \sigma_y^2 & \rho\sigma_y\sigma_w \\ \rho\sigma_y\sigma_w & \sigma_w^2 \end{bmatrix} \right), \tag{10}$$

where $\mu_y = \mu_{v_{t_1}} - \mu_{v_{t2}}$, $\mu_w = \mu_{u_{t_1}} - \mu_{u_{t_2}}$, $\sigma_w = \sqrt{2}\sigma_u$ and $\sigma_y = \sqrt{2}\sigma_v$, and ρ is the correlation coefficient between y and w. We can actually derive a closed-form solution of the probability density function [24] of z as

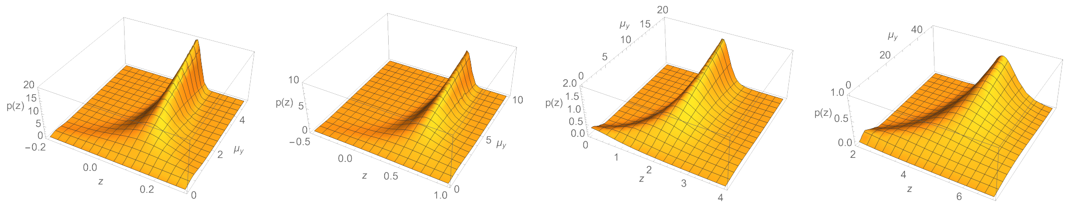
$$\begin{aligned} p(z) =& \frac{g(z)e^{\frac{g(z)^2 - \alpha r(z)^2}{2\beta^2 r(z)^2}}}{\sqrt{2\pi}\sigma_w\sigma_y r(z)^3} \left[\Phi\left(\frac{g(z)}{\beta r(z)}\right) - \Phi\left(-\frac{g(z)}{\beta r(z)}\right) \right] \\ &+ \frac{\beta e^{-2\alpha/\beta}}{\pi\sigma_w\sigma_y r(z)^2} \end{aligned} \tag{11}$$

where

$$\begin{aligned} r(z) &= \sqrt{\frac{z^2}{\sigma_y^2} - \frac{2\rho z}{\sigma_y\sigma_w} + \frac{1}{\sigma_w^2}}, \\ g(z) &= \frac{\mu_y z}{\sigma_y^2} - \frac{\rho(\mu_y + \mu_w z)}{\sigma_y\sigma_w} + \frac{\mu_w}{\sigma_w^2}, \\ \alpha &= \frac{\mu_w^2 + \mu_y^2}{\sigma_y^2} - \frac{2\rho\mu_y\mu_w}{\sigma_w\sigma_y}, \qquad \beta = \sqrt{1-\rho^2}, \end{aligned} \tag{12}$$

and Φ is the cumulative distribution function of the standard normal. Without loss of generality, we can assume $\mu_w > 0$ and $\mu_y > 0$ because negative ground-truth displacements enjoy the same property. This solution has a good property that larger μ_w or μ_y makes the probability density at the true value, i.e. $\mu_z = \frac{\mu_y}{\mu_w}$, higher, and the tails decay more rapidly. So the estimation of $arctan\theta$, also θ, has smaller noise when μ_w or μ_y is larger. Under the assumption of linear motion, we thus should select two observations with a large temporal difference to estimate the direction.

It is reasonable to assume the noise of detection along the u-axis and v-axis are independent so $\rho = 0$. And when representing the center position in pixel, it is also moderate to assume $\sigma_w = \sigma_y = 1$ (also for the ease of presentation). Then, with different true value of $\mu_z = \frac{\mu_y}{\mu_w}$, the visualizations of $p(z)$ over z and μ_y are shown in Figure 4. The visualization demonstrates our analysis above. Moreover, it shows that when the value of μ_y or μ_w is small, the cluster peak of the distribution at μ_z is not significant anymore, as the noise σ_y and σ_w can be dominant. Considering the visualization shows that happens when μ_y is close to σ_y, this can actually happen when we estimate the speed by observations from two consecutive frames because the variance of observation can be close to the absolute displacement of object motion. This makes another support to our analysis in the main paper about the sensitivity to state estimation noise.



(a) $\mu_z = 0.1$ **(b)** $\mu_z = 0.5$ **(c)** $\mu_z = 2$ **(d)** $\mu_z = 5$

Figure 4: The probability density of $z = tan\theta$ under different true value of z, i.e. $\mu_z = \frac{\mu_y}{\mu_w}$. We set μ_y and z as two variables. It shows that under different settings of true velocity direction when μ_y is smaller, the probability of estimated value with a significant shift from the true value is higher. As μ_y is proportional to the time difference of the two selected observations under linear motion assumption, it relates to the case that the two steps for velocity direction estimation has a shorter time difference.

Table 8: Ablation study about the interpolation post-processing.

	MOT17-val				DanceTrack-val			
	HOTA↑	AssA↑	MOTA↑	IDF1↑	HOTA↑	AssA↑	MOTA↑	IDF1↑
w/o interpolation	66.5	68.9	74.9	77.7	52.1	35.3	87.3	51.6
Linear Interpolation	**68.0**	**69.9**	**77.9**	**79.3**	**52.8**	**35.6**	**89.8**	**52.1**
GPR Interpolation	65.2	67.0	72.9	75.9	51.6	35.0	86.1	51.2

B INTERPOLATION BY GAUSSIAN PROGRESS REGRESSION

Interpolation as post-processing. Although we focus on developing an online tracking algorithm, we are also interested in whether post-process can further optimize the tracking results in diverse conditions. However, we emphasize that usually we only allow interpolation in video object tracking and this makes the process not online anymore. Moreover, interpolation is typically limited to fill the track states on frames where they are missing. As the contrary, the commonly used smoothing techniques in Bayesian estimation, such as fix-lag smoother and fix-interval smoother, are not allowed for object tracking because they require to modify the states already fixed in the tracks on previous time steps instead of just filling missing data points.

Despite the failure of GPR in online tracking in Table 6, we continue to study if GPR is better suited for interpolation in Table 8. We compare GPR with the widely-used linear interpolation. The maximum gap for interpolation is set as 20 frames and we use the same kernel for GPR as mentioned above. The results suggest that the GPR's non-linear interpolation is simply not efficient. We think this is due to limited data points which results in an inaccurate fit of the object trajectory. Further, the variance in regressor predictions introduces extra noise. Although GPR interpolation decreases the performance on MOT17-val significantly, its negative influence on DanceTrack is relatively minor where the object motion is more non-linear. We believe how to fit object trajectory with non-linear hypothesis still requires more study.

From the analysis in the main paper, the failure of SORT can mainly result from occlusion (lack of observations) or the non-linear motion of objects (the break of the linear-motion assumption). So the question arises naturally whether we can extend SORT free of the linear-motion assumption or at least more robust when it breaks.

One way is to extend from KF to non-linear filters, such as EKF [30,51] and UKF [28]. However, for real-world online tracking, they can be hard to be adopted as they need knowledge about the motion pattern or still rely on the techniques fragile to non-linear patterns, such as linearization [29]. Another choice is to gain the knowledge beyond linearity by regressing previous trajectory, such as combing Gaussian Process (GP) [61,47,32]: given a observation $\mathbf{z}_\star$ and a kernel function $k(\cdot,\cdot)$, GP defines gaussian functions with mean $\mu_{\mathbf{z}_\star}$ and variance $\Sigma_{\mathbf{z}_\star}$ as

$$\begin{aligned}\mu_{\mathbf{z}_\star} &= \mathbf{k}_\star^\top[\mathbf{K}+\sigma^2\mathbf{I}]^{-1}\mathbf{y},\\ \Sigma_{\mathbf{z}_\star} &= k(\mathbf{z}_\star,\mathbf{z}_\star) - \mathbf{k}_\star^\top[\mathbf{K}+\sigma^2\mathbf{I}]^{-1}\mathbf{k}_\star,\end{aligned} \tag{13}$$

Table 9: Ablation study about using Gaussian Process Regression for object trajectory interpolation. LI indicates Linear Interpolation, which is used to interpolate the trajectory before smoothing the trajectory by GPR. MT indicates Median Trick for kernel choice in regression. L_τ is the length of trajectory.

	MOT17-val				DanceTrack-val			
Interpolation Method	HOTA	AssA	MOTA	IDF1	HOTA	AssA	MOTA	IDF1
w/o interpolation	66.5	68.9	74.9	77.7	52.1	35.3	87.3	51.6
Linear Interpolation	**69.6**	**69.9**	**77.9**	**79.3**	52.8	**35.6**	89.8	**52.1**
GPR Interp, $l = 1$	66.2	67.6	74.3	76.6	51.8	35.0	86.6	50.8
GPR Interp, $l = 5$	66.3	67.0	72.9	75.9	51.8	35.1	86.5	51.1
GPR Interp, $l = L_\tau$	66.1	67.0	73.1	77.8	51.6	35.1	86.4	50.7
GPR Interp, $l = 1000/L_\tau$	65.9	67.0	73.0	77.8	51.8	35.0	86.9	51.0
GPR Interp, $l =$ MT(τ)	65.9	67.0	73.1	77.8	51.7	35.1	86.7	50.9
LI + GPR Smoothing, $l = 1$	69.5	69.6	77.8	**79.3**	52.8	**35.6**	**89.9**	**52.1**
LI + GPR Smoothing, $l = 5$	69.5	69.7	77.8	**79.3**	52.9	34.9	89.7	**52.1**
LI + GPR Smoothing, $l = L_\tau$	69.6	69.5	77.8	79.2	52.9	**35.6**	**89.9**	**52.1**
LI + GPR Smoothing, $l = 1000/L_\tau$	69.5	**69.9**	77.8	**79.3**	**53.0**	**35.6**	**89.9**	**52.1**
LI + GPR Smoothing, $l =$ MT(τ)	69.5	69.6	77.8	**79.3**	52.8	**35.6**	89.8	**52.1**

where $\mathbf{k}_\star$ is the kernel matrix between the input and training data and $\mathbf{K}$ is the kernel matrix over training data, $\mathbf{y}$ is the output of data. In the main paper, we show a primary study of using Gaussian Process Regression (GPR) in the online generation of the virtual trajectory in OOS and offline interpolation. But neither of them successfully boosts the tracking performance. In this section, We investigate in detail the chance of combining GPR and SORT for multi-object tracking for interpolation as some designs are worth more study.

B.1 Choice of Kernel Function in Gaussian Process

The kernel function is a key variable of GPR. There is not a generally efficient guideline to choose the kernel for Gaussian Process Regression though some basic observations are available [14]. When there is no additional knowledge about the time sequential data to fit, the RBF kernel is one of the most common choices:

$$k(\mathbf{x}, \mathbf{x}') = \sigma^2 \exp\left(-\frac{||\mathbf{x} - \mathbf{x}'||^2}{2l^2}\right), \tag{14}$$

where l is the lengthscale of the data to be fit. It determines the length of the "wiggles" of the target function. σ^2 is the output variance that determines the average distance of the function away from its mean. This is usually just a scale factor [14]. GPR is considered sensitive to l in some situations. So we conduct an ablation study over it in the offline interpolation to see if we can use GPR to outperform the linear interpolation widely used in multi-object tracking.

Table 10: Results on CroHD Head Tracking dataset[55]. Our method uses the detections from HeadHunter[55] or FairMOT[69] to generate new tracks.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP(10^4)↓	FN(10^4)↓	IDs↓	Frag↓
HeadHunter [55]	36.8	57.8	53.9	**5.18**	30.0	4,394	15,146
HeadHunter dets + OC-SORT	39.0	60.0	56.8	**5.18**	28.1	**4,122**	10,483
FairMOT [69]	43.0	60.8	62.8	11.8	19.9	12,781	41,399
FairMOT dets + OC-SORT	**44.1**	**67.9**	**62.9**	10.2	**16.4**	4,243	**10,122**

Table 11: Results on DanceTrack test set. "Ours (MOT17)" uses the YOLOX detector trained on MOT17-training set.

Tracker	HOTA↑	DetA↑	AssA↑	MOTA↑	IDF1↑
SORT	47.9	72.0	31.2	91.8	50.8
Ours	55.1	80.3	38.0	89.4	54.2
Ours (MOT17)	48.6	71.0	33.3	84.2	51.5

Table 12: Results on MOT17 test set with the public detections. LI indicates Linear Interpolation.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP(10^4)↓	FN(10^4)↓	IDs↓	Frag↓	AssA↑	AssR↑
CenterTrack [71]	-	61.5	59.6	1.41	20.1	2,583	-	-	-
QDTrack [41]	-	64.6	65.1	1.41	18.3	2,652	-	-	-
Lif_T [25]	51.3	60.5	65.6	1.50	20.7	1,189	3,476	54.7	59.0
TransCt [66]	51.4	68.8	61.4	2.29	**14.9**	4,102	8,468	47.7	52.8
TrackFormer [38]	-	**62.5**	60.7	3.28	17.5	2,540	-	-	-
Ours	52.4	58.2	65.1	**0.44**	23.0	**784**	2,006	**57.6**	63.5
Ours + LI	**52.9**	59.4	65.7	0.66	22.2	801	**1,030**	57.5	**63.9**

Table 13: Results on MOT20 test set with the public detections. LI indicates Linear Interpolation.

Tracker	HOTA↑	MOTA↑	IDF1↑	FP(10^4)↓	FN(10^4)↓	IDs↓	Frag↓	AssA↑	AssR↑
MPNTrack [6]	46.8	57.6	59.1	17.0	20.1	1,210	1,420	47.3	52.7
TransCt [66]	43.5	61.0	49.8	4.92	**14.8**	4,493	8,950	36.1	44.5
ApLift [26]	46.6	58.9	56.5	1.77	19.3	2,241	2,112	45.2	48.1
TMOH [52]	48.9	60.1	61.2	3.80	16.6	2,342	4,320	48.4	52.9
LPC_MOT [12]	49.0	56.3	62.5	1.17	21.3	1,562	1,865	52.4	54.7
Ours	54.3	59.9	67.0	**0.44**	20.2	554	2,345	59.5	65.1
Ours + LI	**55.2**	**61.7**	**67.9**	0.57	19.2	**508**	**805**	**59.8**	**65.9**

B.2 GPR for Offline Interpolation

In the main paper, we present the use of GPR in online virtual trajectory fitting (Table 8) and offline interpolation (Table 10) where we use $l^2 = 25$ and $\sigma = 1$ for the kernel in Eq. 14. Further, we make a more thorough study of the setting of GPR. We follow the settings of experiments in the main paper that only trajectories longer than 30 frames are put into interpolation. And the interpolation is only applied to the gap shorter than 20 frames. We conduct the experiments on the validation set of MOT17 and DanceTrack.

For the value of l, we try fixed values, i.e. $l = 1$ and $l = 5$ ($2l^2 = 50$), value adaptive to trajectory length, i.e. $l = L_\tau$ and $l = 1000/L_\tau$, and the value output by Median Trick (MT) [18]. The training data is a series of quaternary $[u, v, w, h]$, normalized to zero-mean before being fed into training. The results are shown in Table 9. Linear interpolation is simple but builds a strong baseline as it can stably improve the tracking performance concerning multiple metrics. Directly using GPR to interpolate the missing points hurts the performance and the results of GPR are not sensitive to the setting of l.

There are two reasons preventing GPR from accurately interpolating missing segments. First, the trajectory is usually limited to at most hundreds of steps, providing very limited data points for GPR training to converge. Besides, the missing intermediate data points make the data series discontinuous, causing a huge challenge. We can fix the second issue by interpolating the trajectory with Linear Interpolation (LI) first and then smoothing the interpolated steps by GPR. This outperforms LI on DanceTrack but still regrades over LI on MOT17. This is likely promoted by the non-linear motion on DanceTrack. By fixing the missing data issue of GPR, GPR can have a more accurate trajectory fitting over LI for the non-linear trajectory cases. But considering the outperforming from GPR is still minor compared with the Linear Interpolation-only version and GPR requires much heavier computation overhead, we do not recommend using such a practice in most multi-object tracking tasks. More careful and deeper study is still required on this problem.

C Results on More Benchmarks

Results on HeadTrack [55]. When considering tracking in the crowd, focusing on only a part of the object can be beneficial as it usually suffers less from occlusion than the full body. This line of study is conducted over hand tracking [39,49], human pose [65] and head tracking [55,3,42] for a while. Moreover, with the knowledge of more fine-grained part trajectory, it can be useful in downstream tasks, such as action recognition [15,17] and forecasting [31,7,33,8]. As we are interested in the multi-object tracking in the crowd, we also evaluate the proposed OC-SORT on a recently proposed human head tracking dataset CroHD [55]. To make a fair comparison on only the association performance, we adopt OC-SORT by directing using the detections from existing tracking algorithms. The results are shown in Table 10. The detections of FairMOT [69] and HeadHunter [55] are extracted

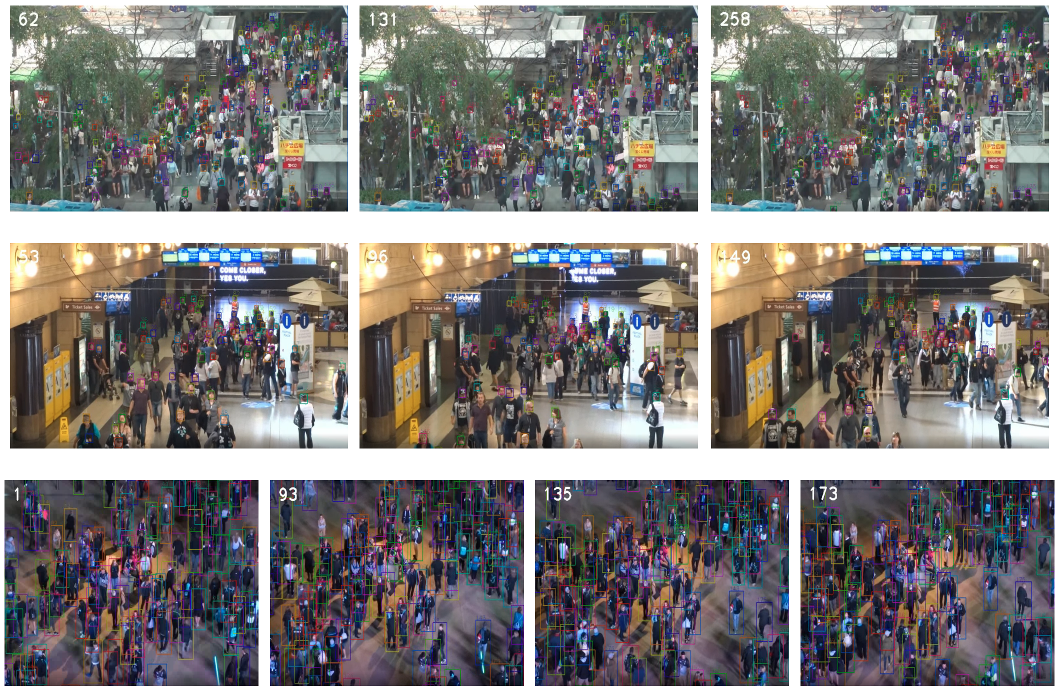


Figure 5: The visualization of the output of OC-SORT on randomly selected samples from the test set of HeadTrack [55] (the first two rows) and MOT20 [13] (the bottom row). These two datasets are both challenging because of the crowded scenes where pedestrians have heavy occlusion with each other. OC-SORT achieves superior performance on both datasets.

from their tracking results downloaded from the official leaderboard [1]. We use the same parameters for OC-SORT as on the other datasets we evaluate on. The results suggest a significant tracking performance improvement compared with the previous methods [55,69] for human body part tracking. But considering the tracking performance is still relatively low (HOTA=$\tilde{4}$0) which is highly related to the tiny size of head targets. Some samples from the test set of HeadTrack are shown in the first two rows of Figure 5.

Public Tracking on MOT17 and MOT20. Although we use the same object detectors as baselines, there is still some variance in detections. Therefore, we also report with the public detections on MOT17/MOT20 in Table 12 and Table 13. OC-SORT still outperforms the existing state-of-the-arts in the public tracking setting. And the outperforming of OC-SORT is more significant on MOT20 which has more severe occlusion scenes. Some samples from the test set of MOT20 are shown in the last row in Figure 5.

D Pseudo-code of OC-SORT

The pseudo-code of OC-SORT is provided in Algorithm. 1 for reference.

E More Results on DanceTrack

To gain more intuition about the improvement of OC-SORT over SORT, we provide more comparisons. In Figure 6, we show more samples where SORT suffers from ID switch or Fragmentation caused by non-linear motion or occlusion but OC-SORT survives. Furthermore, in Figure 7, we show more samples of trajectory visualizations from SORT and OC-SORT on DanceTrack-val set.

As DanceTrack [54] is proposed to emphasize association algorithm so the object detection is relatively easy on it. We train to use the YOLOX [19] trained from the MOT17 training set to provide

[1]https://motchallenge.net/results/Head_Tracking_21/

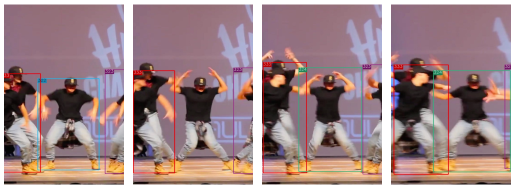
Algorithm 1: Pseudo-code of OCSORT.

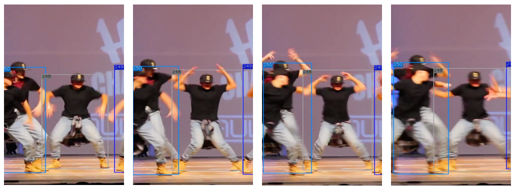
```
Input: Detections $\mathcal{Z} = \{\mathbf{z}_k^i | 1 \leq k \leq T, 1 \leq i \leq N_k\}$; Kalman Filter KF; threshold to remove
       untracked tracks $t_{\text{expire}}$
Output: The set of tracks $\mathcal{T} = \{\tau_i\}$
1  Initialization: $\mathcal{T} \leftarrow \emptyset$ and KF;
2  for timestep $t \leftarrow 1 : T$ do
       /* Step 1: match track prediction with observations */
3      $\mathbf{Z}_t \leftarrow [\mathbf{z}_t^1, ..., \mathbf{z}_t^{N_t}]^\top$ /* Obervations */
4      $\hat{\mathbf{X}}_t \leftarrow [\hat{\mathbf{x}}_t^1, ..., \hat{\mathbf{x}}_t^{|\mathcal{T}|}]^\top$ from $\mathcal{T}$ /* Estimations by KF.predict */
5      $\mathbf{V}_t \leftarrow$ estimated velocity direction from $\mathcal{T}$
6      $C_t \leftarrow C_{\text{IoU}}(\hat{\mathbf{X}}_t, \mathbf{Z}_t) + \lambda C_v(\hat{\mathbf{X}}_t, \mathbf{Z}_t, \mathbf{V}_t)$ /* Cost Matrix with OCM term */
7      Linear assignment by Hungarians with cost $C_t$
8      $\mathcal{T}_t^{\text{matched}} \leftarrow$ tracks matched to an observation
9      $\mathcal{T}_t^{\text{remain}} \leftarrow$ tracks not matched to any observation
10     $\mathbf{Z}_t^{\text{remain}} \leftarrow$ observations not matched to any track
       /* Step 2: perform OCR to find lost tracks back */
11     $\mathbf{Z}^{\mathcal{T}_t^{\text{remain}}} \leftarrow$ last matched observations of tracks in $\mathcal{T}_t^{\text{remain}}$
12     $C_t^{\text{remain}} \leftarrow C_{\text{IoU}}(\mathbf{Z}^{\mathcal{T}_t^{\text{remain}}}, \mathbf{Z}_t^{\text{remain}})$
13     Linear assignment by Hungarians with cost $C_t^{\text{remain}}$
14     $\mathcal{T}_t^{\text{recovery}} \leftarrow$ tracks from $\mathcal{T}_t^{\text{remain}}$ and matched to observations in $\mathbf{Z}^{\mathcal{T}_t^{\text{remain}}}$
15     $\mathbf{Z}_t^{\text{unmatched}} \leftarrow$ observations from $\mathbf{Z}^{\mathcal{T}_t^{\text{remain}}}$ that are still unmatched to tracks
16     $\mathcal{T}_t^{\text{unmatched}} \leftarrow$ tracks from $\mathcal{T}_t^{\text{remain}}$ that are still unmatched to observations
17     $\mathcal{T}_t^{\text{matched}} \leftarrow \{\mathcal{T}_t^{\text{matched}}, \mathcal{T}_t^{\text{recovery}}\}$
       /* Step 3: update status of matched tracks */
18     for $\tau$ in $\mathcal{T}_t^{matched}$ do
19         if $\tau.tracked = False$ then
               /* Perform OOS for track from untracked to tracked */
20             $\mathbf{z}_{t'}^\tau, t' \leftarrow$ The last observation matched to $\tau$ and the time step
21             Rollback KF parameters to $t'$
               /* Generate virtual observation trajectory */
22             $\hat{\mathbf{Z}}_t^\tau \leftarrow [\hat{\mathbf{z}}_{t'+1}^\tau, ..., \hat{\mathbf{z}}_{t-1}^\tau]$
23             Online smooth KF parameters along $\hat{\mathbf{Z}}_t^\tau$
24         end
25         $\tau.tracked = True$
26         $\tau.untracked = 0$
27         Append the new matched associated observation $\mathbf{z}_t^\tau$ to $\tau$'s observation history
28         Update KF parameters for $\tau$ by $\mathbf{z}_t^\tau$
29     end
       /* Step 4: initialize new tracks and remove expired tracks
          */
30     $\mathcal{T}_t^{new} \leftarrow$ new tracks generated from $\mathbf{Z}_t^{\text{unmatched}}$
31     for $\tau$ in $\mathcal{T}_t^{unmatched}$ do
32         $\tau.tracked = False$
33         $\tau.untracked = \tau.untracked + 1$
34     end
35     $\mathcal{T}_t^{\text{reserved}} \leftarrow \{\tau | \tau \in \mathcal{T}_t^{\text{unmatched}}$ and $\tau.untacked < t_{\text{expire}}\}$ /* remove expired
          unmatched tracks */
36     $\mathcal{T} \leftarrow \{\mathcal{T}_t^{\text{new}}, \mathcal{T}_t^{\text{matched}}, \mathcal{T}_t^{\text{reserved}}\}$ /* Conclude */
37 end
38 $\mathcal{T} \leftarrow$ Postprocess($\mathcal{T}$) /* [Optional] offline post-processing */
39 Return: $\mathcal{T}$
```

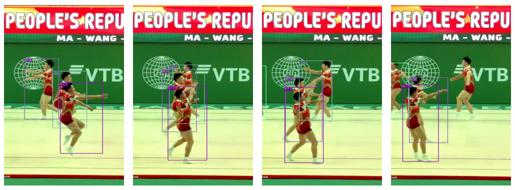
detections on DanceTrack and find the tracking performance of OC-SORT is already higher than the baselines. The results are shown in Table 11.

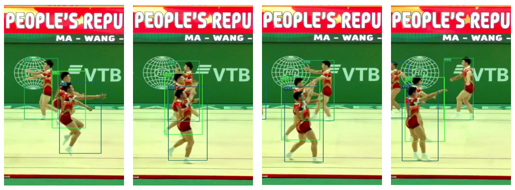
F Reproducibility Statement

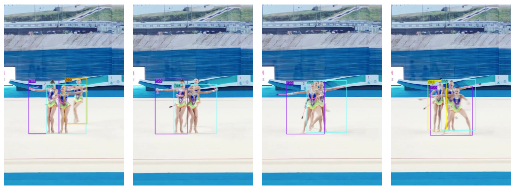
We include the source code, training, and inference scripts as well as the pretrained weights in the supplementary materials to re-produce all the results we report in this paper.

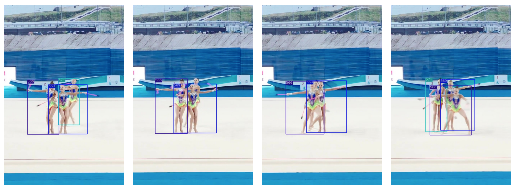

(a) SORT: dancetrack0036

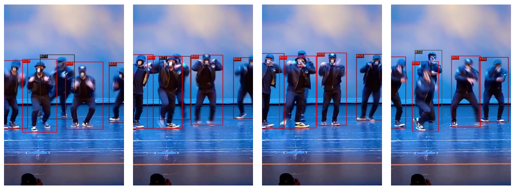

(b) OC-SORT: dancetrack0036

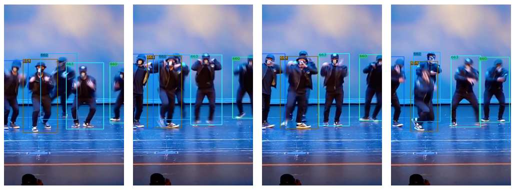

(c) SORT: dancetrack0054

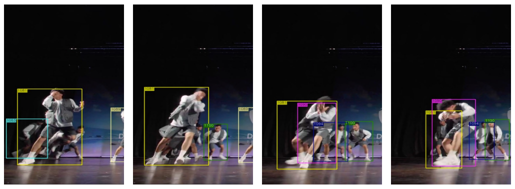

(d) OC-SORT: dancetrack0054

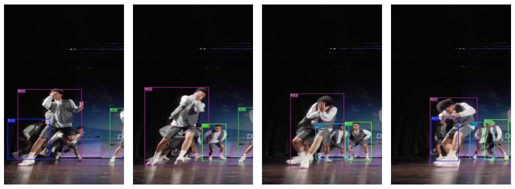

(e) SORT: dancetrack0064

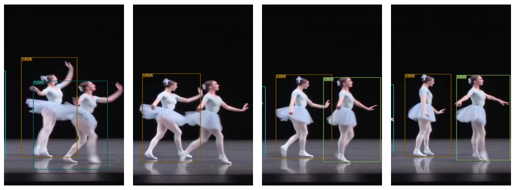

(f) OC-SORT: dancetrack0064


(g) SORT: dancetrack0078


(h) OC-SORT: dancetrack0078


(i) SORT: dancetrack0089


(j) OC-SORT: dancetrack0089


(k) SORT: dancetrack0100

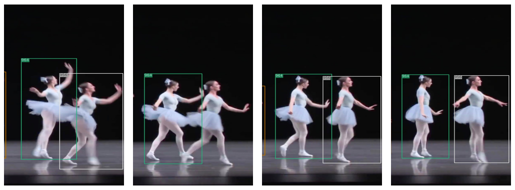
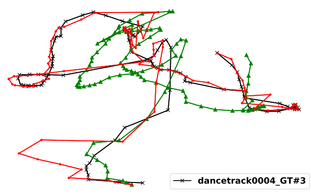
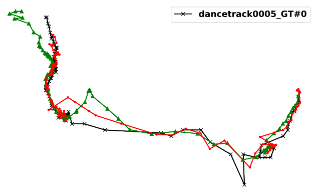

(l) OC-SORT: dancetrack0100

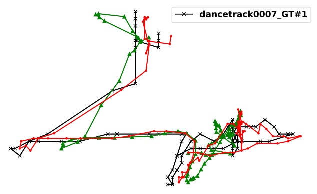
Figure 6: More samples where SORT suffers from the fragmentation and ID switch of tracks from occlusion or non-linear motion but OC-SORT survives. To be precise, the issue happens on the objects by SORT at: (a) #322 → #324; (c) ID switch between #672 and #673, later #673 being lost; (e) #760 → #761; (g) #871 → #872; (i) #1063 → #1090, then ID switch with #1081; (l) #1295 → #1304. We select samples from diverse scenes, including street dance, classic dance and gymnastics. Best viewed in color and zoomed in.



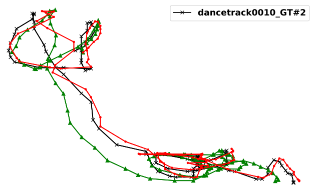
(a) GT #3 on video #0003



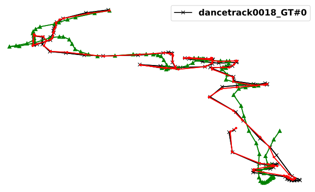
(b) GT #0 on video #0005



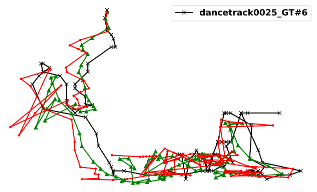
(c) GT #1 on video #0007



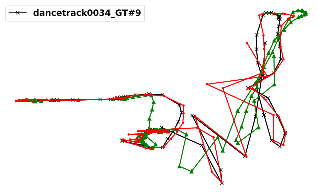
(d) GT #2 on video #0010



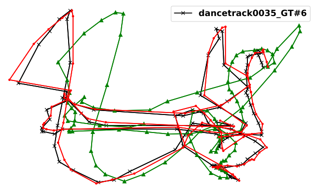
(e) GT #0 on video #0018



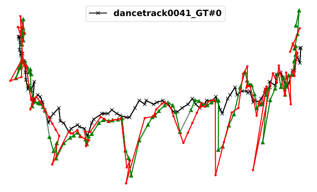
(f) GT #6 on video #0025



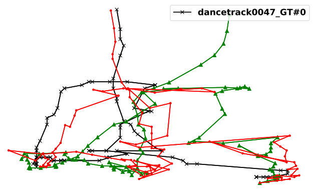
(g) GT #9 on video #0034



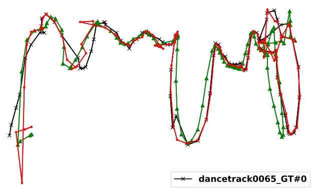
(h) GT #6 on video #0035



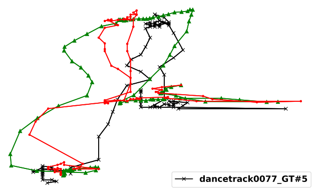
(i) GT #0 on video #0041



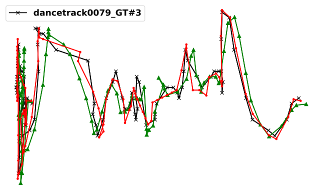
(j) GT #0 on video #0047



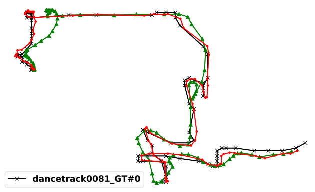
(k) GT #0 on video #0065



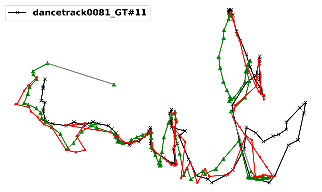
(l) GT #5 on video #0077



(m) GT #3 on video #0079



(n) GT #0 on video #0081



(o) GT #11 on video #0081

Figure 7: Randomly selected object trajectories on the videos from Dance-val set. The **black cross** indicates the ground truth trajectory. The **red dots** indicate the trajectory output by OC-SORT and associated to the selected GT trajectory. The **green triangles** indicate the trajectory output by SORT and associated to the selected GT trajectory. SORT and OC-SORT use the same hyperparameters and detections. Trajectories are sampled at the first 100 frames of each video sequence.