Observation-Centric SORT: Rethinking SORT for Robust Multi-Object Tracking

Jinkun Cao\textsuperscript{1} Jiangmiao Pang\textsuperscript{2} Xinshuo Weng\textsuperscript{3} Rawal Khirodkar\textsuperscript{1} Kris Kitani\textsuperscript{1}
\textsuperscript{1}Carnegie Mellon University \textsuperscript{2}Shanghai AI Laboratory \textsuperscript{3}Nvidia

Abstract

Kalman filter (KF) based methods for multi-object tracking (MOT) make an assumption that objects move linearly. While this assumption is acceptable for very short periods of occlusion, linear estimates of motion for prolonged time can be highly inaccurate. Moreover, when there is no measurement available to update Kalman filter parameters, the standard convention is to trust the priori state estimations for posteriori update. This leads to the accumulation of errors during a period of occlusion. The error causes significant motion direction variance in practice. In this work, we show that a basic Kalman filter can still obtain state-of-the-art tracking performance if proper care is taken to fix the noise accumulated during occlusion. Instead of relying only on the linear state estimate (i.e., estimation-centric approach), we use object observations (i.e., the measurements by object detector) to compute a virtual trajectory over the occlusion period to fix the error accumulation of filter parameters. This allows more time steps to correct errors accumulated during occlusion. We name our method Observation-Centric SORT (OC-SORT). It remains Simple, Online, and Real-Time but improves robustness during occlusion and non-linear motion. Given off-the-shelf detections as input, OC-SORT runs at 700+ FPS on a single CPU. It achieves state-of-the-art on multiple datasets, including MOT17, MOT20, KITTI, head tracking, and especially DanceTrack where the object motion is highly non-linear. The code and models are available at https://github.com/noahcao/OC_SORT.

1. Introduction

We aim to develop a motion model-based multi-object tracking (MOT) method that is robust to occlusion and non-linear motion. Most existing motion model-based algorithms assume that the tracking targets have a constant velocity within a time interval, which is called the linear motion assumption. This assumption breaks in many practical scenarios, but it still works because when the time interval is small enough, the object’s motion 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, we first rethink current motion models and recognize some limitations. Then, we propose addressing them for more robust tracking performance, especially in occlusion.

As the main branch of motion model-based tracking, filtering-based methods assume a transition 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 measurements of target objects, also called “observations”. Observations usually serve as auxiliary information to help update the posteriori parameters of the filter. The trajectories are still extended by the state estimations. Among this line of work, the most widely used one is SORT [3], which uses a Kalman filter (KF) to estimate object states and a linear motion function as the transition
function between time steps. However, SORT shows insufficient tracking robustness when the object motion is non-linear, and no observations are available when updating the filter posteriori parameters.

In this work, we recognize three limitations of SORT. First, 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 significant variance. Also, the noise in the velocity estimate will accumulate into the position estimate by the transition process. Second, the noise of state estimations by KF is accumulated along the time when there is no observation available in the update stage of KF. We show that the error accumulates very fast with respect to the time of the target object’s being untracked. The noise’s influence on the velocity direction often makes the track lost again even after re-association. Last, given the development of modern detectors, the object state by detections usually has lower variance than the state estimations propagated along time steps by a fixed transition function in filters. However, SORT is designed to prolong the object trajectories by state estimations instead of observations.

To relieve the negative effect of these limitations, we propose two main innovations in this work. First, we design a module to use object state observations to reduce the accumulated error during the track’s being lost in a backcheck fashion. To be precise, besides the traditional stages of predict and update, we add a stage of re-update to correct the accumulated error. The re-update is triggered when a track is re-activated by associating to an observation after a period of being untracked. The re-update uses virtual observations on the historical time steps to prevent error accumulation. The virtual observations come from a trajectory generated using the last-seen observation before untracked and the latest observation re-activating this track as anchors. We name it Observation-centric Re-update (ORU).

Besides ORU, the assumption of linear motion provides the consistency of the object motion direction. But this cue is hard to be used in SORT’s association because of the heavy noise in direction estimation. But we propose an observation-centric manner to incorporate the direction consistency of tracks in the cost matrix for the association. 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 Observation-Centric 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, i.e. sensitivity to the noise of state estimations, error accumulation over time, and being estimation-centric;
2. We propose OC-SORT for tracking under occlusion and non-linear motion by fixing SORT’s limitations. It achieves state-of-the-art performance on multiple datasets in an online and real-time fashion.

2. Related Works

Motion Models. Many modern MOT algorithms [3, 11, 63, 70, 73] 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, 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 from a hidden Markov model [44]. Given that the linear motion assumption limits KF, follow-up works like Extended KF [52] 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 and require motion pattern assumption. On the other hand, particle filters [22] solve the non-linear motion by sampling-based posterior estimation but require exponential order of computation. Therefore, these variants of Kalman filter and particle filters are rarely adopted in the visual multi-object tracking and the mostly adopted motion model is still based on Kalman filter [3].

Multi-object Tracking. As a classic computer vision task, visual multi-object tracking is traditionally approached from probabilistic perspectives, e.g. joint probabilistic association [1]. And modern video object tracking is usually built upon modern object detectors [46, 48, 74]. SORT [3] adopts the Kalman filter for motion-based multi-object tracking given observations from deep detectors. DeepSORT [63] further introduces deep visual features [23, 51] into object association under the framework of SORT. Re-identification-based object association [42, 63, 71] has also become popular since then but falls short when scenes are crowded and objects are represented coarsely (e.g. enclosed by bounding boxes), or object appearance is not distinguishable. More recently, transformers [58] have been introduced to MOT [8, 39, 55, 69] 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 both accuracy and time efficiency.
3. Rethink the Limitations of SORT

In this section, we review Kalman filter and SORT [3]. We recognize some of their limitations, which are significant with occlusion and non-linear object motion. We are motivated to improve tracking robustness by fixing them.

3.1. Preliminaries

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 to estimate the target state on the next time step. The filter maintains two variables, the posteriori state estimate \( \hat{x} \) and the posteriori estimate covariance matrix \( P \). In the task of object tracking, we describe the KF process with the transition model \( F \), the observation model \( H \), the process noise \( Q \), and the observation noise \( R \). At each step \( t \), given observations \( z_t \), KF works in an alternation of predict and update stages:

\[
\begin{align*}
\text{predict} & : \quad \hat{x}_{t|t-1} = F \hat{x}_{t-1|t-1} \\
& \quad P_{t|t-1} = F P_{t-1|t-1} F^T + Q_t \\
\text{update} & : \quad \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t (z_t - H_t \hat{x}_{t|t-1}) \\
& \quad P_{t|t} = (I - K_t H_t) P_{t|t-1}
\end{align*}
\]

The stage of predict is to derive the state estimations on the next time step \( t \). Given a measurement of target states on the next step \( t \), the stage of update aims to update the posteriori parameters in KF. Because the measurement comes from the observation model \( H \), it is also called “observation” in many scenarios.

SORT [3] is a multi-object tracker built upon KF. The KF’s state \( x \) in SORT is defined as \( x = [u, v, s, r, \tilde{u}, \tilde{v}, \tilde{s}]^T \), 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, \( \tilde{u}, \tilde{v} \) and \( \tilde{s} \) are the corresponding time derivatives. The observation is a bounding box \( z = [u, v, w, h, c]^T \) with object center position \((u, v)\), object width \( w \), and height \( h \) and the detection confidence \( c \) respectively. SORT assumes linear motion as the transition model \( F \) which leads to the state estimation as

\[
\begin{align*}
u_{t+1} &= u_t + \tilde{u}_t \Delta t, \\
v_{t+1} &= v_t + \tilde{v}_t \Delta t.
\end{align*}
\]

To leverage KF (Eq 1) in SORT for visual MOT, the stage of predict corresponds to estimating the object position on the next video frame. And the observations used for the update stage usually come from a detection model. The update stage is to update Kalman filter parameters and does not directly edit the tracking outcomes.

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. However, 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. SORT uses the priori estimations directly as posterior. We call this “dummy update”, namely

\[
\begin{align*}
\hat{x}_{t|t} &= \hat{x}_{t|t-1}, \\
P_{t|t} &= P_{t|t-1}.
\end{align*}
\]

The philosophy behind such a design is to trust estimations when no observations are available to supervise them. We thus call the tracking algorithms following this scheme “estimation-centric”. However, we will see that this estimation-centric mechanism can cause trouble when non-linear motion and occlusion happen together.

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 show that SORT is sensitive to the noise from KF’s state estimations. To begin with, we 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 \((\mu_u, \mu_v)\) is the underlying true position. Then, if we assume that the state noises are independent on different steps, by Eq.2, the object speed between two time steps, \( t \mapsto t + \Delta t \), is

\[
\begin{align*}
\dot{u} &= \frac{u_{t+\Delta t} - u_t}{\Delta t}, \\
\dot{v} &= \frac{v_{t+\Delta t} - v_t}{\Delta t},
\end{align*}
\]

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, a small \( \Delta t \) will amplify the noise. This suggests that SORT will suffer from the heavy noise of velocity estimation on high-frame-rate videos. The analysis above is simplified from the reality. In practice, velocity won’t be determined by the state on future time steps. For a more strict analysis, please refer to Appendix G.

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 [41] 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. 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 the time, will be able to fix the noise when updating the posteriori parameters. However, we find that such a high sensitivity to state noise introduces significant problems in practice after being amplified by 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. 4, we assume the noise of the object state is i.i.d on different time steps (this is a simplified version, a more detailed analysis is provided in Appendix G). 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 in update to prevent the posteriori state estimation and co-variance, \( \hat{x}_{||t} \) and \( P_{||t} \), deviating from the true value too far away. However, when no observations are provided to KF, it cannot use observation to update its parameters. Then it has to follow Eq. 3 to prolong the estimated trajectory to the next time step. Consider a track is occluded on the time step between \( t \) and \( t + T \) and the noise of speed estimate follows \( \delta u_t \sim N(0, 2\sigma_u^2) \), \( \delta v_t \sim N(0, 2\sigma_v^2) \) for SORT. On the step \( t + T \), state estimation would be

\[
\begin{align*}
    u_{t+T} &= u_t + T\dot{u}_t, \\
    v_{t+T} &= v_t + T\dot{v}_t,
\end{align*}
\]

(5)

whose noise follows \( \delta u_{t+T} \sim N(0, 2T^2\sigma_u^2) \) and \( \delta v_{t+T} \sim N(0, 2T^2\sigma_v^2) \). So without the observations, the estimation from the linear motion assumption of KF results in a fast error accumulation with respect to time. Given \( \sigma_v \) and \( \sigma_u \) 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 \( 50 \times 300 \) pixels. So even assuming the variance of position estimation is only 1 pixel, 10-frame occlusion can accumulate a shift in final position estimation 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 it follows KF to be estimation-centric. It allows update without the existence of observations and purely trusts the estimations. A key difference between state estimations and observations is that we can assume that the observations by an object detector in each frame are affected by i.i.d. noise \( \delta u \sim N(0, \sigma_u^2) \) while the noise in state estimations can be accumulated along the hidden Markov process. Moreover, modern object detectors use powerful object visual features [48, 51]. It makes that, even on a single frame, it is usually safe to assume \( \sigma' < \sigma_u \) and \( \sigma' < \sigma_v \) because the object localization is more accurate by detection than from the state estimations through linear motion assumption. Combined with the previously mentioned two limitations, being estimation-centric makes SORT suffer from heavy noise when there is occlusion and the object motion is not perfectly linear.

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 Re-Update (ORU) 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 Re-Update (ORU)

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 Re-Update (ORU) to reduce the accumulated error. Once a track is associated with an observation again after a period of being untracked (“re-activation”), we backcheck the period of its being lost and re-update the parameters of KF. The re-update is based on “observations” from a virtual trajectory. The virtual trajectory is generated referring to the observations on the steps starting and ending the untracked period. For example, by denoting the last-seen observation before being untracked as $z_{t_1}$ and the observation triggering the re-association as $z_{t_2}$, the virtual trajectory is denoted as

$$\tilde{z}_t = \text{Traj}_{\text{virtual}}(z_{t_1}, z_{t_2}, t), t_1 < t < t_2. \quad (6)$$

Then, along the trajectory of $\tilde{z}_t(t_1 < t < t_2)$, we run the loop of predict and re-update. The re-update operation is

$$\text{re-update} \left\{ \begin{array}{l} K_t = P_{t[t-1]}H_t^T(H_tP_{t[t-1]}H_t^T + R_t)^{-1} \\ \hat{x}_{t|t} = \hat{x}_{t|t-1} + K_t(z_t - H_t\hat{x}_{t|t-1}) \\ P_{t|t} = (I - K_tH_t)P_{t|t-1} \end{array} \right. \quad (7)$$

As the observations on the virtual trajectory match the motion pattern anchored by the last-seen and the latest associated real observations, the update will not suffer from the error accumulated through the dummy update anymore. We call the proposed process Observation-centric Re-Update. It serves as an independent stage outside the predict-update loop and is triggered only a track is re-activated from a period of having no observations.

4.2. Observation-Centric Momentum (OCM)

In a reasonably short time interval, we can approximate the motion as linear. And the linear motion assumption also asks for consistent motion direction. But the noise prevents us from leveraging the consistency of direction. To be precise, to determine the motion direction, we need the object state on two steps with a time difference $\Delta t$. If $\Delta t$ is small, the velocity noise would be significant because of the estimation’s sensitivity to state noise. If $\Delta t$ is big, the noise of direction estimation can also be significant because of the temporal error magnification and the failure of linear motion assumption. As state observations have no problem of temporal error magnification that state estimations suffer from, we propose to use observations instead of estimations to reduce the noise of motion direction calculation and introduce the term of velocity consistency to help the association.

With the new term, given $N$ existing tracks and $M$ de-
...lations on the new-coming time step, the association cost matrix is formulated as

\[
C(\tilde{X}, Z) = C_{\text{IoU}}(\tilde{X}, Z) + \lambda C_{v}(Z, Z),
\]

where \( \tilde{X} \in \mathbb{R}^{N \times 7} \) is the set of object state estimations and \( Z \in \mathbb{R}^{M \times 5} \) is the set of observations on the new time step. \( \lambda \) is a weighting factor. \( Z \) contains the trajectory of observations of all existing tracks. \( C_{\text{IoU}}(\cdot, \cdot) \) calculates the negative pairwise IoU (Intersection over Union) and \( C_{v}(\cdot, \cdot) \) calculates the consistency between the directions of i) linking two observations on an existing track (\( \theta^{\text{track}} \)) and ii) linking a track’s historical observation and a new observation (\( \theta^{\text{intention}} \)). \( C_{v} \) contains all pairs of \( \Delta \theta = |\theta^{\text{track}} - \theta^{\text{intention}}| \).

In our implementation, we calculate the motion direction in radians, namely \( \theta = \arctan\left(\frac{v_{y} - v_{y}^{0}}{u_{x} - u_{x}^{0}}\right) \) where \((u_{1}, v_{1})\) and \((u_{2}, v_{2})\) are the observations on two different time steps. The calculation is also illustrated in Figure 4.

Following the assumptions of noise distribution mentioned before, we can derive a closed-form probability density function of the distribution of the noise in the direction estimation. The derivation 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 the noise of direction estimation is negatively correlated to the time difference between the two observation points, i.e. \( \Delta t \). This suggests increasing \( \Delta t \) to achieve a low-noisy estimation of \( \theta \). However, the assumption of linear motion typically holds only when \( \Delta t \) is small enough. Therefore, the choice of \( \Delta t \) requires a trade-off.

Besides ORU and OCM, we also find it empirically helpful to check a track’s last presence to recover it from being lost. We thus apply a heuristic Observation-Centric Recovery (OCR) technique. OCR will start a second attempt of associating between the last observation of unmatched tracks to the unmatched observations after the usual association stage. It can 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 [41], MOT20 [14], KITTI [20], DanceTrack [54] and CroHD [56]. MOT17 [41] and MOT20 [14] 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. CroHD is a dataset for head tracking in the crowd and the results on it are included in the appendix. DanceTrack [54] is a recently proposed dataset for human tracking. For the data in DanceTrack, 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 under occlusion and non-linear object motion, we would emphasize the comparison on DanceTrack.

Implementation. 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 [70]. For KITTI [20], we use the detections from PermaTrack [57] publicly available in the official release¹. For ORU, we generate the virtual trajectory during occlusion with the constant-velocity assumption. Therefore, Eq. 6 is adopted as \( \tilde{z}_{t} = z_{t_{1}} + \frac{t_{2} - t_{1}}{t_{2} - t_{1}} (z_{t_{2}} - z_{t_{1}}) \), \( t_{1} < t < t_{2} \). 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. 8. 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 proper balance between the accuracy of object detection and association [37]. We also emphasize AssA to evaluate the association performance. IDF1 is also used for association performance evaluation. Other metrics we report, such as MOTA, are highly related to detection performance. It is fair to use these metrics only when all methods use the same detections for tracking, which is referred to as “public tracking” as reported in Appendix C.

5.2. Benchmark Results

Here we report the benchmark results on multiple datasets. We put all methods that use the shared detection results in the blue blocks 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 [70]. 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. Also, ByteTrack uses more detections of low-confidence to achieve higher MOTA scores but we keep the detection confidence threshold the same as on other datasets, which is the common practice in the community. We inherit the linear interpolation on the two datasets by baseline methods 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

---

¹https://github.com/TRI-ML/permatrack/
Table 1. Results on MOT17-test with the private detections. ByteTrack and OC-SORT share detections.

<table>
<thead>
<tr>
<th>Tracker</th>
<th>HOTA↑</th>
<th>MOTA↑</th>
<th>IDF1↑</th>
<th>FP(10^4)↓</th>
<th>FN(10^3)↓</th>
<th>IDs↓</th>
<th>Frags↓</th>
<th>AssA↑</th>
<th>AssR↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fair MOT [71]</td>
<td>59.3</td>
<td>73.7</td>
<td>72.3</td>
<td>2.75</td>
<td>11.7</td>
<td>3,303</td>
<td>8,073</td>
<td>58.0</td>
<td>63.6</td>
</tr>
<tr>
<td>Trans CT [67]</td>
<td>54.5</td>
<td>73.2</td>
<td>62.2</td>
<td>2.31</td>
<td>12.4</td>
<td>4,614</td>
<td>9,519</td>
<td>49.7</td>
<td>54.2</td>
</tr>
<tr>
<td>TransTrk [55]</td>
<td>54.1</td>
<td>75.2</td>
<td>63.5</td>
<td>5.02</td>
<td>8.64</td>
<td>3,603</td>
<td>4,872</td>
<td>47.9</td>
<td>57.1</td>
</tr>
<tr>
<td>GRTU [60]</td>
<td>62.0</td>
<td>74.9</td>
<td>75.0</td>
<td>3.20</td>
<td>10.8</td>
<td>1,812</td>
<td>1,824</td>
<td>62.1</td>
<td>65.8</td>
</tr>
<tr>
<td>QDTrack [42]</td>
<td>53.9</td>
<td>68.7</td>
<td>66.3</td>
<td>2.66</td>
<td>14.7</td>
<td>3,578</td>
<td>8,091</td>
<td>52.7</td>
<td>57.2</td>
</tr>
<tr>
<td>MOTr [69]</td>
<td>57.2</td>
<td>71.9</td>
<td>68.4</td>
<td>2.11</td>
<td>13.6</td>
<td>2,115</td>
<td>3,897</td>
<td>55.8</td>
<td>59.2</td>
</tr>
<tr>
<td>PermaTr [57]</td>
<td>55.5</td>
<td>73.8</td>
<td>68.9</td>
<td>2.90</td>
<td>11.5</td>
<td>3,699</td>
<td>6,132</td>
<td>53.1</td>
<td>59.8</td>
</tr>
<tr>
<td>TransMOT [12]</td>
<td>61.7</td>
<td>76.7</td>
<td>75.1</td>
<td>3.62</td>
<td>9.32</td>
<td>2,346</td>
<td>7,719</td>
<td>59.9</td>
<td>66.5</td>
</tr>
<tr>
<td>GTR [75]</td>
<td>59.1</td>
<td>75.3</td>
<td>71.5</td>
<td>2.68</td>
<td>11.0</td>
<td>2,859</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>DST-Tracker [8]</td>
<td>60.1</td>
<td>75.2</td>
<td>72.3</td>
<td>2.42</td>
<td>11.0</td>
<td>2,724</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>McMOT [5]</td>
<td>56.9</td>
<td>72.5</td>
<td>69.0</td>
<td>2.72</td>
<td>11.5</td>
<td>2,724</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>UniCon [68]</td>
<td>61.7</td>
<td>77.2</td>
<td>75.5</td>
<td>5.01</td>
<td>7.33</td>
<td>5,379</td>
<td>-</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>ByteTrack [70]</td>
<td>63.1</td>
<td>80.3</td>
<td>77.3</td>
<td>2.55</td>
<td>8.37</td>
<td>2,196</td>
<td>2,277</td>
<td>62.0</td>
<td>68.2</td>
</tr>
<tr>
<td>OC-SORT</td>
<td>63.2</td>
<td>78.0</td>
<td>77.5</td>
<td>1.51</td>
<td>10.8</td>
<td>1,950</td>
<td>2,040</td>
<td>63.2</td>
<td>67.5</td>
</tr>
</tbody>
</table>

Table 2. Results on MOT20-test with private detections. ByteTrack and OC-SORT share detections.

<table>
<thead>
<tr>
<th>Tracker</th>
<th>HOTA↑</th>
<th>MOTA↑</th>
<th>IDF1↑</th>
<th>FP(10^4)↓</th>
<th>FN(10^3)↓</th>
<th>IDs↓</th>
<th>Frags↓</th>
<th>AssA↑</th>
<th>AssR↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Fair MOT [71]</td>
<td>54.6</td>
<td>61.8</td>
<td>67.3</td>
<td>10.3</td>
<td>8.89</td>
<td>5,243</td>
<td>7,874</td>
<td>54.7</td>
<td>60.7</td>
</tr>
<tr>
<td>Trans CT [67]</td>
<td>43.5</td>
<td>58.5</td>
<td>49.6</td>
<td>6.42</td>
<td>14.6</td>
<td>4,695</td>
<td>9,581</td>
<td>37.0</td>
<td>45.1</td>
</tr>
<tr>
<td>Semi-TCL [35]</td>
<td>55.3</td>
<td>65.2</td>
<td>70.1</td>
<td>6.12</td>
<td>11.5</td>
<td>4,139</td>
<td>8,508</td>
<td>56.3</td>
<td>60.9</td>
</tr>
<tr>
<td>CSTrack [36]</td>
<td>54.0</td>
<td>66.6</td>
<td>68.6</td>
<td>2.54</td>
<td>14.4</td>
<td>3,196</td>
<td>7,632</td>
<td>54.0</td>
<td>57.6</td>
</tr>
<tr>
<td>GSDT [61]</td>
<td>53.6</td>
<td>67.1</td>
<td>67.5</td>
<td>3.19</td>
<td>13.5</td>
<td>3,131</td>
<td>9,875</td>
<td>52.7</td>
<td>58.5</td>
</tr>
<tr>
<td>TransMOT [12]</td>
<td>61.9</td>
<td>77.5</td>
<td>75.2</td>
<td>3.42</td>
<td>8.08</td>
<td>1,615</td>
<td>2,421</td>
<td>60.1</td>
<td>66.3</td>
</tr>
<tr>
<td>McMOT [5]</td>
<td>54.1</td>
<td>63.7</td>
<td>66.1</td>
<td>4.79</td>
<td>13.8</td>
<td>1,938</td>
<td>-</td>
<td>55.0</td>
<td>-</td>
</tr>
<tr>
<td>ByteTrack [70]</td>
<td>61.3</td>
<td>77.8</td>
<td>75.2</td>
<td>2.62</td>
<td>8.76</td>
<td>1,223</td>
<td>1,460</td>
<td>59.6</td>
<td>66.2</td>
</tr>
<tr>
<td>OC-SORT</td>
<td>62.1</td>
<td>75.5</td>
<td>75.9</td>
<td>1.80</td>
<td>10.8</td>
<td>913</td>
<td>1,198</td>
<td>62.0</td>
<td>67.5</td>
</tr>
</tbody>
</table>

Table 3. Results on DanceTrack test set. Methods in the blue block share the same detections.

<table>
<thead>
<tr>
<th>Tracker</th>
<th>HOTA↑</th>
<th>DetA↑</th>
<th>AssA↑</th>
<th>MOTA↑</th>
<th>IDF1↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>CenterTrack [73]</td>
<td>41.8</td>
<td>78.1</td>
<td>22.6</td>
<td>86.8</td>
<td>35.7</td>
</tr>
<tr>
<td>Fair MOT [71]</td>
<td>39.7</td>
<td>66.7</td>
<td>23.8</td>
<td>82.2</td>
<td>40.8</td>
</tr>
<tr>
<td>QDTrack [42]</td>
<td>45.7</td>
<td>72.1</td>
<td>29.2</td>
<td>83.0</td>
<td>44.8</td>
</tr>
<tr>
<td>TransTrk [55]</td>
<td>45.5</td>
<td>75.9</td>
<td>27.5</td>
<td>88.4</td>
<td>45.2</td>
</tr>
<tr>
<td>TraDes [64]</td>
<td>43.3</td>
<td>74.5</td>
<td>25.4</td>
<td>86.2</td>
<td>41.2</td>
</tr>
<tr>
<td>MOTr [69]</td>
<td>54.2</td>
<td>73.5</td>
<td>42.0</td>
<td>79.7</td>
<td>51.5</td>
</tr>
<tr>
<td>GTR [75]</td>
<td>48.0</td>
<td>72.5</td>
<td>31.9</td>
<td>84.7</td>
<td>50.3</td>
</tr>
<tr>
<td>DST-Tracker [8]</td>
<td>51.9</td>
<td>72.3</td>
<td>34.6</td>
<td>84.9</td>
<td>51.0</td>
</tr>
<tr>
<td>SORT [3]</td>
<td>47.9</td>
<td>72.0</td>
<td>31.2</td>
<td>91.8</td>
<td>50.8</td>
</tr>
<tr>
<td>DeepSORT [63]</td>
<td>45.6</td>
<td>71.0</td>
<td>29.7</td>
<td>87.8</td>
<td>47.9</td>
</tr>
<tr>
<td>ByteTrack [70]</td>
<td>47.3</td>
<td>71.6</td>
<td>31.4</td>
<td>89.5</td>
<td>52.5</td>
</tr>
<tr>
<td>OC-SORT</td>
<td>54.6</td>
<td>80.4</td>
<td>40.2</td>
<td>89.6</td>
<td>54.6</td>
</tr>
<tr>
<td>OC-SORT + Linear Interp</td>
<td>55.1</td>
<td>80.4</td>
<td>40.4</td>
<td>92.2</td>
<td>54.9</td>
</tr>
</tbody>
</table>

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. 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. 8 in Appendix E. We also visualize the output trajectories by OC-SORT and SORT on randomly selected DanceTrack video clips in Fig. 9 in Appendix E. For multi-object tracking in occlusion and non-linear motion, the results on DanceTrack are strong evidence of the effectiveness 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 [57] and report its performance in the table as well. 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 problem, we perform offline head padding (HP) post-processing by writing these entries back after finishing the online tracking stage. The results of the car category on KITTI show an essential shortcoming of the default implementation version of OC-SORT that it chooses the IoU matching for the association. When the objects move fast or the frame rate is low, the IoU of bounding boxes between consecutive frames can be very low or even zero. This issue does not come from the intrinsic design of OC-SORT and is widely observed when using IoU as the association cue. Adding other cues [49, 72, 73] and appearance similarity [38, 63] have been demonstrated [63] effective to solve this. In contrast to the relatively inferior car tracking performance, OC-SORT improves pedestrian tracking performance to a new state-of-the-art. Using the same detections, OC-SORT achieves a large performance gap over PermaTr with 10x faster speed.
Table 4. Results on KITTI-test. Our method uses the same detections as PermaTr [57]

<table>
<thead>
<tr>
<th>Tracker</th>
<th>MOT17</th>
<th>DanceTrack-val</th>
</tr>
</thead>
<tbody>
<tr>
<td>OC-SORT</td>
<td>HOTA↑</td>
<td>MOTA↑</td>
</tr>
<tr>
<td>IMMIDP [65]</td>
<td>68.66</td>
<td>82.75</td>
</tr>
<tr>
<td>SMAT [21]</td>
<td>71.88</td>
<td>83.64</td>
</tr>
<tr>
<td>TrackMPN [45]</td>
<td>72.30</td>
<td>87.33</td>
</tr>
<tr>
<td>MPNetTrack [4]</td>
<td>-</td>
<td>-</td>
</tr>
<tr>
<td>CenterTr [73]</td>
<td>73.02</td>
<td>88.83</td>
</tr>
<tr>
<td>LGM [59]</td>
<td>73.14</td>
<td>87.60</td>
</tr>
<tr>
<td>TuSimple [11]</td>
<td>71.55</td>
<td>86.31</td>
</tr>
<tr>
<td>PermaTr [57]</td>
<td>77.42</td>
<td>90.85</td>
</tr>
<tr>
<td>OC-SORT</td>
<td>74.64</td>
<td>87.81</td>
</tr>
<tr>
<td>OC-SORT + HP</td>
<td>76.54</td>
<td>90.28</td>
</tr>
</tbody>
</table>

Table 5. Ablation on MOT17-val and DanceTrack-val.

<table>
<thead>
<tr>
<th></th>
<th>MOT17-val</th>
<th>DanceTrack-val</th>
</tr>
</thead>
<tbody>
<tr>
<td>ORU</td>
<td>HOTA↑</td>
<td>AssA↑</td>
</tr>
<tr>
<td>OCM</td>
<td>OCR</td>
<td>HOTA↑</td>
</tr>
<tr>
<td>✓</td>
<td>64.9</td>
<td>66.8</td>
</tr>
<tr>
<td>✓</td>
<td>66.3</td>
<td>68.0</td>
</tr>
<tr>
<td>✓✓</td>
<td>66.4</td>
<td>69.0</td>
</tr>
<tr>
<td>✓✓✓</td>
<td>66.5</td>
<td>68.9</td>
</tr>
</tbody>
</table>

Table 6. Ablation on the trajectory hypothesis in ORU.

<table>
<thead>
<tr>
<th></th>
<th>MOT17-val</th>
<th>DanceTrack-val</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>HOTA↑</td>
<td>AssA↑</td>
</tr>
<tr>
<td>✓</td>
<td>Const. Speed</td>
<td>66.5</td>
</tr>
<tr>
<td>✓</td>
<td>GPR</td>
<td>63.1</td>
</tr>
<tr>
<td>✓</td>
<td>Linear Regression</td>
<td>64.3</td>
</tr>
<tr>
<td>✓</td>
<td>Const. Acceleration</td>
<td>66.2</td>
</tr>
</tbody>
</table>

Table 7. Influence from the value of $\Delta t$ in OCM.

<table>
<thead>
<tr>
<th></th>
<th>MOT17-val</th>
<th>DanceTrack-val</th>
</tr>
</thead>
<tbody>
<tr>
<td>✓</td>
<td>HOTA↑</td>
<td>AssA↑</td>
</tr>
<tr>
<td>✓</td>
<td>$\Delta t = 1$</td>
<td>66.1</td>
</tr>
<tr>
<td>✓</td>
<td>$\Delta t = 2$</td>
<td>66.3</td>
</tr>
<tr>
<td>✓</td>
<td>$\Delta t = 3$</td>
<td>66.5</td>
</tr>
<tr>
<td>✓</td>
<td>$\Delta t = 6$</td>
<td>66.0</td>
</tr>
</tbody>
</table>

can still run in an online and real-time fashion.

5.3. Ablation Study

Component Ablation. We ablate the contribution of proposed modules on the validation sets of MOT17 and DanceTrack in Table 5. The splitting of the MOT17 validation set follows a popular convention [73]. The results demonstrate the effectiveness of the proposed modules in OC-SORT. The results show that the performance gain from ORU is significant on both datasets but OCM only shows significant help on DanceTrack dataset where object motion is more complicated and the occlusion is heavy. The ablation study proves the effectiveness of our proposed method to improve tracking robustness in occlusion and non-linear motion.

Virtual Trajectory in ORU. For simplicity, we follow the naive hypothesis of constant speed to generate a virtual trajectory in ORU. 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 $[10] \kappa(x, x') = \exp(-||x-x'||^2/\sigma^2)$.

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.

$\Delta t$ in OCM. There is a trade-off when choosing the time difference $\Delta t$ in OCM (Section 4). A large $\Delta t$ decreases the noise of velocity estimation, but is also likely to discourage approximating object motion as linear. Therefore, we study the influence of varying $\Delta t$ in Table 7. Our results agree with our analysis that increasing $\Delta t$ from $\Delta t = 1$ can boost performance. But increasing $\Delta t$ higher than a best-practice value instead hurts the performance because of the difficulty of maintaining the approximation of linear motion.

6. Conclusion

We analyze the popular motion-based tracker SORT and recognize its intrinsic limitations from using Kalman filter. These limitations significantly 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 keeping simple, online, and real-time. In our experiments on diverse datasets, OC-SORT significantly outperforms the state-of-the-art. The gain is especially significant for multi-object tracking under occlusion and non-linear object motion.

Acknowledgement. We thank David Held, Deva Ramanan, and Siddharth Ancha for the discussion about the theoretical modeling of OC-SORT. We thank Yuda Song for the help with the analysis of OCM error distribution. We also would like to thank Zhengyi Luo, Yuda Song, and Erica Weng for the detailed feedback on the paper writing. We also thank Yifu Zhang for sharing his experience with ByteTrack. This project was sponsored in part by NSF NRI award 2024173.
References


[61] 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

[62] Christopher Williams and Carl Rasmussen. Gaussian pro-
cesses for regression. Advances in neural information pro-
cessing systems, 8, 1995.

[63] 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

[64] 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,

[65] 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 com-

[66] Yuliang Xiu, Jiefeng Li, Haoyu Wang, Yinghong Fang, and
Cewu Lu. Pose flow: Efficient online pose tracking. arXiv

[67] Yihong Xu, Yutong Ban, Guillaume Delorme, Chuang Gan,
Daniela Rus, and Xavier Alameda-Pineda. Transcenter:
Transformers with dense queries for multiple-object track-

[68] Bin Yan, Yi Jiang, Peize Sun, Dong Wang, Zehuan Yuan,
Ping Luo, and Huchuan Lu. Towards grand unification of

[69] Fangao Zeng and et al. Motr: End-to-end multiple-object

[70] 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

[71] Zhaohui Zheng, Ping Wang, Wei Liu, Jinze Li, Rongguang
Ye, and Dongwei Ren. Distance-iou loss: Faster and better
learning for bounding box regression. In Proceedings of the
AAAI conference on artificial intelligence, volume 34, pages

Tracking objects as points. In ECCV, pages 474–490.

[73] Xingyi Zhou, Dequan Wang, and Philipp Krähenbühl. Ob-

[74] Xingyi Zhou, Tianwei Yin, Vladlen Koltun, and Philipp